Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

ЛАБОРАТОРНА РОБОТА №4

з дисципліни

«МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ»

на тему:

«Дослідження особливостей реалізації існуючих програмних систем,які використовують криптографічні механізми захисту інформації.»

Виконав:

студент групи ФБ-21мн

Проноза Андрій

Перевірила:

Селюх П.В.

Дослідити основні задачі, що виникають при програмній реалізації криптосистем. Запропонувати методи вирішення задачі контролю доступу до ключової інформації, що зберігається в оперативній пам'яті ЕОМ для різних (обраних) операційних систем. Запропонувати методи вирішення задачі контролю правильності функціонування програми криптографічної обробки інформації. Порівняти з точки зору вирішення цих задач інтерфейси Crypto API, PKCS 11. основні задачі, що виникають при програмній реалізації криптосистем Підгрупа 1С. Реалізація для серверу в системі клієнт-сервер

Головним завданням кібербезпеки є забезпечення конфіденційності, цілісності та доступності інформаційних ресурсів. Це є особливо важливим на сьогоднішній день для серверів, а особливо для хмарних технологій. Вади в безпеці можуть з'явитися на будь-якому етапі й можуть бути виявлені не тільки в програмному забезпеченні, а й і у інфраструктурі.

У випадку серверної частини дуже важливо переконатися у тому, що конфіденційна інформація передається та зберігається належним чином, до основних завдань можна віднести:

- запровадити надійне шифрування для всіх даних, що передаються
- Шифрувати конфіденційну інформацію, такі як дані для автентифікації на серверній частині
- Не зберігати паролі або ж будь-яку конфіденційну інформацію на на стороні клієнта у відкритому вигляді
- Не зберігати конфіденційну інформацію в логах серверу
- Перевірити правильність SSL-сертифікатів, особливо чи їх термін дії не закінчився
- Використовувати алгоритми хешування для перевірки цілісності інформації, особливо логів

Для того, щоб гарантувати, що криптографічні механізми відповідають бажаному рівню безпеки необхідне виконання наступних задач

1. Усі криптографічні алгоритми та режими їх роботи повинні бути відкритими, перевіреними та схваленими великою кількістю експертів в області криптографії. Приклади таких алгоритмів можна знайти в, наприклад, стандартах створених Національним інститутом стандартів та технологій (NIST) в США, Федеральним офісом інформаційної безпеки (BSI) в Німеччині, мережі та інформаційні системи (NIS) в ЄС та також ряду державних стандартів України

- ДСТУ 4145-2002 Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка
- ДСТУ 7564:2014 Інформаційні технології. Криптографічний захист інформації. Функція гешування
- ДСТУ 7624:2014 Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення
- ДСТУ 8845:2019 Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення
- ДСТУ ГОСТ 28147:2009. Системи обробки інформації. Захист криптографічний. Алгоритм криптографічного перетворення

2. Усі криптографічні протоколи для захищеного передавання даних мають бути перевіреними та схваленими незалежними експертами. Прикладами таких протоколів можуть бути Secure Shell (SSH) та Transport Layer Security (TLS), оскільки вони були перевірені міжнародним співтовариством проектувальників, провайдерів та операторів IETF
3. Усі реалізації використання криптографічних алгоритмів або протоколів мають пройти повне і незалежне тестування для перевірки правильності їх впровадження.

Загалом же навіть у випадку використання надійних алгоритмів шифрування, керування й зберігання криптографічних ключів є надзвичайно важливим завданням, особливо в сфері хмарних технологій й великих дата-центрів, де кількість таких ключів може бути дуже великою.
Загалом усі ключі можна поділити на дві великі категорії

1. Симетричні
   Використовуються для шифрування/розшифрування з використанням симетричних алгоритмів шифрування, а також для перевірки цілісності повідомлень, що передаються, для цього використовують HMAC, що використовує хеш-функцію разом з секретним ключем, що дозволяє створити унікальний код для кожного повідомлення, який можна використовувати для перевірки автентичності повідомлення. Хеш-функція використовується для створення контрольної суми повідомлення, а секретний ключ додає додатковий рівень безпеки, так як лише особа з доступом до секретного ключа зможе створити коректний HMAC. При передачі даних даний тип ключів може бути одноразовим, тобто для кожного повідомлення, що передається, створюється окремий ключ, або ж такий ключ може створюватися для кожного сеансу зв'язку, наприклад в TLS. Якщо ж ключ використовується для збереження конфіденційності даних, то такий ключ може зберігатися настільки ж довго, як і дані, що шифруються. Також дані ключі можуть бути використані для шифрування інших симетричних, або ж закритих ключів.

2. Пара відкритого та закритого ключів.
   Пара ключів, що використовуються в алгоритмах асиметричної криптографії, а також для створення цифрового підпису. Дані ключі можуть бути використані для аутентифікації сторін при комунікації, наприклад в Transport Layer Security (TLS), або ж в технології Virtual Private Network (VPN). В даному випадку ключі можуть зберігатися достатньо довго, термін дії може сягати трьох років. закритий ключ може бути використаний для створення цифрового підпису, в той час як відкритий ключ може бути використаний для перевірки. Прикладом такого використання можуть бути Secure/Multipart Internet Mail Extensions (S/MIME), що використовується для для забезпечення криптографічної безпеки електронної пошти, воно забезпечує аутентифікацію, цілісність повідомлення і гарантію збереження авторства, безпеку даних. Також може використовуватись для верифікації підпису на збережених даних. В даному випадку термін використання теж може сягати трьох років. Дані ключі в тому числі можуть використовуватись в TLS при передачі конфіденційних даних від клієнту до серверу.

Загальні вимоги до безпек ключів

- Сторони, що здійснюють управління ключами належним чином ідентифікуються й результати процесу авторизації для проведення операції з ключами добре перевірені.
- Перед виконанням команд пов’язаних з ключами й пов’язаних з цим даними сторона, що намагається її виконати, має бути перевірене
- Всі команди управління ключами є захищеними від невиявлених, неавторизованих модифікацій, тобто відбувається перевірка цілісності.
- Відбувається перевірка джерела, що намагається отримати доступ до ключів та метаданих
- Всі ключі та метадані захищені від невиявлених та неавторизованих модифікацій
- Криптостійкість обраних криптографічних механізмів забезпечення конфіденційності, цілісності та доступності усіх операцій з ключами має бути такою ж як і при їх зберіганні.

Контроль доступу до ресурсів

В якості прикладу задач, що необхідно виконати для захисту даних можна навести рекомендації, що надають Amazon Web Services для використання Resource Access Manager

Так з метою захисту даних пропонується:

- надавати користувачам тільки ті дозволи, які дійсно необхідні для виконання їх обов’язків
- використовувати мультифакторну аутентифікацію для кожного з акаунтів
- використовувати SSL/TLS для комунікації з ресурсами AWS. Сертифікат SSL/TLS - цифровий об'єкт, який дозволяє системам перевіряти особистість та внаслідок цього встановлювати зашифроване мережеве з'єднання з іншою системою з допомогою протоколу Secure Sockets Layer/Transport Layer Security (SSL/TLS).
- налаштувати журнал дій в системі
- використовувати рішення шифрування AWS разом з іншими засобами забезпечення безпеки

До основних методів захисту ключової інформації в оперативній пам’яті можна віднести

1. Шифрування
   Шифрування конфіденційної інформації в оперативній пам’яті може попередити неавторизованому доступу до неї, навіть у випадку, якщо атакуюча сторона отримає доступ до пам’яті. Це може бути здійснено як апаратними, так і програмними засобами. В даному випадку також може використовуватись Transparent Data Encryption (TDE), що в основному використовується для шифрування баз даних, коли сегменти пам’яті перед записуванням на диск зашифровуються. Windows надає також Data-Protection API, що дозволяє зашифрувати дані використовуючи облікові дані входу користувача. Проте даний підхід призводить до зменшення продуктивності.
2. Ізоляція пам’яті
   Використовуючи сегментацію пам’яті можна ізолювати різні процеси та програми від несанкціонованого доступу до відповідних сегментів пам’яті, що містять конфіденційну інформацію

3. Використання довіреного середовища виконання
   TEE забезпечують ізольоване та безпечне середовище виконання в головному процесорі, захищаючи критично важливі операції та конфіденційні дані від несанкціонованого доступу.
4. Використання апаратних модулів безпеки
   Сюди відносяться спеціальні пристрої, що надійно зберігають ключі шифрування. У Windows прикладом цього може слугувати Trusted Platform Module, що є криптопроцессором, в якому зберігаються криптографічні ключі
5. Мінімізація даних, що використовуються
   Необхідно зменшити кількість конфіденційних даних, що зберігаються в оперативній пам'яті, це можна зробити шляхом використання таких технік як "laze loading", або ж ліниве завантаження, що фактично означає завантаження до оперативної пам'яті, наприклад ключів, лише коли вони безпосередньо необхідні й хорошою практикою буде одразу після їх використання видаляти їх
6. Постійна перевірка
   Можна використовувати інструменти в операційній системі для виявлення незвичної та підозрілої активності при використанні пам'яті, прикладом таких інструментів в Windows може бути RAMMap, що є розширеною утилітою для аналізу використання фізичної пам'яті, в операційній системі Linux можна скористатися cat/proc/meminfo, що виведе на екран віртуальний файл з інформацією про пам'ять, що використовується
7. Постійні оновлення
   Необхідно регулярно оновлювати операційну систему та використовувані застосунки
8. Модель нульової довіри
   Дана модель передбачає постійну автентифікацію користувачів і пристроїв, шифрування всіх ресурсів, надання мінімального доступу та обмеження його тривалості, сегментація для мінімізації наслідків порушень безпеки даних.

Прикладом для використання контролю доступу та передачі даних може бути "in-memory", або як їх ще називають резидента, nosql база даних Hazelcast. Вона зберігає дані у формі ключ-значення в оперативній пам'яті. Hazelcast не шифрує дані, бо це "дані, що використовуються", а не "дані, що зберігаються" й для безпечного зберігання даних, необхідно забезпечити безпеку самого хоста. Проте він забезпечує захищеність при передачі даних. Це може бути досягнуто двома шляхами. Першим є використання TLS протоколу, другим є використання симетричного шифрування. Хоча перевага надається TLS.

Для вирішення задачі контролю правильності функціонування необхідно ввести вимоги до менеджменту ключів, перевірку яких необхідно проводити.
Процес менеджменту ключів можна умовно поділити на наступні етапи, кожен з яких повинен бути протестований

1. Генерація та збереження ключів
2. Передача ключів потрібним особам
3. Розгортання ключів на серверах застосунків
4. Заміна та виведення з ужитку старих ключів

Генерація ключів
Ключі повинні бути згенерованими з використанням криптографічно стійких алгоритмів. Ключі не повинні бути засновані на загальних словах або фразах, якщо декілька ключів використовуються, то вони повинні бути незалежними один від одного

Термін використання та виведення ключів з ужитку
Ключі повинні бути змінені на основі наступних критеріїв:

- якщо є підозри, що ключ, який нині використовується, може бути скомпрометований, або ж така необхідність може виникнути, якщо хтось, хто мав доступ до цих ключів покинув організацію
- коли запланований термін використання ключі закінчився
- після того, як даним ключем було зашифрована певна кількість даних, так для 64-бітного ключа це може бути 2^35 байтів даних, а для 128-бітного ключа це може бути близько 2^68 байтів даних
- якщо, наприклад було знайдено нову атаку, яка робить використовуваний алгоритм шифрування ненадійним з ключем певної довжини, що використовується

У випадку заміни ключа існують два основні підходи для роботи з уже зашифрованими ключем, що намагатимуться замінити, даними. Перший шлях має на меті розшифрування усіх даних старим ключем та зашифрування їх новим ключем. Другий підхід полягає у маркуванні записів з певним ID ключа, яким його було зашифровано й продовження зберігання старого ключа, щоб мати змогу розшифрувати ці дані. Перший варіант є більш бажаним, оскільки спрощує процес керування ключами й надає загалом більше захищеність, але він не завжди є можливим. Також варто пам’ятати, що старі ключі, навіть після виведення їх з ужитку, мають зберігатися ще певний час у випадку, якщо старі резервні копії даних мають бути розшифровані. Також необхідно готуватися до процесу зміни ключа заздалегідь, щоб мати змогу швидко змінити ключ у випадку його компрометації.

Зберігання ключів
Надійне збереження ключів є дуже важкою для вирішення задачею, оскільки застосунки постійно в тій чи іншій мірі мають мати доступ до ключів, щоб встановити з’єднання, перевірити цифровий підпис чи розшифрувати дані.
Якщо це можливо, то мають використовуватися механізми безпечного зберігання даних, що забезпечуються операційною системою, надавачем хмарних послуг тощо, сюди відносять

- апаратні модулі безпеки - Hardware Security Module (HSM), що є фізичним обчислювальним пристроєм, який захищає та керує секретами, виконує функції шифрування та дешифрування для цифрових підписів, надійну автентифікацію.
- віртуальні HSM, які є програмою, що надає подібні до фізичного HSM послуги
- Сховища ключів, наприклад Azure Key Vault, що є хмарною службою безпеки, що надає надійне та централізоване рішення для зберігання криптографічних ключів
- зовнішні сервіси управління секретами, наприклад HashiCorp Vault, він надає послуги з шифрування даних перед записом їх на жорсткий диск, шифрування та розшифрування даних без їхнього зберігання, що допомагає розробникам не реалізовувати це, також він автоматично слідкує за терміном дії секрету. Також

він здатен відкликати групи секретів, які, наприклад переглядалися або використовувалися певним користувачем

Іншим важливим фактором є зберігання зашифрованих даних та ключів окремо один від одного. Якщо інформація зберігається в базі даних, то ключів мають зберігатися, наприклад, в файловій системі, якщо це можливо, то ключі ліпше зберігати на відокремленій системі. Також самі ключі шифрування ліпше зберігати в зашифрованому вигляді, у цьому випадку їх навіть можна зберігати разом з зашифрованими даними.

PKCS 11 та KMIP
PKCS 11- є одним з Public-key Cryptography Standards, він надає API, що описує функції, які необхідно використовувати для створення, генерації, модифікації та видалення об’єктів типу RSA ключів , DES ключів та x.509 сертифікатів при роботі з апаратними модулями безпек.
KMIP(Key Management Interoperability Protocol) - протокол, що описує маніпуляції з криптографічними ключами на віддаленому сховищі ключів. Він включає симетричні ключі, публічні та приватні ключі, сертифікати , секрети, підписання сертифікатів та описує операції пов’язані з створенням ключів, шифруванням ключів, додаванні та зберіганні ключів на сервері, перевірці сертифікатів, експорт та імпорт ключів при роботі з іншими серверами. В ньому у тому числі контролюється час використання ключів.