Лекція №10. Безпека

Загальні принципи безпеки

Інформаційна безпека — це безперервний процес захисту інформаційних систем від погроз трьох видів:

- несанкціонований доступ (НСД) і використання
- порушення цілісності, конфіденційності, автентичності та інших характеристик даних
- порушення доступності та/або повноцінного функціонування інформаційної системи

Підсистема безпеки не є виділеним компонентом ОС і її практично неможливо додати в систему пост-фактум, тобто вона повинна бути вбудована із самого початку.

Принципи створення безпечної системи:

- принцип безпечних налаштувань за замовчуванням
- принцип валідації даних, що надходять від інших учасників системи
- принцип повної медації — завжди перевірка поточних прав
- принцип найменших привілеїв — видавати права, достатні для виконання тільки необхідних операцій і не більше того
- принцип поділу привілеїв — по можливості, вимагати згоди декількох учасників для виконання операцій
- принцип економії на механізмі — механізми захисту мають бути максимально простими з можливих і реалізовуватися на найнижчому з можливих рівні
- принцип мінімального спільного між різними учасниками системи
- прицип відкритого дизайну — архітектура, реалізація і використовувані алгоритми в системі повинні бути відомі, а в секреті можуть триматися тільки обмежені за обсягом авторизаційні дані (ключі, паролі і т.п.)
- принцип психологічної прийнятності

Основні сервіси системи безпеки описуються абревіатурою AAA:

- аутентифікація (Authentication) — встановлення "особистості" сторони, з якої відбувається взаємодія

* pdf generated on 2015-12-31 from markdown sources with SBCL 1.2.16 on Linux 3.16.0-38-generic with 3bmd, cl-pdf & cl-typesetting

- авторизація (Authorization) — перевірка прав на виконання будь-яких операцій у системі
- облік (Accounting) — облік операцій, пов'язаних з системою безпеки (для можливості подальшого розслідування та встановлення причин проблеми)

Способи (фактори) аутентифікації:

- за паролем
- за питанням безпеки
- за одноразовим паролем
- за жетоном (унікальним числом)
- за допомогою сертифіката ЕЦП

Аутентифікація може бути як однофакторноб, так і багатофакторною.

Механізми роботи системи безпеки

В системі безпеки розглядаються 3 сутності: учасники системи (суб'єкти, користувачі), ресурси та права доступу. Одиницею управління є **домен захисту** — це пара об'єкт і право доступу. Домен може відповідати одному суб'єкту або їх групі.

Матриця контролю доступу — це теоретична модель, яка описує матрицю, яка приводить у відповідність всі ресурси системи з усіма її суб'єктами. В комірках цієї матриці знаходяться права доступу конкретного користувача/ролі/групи до конкретного ресурсу. Така матриця дозволяє описати всі права доступу в системі, однак її практичне застосування не ефективно.

На практиці використовуються 2 наступних підходу:

- списки контролю доступу (Access Control Lists, ACL), які реалізують зберігання та облік прав на рівні ресурсів, тобто, фактично, по стовпцях цієї матриці
- мандатні системи (Capability або C-list), які реалізують зберігання прав доступу у суб'єктів системи, тобто по рядках матрицям

У системі, заснованій на ACL, для кожного ресурсу визначений список суб'єктів з їх правами. Наприклад, у ФС Unix в кожної директорії і файлу визначені 3 типи суб'єктів: користувач-власник, група-власник і всі інші,— а також 3 типи прав: читання, запис і виконання. Іншим прикладом ACL є список правил міжмережевого екрану, ресурсами в яких є хости/підмережі та/або можливість звернення до певних портів/використання певних протоколів. У такій системі

* pdf generated on 2015-12-31 from markdown sources with SBCL 1.2.16 on Linux 3.16.0-38-generic with 3bmd, cl-pdf & cl-typesetting

замість прав доступу встановлюються дії екрану при зверненні: дозволити, заборонити, обмежити і т.д. При цьому, враховуючи потенційну необмеженість різних суб'єктів-учасників мережі, в такій системі в основному використовуються узагальнені суб'єкти: всі хости, всі зовнішні хости, всі хости з певної підмережі і т.д.

У системах на основі мандатів мандат видається окремо для кожного суб'єкта на кожне право доступу. Мандат, як правило, реалізується як числовий жетон (token), який видається суб'єкту системою безпеки. Цей жетон може бути:

- просто унікальним числом, яке записується в базу даних для кортежу користувач, домен безпеки. Проблема такого способу в потенційно необмеженій кількості записів в системі з великою кількістю ресурсів та/або суб'єктів. У такій системі аутентифікованому суб'єкту достатньо надати жетон для того, щоб ідентифікувати ресурс, до якого він хоче отримати доступ, і отримати доступ
- криптографічного величиною, отриманою застосуванням односторонньої функції до кортежу, що включає домен безпеки і секретний ключ, відомий тільки ядру системи, `c = f(domain, key)`. У цьому випадку для перевірки мандата суб'єкт повинен надати не тільки сам жетон, але й ідентифікатор ресурсу і права доступу. Перевірка буде проводиться повторним обчисленням значення функції над тими ж аргументами. Безпека системи заснована на неможливості отримати те ж значення жетона без знання секретного ключа. У цій системі утруднений відкликання окремих мандатів, оскільки для відкликання мандата потрібно або змінити ідентифікатор ресурсу, що вплине на всіх суб'єктів, що мають до нього доступ, або змінити ключ, який, як правило, унікальний для користувача, але відкликання ключа призведе до відкликання усіх мандатів цього користувача. Для вирішення цієї проблеми використовуються т.зв. непрямі об'єкти.

Хоча з точки зору моделі Матриці прав доступу в обох системах зберігається одна і та ж інформація, ця модель описує тільки статичні характеристики системи і не описує її поведінки в динаміці.

При розгляді динаміки роботи системи безпеки на основі мандатів володіють наступними перевагами перед списками контролю доступу:

- відсутність необхідності використання загального простору імен ресурсів, відомого всім суб'єктам системи
- можливість більш гранулярного обліку прав: в системі на основі списку контролю доступу адміністраторам системи необхідно вичерпне знання

* pdf generated on 2015-12-31 from markdown sources with SBCL 1.2.16 on Linux 3.16.0-38-generic with 3bmd, cl-pdf & cl-typesetting

про всіх суб'єктів системи — оскільки це психологічно неприйнятно, суб'єкти зазвичай агрегуються більш загальними сутностями — **принципалами** безпеки

В той же час в мандатних системах важче вирішити наступні проблеми:

- обмежити передачу мандату від одного суб'єкта іншому (така можливість також може бути і корисною властивістю системи)
- відкликання мандатів, особливо вибіркове відкликання окремих прав, а не всіх мандатів для якогось суб'єкта

У багатьох реальних системах використовується комбінація обох підходів: наприклад, у файловій системі Unix ACL використовуються для первинного контролю прав, а для перевірки поточних прав використовуються мандат, який видається після первинної авторизації, перевірка якого набагато ефективніше.

Системи на основі мандата часто використовуються для створення т.зв. "пісочниць", наприклад для виконання коду, отриманого з недовірених джерел — оскільки в такій системі простіше реалізувати принцип "по-замовчуванню без доступу".

Реализація системи безпеки

Апаратна платформа надає такі базові механізми для підтримки системи безпеки:

- кільця процесора (CPU rings), які використовуються в сегментной організації пам'яті
- привілейовані інструкції (в деяких платформах)
- рандомізація адресного простору програми, захист стеку і т.п.

Використовуючи ці примітиви ОС вибудовує систему безпеки. Основа цієї системи в більшості ОС:

- виконання всіх критичних операцій в ядрі ОС та надання обмеженого інтерфейсу до них через механізм системних викликів
- разделение пользователей на администраторов (в Unix-системах: особый пользователь `root`) и обычных пользователей

* pdf generated on 2015-12-31 from markdown sources with SBCL 1.2.16 on Linux 3.16.0-38-generic with 3bmd, cl-pdf & cl-typesetting

Література

- Secure Systems Design Principles
- Defensive Programming
- CWE/SANS Top 25 Most Dangerous Software Errors
- Capability Myths Demolished
- Classic Buffer Overflow Explained
- Priviledge Escalation Bug in Linux
- How to Exploit an XSS
- Server compromised due to publicly accessible Redis

* pdf generated on 2015-12-31 from markdown sources with SBCL 1.2.16 on Linux 3.16.0-38-generic with 3bmd, cl-pdf & cl-typesetting