Лабораторна робота №8

Аутентифікація користувачів на основі токенів безпеки.

Мета:
Вивчити роботу системи аутентифікації на основі токенів безпеки.

Обладнання:
- персональний комп'ютер з встановленою операційною системою Windows;
- встановлені бібліотеки криптографічного інтерфейсу від Microsoft - Microsoft CryptoAPI;
- флеш-диск для збереження інформації.

Завдання:
1. Будь-якою мовою програмування розробити просту систему аутентифікації користувачів на основі токена безпеки (флеш-диск) з використанням Microsoft CryptoAPI.
2. Перевірити її роботу.

Література:

1. Галицкий А.В., Рябко С.Д., Шаньгин В.Ф. Защита информации в сети. – М.:ДМК Пресс, 2004.
2. Щеглов А.Ю. Защита компьютерной информации от несанкционированного доступа. – СПб.:Наука и техника, 2004.
3. Проскурин В.Г., Крутов С.В., Мацкевич И.В. Защита в операционных системах. – М.: «Радио и связь», 2000.
4. Щербаков А, Домашев А. Прикладная криптография. Использование и синтез криптографических интерфейсов. М.:Русская редакция, 2003.
5. М.А.Деднев, Д.В.Дыльнов, М.А.Иванов Защита информации в банковском деле и электронном бизнесе. М.:Кудиц-образ, 2004. – 512 с.

Теоретичні відомості

Правильне функціонування підсистеми безпеки комп'ютерної системи вимагає реалізації ряду функцій загального призначення, пов'язаних з перетворенням вмісту об'єктів системи (файлів, записів бази даних тощо) або з обчислення деяких спеціальних функцій, які суттєво залежать від вмісту об'єктів. До таких функцій належать алгоритми контролю цілісності об'єктів, аутентифікації та авторизації об'єктів, що керують процесами, а також алгоритми підтримання конфіденційності інформації, що міститься в об'єктах комп'ютерної системи.

Міжнародні та національні стандарти описують ряд добре відомих та вивчених функцій захисного характеру, зокрема алгоритми хешування MD5, MD2, SHA тощо; алгоритми генерування та перевірки електронного цифрового підпису RSA, DSS та інших. Усі ці алгоритми мають різні механізми викликів (зокрема, різну довжину аргументів). Це, у свою чергу, означає, що вони несумісні між собою.

Тому задача вбудовування тих чи інших захисних механізмів в операційну систему на основі якогось одного алгоритму буде виглядати неефективною, особливо, якщо ця ОС розповсюджується в різних регіонах земної кулі. В цьому випадку логічним є побудова «шаруватої» структури, де окремий шар, реалізований, скажемо, як набір динамічних бібліотек, відповідає за захист інформації. Цей спосіб досить універсальний і широко застосовується у сімействі операційних систем Windows. Таким способом можна розв‘язати великий клас задач, пов‘язаних з універсалізацією ОС: від національних налаштувань системи до реалізації різноманітних засобів безпеки.

Зрозуміло, що такі структури повинні мати т.зв. «відкритий інтерфейс», тобто бути детально документованими для того, щоби програмісти могли використати засоби цієї структури при створенні прикладного програмного забезпечення, в тому числі і для захисту інформації.

Сьогодні є достатня кількість криптографічних інтерфейсів, однак найбільшої популярності набув інтерфейс від Microsoft - Microsoft CryptoAPI. Зараз використовується CryptoAPI версії 2.0. Причина популярності цього інтерфейсу полягає в тому, що Microsoft інтенсивно впровадила захисні механізми CryptoAPI у свої операційні системи та прикладне програмне забезпечення. Сучасні ОС сімейства Windows містять багато криптографічних підсистем різного призначення як прикладного рівня, так і рівня ядра. Провідну роль в цьому грають якраз функції CryptoAPI, зокрема базові криптографічні функції, сукупність яких створює інтерфейс CryptoAPI 1.0.

Інтерфейс CryptoAPI 2.0 містить як базові криптографічні функції, так і функції, що реалізують перетворення вищого рівня – роботу з сертифікатами X.509, обробку криптографічних повідомлень PKCS#7 та інші функції, що підтримують інфраструктуру відкритих ключів. Однак набір базових криптографічних функцій цього інтерфейсу утворює CryptoAPI 1.0. Таким чином, функції CryptoAPI 1.0 утворюють криптографічне ядро прикладного рівня для сучасних операційних систем лінійки Windows.

Загальну архітектуру CryptoAPI 1.0 подано на рис. 1.

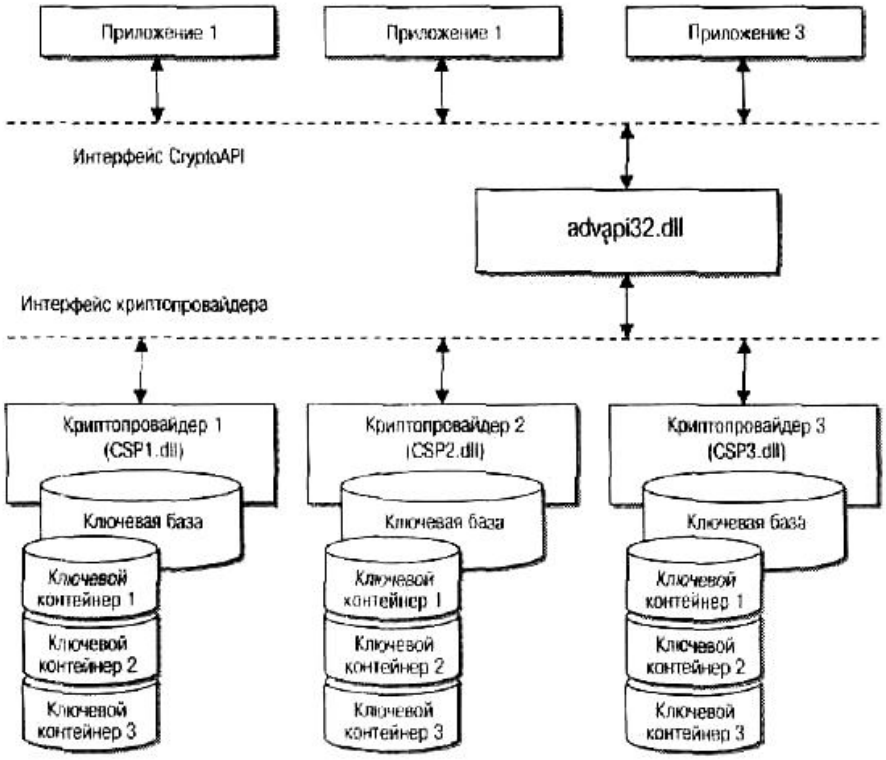

Рис. 1. Загальна архітектура CryptoAPI 1.0.

Усі функції інтерфейсу зосереджено у бібліотеці advapi32.dll. Ці процедури виконують ряд допоміжних функції та викликають бібліотеки, де безпосередньо реалізовано відповідні криптографічні перетворення. Такі бібліотеки називають ***криптопровайдерами***. Криптопровайдери мають стандартний набір функцій, який налічує 23 обов‘язкових та 2 необов‘язкових процедури. Ці процедури подано у таблиці 1.

Таблиця 1

Функції криптопровайдерів

Функція	**Короткий опис**
CryptAcquireContext	Використовується для створення дескриптора ключового контейнера у рамках визначеного криптопровайдера (КП).
CryptContextAddRef	Збільшує на одиницю лічильник посилань на дескриптор КП.
CryptEnumProviders	Використовується для отримання першого та наступних доступних КП.
CryptEnumProviderTypes	Використовується для отримання першого та наступних доступних типів КП.
CryptGetDefaultProvider	Повертає КП, встановлений за замовчуванням

	для вказаного типу КП.
CryptGetProvParam	Повертає параметри КП
CryptReleaseContext	Вивільняє дескриптор КП
CryptSetProvider *CryptSetProviderEx*	Задає тип та назву КП за замовчуванням
CryptSetProvParam	Встановлює параметри КП
CryptDeriveKey	Створює сесійні криптографічні ключі з ключового матеріалу
CryptDestroyKey	Звільняє дескриптор ключа
CryptDuplicateKey	Робить копію криптографічного ключа
CryptExportKey	Експортує криптографічні ключі із заданого контейнера
CryptGenKey	Генерує випадкові криптографічні ключі та ключові пари
CryptGenRandom	Генерує випадкову послідовність та зберігає її в буфері
CryptGetKeyParam	Повертає параметри ключа
CryptImportKey	Імпортує криптографічні ключі з ключового блоба у контейнер КП
CryptSetKeyParam	Встановлює параметри ключа
CryptDecrypt	Виконує операцію розшифрування даних
CryptEncrypt	Виконує операцію за шифрування даних
CryptCreateHash	Створює хешований потік даних
CryptDestroyHash	Знищує об‘єкт хеш функції
CryptDuplicateHash	Створює точну копію хеш-об‘єкта
CryptGetHashParam	Повертає параметри хеш-об‘єкта
CryptHashData	Додає дані до хеш-об‘єкта
CryptHashSessionKey	Підмішує до хеш-об‘єкта сесійний ключ
CryptSetHashParam	Встановлює параметри хеш-об‘єкту
CryptSignHash	Обчислює значення ЕЦП від значення хешу
CryptVerifySignature	Перевіряє ЕЦП заданого значення хешу

Як бачимо з таблиці, CryptoAPI 1.0 підтримує усі основні методи криптографічного перетворення даних: від генерування криптографічних послідовностей випадкових чисел до операцій з електронним цифровим підписом. Таким чином, знаючи інтерфейс CryptoAPI 1.0, програміст може

досить легко реалізувати усі популярні криптографічні алгоритми у своїх прикладних програмах.

Програміст, який працює з цим інтерфейсом, може отримати усю необхідну інформацію про певного криптопровайдера засобами функції *CryptGetProvParam*. Перше, що необхідно знати при цьому – це набір криптографічних стандартів, які реалізують встановлені у системі крипто-провайдери.

Окрім різниці у стандартах, криптопровайдери відрізняються способом фізичної організації збереження ключової інформації. З точки зору програмування спосіб зберігання ключів значення не має, однак він дуже важливий з точки зору експлуатації та безпеки комп‘ютерної системи. Існуючі криптопровайдери Microsoft зберігають ключову інформацію на жорсткому диску (у реєстрі або у файлах), а провайдери інших фірм (GemPlus, Schlumberger та Infineon) – на смарт-картках.

Якщо способи фізичної організації збереження ключової інформації у криптопровайдерів відрізняється, то логічна структура, яка визначається інтерфейсами та з якою мають справу програмісти, однакова для будь-якого типу провайдера. Ключова база визначається набором ключових контейнерів, кожен з яких має ім‘я, що привласнюється йому при створенні, а потім використовується для роботи з ним. У ключовому контейнері зберігається довготривала ключова інформація, наприклад, ключові пари для цифрового підпису або несиметричної системи шифрування.

Тепер розглянемо детально, як функції інтерфейсу CryptoAPI викликають бібліотеки конкретного криптопровайдера. Кожен криптопровайдер має своє власне ім‘я та тип. Його ім‘я – просто рядок, за допомогою якого система його ідентифікує. Так, базовий криптопровайдер Microsoft має назву Microsoft Base Cryptographic Provider v1.0. Тип криптопровайдера – ціле число (у нотації C – DWORD), значення якого ідентифікує набір криптографічних алгоритмів, що підтримуються. Криптопровайдер Microsoft має тип 1, цей тип провайдера реалізує в якості алгоритмів цифрового підпису та обміну ключів алгоритм RSA. Інший базовий криптопровайдер Microsoft, „Microsoft Base DSS and Diffie-Hellman Cryptographic Provider”, має тип 13. Цей тип криптопровайдера реалізує алгоритм цифрового підпису DSS, а в якості алгоритму обміну ключами – протокол Діффі-Хелмана.

Отже, для роботи з набором криптопровайдерами у системному реєстрі міститься список імен усіх криптопровайдерів. З кожним ім‘ям пов‘язаний тип криптопровайдера та ім‘я бібліотеки, яка реалізує його алгоритми.

Окрім цього в системі міститься інформація про те, який криптопровайдер треба застосовувати, якщо користувач явно не вказав конкретне його ім‘я, лише визначивши тип провайдера. Такий криптопровайдер називають провайдером за замовчуванням для заданого типу. Наприклад, для типу 1 провайдером за замовчуванням є Microsoft Base Cryptographic Provider v1.0, а для типу 13 - Microsoft Base DSS and Diffie-Hellman Cryptographic Provider. Для визначення криптопровайдерів за замовчуванням використовують

функцію *CryptGetDefaultProvider,* а для зміни цього параметру – функції *CryptSetProvider* або *CryptSetProviderEx*. Функції дозволяють встановити провайдера за замовчуванням як для поточного користувача, так і для системи в цілому (усіх користувачів). Ці параметри зберігаються у вулику реєстру HKEY_LOCAL_MACHINE. Параметри, встановлені для поточного користувача, мають пріоритет над параметрами, встановленими для усієї системи, та зберігаються у вулику реєстру HKEY_CURRENT_USER. Якщо параметри для поточного користувача відсутні, застосовуються загальносистемні.

Тепер розглянемо, яким чином користувач починає працювати з конкретним криптопровайдером, і як система викликає конкретну бібліотеку, що відповідає обраному криптопровайдеру.

Робота з певним провайдером починається з виклику функції *CryptAcquireContext*, де користувач визначає тип потрібного криптопровайдера, його назву та назву робочого ключового контейнера. В результаті роботи функція повертає користувачу дескриптор криптопровайдера (handle), за допомогою якого користувач в подальшому буде звертатися до нього та передавати його у процедури для виконання усіх необхідних криптографічних операцій.

Детальний опис контексту роботи з криптопровайдерами та приклади (мовою програмування *C*) дивіться у книжці Щербакова Л.Ю., Домашева А.В. «Прикладная криптография».

Власне бібліотеки CryptoAPI разом з файлами заголовків та допомоги постачаються у складі бібліотек MSDN.

Відомості про способи аутентифікації.

Однією з основних функцій систем захисту від несанкціонованого доступу є ідентифікація та аутентифікація. Вона полягає в тому, що жоден суб’єкт (сутність обчислювальної системи, здатна ініціювати виконання операцій) не може отримати доступ до об’єктів (сутностей обчислювальної системи, що захищаються) без надання системі захисту певного обсягу інформації про себе.

При цьому *ідентифікація* суб’єкта полягає в тому, що суб’єкт повідомляє системі захисту свій унікальний ідентифікатор в обчислювальній системі; *аутентифікація* суб'єкта полягає в тому, що суб’єкт надає системі захисту окрім ідентифікуючої інформації ще й певну інформацію, за допомогою якої система перевіряє, що він дійсно є тим суб’єктом, якого стосується ідентифікуюча інформація; *авторизація* суб’єкта відбувається після вдалих ідентифікації та аутентифікації і полягає в тому, що обчислювальна система виконує дії, необхідні для того, щоб суб’єкт мав можливість почати роботу.

Таким чином, щоб отримати доступ в обчислювальну систему, користувач має спочатку ідентифікувати себе, а механізми захисту, в свою чергу, мають підтвердити істинність користувача, тобто підтвердити, що він

дійсно є тим, кого з себе удає. Існує три групи способів підтвердження істинності користувача. Відповідно, для кожної групи механізми підсистеми ідентифікації та аутентифікації мають перевірити:

1) **щось, що користувач знає** (паролі, ідентифікаційні коди, інші відомості);

2) **щось, що користувач має** (ключі, магнітні чи смарт-картки і т.п.);

3) **щось, чим користувач є** (особисті характеристики користувача: відбитки пальців, малюнок сітківки ока, характеристики голосу, особливості користування клавіатурою та маніпуляторами).

Далі розглядатимуться способи, що належать до першої групи, як найбільш поширені.

Якщо перевіряється істинність тільки користувача, то таку процедуру називають одностороннім (peer-entity) підтвердженням істинності. В іншому випадку, тобто коли користувач має підтвердити свою істинність системі, а система, в свою чергу, має підтвердити свою істинність користувачеві, така процедура носить назву двосторонньої (peer-to-peer) аутентифікації.

В разі використання аутентифікації **за простим паролем** кожен користувач обчислювальної системи отримує пару значень – ідентифікатор (ім'я в системі) та пароль. Користувач отримує доступ, якщо вказаний ним в процесі входу в систему ідентифікатор є зареєстрованим, а відповідний пароль – вірним.

Така схема вразлива щодо втрати або розголошення пароля, внаслідок чого одні користувачі можуть видавати себе за інших, тим самим здійснюючи несанкціонований доступ.

Іншим способом є аутентифікація **на основі списку паролів**. При цьому користувачеві разом з ідентифікатором надається список паролів. Перший пароль використовується при першому входів систему, другий – при другому і т. д. Незважаючи на те, що така схема є більш стійкою до втрати окремих паролів, вона має суттєві недоліки, а саме:

- користувачеві незручно запам'ятовувати список паролів;
- у випадку помилки або збою при аутентифікації користувач не знає, користуватись йому поточним чи наступним паролем.

Ще одним способом аутентифікації є **метод одноразових паролів**.

Під час реєстрації користувач генерує певну послідовність, наприклад:

$$F^{999}(x),\ \ldots,\ F(F(F(x))),\ F(F(x)),\ F(x),\ x,$$

де x – випадкове число.

При цьому в системі в цей час зберігається значення $F^{1000}(x)$, на першому кроці в якості паролю користувач використовує значення $F^{999}(x)$. Отримавши його, система обчислює $F(F^{999}(x))$ та перевіряє його на відповідність тому $F^{1000}(x)$, що зберігається. В разі відповідності користувач отримує доступ до системи, а в системі в якості поточного зберігається вже

значення $F^{999}(x)$. На другому кроці перевіряється $F(F^{998}(x)) = F^{999}(x)$ і так далі. Таким чином, пароль, що вже був використаний, а також всі інші, що знаходяться у списку перед ним, стають недійсними. При цьому у випадку порушення синхронізації користувач має можливість перейти до наступного в списку значення, або навіть "перескочити" через один чи кілька паролів, а система вираховує $F(F(\ldots F^{n}(x)\ldots))$ поки не отримає значення, відповідне тому, що зберігається.

Перевірити істинність користувача також можна за допомогою **методу рукостискання** (handshake). При цьому існує процедура f, що відома лише користувачеві та обчислювальній системі. При вході в систему генерується випадкове значення x і обчислюється $f(x)$. Користувач, отримавши x, також обчислює $y = f(x)$ та надсилає його системі. Система порівнює власне значення з отриманим від користувача і робить висновок про його (користувача) істинність. При використанні методу рукостискання ніякої конфіденційної інформації між користувачем і обчислювальною системою не передається взагалі, навіть у шифрованому вигляді. Щодо самої функції $f(x)$, то вона має бути досить складною, щоб зловмисник не міг її вгадати, навіть накопичивши велику кількість пар $(x, f(x))$. В якості процедури $f(x)$ можна використовувати шифрування x на таємному ключі, який є спільним секретом (або шифрування таємного "магічного рядка" на ключі x).

Практична частина

1. Ознайомтеся з криптографічним інтерфейсом Microsoft CryptoAPI.
2. Бригадою з двох студентів напишіть програму, що реалізує аутентифікацію згідно наданому варіанту. При цьому один студент з бригади створює серверну частину програми (ту, що перевіряє), а другий – клієнтську (ту, яку перевіряють). Клієнтська та серверна частини будуть здійснювати взаємодію за протоколом TCP/IP. В якості токена безпеки використайте власний флеш-диск. Уся реєстраційна інформація, необхідна для аутентифікації, повинна міститися у файлі на токені.
3. Відладьте програму, користуючись інтерфейсом "зворотної петлі" (loopback), IP-адреса 127.0.0.1.
4. Виконайте програму з фізично розподіленими по двох комп'ютерах клієнтом і сервером.
5. Оформіть звіт з лабораторної роботи.

Звіт має містити:

1) код клієнтської або серверної частин програми (в залежності від ролі студента у бригаді);
2) склад інформації у файлі на токені безпеки;
3) обговорення стійкості застосованих до файлу засобів захисту;
4) протокол роботи вдалої та невдалої аутентифікації;
5) блок-схему; діаграму класів або модулів, або data-flow diagram (на вибір) та діаграму прецедентів розробленого ПЗ (виконуються за допомогою UML – ПЗ);

6) відповіді на контрольні запитання;
7) висновки з лабораторної роботи.

Варіанти завдань

Номер варіанту	Тип аутентифікації
1	***Проста парольна аутентифікація***. Файл на токені безпеки має містити ідентифікатор користувача і відповідний йому пароль, які заносяться туди під час реєстрації користувача. Під час аутентифікації система зчитує дані з файлу на токені і співставляє з серверною базою даних реєстраційної інформації.
2	***Проста парольна аутентифікація за хеш-образами***. Файл на токені безпеки містить ідентифікатор користувача та хеш-образ пароля, які заносяться туди під час реєстрації користувача. При аутентифікації система зчитує дані з файлу на токені, співставляє з хеш-образом, що міститься у серверній базі даних і повідомляє результат аутентифікації. Алгоритм хешування – довільний з набору CryptoAPI.
3	***Аутентифікація на основі списку паролів***. Список паролів та ідентифікатор користувача містяться у файлі на токені та у системній базі даних серверної частини. Під час реєстрування користувача генерується список паролів, що зберігається як у системній базі даних серверної частини, так і у файлі на токені. При аутентифікації за допомогою генератора випадкових чисел зі списку обирається певний пароль, який надається серверній частині разом з ідентифікатором користувача. Серверна частина перевіряє пару ідентифікатор-пароль і повертає висновок про успішність аутентифікації. Використайте довільний генератор випадкових чисел з набору CryptoAPI.

4	***Аутентифікація з використанням одноразових паролів***. Під час реєстрації за допомогою одного з алгоритмів хешування з набору CryptoAPI знаходять 100 послідовних хеш-образів згенерованого випадкового числа. Останній хеш-образ залишається на серверній частині разом з ідентифікатором користувача, а решта 99 образів зберігаються на токені. При аутентифікації клієнтська частина обирає останній образ з файлу і передає його системі. Система обчислює хеш-образ від отриманого образу, порівнює його з тим, що зберігається на сервері і робить висновок про успішність аутентифікації. Використайте довільний генератор випадкових чисел з набору CryptoAPI.
5	***Аутентифікація з використанням методу "рукостискання".*** Під час реєстрації користувача узгоджується процедура перетворення випадкових чисел, причому в системній базі даних зберігається ідентифікатор та процедура перетворення. При аутентифікації серверна частина генерує випадкове число та передає його клієнтській. Клієнтська частина перетворює отримане число згідно з узгодженим алгоритмом і повертає на сервер. Сервер виконує ту ж операцію, порівнює результати і робить висновок про успішність аутентифікації. В якості процедури перетворення використайте один з алгоритмів хешування з набору CryptoAPI. Генератор випадкових чисел – довільний з CryptoAPI.

Для ускладнення завдання використайте алгоритми шифрування бібліотек CryptoAPI для суцільного шифрування даних, що передаються мережею.

ОБОВ'ЯЗКОВО захистіть файл з реєстраційною інформацією на токені безпеки будь-яким відомим Вам способом.

Контрольні запитання

1. Дайте означення процесів ідентифікації, аутентифікації та авторизації.
2. Які способи підтвердження особи користувача існують? Чим вони відрізняються? Охарактеризуйте їх переваги та недоліки.
3. Де використовуються токени безпеки? Які типи токенів Ви знаєте?
4. Охарактеризуйте призначення та можливості інтерфейсу CryptoAPI.
5. Охарактеризуйте основні способи аутентифікації, розглянуті у цій лабораторній роботі. Назвіть основні області їх застосування, переваги та недоліки.
6. Опишіть та дайте характеристику стійкості Вашого способу захисту файлу на токені безпеки.
7. Опишіть криптографічні функції з набору CryptoAPI, які Ви використали у Вашій програмі.
8. Охарактеризуйте основні атаки на метод аутентифікації, використаний Вами.
9. Запропонуйте способи підсилення стійкості використаного Вами методу аутентифікації до атак п.8.