Лабораторна робота №3

Скоробагатько Максим, ФБ-31мн

Тема: Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації.

Мета: Отримання практичних навичок побудови гібридних криптосистем.

Програмна реалізація криптосистем є складним завданням, яке охоплює широкий спектр аспектів, таких як безпека, продуктивність, сумісність і зручність використання. У сучасному світі інформаційних технологій захист даних відіграє вирішальну роль, а криптографія стає ключовою складовою цієї системи. У даному огляді розглянемо основні задачі, які виникають при розробці криптографічного програмного забезпечення, та методи їх вирішення.

Основні задачі програмної реалізації криптосистем

1. Захист ключової інформації

Ключова інформація є критичним елементом будь-якої криптосистеми. Її компрометація може призвести до повного розкриття захищених даних. Основні виклики:

- Безпечне зберігання ключів у пам’яті під час виконання.
- Захист від атак на оперативну пам’ять, таких як cold boot attack або витоки через механізми кешування.
- Контроль доступу до ключів з боку процесів або користувачів із низьким рівнем привілеїв.

2. Контроль правильності функціонування програми

Помилки у програмній реалізації криптографічних алгоритмів можуть призвести до вразливостей:

- Неправильна реалізація криптографічних протоколів.
- Атаки на основі часу виконання операцій або витоків через побічні канали (side-channel attacks).
- Недостатнє тестування і верифікація функціональності.

3. Сумісність і продуктивність

Програмне забезпечення повинно бути сумісним із різними платформами й операційними системами, забезпечуючи при цьому високий рівень продуктивності.

Методи вирішення задач

Контроль доступу до ключової інформації

1. Шифрування ключів у пам’яті:
    - Використання апаратних модулів безпеки (HSM) для зберігання ключів.
    - Використання механізмів, таких як Secure Enclave (MacOS) або Trusted Platform Module (TPM, Windows).
2. Обмеження доступу:
    - Використання SELinux або AppArmor для встановлення політик доступу до пам’яті процесів.
    - Ізоляція критичних процесів у контейнерах або віртуальних машинах.
3. Механізми затирання пам’яті:
    - Затерти чутливу інформацію після використання через функції secure_zero_memory.

Контроль правильності функціонування

1. Верифікація алгоритмів:
    - Використання формальних методів перевірки правильності реалізації.
    - Автоматичне тестування із застосуванням бібліотек на зразок Fuzzing для пошуку прихованих багів.
2. Захист від побічних каналів:

    - Використання алгоритмів із постійним часом виконання (constant-time algorithms).
    - Апаратні контрзаходи, такі як ізоляція кешу.
3. Моніторинг і логування:
    - Застосування спеціалізованих бібліотек для журналювання подій (напр., syslog).
    - Моніторинг аномалій у виконанні програми.

Реалізація для User Endpoint Terminal

User Endpoint Terminal (термінал користувача) є кінцевою точкою, яка потребує спеціалізованого підходу до реалізації криптосистем. Основні аспекти реалізації:

1. Захист ключової інформації:
    - Застосування механізмів ізоляції процесів для захисту ключів. Наприклад, у Windows можна використовувати ізоляцію через Credential Guard.
    - Використання апаратних модулів, таких як TPM, для генерації та зберігання ключів.
    - Шифрування оперативної пам’яті та використання технологій, таких як BitLocker, для захисту збережених даних.
2. Аутентифікація користувачів:
    - Реалізація багатофакторної аутентифікації (наприклад, паролі, апаратні токени, біометрія).
    - Інтеграція з платформами єдиного входу (SSO), такими як OAuth або Active Directory.
3. Контроль правильності функціонування:
    - Постійний моніторинг програмного забезпечення на предмет аномалій і збоїв за допомогою спеціалізованих агентів безпеки (наприклад, Windows Defender Application Control).
    - Автоматичні оновлення програмного забезпечення для виправлення відомих вразливостей.
4. Зручність для користувача:

- Інтуїтивно зрозумілий інтерфейс для керування ключами та сертифікатами.
- Мінімізація затримок у криптографічних операціях через оптимізацію використання апаратних ресурсів.

Приклади реалізації для User Endpoint Terminal

1. BitLocker (Windows):
   - Розроблено корпорацією Microsoft.
   - Використання TPM для зберігання ключів шифрування.
   - Автоматична інтеграція з системними механізмами захисту пам’яті.
   - BitLocker шифрує цілі диски, захищаючи дані від несанкціонованого доступу навіть у разі фізичного викрадення пристрою.
2. macOS Secure Enclave:
   - Реалізовано компанією Apple для пристроїв iOS та macOS.
   - Secure Enclave — це співпроцесор, що забезпечує апаратний захист ключів та інших конфіденційних даних.
   - Інтеграція з API для розробки додатків, які потребують збереження секретів, таких як Touch ID або Face ID.
   - Secure Enclave унікальний своєю архітектурою, яка ізолює конфіденційні дані від основної ОС.
3. Ubuntu AppArmor:
   - Розроблено спільнотою Ubuntu та Canonical.
   - AppArmor надає контроль доступу на основі політик, які обмежують дії процесів.
   - Використовується для захисту криптографічних додатків через ізоляцію доступу до системних ресурсів.
   - Конкурентна перевага — простота налаштування у порівнянні з SELinux, зберігаючи при цьому високий рівень безпеки.

Інтеграція з політиками доступу і каталогами

Реалізація криптографічних систем на терміналах користувачів часто інтегрується з централізованими політиками безпеки через менеджери каталогів, такі як Active Directory (AD). Наприклад:

- BitLocker може застосовувати політики шифрування дисків, встановлені через групові політики AD.
- AD інтегрується з сертифікаційними центрами (CA) для автоматичної видачі та відкликання сертифікатів.
- AppArmor може використовувати централізовані профілі для обмеження доступу додатків відповідно до корпоративних політик.

Порівняння Crypto API та PKCS #11

Crypto API (Microsoft): Crypto API — це набір інтерфейсів, що дозволяють додаткам виконувати криптографічні операції, такі як шифрування, розшифрування, генерація ключів і підпис цифровими сертифікатами. Crypto API інтегрується з інфраструктурою Windows і підтримує CertEnroll API для роботи з сертифікатами. Докладніше: CryptoAPI.

PKCS #11: PKCS #11 — стандартний інтерфейс для роботи з криптографічними токенами, такими як HSM, смарт-карти та USB-токени. PKCS #11 надає уніфікований підхід для роботи з ключами, незалежно від платформи. Докладніше: PKCS #11.

Порівняння

1. Сумісність:
    - Crypto API — працює лише в екосистемі Windows.
    - PKCS #11 — міжплатформений стандарт.
2. Гнучкість:
    - PKCS #11 дозволяє працювати з різними пристроями, такими як смарт-карти та HSM.
    - Crypto API орієнтований на використання в рамках Windows.
3. Простота використання:

    - Crypto API забезпечує просту інтеграцію з системними механізмами Windows.
    - PKCS #11 вимагає додаткової конфігурації для підтримки апаратних пристроїв.
4. Продуктивність:
    - Crypto API інтегрується з апаратними можливостями Windows, що забезпечує високий рівень продуктивності.
    - PKCS #11 може мати варіації продуктивності залежно від конкретної реалізації драйверів для пристроїв.
5. Безпека:
    - Crypto API підтримує сучасні алгоритми шифрування та інтегрується з функціями Windows для захисту пам’яті.
    - PKCS #11 дозволяє реалізовувати ізольовану обробку ключів на апаратному рівні, що робить його ідеальним для HSM.

Приклади використання

Crypto API використовується, наприклад, у Microsoft BitLocker для шифрування дисків. Цей інтерфейс також дозволяє легко інтегруватися з Active Directory для централізованого управління сертифікатами. У свою чергу, PKCS #11 широко застосовується в апаратних модулях безпеки (HSM) таких виробників, як Thales Luna або SafeNet, а також у двофакторній аутентифікації за допомогою смарт-карт, як-от eToken Aladdin.

Висновок

Програмна реалізація криптосистем вимагає ретельного підходу до захисту ключової інформації, забезпечення правильності функціонування алгоритмів і сумісності. Crypto API та PKCS #11 мають свої переваги і недоліки залежно від конкретних вимог до реалізації. Crypto API забезпечує простоту інтеграції в екосистемі Windows, тоді як PKCS #11 надає більше гнучкості для роботи з різними пристроями і платформами.

Реалізація для User Endpoint Terminal додає важливий контекст до вибору інструментів і методів, забезпечуючи безпеку та зручність для кінцевих користувачів. Конкретні приклади, такі як BitLocker, macOS Secure Enclave та Ubuntu AppArmor, демонструють сучасні підходи до захисту ключів та ізоляції процесів. BitLocker надає цілісний захист даних на рівні дисків у Windows. Secure Enclave від Apple відрізняється апаратною ізоляцією конфіденційної інформації, що робить її унікальною для мобільних і десктопних пристроїв. Ubuntu AppArmor виділяється гнучкістю у створенні політик доступу, забезпечуючи високу безпеку в середовищах з відкритим кодом.

Інтеграція з інструментами на зразок Active Directory дозволяє централізувати управління політиками безпеки та доступом, що підвищує ефективність реалізації криптосистем у великих організаціях. У підсумку, вибір між Crypto API та PKCS #11 повинен базуватися на конкретних потребах проекту, з урахуванням платформи, типу пристроїв і рівня безпеки, що вимагається.