

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

Технології захисту інформації

**методичні вказівки
до виконання лабораторних робіт**

Київ 2015

Зміст

Лабораторна робота 1. Розробка програми розмежування повноважень користувачів на основі парольної автентифікації

Лабораторна робота 2 Захист додатків від несанкціонованого використання і копіювання

Лабораторная работа №3 Використання функцій криптографічного інтерфейсу Windows для захисту інформації

Лабораторна робота 4 Аналіз механізмів захисту додатку та їх блокування

Лабораторна робота 5 Автоматизований пошук вразливостей у вихідних текстах програмного забезпечення, що написані на мові високого рівня.

Лабораторна робота № 6 Механізми безпеки баз даних

Лабораторна робота 7. Побудова моделі порушника в АС класу 3

Підходи до моделювання атак

Лабораторна робота 8. Механізми захисту операційних систем

Лабораторна робота 1. Розробка програми розмежування повноважень користувачів на основі парольної автентифікації

Мета роботи

Реалізувати алгоритм розмежування користувачів за допомогою парольної автентифікації.

Вміст завдання

Необхідно розробити додаток, який відповідає наступним вимогам:

1. Програма повинна забезпечувати роботу в двох режимах: адміністратора (користувача з фіксованим ім'ям ADMIN) і звичайного користувача.
2. У режимі адміністратора програма повинна підтримувати наступні функції (при правильному введенні пароля):
 - зміна пароля адміністратора (при правильному введенні старого пароля);
 - перегляд списку імен зареєстрованих користувачів і встановлених для них параметрів (блокування облікового запису, включення обмежень на вибрані паролі) – всього списку цілком в одному вікні або по одному елементу списку з можливістю переміщення до його початку або кінця;
 - додавання унікального імені нового користувача до списку з порожнім паролем (рядком нульової довжини);
 - блокування можливості роботи користувача із заданим ім'ям;
 - включення або відключення обмежень на вибрані користувачем паролі (відповідно до індивідуального завдання, визначуваного номера варіанту);
 - завершення роботи з програмою.
3. У режимі звичайного користувача програма повинна підтримувати лише функції зміни пароля користувача (при правильному введенні старого пароля) і завершення роботи, а всі останні функції мають бути заблоковані.
4. У режимі звичайного користувача програма повинна підтримувати лише функції зміни пароля користувача (при правильному введенні старого пароля) і завершення роботи, а всі останні функції мають бути заблоковані '*’.
5. За відсутності введеного у вікні входу імені користувача в списку зареєстрованих адміністратором користувачів програма повинна видавати відповідне повідомлення і надавати користувачеві можливість повторного введення імені або завершення роботи з програмою.
6. При неправильному введенні пароля програма повинна видавати відповідне повідомлення і надавати користувачеві можливість

повторного введення. При трикратному введенні невірною пароля робота програми повинна завершуватися.

7. При заміні пароля програма повинна просити користувача підтвердити введений пароль шляхом його повторного введення.
8. Якщо вибраний користувачем пароль не відповідає необхідним обмеженням (при установці відповідного параметра облікового запису користувача), то програма повинна видавати відповідне повідомлення і надавати користувачеві можливість введення іншого пароля, завершення роботи з програмою (при першому вході даного користувача) або відмови від зміни пароля.
9. Інформація про зареєстрованих користувачів, їх паролі, відсутність блокування їх роботи з програмою, а також включенні або відключенні обмежень на вибрані паролі повинна зберігатися в спеціальному файлі. При першому запуску програми цей файл повинен створюватися автоматично і містити інформацію лише про адміністратора, що має порожній пароль.
10. Інтерфейс з програмою має бути організований на основі меню, обов'язковою частиною якого має бути підменю «Довідка» з командою «Про програму». При виборі цієї команди повинна видаватися інформація про автора програми і видане індивідуальне завдання. Інтерфейс користувача програми може також включати панель управління з дублюючими команди меню графічними кнопками і рядок стану.
11. Для реалізації вказаних в пунктах 2-3 функцій в програмі повинні використовуватися спеціальні діалогові форми, що дозволяють користувачеві (адміністраторові) вводити необхідну інформацію.

12.

Індивідуальні варіанти завдань (обмеження на вибрані паролі)

1. Довжина не менше мінімальної довжини, що встановлюється адміністратором і що зберігається в обліковому записі користувача.
2. Наявність букв і знаків арифметичних операцій.
3. Наявність цифр і знаків арифметичних операцій.
4. Наявність латинських букв і символів кирилиці.
5. Наявність букв, цифр і розділових знаків.
6. Наявність латинських букв, символів кирилиці і цифр.
7. Наявність латинських букв, символів кирилиці і розділових знаків.
8. Наявність рядкових і прописних букв, а також цифр.
9. Наявність рядкових і прописних букв, а також розділових знаків.
10. Наявність рядкових і прописних букв, а також знаків арифметичних операцій.
11. Наявність латинських букв, символів кирилиці і знаків арифметичних операцій.
12. Наявність букв, цифр і знаків арифметичних операцій.

- 13.Наявність букв, розділового і знаків арифметичних операцій знаків.
- 14.Наявність цифр, розділового і знаків арифметичних операцій знаків.
- 15.Відсутність символів, що повторюються.
- 16.Відсутність підряд розташованих однакових символів.
- 17.Неспівпадання з ім'ям користувача.
- 18.Неспівпадання з ім'ям користувача, записаним в зворотному порядку.
- 19.Наявність рядкових і прописних латинських букв, цифр і символів кирилиці.
- 20.Наявність рядкових і прописних букв, цифр і знаків арифметичних операцій.
- 21.Наявність латинських букв, символів кирилиці, цифр і знаків арифметичних операцій.
- 22.Наявність латинських букв, символів кирилиці, цифр і розділових знаків.
- 23.Наявність рядкових і прописних букв, цифр і розділових знаків.
- 24.Наявність рядкових і прописних символів кирилиці, цифр і розділових знаків.
- 25.Наявність рядкових і прописних латинських букв, цифр і знаків арифметичних операцій.

Теоретичні відомості

Засоби мов програмування, що необхідні для виконання роботи

Засоби мови C++, що рекомендуються для розробки програми

1. Тип даних для представлення інформації про обліковий запис користувача програми:

```
Struct {  
//ім'я користувача – рядок в стилі Cі (масив символів)  
//довжина пароля  
//пароль користувача – масив символів  
//признак блокування облікового запису  
//признак включення обмежень на вибрані паролі  
} імя_структури;
```

2. Об'єкт класу fstream файлового потокового вводу-виводу, відкритий в бінарному режимі і такий, що складається із структур приведенного вище типу (визначений в заголовному файлі fstream.h) для представлення файлу облікових записів користувачів програми:

```
fstream імя_файлової_змінної;
```

3. Методи класу fstream для роботи з файлом облікових записів:

/ відкриття існуючого файлу під ім'ям Filename для читання і запису в двійковому режимі */*

*void open(const char *FileName, ios::in|ios::out|ios::binary);*

// створення нового файлу

*void open(const char *FileName, ios::out|ios::binary);*

! – переобтяжена операція, що повертає true, якщо остання операція вводу або виводу завершилася з помилкою

// перевірка існування файлу з ім'ям Filename (функція Borland C++ Builder)

bool FileExists(const AnsiString& FileName);

/ скидання прапора помилки для потоку введення або виводу (необхідний для продовження роботи в програмі з цим потоком) */*

void clear(int=0);

// закриття файлу

void close();

/ переміщення покажчика поточної позиції файлу на off байт відносно dir (можливі значення ios::beg, ios::cur, ios::end) */*

ostream& seekp(long off, seek_dir dir);

/ читання даних з файлу в буфер buf довжини n, рівній довжині структури приведенного вище типу */*

*istream& read(char *buf, int n);*

/ запис даних з буфера buf довжини n, що містить структуру приведенного вище типу, у файл */*

*ostream& write(const char *buf, int n);*

// перевірка досягнення кінця файлу

bool eof();

4. Засоби перевірки встановлених обмежень на вибрані паролі (прототипи функцій визначені в заголовних файлах *string.h* и *stdlib.h*):
/* перетворення об'єкту класу *AnsiString* (значення властивості *Text* об'єкту класу *Cedit*, відповідного компоненту діалогової форми - однорядковому редакторові) в рядок-масив символів, метод класу *AnsiString* */

char c_str();*

// здобуття поточної довжини рядка *S*

*unsigned strlen(const char *S);*

/* здобуття покажчика на символ в рядку *S*, з якого починається перше входження підрядка *Substr*, або NULL, якщо *Substr* не входить в *S* */

char strstr(const char *S, const char *Substr);*

// перетворення рядка *S* в ціле число

*int atoi(const char *S);*

/* здобуття покажчика на перший символ рядка *s1*, що збігся з одним з символів рядка *s2*, або NULL */

*char *strpbrk(char *s1, const char *s2);*

/* здобуття довжини початкового сегменту *s1*, що складається лише з символів, що входять в *s2*, або 0 */

*unsigned strspn(const char *s1, const char *s2);*

/* зміна порядку дотримання символів рядка на зворотний (останній стає першим і так далі) */

*char *strrev(char *s);*

// здобуття дублікату рядка

*char *strdup(const char *s);*

// перевірка символу *ch*

BOOL IsCharAlpha(char ch); // TRUE, якщо *ch* – буква

BOOL IsCharUpper(char ch); // TRUE, якщо *ch* – прописна буква

BOOL IsCharLower(char ch); // TRUE, якщо *ch* – рядкова буква

bool isalpha(char ch); // true, якщо *ch* – латинська буква

bool isdigit(char ch); // true, якщо *ch* – латинська цифра

bool isalnum(char ch); // true, якщо *ch* – латинська буква або цифра

bool isupper(char ch); // true, якщо *ch* – прописна латинська буква

bool islower(char ch); // true, якщо *ch* – рядкова латинська буква

bool ispunct(int ch); /* true, якщо *ch* – друкований символ, що не є латинською буквою, цифрою або пропуском */

‘+’ ‘-’ ‘*’ ‘/’ ‘%’ – символи знаків арифметичних операцій

5. Заміна на екрані символом ‘*’ символів пароля, що вводиться:
Властивість *PasswordChar* компонента Edit (редагований рядок)=’*’

Лабораторна робота 2 Захист додатків від несанкціонованого використання і копіювання

Мета роботи

Оволодіти навиками захисту програмного забезпечення від несанкціонованого використання і копіювання.

Вміст завдання

1. Для програми, розробленої при виконанні лабораторної роботи № 1, написати програму-інсталятор, яка:

- запрошує у користувача папку для установки програми, яка захищається;
- записує туди файл з виконуваним кодом програми;
- збирає інформацію про комп'ютер, на якому встановлюється програма;
- хешує цю інформацію,
- підписує її особистим ключем користувача програми і записує підпис до реєстру Windows в розділ HKEY_CURRENT_USER \ Software \ *Фамілія_студента* як значення параметра Signature.

2. У саму програму, що захищається, включити фрагмент, в якому:

- збирається інформація про комп'ютер, на якому запускається програма;
- обчислюється хеш-значення цієї інформації;
- зчитується підпис із зазначеного вище розділу реєстру, яка перевіряється за допомогою відкритого ключа користувача.

3. При невдалій перевірці робота програми, яка захищається, повинна завершуватися з видачею відповідного повідомлення.

4. Зібрана про комп'ютер інформація включає в себе:

- ім'я користувача,
- ім'я комп'ютера,
- шлях до папки з ОС Windows,
- шлях до папки з системними файлами ОС Windows,
- а також дані, обрані відповідно до виданого завдання.

Індивідуальні варіанти завдань

Інформація, що збирається про комп'ютер

№	Тип та підтип клавіатур	К-ть кнопок миші	Ширина екрану	Висота екрану	Набір дисків пристроїв	Обсяг пам'яті	Дані про диск, на якому встановлена програма
1	Ні	Да	Ні	Да	Ні	Да	Обсяг
2	Ні	Да	Да	Ні	Ні	Да	Файлова система
3	Да	Ні	Ні	Да	Ні	Да	Обсяг
4	Да	Ні	Да	Ні	Ні	Да	Обсяг
5	Ні	Да	Ні	Да	Да	Ні	Файлова система
6	Ні	Да	Да	Ні	Да	Ні	Обсяг
7	Да	Ні	Ні	Да	Да	Ні	Обсяг
8	Да	Ні	Да	Ні	Да	Ні	Обсяг
9	Ні	Да	Ні	Да	Ні	Да	Метка тома
10	Ні	Да	Да	Ні	Ні	Да	Метка тома
11	Да	Ні	Ні	Да	Ні	Да	Метка тома
12	Да	Ні	Да	Ні	Ні	Да	Метка тома
13	Ні	Да	Ні	Да	Да	Ні	Метка тома
14	Ні	Да	Да	Ні	Да	Ні	Метка тома
15	Да	Ні	Ні	Да	Да	Ні	Метка тома
16	Да	Ні	Да	Ні	Да	Ні	Метка тома
17	Ні	Да	Ні	Да	Ні	Да	Серійний №
18	Ні	Да	Да	Ні	Ні	Да	Серійний №
19	Да	Ні	Ні	Да	Ні	Да	Серійний №
20	Да	Ні	Да	Ні	Ні	Да	Серійний №
21	Ні	Да	Ні	Да	Да	Ні	Серійний №
22	Ні	Да	Да	Ні	Да	Ні	Серійний №
23	Да	Ні	Ні	Да	Да	Ні	Серійний №
24	Да	Ні	Да	Ні	Да	Ні	Серійний №

Теоретичні відомості

Засоби мов програмування, що необхідні для виконання роботи

Засоби мови Сі++, що рекомендуються для розробки програми

. Збір інформації про комп'ютер:

// Отримання в буфері lpBuffer довжини nSize імені користувача поточного сеансу

BOOL GetUserName(LPTSTR lpBuffer, LPDWORD nSize);

*/ * Отримання імені комп'ютера в буфері lpBuffer довжини nSize >= MAX_COMPUTERNAME_LENGTH+1 */*

BOOL GetComputerName(LPTSTR lpBuffer, LPDWORD nSize);

*/ * Одержання в буфері lpBuffer довжини uSize >= MAX_PATH шляху до каталогу з ОС Windows */*

UINT GetWindowsDirectory(LPTSTR lpBuffer, UINT uSize);

*/ * Одержання в буфері lpBuffer довжини uSize >= MAX_PATH шляху до системного каталогу Windows */*

UINT GetSystemDirectory(LPTSTR lpBuffer, UINT uSize);

// Отримання типу (nTypeFlag = 0) або підтипу (nTypeFlag = 1) клавіатури

int GetKeyboardType(int nTypeFlag);

*/ * Отримання кількості кнопок миші (nIndex = SM_CMOUSEBUTTONS), ширини (nIndex = SM_CXSCREEN) або висоти (nIndex = SM_CYSCREEN) екрану */*

int GetSystemMetrics(int nIndex);

*/ * Одержання в буфері lpBuffer довжини nBufferLength рядки з кореневими каталогами всіх дисків, розділених 0-символами; результат - довжина отриманого рядка без заключного 0-символу */*

DWORD GetLogicalDriveStrings(DWORD nBufferLength, LPTSTR lpBuffer);

*/ * Одержання в буфері * lpBuffer структури типу MEMORYSTATUS з характеристиками пам'яті комп'ютера (поле dwTotalPhys містить ціле число, рівне загальному обсягу фізичної пам'яті в байтах) */*

VOID GlobalMemoryStatus(LPMEMORYSTATUS lpBuffer);

*/ * Отримання інформації про обсяг поточного диска (lpRootPathName = NULL): кількості секторів в кластері (lpSectorsPerCluster), розмірі сектора (lpBytesPerSector), загальній кількості кластерів (lpTotalNumberOfClusters), lpNumberOfFreeClusters = NULL * /*

*BOOL GetDiskFreeSpace(LPCTSTR lpRootPathName,
LPDWORD lpSectorsPerCluster, LPDWORD lpBytesPerSector,
LPDWORD lpNumberOfFreeClusters,
LPDWORD lpTotalNumberOfClusters);*

*/ * Отримання інформації про поточний диск (lpRootPathName = NULL): мітці томи (в буфері lpVolumeNameBuffer довжини nVolumeNameSize), серійному номері (у змінній * lpVolumeSerialNumber), файлової системи (в буфері lpFileNameBuffer довжини nFileNameSize), lpMaximumComponentLength = NULL, lpFileSystemFlags = NULL * /*

*BOOL GetVolumeInformation(LPCTSTR lpRootPathName,
LPTSTR lpVolumeNameBuffer, DWORD nVolumeNameSize,
LPDWORD lpVolumeSerialNumber,
LPDWORD lpMaximumComponentLength, LPDWORD
lpFileSystemFlags,
LPTSTR lpFileNameBuffer, DWORD nFileNameSize);*

Отримання і перевірка електронного цифрового підпису (ЕЦП) (константи, типи даних і прототипи функцій межах у файлі wincrypt.h):
HCRYPTPROV, HCRYPTKEY, HCRYPTHASH - типи даних для дескрипторів криптопровайдером (CSP), криптографічного ключа, хеш-об'єкта

ALG_ID - тип даних для кодів криптографічних алгоритмів

*/ * Ініціалізація криптопровайдером:
в *phProv записується його дескриптор,
pszContainer=NULL,
pszProvider=NULL,
dwProvType=PROV_RSA_FULL,
dwFlags=0 або (якщо при першому запуску програми CryptAcquireContext повертає FALSE) реєстрація нового користувача в криптопровайдері
dwFlags= CRYPT_NEW_KEYSET */
BOOL CryptAcquireContext(HCRYPTPROV *phProv, LPCSTR pszContainer,
LPCSTR pszProvider, DWORD dwProvType, DWORD dwFlags);*

*/ * Створення в криптопровайдером з дескриптором hProv пари ключів ЕЦП (AlgId = AT_SIGNATURE, dwFlags = 0) і запис дескриптора відкритого ключа в * phKey * /*

*BOOL CryptGenKey(HCRYPTPROV hProv, ALG_ID AlgId,
DWORD dwFlags, HCRYPTKEY *phKey);*

*/ * Отримання у криптопровайдера з дескриптором hProv дескриптора відкритого ключа ЕЦП (dwKeySpec = AT_SIGNATURE) в змінній * phUserKey (якщо функція повертає FALSE, то пару ключів ЕЦП потрібно створити за допомогою функції CryptGenKey) * /*

*BOOL CryptGetUserKey(HCRYPTPROV hProv, DWORD dwKeySpec,
HCRYPTKEY *phUserKey);*

*/ * Створення порожнього хеш-об'єкта (hProv - дескриптор ініціалізувати криптопровайдером, AlgId - код алгоритму хешування, hKey = 0, dwFlags = 0, в * phHash записується дескриптор хеш-об'єкта) * /*

*BOOL CryptCreateHash(HCRYPTPROV hProv, ALG_ID AlgId,
HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH *phHash);*

*/ * Додавання в хеш-об'єкт даних з буфера * pbData довжини dwDataLen (hHash - дескриптор хеш-об'єкта, dwFlags = 0) * /*

*BOOL CryptHashData(HCRYPTHASH hHash, CONST BYTE *pbData,
DWORD dwDataLen, DWORD dwFlags);*

*/ * Отримання для хеш-об'єкта з дескриптором hHash ЕЦП в буфері pbSignature довжини * pdwSigLen (після виконання функції в цю змінну записується фактична довжина ЕЦП); dwKeySpec = AT_SIGNATURE, sDescription = NULL, dwFlags = 0 * /*

*BOOL CryptSignHash(HCRYPTHASH hHash, DWORD dwKeySpec,
LPCTSTR sDescription, DWORD dwFlags, BYTE *pbSignature,
DWORD *pdwSigLen);*

*/ * Перевірка ЕЦП з буфера * pbSignature довжини dwSigLen для хеш-об'єкта з дескриптором hHash за допомогою відкритого ключа hPubKey (sDescription = NULL, dwFlags = 0) * /*

*BOOL CryptVerifySignature(HCRYPTHASH hHash, BYTE *pbSignature,
DWORD dwSigLen, HCRYPTKEY hPubKey, LPCTSTR sDescription,
DWORD dwFlags);*

// Руйнування хеш-об'єкта з дескриптором hHash

BOOL CryptDestroyHash(HCRYPTHASH hHash);

// Руйнування ключа шифрування з дескриптором hKey

BOOL CryptDestroyKey(HCRYPTKEY hKey);

// Звільнення криптопровайдера з дескриптором hProv (dwFlags = 0)

BOOL CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags);

// Робота з реєстром Windows:

Клас TRegistry (визначений у файлі vcl \ registry.hpp):

конструктор без параметрів;

властивості:

HKEY RootKey (кореневий розділ реєстру, за замовчуванням

HKEY_CURRENT_USER);

HKEY CurrentKey (поточний розділ реєстру, тільки для читання);

AnsiString CurrentPath (шлях до поточного розділу реєстру, тільки для читання).

• методи:

/ Відкриття або (якщо CanCreate = true) при необхідності створення
поточного розділу реєстру Key */*

bool OpenKey (const AnsiString Key, bool CanCreate);

/ Запис (перезапис) в поточний розділ реєстру значення параметра Name з
буфера Buffer довжини BufSize */*

*void WriteBinaryData (const AnsiString Name, void * Buffer, int BufSize);*

*// Запис і закриття поточного розділу реєстру
void CloseKey ();*

*// Перевірка існування в реєстрі розділу Key
bool KeyExists (const AnsiString Key);*

/ Читання з поточного розділу реєстру значення параметра Name в буфер*

Buffer довжини BufSize * /

int ReadBinaryData (const AnsiString Name, void * Buffer, int BufSize);

Засоби PHP, що рекомендуються для розробки програми

Php – скрипт для определения моделі та версії браузера:

```
function user_browser($agent) {  
    preg_match("/(MSIE|Opera|Firefox|Chrome|Version|Opera  
Mini|Netscape|Konqueror|SeaMonkey|Camino|Minefield|Iceweasel|K-Meleon|Ma  
xthon)(?:\| | )([0-9.]+)/", $agent, $browser_info);  
    list($browser,$version) = $browser_info;  
    if (preg_match("/Opera ([0-9.]+)/i", $agent, $opera)) return 'Opera  
'. $opera[1];  
    if ($browser == 'MSIE') {  
        preg_match("/(Maxthon|Avant Browser|MyIE2)/i", $agent, $ie);  
        if ($ie) return $ie[1]. ' based on IE ' . $version;  
        return 'IE ' . $version;  
    }  
    if ($browser == 'Firefox') {  
        preg_match("/(Flock|Navigator|Epiphany)\|([0-9.]+)/", $agent, $ff);  
        if ($ff) return $ff[1]. ' ' . $ff[2];  
    }  
    if ($browser == 'Opera' && $version == '9.80') return 'Opera  
' . substr($agent,-5);  
    if ($browser == 'Version') return 'Safari ' . $version;  
    if (!$browser && strpos($agent, 'Gecko')) return 'Browser based on  
Gecko';  
    return $browser. ' ' . $version;  
}
```

PHP та WMI (COM-objects)

Як здійснюється моніторинг або керування машинами через сайт – для цього використовуємо підключення з PHP до підсистеми WMI шляхом створення нових COM-об’єктів.

WMI (інструментарій керування Windows) надає можливість підключення до таких провайдерів:

- Dsprov.dll, провайдер каталога Active Directory (Active Directory provider), дозволяє звертання до Active Directory як до об’єкта WMI;
- Ntevt.dll, провайдер журналу подій (Event Log provider), дає змогу керувати журналом подій;
- Wbemperf.dll, провайдер системних лічильників (Performance Counter provider) — надає доступ до лічильників продуктивності;
- Stdprov.dll, провайдер реєстра (Registry provider), дозволяє здійснювати читання та зміну реєстру;
- Snmpincl.dll, провайдер SNMP-пристроїв (SNMP provider), відкриває шлях доступу до SNMP (Simple Network Management Protocol);
- Wmiprov.dll, провайдер драйверів пристроїв (WDM provider), дає можливість одержувати інформацію низького рівня про драйвери пристроїв Windows Driver Model (WDM);
- Cimwin32.dll, провайдер підсистеми Win32 (Win32 provider), забезпечує доступ до інформації про комп’ютер, ОС, підсистему безпеки, диски, периферійні пристрої, файлові системи, файли, папки, мережні ресурси, принтери, процеси, сервіси і т.д.;
- Msiprov.dll, провайдер встановленого ПЗ (Windows Installer provider) — дозволяє одержувати інформацію про встановлене ПЗ.

Найбільш цікаві та використовані в рамках цієї роботи функції будуть виконувати звертання до провайдерів підсистеми Win32 та провайдеру реєстра.

Наприклад, функція одержання та виводу інформації про версію BIOS, виробника та модель материнської плати. Даний функціонал реалізується за допомогою створення нового COM-об’єкту підключенням до провайдера StdRegProv підсистеми WMI. Всі розглянуті функції визиваються зі змінною \$server, що містить ім’я керованої віддаленої машини в локальній мережі або її IP-адресу:

```
$obj = new COM('winmgmts:{impersonationLevel=impersonate}//$server./root/default:StdRegProv');  
$obj->getStringValue(HKLM,$keypath1,$keyvalue_def,$key);  
echo "BIOS release date: ".$key."r\n";
```

```
$obj->getStringValue(HKLM, $keypath2,$keyvalue_mb_model, $key);  
echo "Mainboard model: ".$key ."\r\n";
```

Після створення нового COM-об'єкта методом `getStringValue` читаємо розділ реєстра `HKLM` (вибір розділу реєстра задається константою, в цьому випадку так: `define('HKLM',0x80000002);`). При необхідності читати інші розділи реєстру задаємо інші константи:

```
Const HKEY_CLASSES_ROOT = 0x80000000  
Const HKEY_CURRENT_USER = 0x80000001  
Const HKEY_LOCAL_MACHINE = 0x80000002  
Const HKEY_USERS = 0x80000003  
Const HKEY_CURRENT_CONFIG = 0x80000005
```

Крім метода `getStringValue` існують додаткові методи роботи із провайдером `StdRegProv`, в тому числі:

- `GetBinaryValue` — читання значень типу `BINARY`;
- `GetDWORDValue` — читання значень типу `DWORD`;
- `GetExpandedStringValue` — читання значень типу `EXPANDED STRING`;
- `GetMultiStringValue` — читання значень типу `MULTI STRING`;
- `CreateKey` — створення ключа реєстру;
- `SetBinaryValue` — запис значення типу `BINARY`;
- `SetDWORDValue` — запис значення типу `DWORD`;
- `SetExpandedStringValue` — запис значення типу `EXPANDED STRING`;
- `SetMultiStringValue` — запис значення типу `MULTI STRING`;
- `SetStringValue` — запис значення рядка;
- `DeleteKey` — видалення ключа;
- `DeleteValue` — видалення значення ключа;
- `EnumKey` — одержати перелік ключів реєстру;
- `EnumValues` — одержати перелік значень ключів;
- `CheckAccess` — перевірка прав доступу до ключа реєстру.

В даному випадку вони не використовуються, детальніше про них можна подивитись в `MSDN`.

Для здійснення функцій моніторинга системи є потреба працювати з простором імен `WMI (/root/cimv2)`, що дає можливість звертатись до провайдерів `Win32_Processor`, `Win32_OperatingSystem`, `Win32_PerfFormattedData_PerfOS_System`, `Win32_OperatingSystem`, `Win32_Process`, `Win32_Service`.

Інформацію про центральний процесор (CPU), в тому числі його завантаженість у відсотках та дані CUID можна одержати так:

```
$obj = new COM ('winmgmts:{impersonationLevel=impersonate}//$server./root/cimv2');
$pc = 0;
foreach ($obj->instancesof('Win32_Processor') as $mp)
{
    echo "Processor (".++$pc.")\r\n";
    echo "Name: ".trim( $mp->Name )." @ " .
    $mp->CurrentClockSpeed . " MHz\r\n";
    echo "CPU Load: ".$mp->LoadPercentage . "%\r\n";
}
```

Тут створюється новий COM-об'єкт. Звертаючись до методів провайдера Win32_Processor, одержуємо необхідну нам інформацію.

Дані про температуру процесора можна одержати, звернувшись до простору імен /root/WMI та провайдера MSAcpi_ThermalZoneTemperature. Одержання температури CPU працює не на всіх материнських платах ПК, однак, майже завжди працює на ноутбуках.

Робота із простором імен /root/WMI та провайдером MSAcpi_ThermalZoneTemperature:

```
$obj = new COM ('winmgmts:{impersonationLevel=impersonate}//$server./root/WMI');

foreach($obj->instancesof ('MSAcpi_ThermalZoneTemperature') as $mp)
{
    echo "<pre>\r\n";
    $ctemp=($mp->CurrentTemperature);
    echo "<b>Current CPU temperature: " . ( $ctemp — 2732)/10 . "C" . "\r\n";
}
```

Перед виводом на сторінку температуру переведено із кельвінів до шкали Цельсія. За допомогою простору імен /root/cimv2 одержуємо інформацію про ОС, service pack, uptime. В даному випадку використано методи Win32_OperatingSystem та Win32_PerfFormattedData_PerfOS_System. Для переведення секунд при відображенні uptime використовується функція format_time(\$temp).

Вивід параметрів операційної системи та uptime:

```
foreach($obj->instancesof('Win32_OperatingSystem') as $mp)
{
    $temp=($mp->Name);
    echo "OS name: " .substr($temp,0,-40). "\r\n";
    $temp2=($mp->ServicePackMajorVersion);
    echo "Service pack: " . $temp2 . "\r\n";
}

foreach ($obj->instancesof('Win32_PerfFormattedData_PerfOS_System') as
$mp)
{
    $temp=($mp->SystemUpTime);
    echo "System uptime: ".format_time($temp)."\r\n";
}
```

Обмеження при підключенні з PHP до підсистеми WMI.

На Windows XP Home Edition даний функціонал не працює. В XP Professional слід перевірити: якщо на віддаленому комп'ютері в локальних політиках безпеки знаходиться модель "Мережний доступ - модель спільного доступу та безпеки - Гостьова", то слід змінити модель на "Класична (звичайна)".

Налаштувати даний параметр безпеки можна, відкривши відповідну політику та розгорнувши дерево консолі наступним чином:

Панель керування\Адміністрування\Локальна політика безпеки\Параметри безпеки\Локальні політики\Параметри безпеки\Мережний доступ - модель спільного доступу та безпеки -> Класична.

По замовчуванню «Гостьова» модель включена в Windows XP Professional, «Класична» — в сімействі Windows Server 2003 та машинах Windows XP Professional, приєднаних к домену.

Під ОС Windows 7, 8 функціонал працює стабільно.

Лабораторная работа №3 Використання функцій криптографічного інтерфейсу Windows для захисту інформації

Зміст завдання

1. У програму, розроблену при виконанні лабораторних робіт №1 і №2, додати засоби захисту від несанкціонованого доступу до файлу з обліковими даними зареєстрованих користувачів: файл з обліковими записами повинен бути зашифрований за допомогою функцій CryptoAPI з використанням сеансового ключа, згенерованого на

основі введеної адміністратором парольної фрази;
при запуску програми файл з обліковими записами повинен розшифровуватися в тимчасовий файл, який після завершення роботи програми повинен бути знову зашифрований для обліку можливих змін в облікових записах користувачів («старий» вміст файлу облікових записів при цьому стирається).

2. Варіанти використання алгоритмів шифрування і хешування при виклику функцій CryptoAPI вибираються відповідно до виданого викладачем завданням.

Використовувані алгоритми шифрування і хешування

№	Тип симетричного шифрування	Використовуваний режим шифрування	Додавання до ключу випадкового значення	Використовуваний алгоритм хешування
1	Блоковий	Електронна кодова книга	Да	MD2
2	Потоковий	-	Да	MD2
3	Блоковий	Зчеплення блоків шифру	Да	MD2
4	Потоковий	-	Да	MD5
5	Блоковий	Зворотній зв'язок по шифротексту	Да	MD2
6	Потоковий	-	Да	SHA
7	Блоковий	Електронна кодова книга	Да	MD4
8	Потоковий	-	Ні	MD2
9	Блоковий	Зчеплення блоків шифру	Да	MD4
10	Потоковий	-	Ні	MD5
11	Блоковий	Зворотній зв'язок по шифротексту	Да	MD4
12	Потоковий	-	Ні	SHA
13	Блоковий	Електронна кодова книга	Да	MD5
14	Блоковий	Зчеплення блоків шифру	Да	MD5
15	Блоковий	Зворотній зв'язок по шифротексту	Да	MD5
16	Блоковий	Електронна кодова книга	Да	SHA
17	Блоковий	Зчеплення блоків шифру	Да	SHA
18	Блоковий	Зворотній зв'язок по шифротексту	Да	SHA
19	Блоковий	Електронна кодова книга	Ні	MD2
20	Блоковий	Зчеплення блоків шифру	Ні	MD2
21	Блоковий	Зворотній зв'язок по	Ні	MD2

		шифротексту		
22	Блоковий	Електронна кодова книга	Ні	MD4
23	Блоковий	Зчеплення блоків шифру	Ні	MD4
24	Блоковий	Зворотній зв'язок по шифротексту	Ні	MD4
25	Блоковий	Електронна кодова книга	Ні	MD5
26	Блоковий	Зчеплення блоків шифру	Ні	MD5
27	Блоковий	Зворотній зв'язок по шифротексту	Ні	MD5
28	Блоковий	Електронна кодова книга	Ні	SHA
29	Блоковий	Зчеплення блоків шифру	Ні	SHA
30	Блоковий	Зворотній зв'язок по шифротексту	Ні	SHA

Рекомендовані для розробки програми засоби мови Cі ++

1. Файл облікових записів зареєстрованих користувачів для операцій шифрування (розшифрування).

Об'єкт класу `fstream`, відкритий в довічному режимі (визначений у файлі `fstream.h`).

2. Робота з файлом облікових записів (методи класу `fstream`):

`/* Відкриття існуючого файлу під ім'ям FileName для читання в бінарний режимі */`

```
void open (const char * FileName, ios :: in | ios :: binary);
```

`// Створення нового файлу з ім'ям FileName`

```
void open (const char * FileName, ios :: out | ios :: binary);
```

`// Читання даних з файлу в буфер buf довжини n, кратній довжині блоку шифру`

```
I
stream & read (char * buf, int n);
```

`/* Отримання кількості байт, фактично прочитаних під час останньої операції читання з файлу */`

```
int gcount ();
```

`// Запис в файл даних з буфера buf довжини n, кратній довжині блоку шифру`

```
ostream& write(const char *buf, int n);
```

```
// Закриття файлу
```

```
void close ();
```

```
// Перевірка досягнення кінця файлу
```

```
bool eof ();
```

```
// Видалення файлу з ім'ям filename
```

```
int remove (const char * filename);
```

3. шифрування (розшифрування) файлу облікових записів (константа, типі даних і прототипів функцій визначені в заголовному файлі **wincrypt.h**):

HCRYPTPROV, HCRYPTKEY, HCRYPTHASH - типі даних для дескрипторів криптопровайдера (CSP), криптографічного ключа, хеш-об'єкта

ALG_ID - тип даних для кодів криптографічних алгоритмів

```
/ * ініціалізація криптопровайдера:
```

```
в * phProv записується його дескриптор,
```

```
pszContainer = NULL,
```

```
pszProvider = NULL,
```

```
dwProvType = PROV_RSA_FULL,
```

```
dwFlags = 0 або (якщо при першому запуску програми CryptAcquireContext повертає FALSE) реєстрація нового користувача в криптопровайдерів
```

```
= CRYPT_NEW_KEYSET * /
```

```
BOOL CryptAcquireContext (HCRYPTPROV * phProv, LPCSTR pszContainer,  
LPCSTR pszProvider, DWORD dwProvType, DWORD dwFlags);
```

```
/ * Створення пустого хеш-об'єкта (hProv - дескриптор ініціалізованих  
криптопровайдера, AlgId - код алгоритму хешування, hKey = 0, dwFlags = 0, в *  
phHash записується дескриптор хеш-об'єкта) * /
```

```
BOOL CryptCreateHash (HCRYPTPROV hProv, ALG_ID AlgId, HCRYPTKEY hKey,  
DWORD dwFlags, HCRYPTHASH * phHash);
```

```
/ * Хешування паролльної фрази pbData довжини dwDataLen (hHash -  
дескриптор хеш-об'єкта, dwFlags = 0) * /
```

```
BOOL CryptHashData (HCRYPTHASH hHash, CONST BYTE * pbData,  
DWORD dwDataLen, DWORD dwFlags);
```

```

// Руйнування хеш-об'єкта з дескриптором hHash

BOOL CryptDestroyHash (HCRYPTHASH hHash);

/ * Створення ключа шифрування з хеш-об'єкта з пароллю фразою
hBaseData (hProv - дескриптор криптопровайдера, Algid - код алгоритму
шифрування, dwFlags = CRYPT_EXPORTABLE з можливим об'єднанням через |
з ознакою додавання до ключа випадкового значення CRYPT_CREATE_SALT,
в * phKey записується дескриптор ключа) * /

BOOL CryptDeriveKey (HCRYPTPROV hProv, ALG_ID Algid,
HCRYPTHASH hBaseData, DWORD dwFlags, HCRYPTKEY * phKey);

// Руйнування ключа шифрування з дескриптором hKey

BOOL CryptDestroyKey (HCRYPTKEY hKey);

// Звільнення криптопровайдера з дескриптором hProv (dwFlags = 0)

BOOL CryptReleaseContext (HCRYPTPROV hProv, DWORD dwFlags);

/ * Шифрування на ключі з дескриптором hKey порції даних з буфера pbData
довжини dwBufLen (dwDataLen - довжина порції даних, після виконання
функції в цю змінну записується фактична довжина зашифрованих даних;
hHash = 0, dwFlags = 0, Final - ознака останньої порції даних) * /

BOOL CryptEncrypt (HCRYPTKEY hKey, HCRYPTHASH hHash, BOOL Final,
DWORD dwFlags, BYTE * pbData, DWORD * pdwDataLen,
DWORD dwBufLen);

/ * Розшифровка на ключі з дескриптором hKey порції даних з буфера pbData
(dwDataLen - довжина порції даних, після виконання функції в цю змінну
записується фактична довжина розшифрованих даних; hHash = 0, dwFlags = 0,
Final - ознака останньої порції даних) * /

BOOL CryptDecrypt (HCRYPTKEY hKey, HCRYPTHASH hHash, BOOL Final,
DWORD dwFlags, BYTE * pbData, DWORD * pdwDataLen);

/ * Установка режиму шифрування для ключа hKey (dwParam = KP_MODE,
pbData вказує на змінну типу unsigned long, в якій записаний код
встановлюваного режиму, dwFlags = 0) , KP_MODE - режим шифрування * /

BOOL CryptSetKeyParam (HCRYPTKEY hKey, DWORD dwParam, BYTE * pbData,

```



```
DWORD dwFlags);
```

```
.
```

Лабораторна робота 4 Аналіз механізмів захисту додатку та їх блокування

Мета роботи

Навчитися використовувати засоби статичного і динамічного аналізу програм. Отримати навички модифікації бінарного коду додатка.

Вміст завдання

Досліджувати надану програму, що володіє захистом від свого несанкціонованого використання і зламати захист різними способами.

У процесі виконання лабораторної роботи необхідно:

- Виділити в досліджуваній програмі ділянку коду, що виконує функцію прийняття рішення про коректність введенного пароля. Визначити файл або файли, в яких зберігається зашифрований пароль.
- Здійснити блокування встановленої захисту, реалізувавши відключення захисного механізму, шляхом модифікації функції прийняття рішення про коректність введенного пароля.
- Виділити в програмі ділянку коду, відповідальний за формування коректного пароля, відповідного введеному імені користувача. Досліджувати даний код і формально записати алгоритм формування коректного пароля. Використовуючи код програми, відповідальний за формування правильного пароля, створити генератор паролів.
- Здійснити злом встановленої захисту, використовуючи деякий користувальницький ідентифікатор (ім'я користувача) і відповідний йому коректний пароль, сформований по знайденому в п.3 алгоритму.
- Виділити в досліджуваній програмі ділянку коду, що виконує функцію прийняття рішення про перевищення встановленої межі запусків. Визначити ключі реєстру, в яких зберігається лічильник запусків.
- Здійснити блокування встановленої захисту - або підміною функції прийняття рішення про перевищення встановленої межі, або шляхом зміни ключів реєстру.

Для вирішення даних задач рекомендується використовувати наступні засоби динамічного і статичного дослідження ПО:

1. Interactive DisAssembler Pro v 4.51 і вище.
2. Hiew 6.81 і вище.
3. Вбудований відладчик IDE MS Visual Studio 2003/2005, OllyDbg або ж SoftIce

Теоретичні відомості

У процесі виконання лабораторної роботи необхідно використовувати програми, опис яких наведено нижче.

НІЕВ 6.81

Для нормальної роботи з даною програмою необхідно знати деякі керуючі комбінації і режими роботи програми.

При старті програми без параметрів перед користувачем відкривається вікно вибору потрібного файлу (рис. 1) Для визначеності виберемо файл Niew.hlp з рідного каталогу програми (рис. 2). Як можна бачити, за замовчуванням даний редактор працює в режимі звичайного перегляду тексту. Тепер натиснемо клавішу ENTER (аналогічно можна натиснути клавішу F4 і вибрати необхідний режим перегляду інформації), таким чином здійснивши перехід від звичайного перегляду в режимі тексту в шістнадцяткове відображення інформації (рис. 3). Тепер вікно редактора розбито на 3 колонки. Крайня ліва являє собою зсув байтів відносно початку файлу, центральна область - шістнадцяткове представлення інформації, крайня права - текстове відображення інформації.

Для того щоб перейти в режим редагування тексту, необхідно натиснути клавішу F3. Після цього можна міняти всі байти, що знаходяться в центральній області вікна редактора, на необхідні користувачу. Щоб зберегти зміни, потрібно натиснути клавішу F9.

При роботі з редактором потрібно пам'ятати, що він відкриває файли з можливістю читання іншими програмами, але не на запис. Це означає, що якщо ви в даний момент редагуєте за допомогою НІЕВ якийсь файл, то інша програма, що намагається внести зміни в цей же файл, не зможе звернутися до цього файлу. До цього відноситься також спроба запуску програми, якщо ви його в даний момент редагуєте.

Тепер розглянемо практичний приклад - відкриємо файл додатка Windows Notepad. Перед цим рекомендується зробити його резервну копію.

Рис. 1. Вікно НІЕВ при запуску без параметрів.

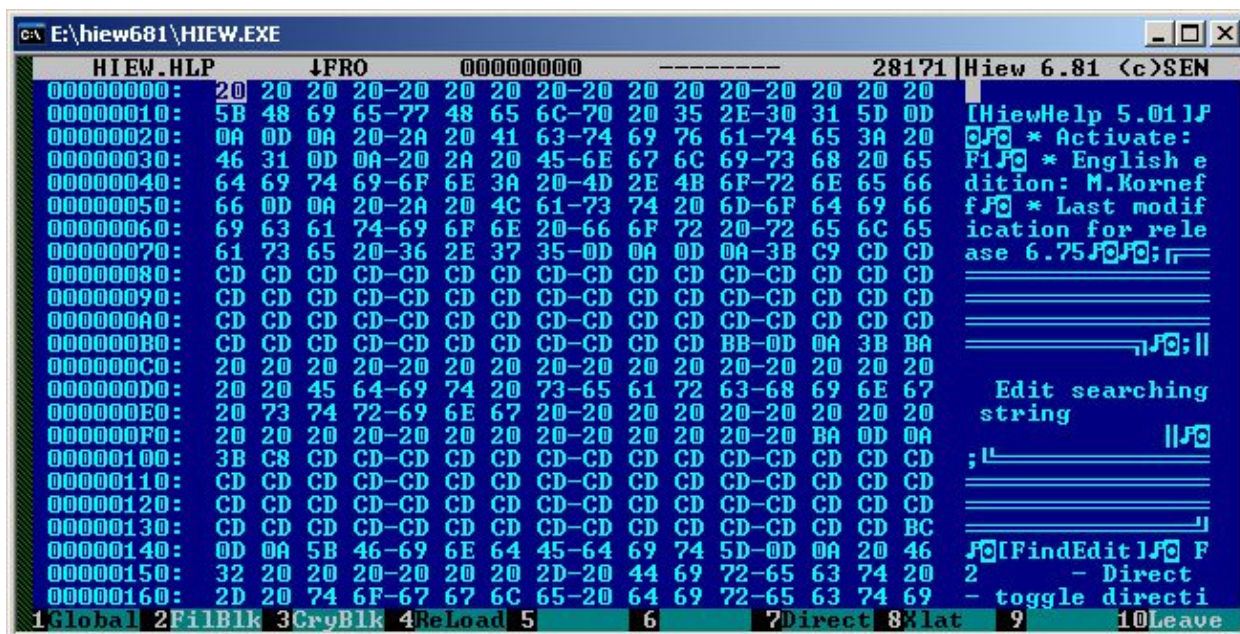
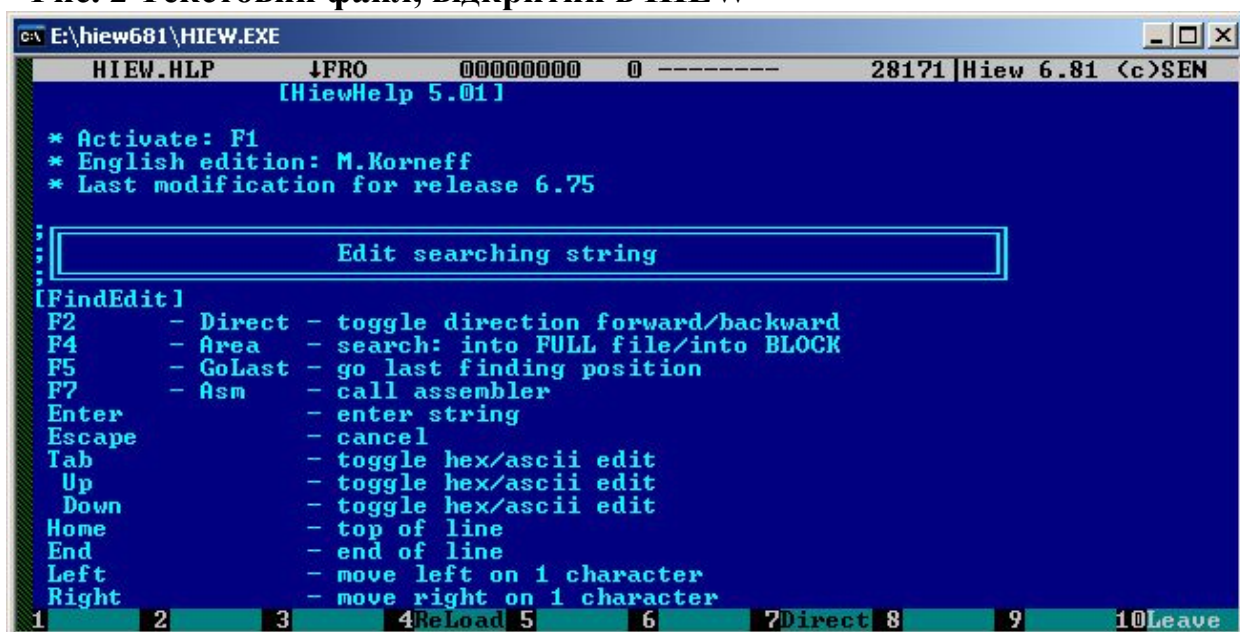


Рис. 2 Текстовий файл, відкритий в HIEW



Натіснемо клавішу F9 и перейдемо в режим Вибори файлів. Після того, Як файл БУВ Вибране, перейдемо в режим дизасемблера (клавіша F4, або два рази ENTER).

Рис. 3 Шістнадцятковий режим відображення інформації


```

C:\ E:\hiew681\HIEW.EXE
notepad.exe  ↓FRO  PE.0100739D a32 ----- 69120 Hiew 6.81 <c>SEN
0100739D: 5A70          push      070
0100739F: 6898180001     push      001001898 -----↑ <1>
010073A4: E8BF010000     call      .001007568 -----↓ <2>
010073A9: 33DB          xor       ebx,ebx
010073AB: 53            push      ebx
010073AC: 8B3DCC100001   mov       edi,GetModuleHandleA ;KERNE
010073B2: FFD7          call      edi
010073B4: 6681384D5A     cmp       w,[eax],05A4D ;"ZM"
010073B9: 751F          jne       .0010073DA -----↓ <3>
010073BB: 8B483C         mov       ecx,[eax][3C]
010073BE: 03C8          add       ecx,eax
010073C0: 813950450000   cmp       d,[ecx],000004550 ;" EP"
010073C6: 7512          jne       .0010073DA -----↓ <4>
010073C8: 0FB74118     movzx     eax,w,[ecx][18]
010073CC: 3D0B010000     cmp       eax,00000010B ;" 08"
010073D1: 741F          je        .0010073F2 -----↓ <5>
010073D3: 3D0B020000     cmp       eax,00000020B ;" 08"
010073D8: 7405          je        .0010073DF -----↓ <6>
010073DA: 895DE4         mov       [ebp][1C],ebx
010073DD: EB27          jmps      .001007406 -----↓ <7>
010073DF: 83B9840000000E cmp       d,[ecx][00000084],00E ;"П"
010073E6: 76F2          jbe       .0010073DA -----↑ <8>
010073E8: 33C0          xor       eax,eax
1Global 2FillBlk 3 4ReLoad 5OrdOff 6Ibyte 7Direct 8Xlat 9Auto 10Leave

```

Рис. 5. Точка входу додатку Блокнот

Тепер «познущатися» над Блокнотом, замінивши виклик однієї процедури на порожні операції. Для цього натиснемо F3, і потім послідовно замінимо байти зі значенням 6A70 на 9090 (0x90 = *por* - відсутність будь-якої операції). Після цього натискаємо F9 для того, щоб зберегти зміни і натискаючи ESC для виходу з HIEW.

Відзначимо, що найчастіше пор'амі доводиться «забивати» не один байт, а цілий шматок програми; для цього зручніше відзначити блок байт (клавіша «*»), а потім виконати команду FillBlk (ALT-F2).

```

C:\ E:\hiew681\hiew.exe
notepad.exe  ↓FRO  PE.0100739D a32 ----- 69120 Hiew 6.81 <c>SEN
0100739D: 90            nop
0100739E: 90            nop
0100739F: 6898180001     push      001001898 -----↑ <1>
010073A4: E8BF010000     call      .001007568 -----↓ <2>
010073A9: 33DB          xor       ebx,ebx
010073AB: 53            push      ebx
010073AC: 8B3DCC100001   mov       edi,GetModuleHandleA ;KERNE
010073B2: FFD7          call      edi
010073B4: 6681384D5A     cmp       w,[eax],05A4D ;"ZM"
010073B9: 751F          jne       .0010073DA -----↓ <3>
010073BB: 8B483C         mov       ecx,[eax][3C]
010073BE: 03C8          add       ecx,eax
010073C0: 813950450000   cmp       d,[ecx],000004550 ;" EP"
010073C6: 7512          jne       .0010073DA -----↓ <4>
010073C8: 0FB74118     movzx     eax,w,[ecx][18]
010073CC: 3D0B010000     cmp       eax,00000010B ;" 08"
010073D1: 741F          je        .0010073F2 -----↓ <5>
010073D3: 3D0B020000     cmp       eax,00000020B ;" 08"
010073D8: 7405          je        .0010073DF -----↓ <6>
010073DA: 895DE4         mov       [ebp][1C],ebx
010073DD: EB27          jmps      .001007406 -----↓ <7>
010073DF: 83B9840000000E cmp       d,[ecx][00000084],00E ;"П"
010073E6: 76F2          jbe       .0010073DA -----↑ <8>
010073E8: 33C0          xor       eax,eax
1Global 2FillBlk 3 4ReLoad 5OrdOff 6Ibyte 7Direct 8Xlat 9Auto 10Leave

```

Рис. 6 Змінена точки входу Блокнота

Запустимо отриману версію Блокнота. Як бачимо, вона не запускається. Таким чином, ми навчилися знищувати корисний код програми шляхом заміни потрібних байтів на пор. Пам'ятайте, що в даній лабораторній роботі

левову частку захисту можна зняти таким методом, головне знати, що і де замінювати. Звідси випливає висновок: ПРИ НЕОБХІДНОСТІ ЗНЕШКОДИТИ ДІЛЯНКУ КОДА ШЛЯХОМ ВИДАЛЕННЯ його функціональних ІЗ ПРОГРАМИ, НЕОБХІДНО ЗАМІНИТИ потрібну операцію еквівалентною кількістю БАЙТ 0x90.

Для пошуку в редакторі потрібного рядка досить натиснути клавішу F7 і ввести те, що вам потрібно знайти або в ASCII, або в шістнадцятиричних кодах. Якщо входжень потрібного рядка більш ніж одне, то для пошуку входжень, що залишилися, необхідно використовувати комбінацію клавіш CTRL + F7.

За більш детальним описом команд звертатися до функції допомоги даної програми F1. Інформація файлів допомоги контекстно-залежна.

Примітка. Можна також використовувати редактор XVI32.

IDA Pro v4.51

Для виконання даної лабораторної роботи нам знадобиться можливість IDA показувати нам таблицю експорту та імпорту функцій програми. Власне, будь-яка програма під Windows не є річчю в собі, а хоч якось повинна взаємодіяти з навколишнім середовищем, в нашому випадку - операційною системою. Тому як мінімум одну функцію будь-яка програма повинна експортувати: точку входу. Можна сказати, що в нашому випадку вона може послужити відправною точкою в дослідженні програми. Крім цього, більшість програм імпортують функції WinAPI, і їхні імена подані в таблиці імпорту. У нашому випадку нам знадобляться функції, відповідальні за виведення інформації на екран і взяття інформації з елементів вікна. Щоб полегшити пошук студента, нижче наводяться функції, задіяні в даній лабораторній роботі:

```
MessageBoxA ();  
GetDlgItemTextA ();
```

З цього випливає, що в роботі необхідно буде шукати місця виклику даних функцій. На цьому теоретичному відступ закінчується.

Практичне використання IDA.

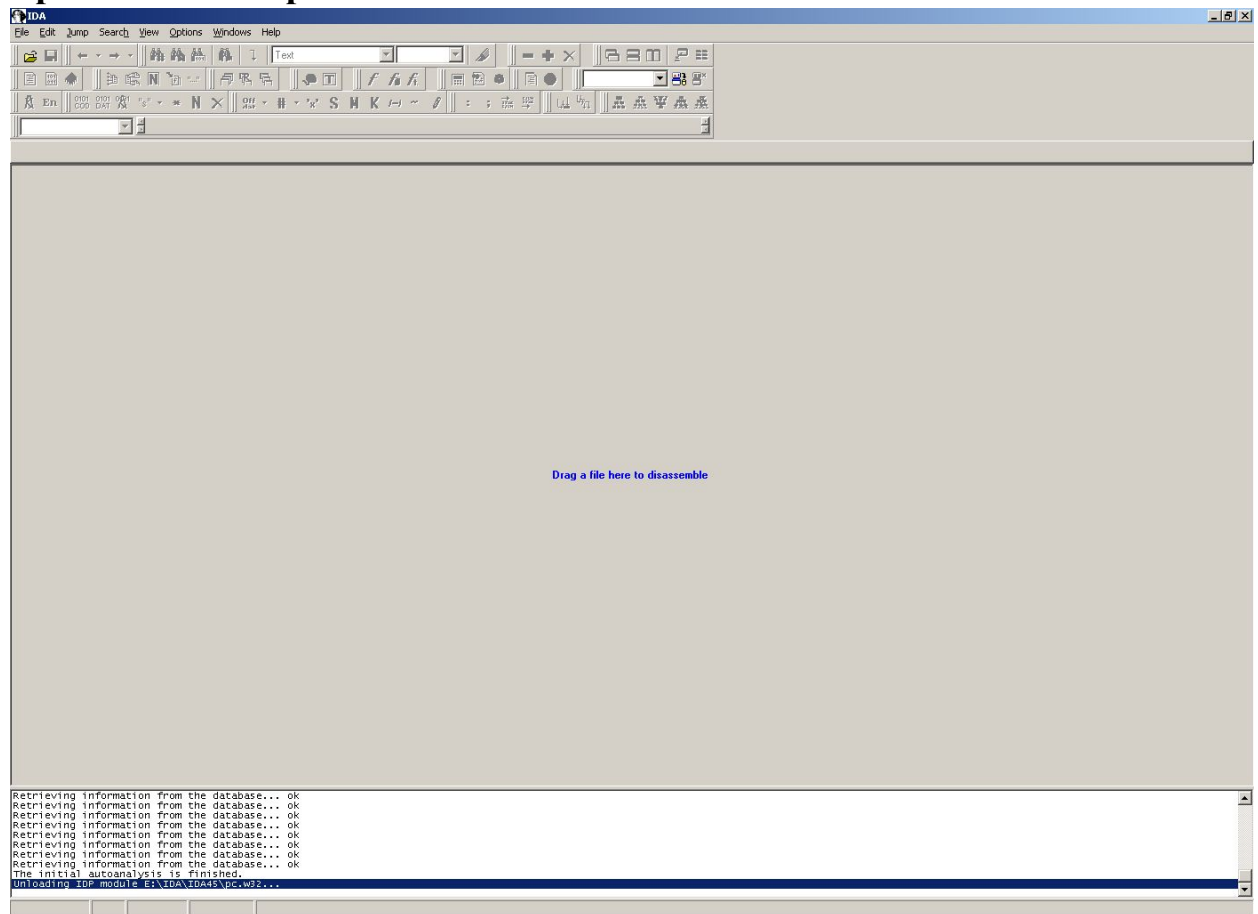


Рис. 7. Основне вікно IDA

При запуску IDA можна спостерігати вікно, представлене на рис. 7. Щоб відчувати, як працює система, виберемо нашу програму для аналізу, і отримаємо результат, зображений на рис. 8.

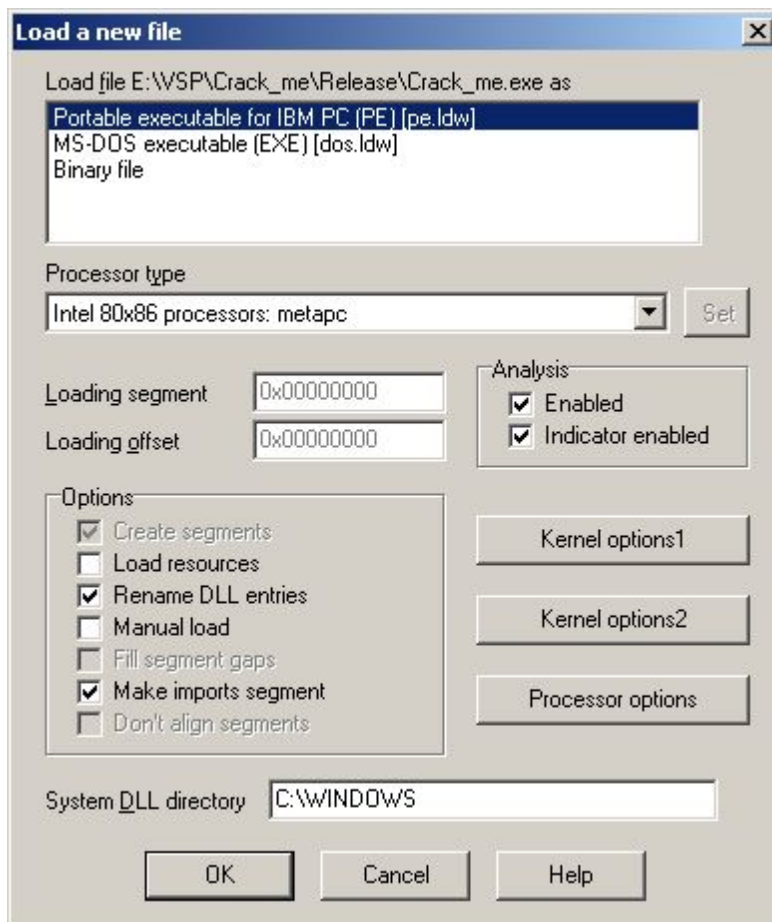


Рис. 8. Вікно запиту про метод аналізу файлу.
Натискаємо на ОК і отримуємо наступне (рис. 9).

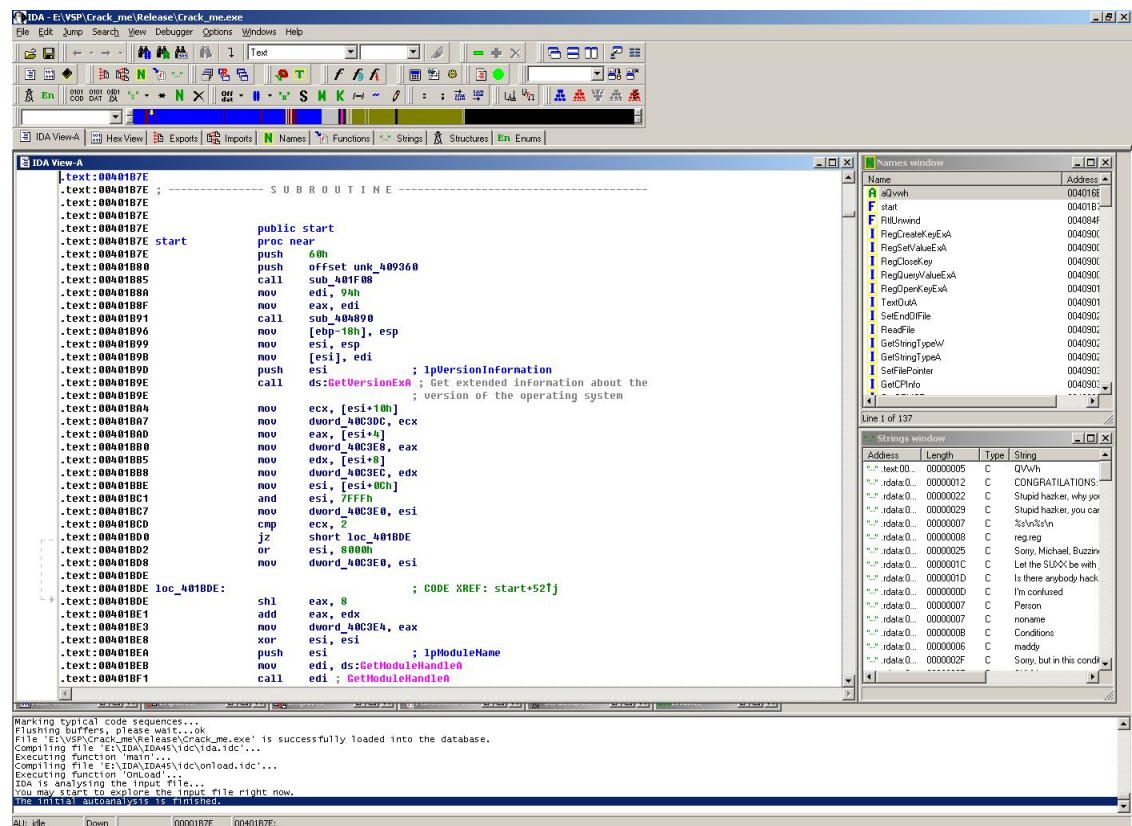


Рис. 9. Завершений аналіз виконуваного файлу

Дане вікно з'являється після остаточного аналізу програми. Після цього вибираємо вкладку Imports, щоб подивитися список функцій, що використовуються програмою (рис. 10).

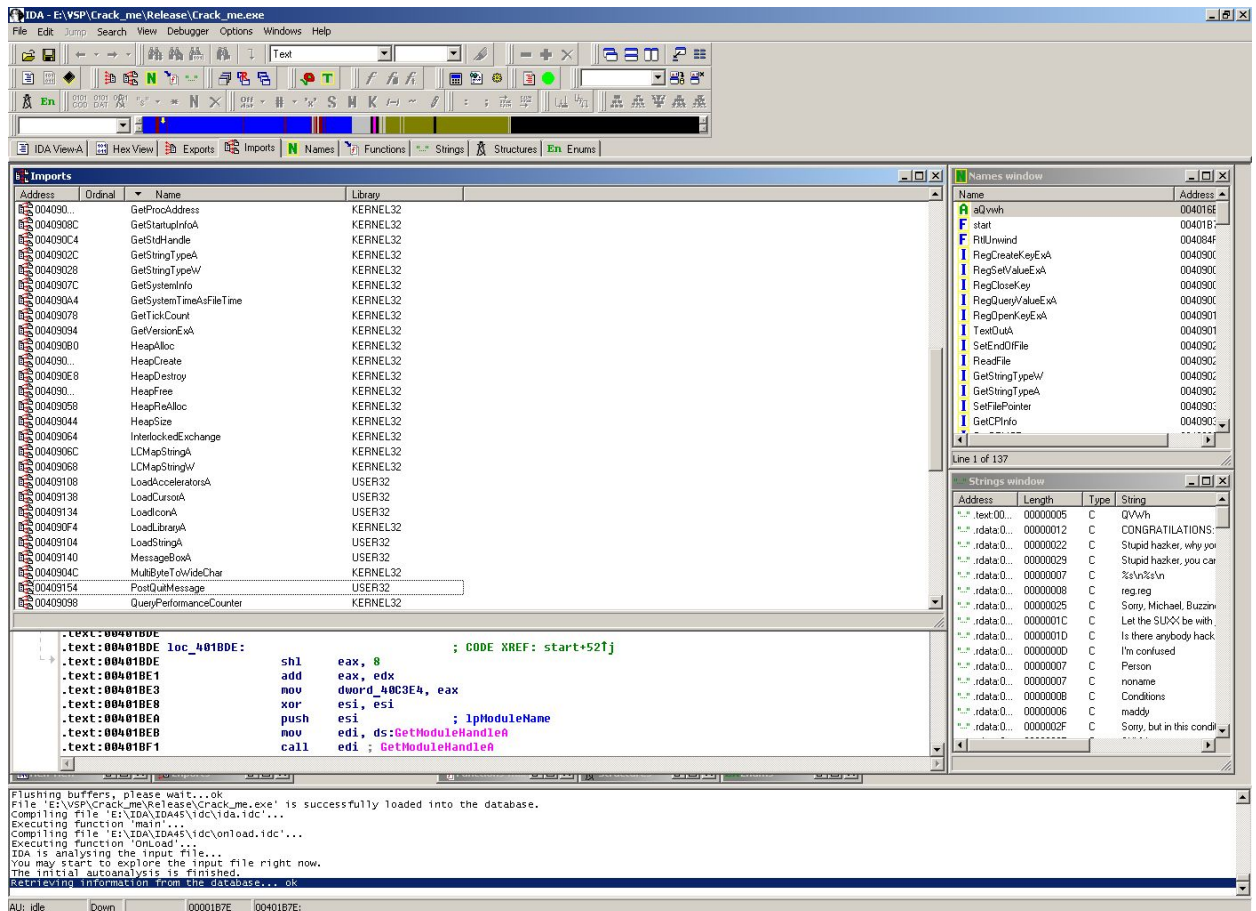


Рис. 10. Відкрите вікно таблиці імпорту

Клацнувши подвійним клацанням на функції GetSystemTimeAsFileTime, ми побачимо картину, зображену на рис. 11.

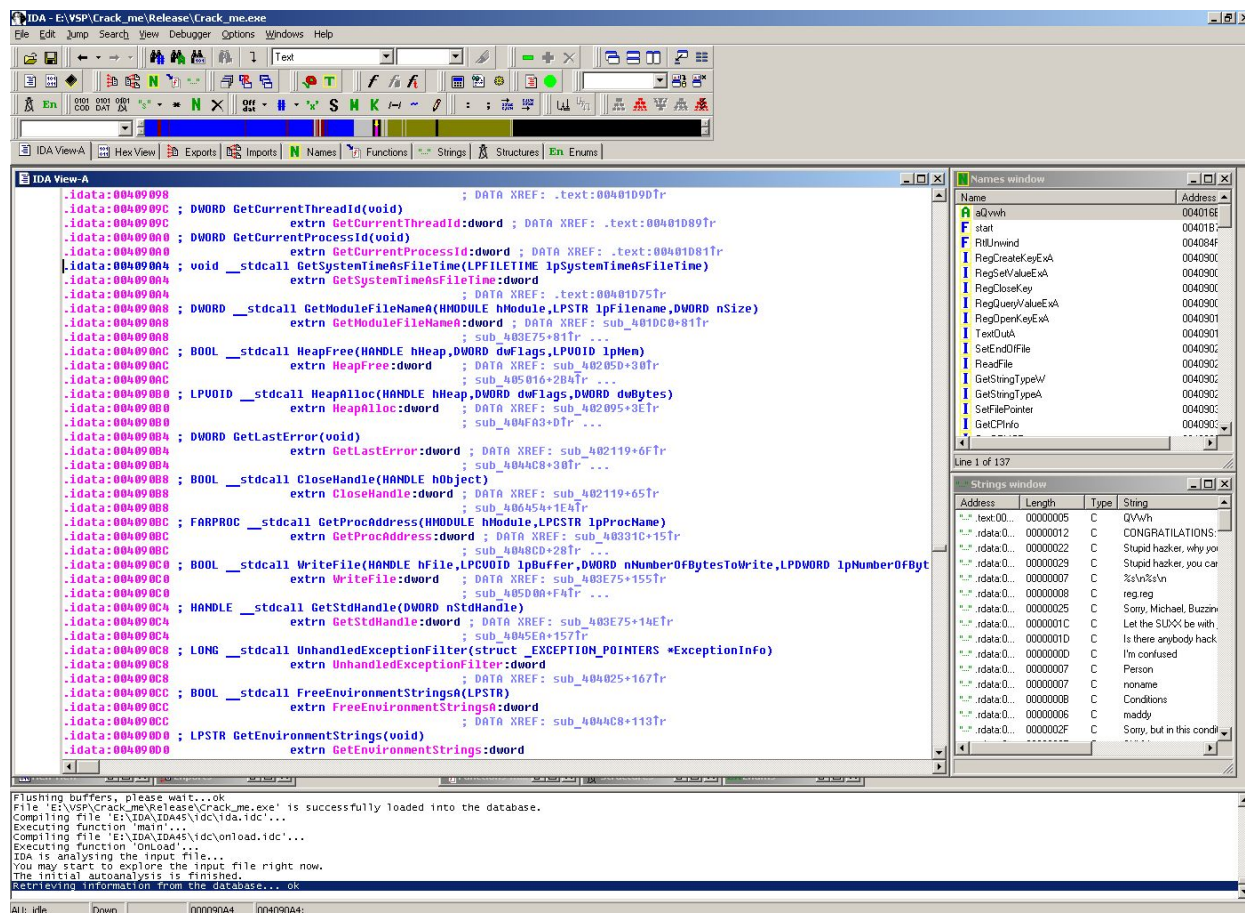


Рис. 11. Інформація про імпортовані функції GetSystemTimeAsFileTime

Як можна бачити з рис. 11, після здійсненої операції перед нами постає опис імпортованої функції з її параметрами і місцем виклику в програмі (; DATA XREF: .Text: 00401D75). Тепер, якщо нам необхідно дізнатися, де в програмі викликається дана функція, то досить зробити подвійне клацання по напису. Text: 00401D75 і IDA автоматично переведе Вас в потрібне місце виклику (рис. 12). Таким чином, можна переглядати будь-яку послідовність викликів функцій.

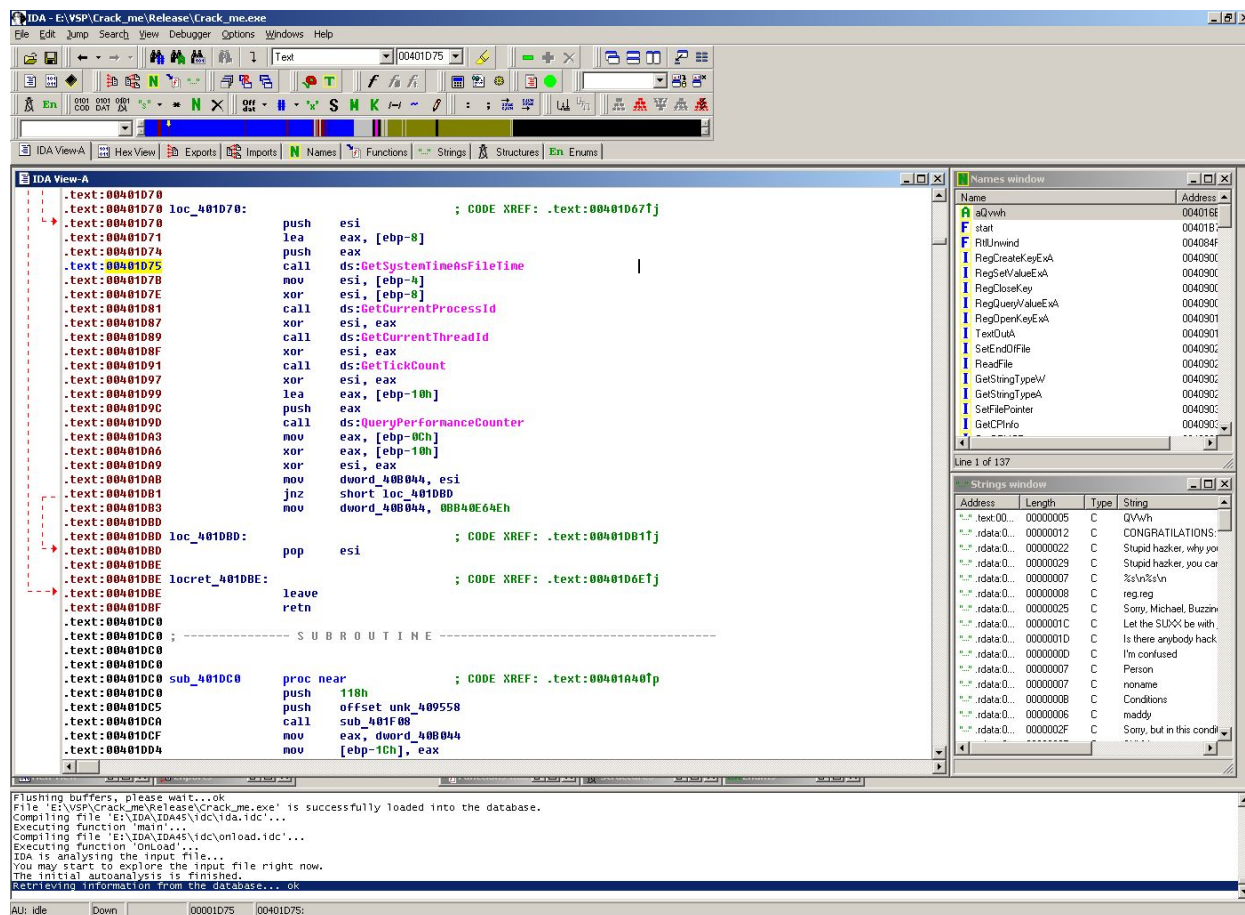


Рис. 12. Місце виклику функції GetSystemTimeAsFileTime

Варто відзначити дуже потужну функціональність IDA в плані навігації по тексті. Щоб повернутися в колишнє місце (тобто момент до переходу по посиланню), достатньо всього лише натиснути клавішу ESC. Крім цього, дуже зручним є графічне відображення переходів в програмі стрілками в правій стороні вікна дизасемблера.

Власне, цієї інформації цілком достатньо для виконання лабораторної роботи.

Робота з відладчиком Visual Studio і OllyDbg

Навіть для вирішення самих тривіальних задач в області реверсінга необхідний відладчик: він дозволяє динамічно відстежувати хід роботи програми, вносити зміни «на льоту» в її образ в пам'яті, стежити за станом регістрів і оперативної пам'яті та багато іншого.

Серед сучасних відладчиків найбільш популярними є SoftIce з комплексу Compuware Driver Studio і OllyDbg. SoftIce працює в режимі ядра і тому дозволяє досліджувати драйвери та компоненти Windows, що працюють в ring0; проте в більшості випадків його потужність є надлишковою і доцільніше використовувати більш "дружній" OllyDbg. До того ж, OllyDbg є freeware і доступний для безкоштовного скачування в Інтерніі.

Крім того, ми також розглянемо вбудований відладчик Microsoft Visual Studio.

Відладчик Visual Studio

На основі даного відладчика дається уявлення про код програми, який необхідно буде шукати. Отже, приступимо.

У що перетворюється код програми, написаної на Сі, після компіляції?

Візьмемо простий приклад:

Створимо простий додаток засоби MS Visual Studio типу Win32 з графічним інтерфейсом.

Переведемо режим проекту в стадію Release (для тих, хто до 4 курсу ще не навчився цього, необхідно вибрати властивості проекту Properties-> Configuration Manager-> Configuration і перевести з режиму Debug в режим Release).

Тепер все має бути готове до застосування.

Для початку подивимося, у що перетворюється звичайний виклик точки входу програми. Для цього поставимо крапку останова на функцію `_tWinMain` (натиснемо F9) в рядку з назвою функції.

Запустимо додаток.

Виконання програми повинно припинитися в поставленої нами точці. Тепер перейдемо від відображення у вигляді Сі до асемблеру шляхом переходу до вкладки Disassembly (рис. 13 і 14).

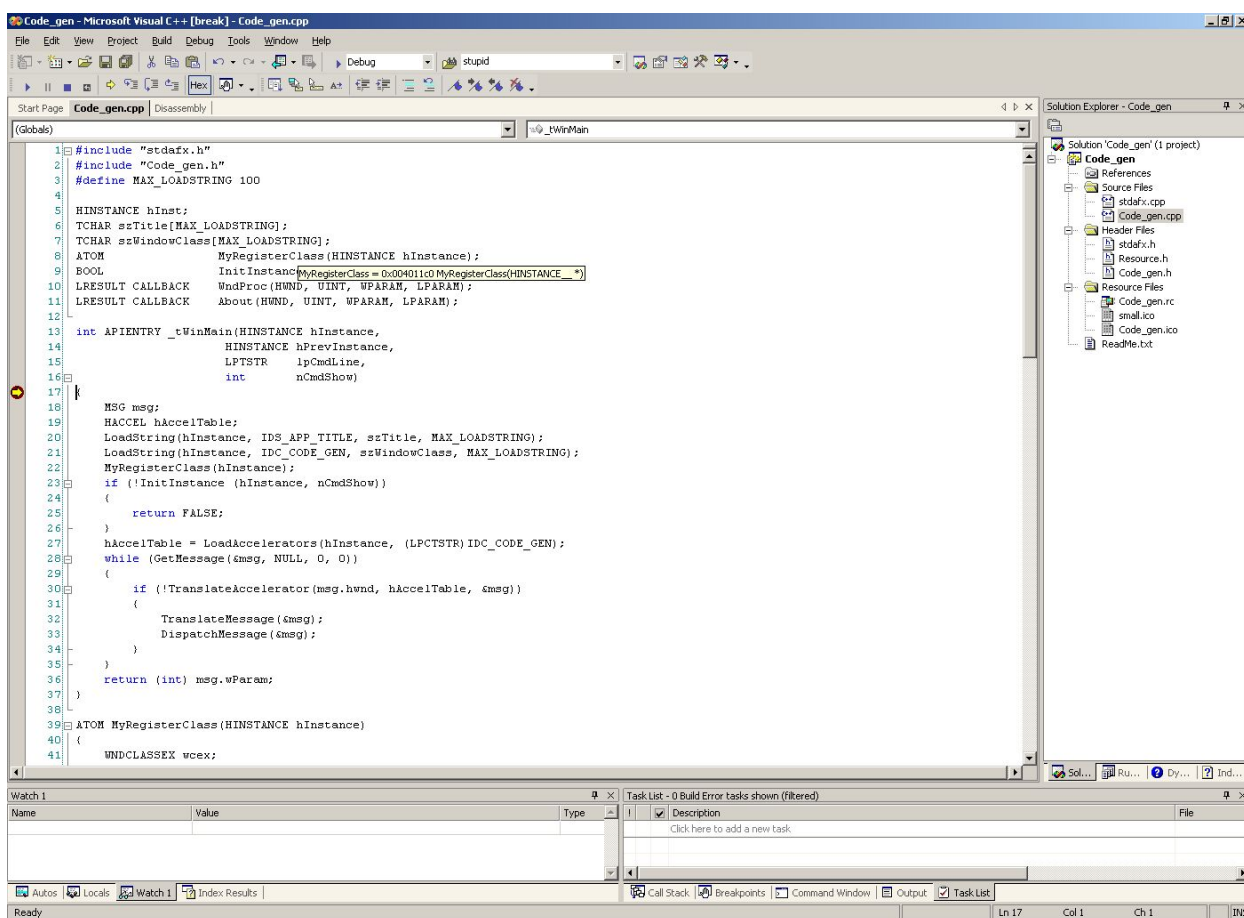


Рис. 13. Лістинг програми.

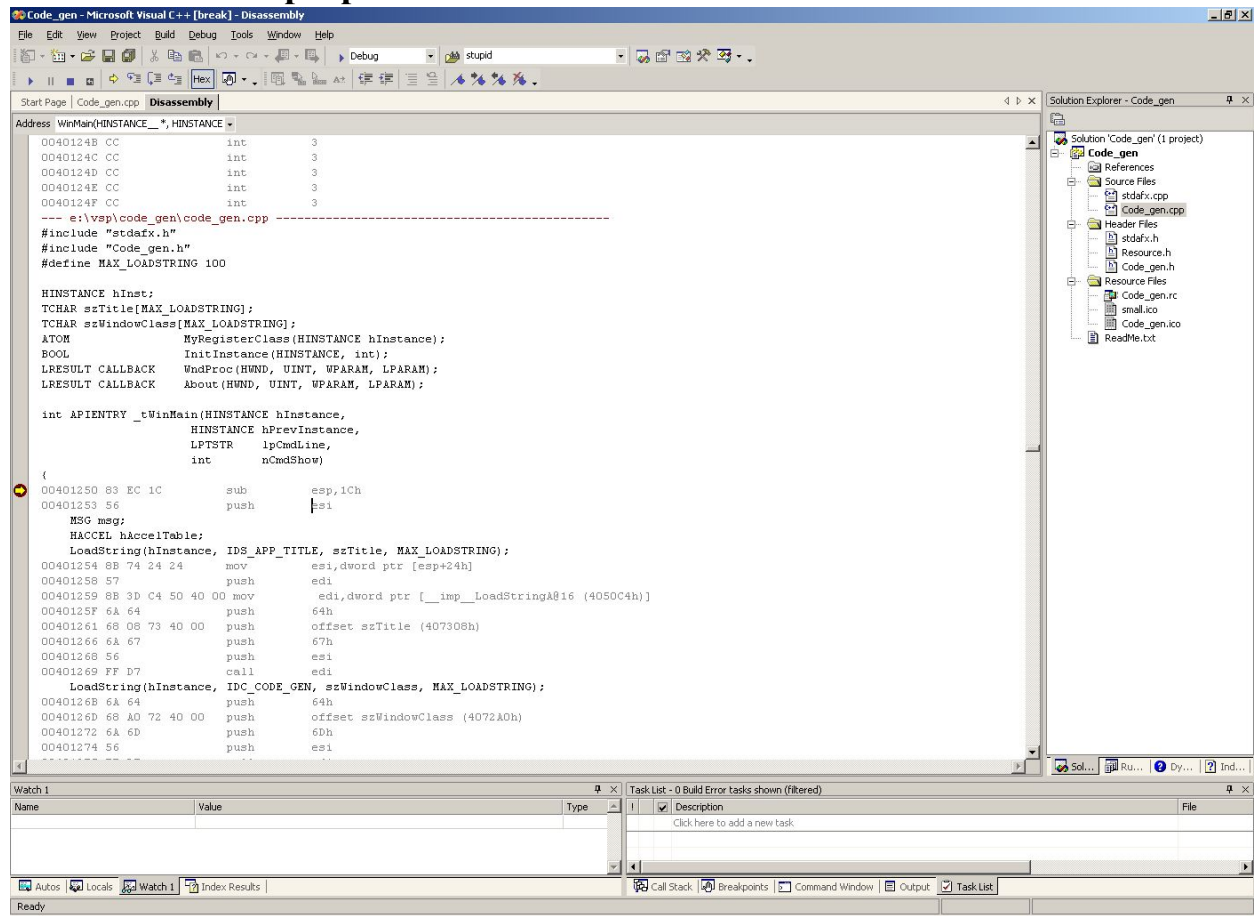


Рис. 14. Вид програми після дизасемблювання

Спробуємо підійти з практичної точки зору і подивимося, на що перетворюються деякі стандартні конструкції мови Сі. Розглянемо комплекс інструкцій:

```

ULONG A, B;
A = GetTickCount();
B = 10;
if (A > B)
{
    A--;
}
for (A = 0; A < 100; A++)
{
    B++;
}
switch(A)
{
case 10:
    B+=100;

```

```

        break;
case 20:
    A+=200;
    break;
}

```

Примітка: якщо Ви захочете переглянути це на реальному прикладі, то або перейдіть в режим DEBUG, або відключіть оптимізацію компілятора, так як в отриманому вами виконуваному файлі дана конструкція, швидше за все, не зустрінеться, тому що компілятор вважатиме її не використовуваною і такою, що не варта включення у виконуваний файл.

Отже:

```

        ULONG A, B;
        A = GetTickCount();
00411B9E 8B F4      mov     esi,esp
00411BA0 FF 15 1C C2 42 00 call    dword ptr [__imp__GetTickCount@0
(42C21Ch)]
00411BA6 3B F4      cmp     esi,esp
00411BA8 E8 72 F8 FF FF call    @ILT+1050(__RTC_CheckEsp)
(41141Fh)
00411BAD 89 45 F8      mov     dword ptr [A],eax
        B = 10;
00411BB0 C7 45 EC 0A 00 00 00 mov     dword ptr [B],0Ah
        if (A > B)
00411BB7 8B 45 F8      mov     eax,dword ptr [A]
00411BBA 3B 45 EC      cmp     eax,dword ptr [B]
00411BBD 76 09      jbe     WinMain+48h (411BC8h)
        {
            A--;
00411BBF 8B 45 F8      mov     eax,dword ptr [A]
00411BC2 83 E8 01      sub     eax,1
00411BC5 89 45 F8      mov     dword ptr [A],eax
        }
        for (A = 0; A < 100; A++)
00411BC8 C7 45 F8 00 00 00 00 mov     dword ptr [A],0
00411BCF EB 09      jmp     WinMain+5Ah (411BDAh)
00411BD1 8B 45 F8      mov     eax,dword ptr [A]
00411BD4 83 C0 01      add     eax,1
00411BD7 89 45 F8      mov     dword ptr [A],eax
00411BDA 83 7D F8 64    cmp     dword ptr [A],64h
00411BDE 73 0B      jae     WinMain+6Bh (411BEBh)
        {
            B++;

```



```

00411BE0 8B 45 EC      mov     eax,dword ptr [B]
00411BE3 83 C0 01      add     eax,1
00411BE6 89 45 EC      mov     dword ptr [B],eax
    }
00411BE9 EB E6      jmp     WinMain+51h (411BD1h)
    switch(A)
00411BEB 8B 45 F8      mov     eax,dword ptr [A]
00411BEE 89 85 F4 FE FF FF mov     dword ptr [ebp-10Ch],eax
00411BF4 83 BD F4 FE FF FF 0A cmp     dword ptr [ebp-10Ch],0Ah
00411BFB 74 0B          je      WinMain+88h (411C08h)
00411BFD 83 BD F4 FE FF FF 14 cmp     dword ptr [ebp-10Ch],14h
00411C04 74 0D          je      WinMain+93h (411C13h)
00411C06 EB 16      jmp     WinMain+9Eh (411C1Eh)
    {
    case 10:
        B+=100;
00411C08 8B 45 EC      mov     eax,dword ptr [B]
00411C0B 83 C0 64      add     eax,64h
00411C0E 89 45 EC      mov     dword ptr [B],eax
        break;
00411C11 EB 0B      jmp     WinMain+9Eh (411C1Eh)
    case 20:
        A+=200;
00411C13 8B 45 F8      mov     eax,dword ptr [A]
00411C16 05 C8 00 00 00 add     eax,0C8h
00411C1B 89 45 F8      mov     dword ptr [A],eax
        break;
    }

```

Отже, ми бачимо наступне: під конструкціями мови високого рівня Сі знаходиться те, у що перетворює компілятор в вихідний код програми.

Настійно рекомендується провести ознайомлення з даним прикладом і усвідомити конструкції асемблера, еквівалентні команді мов високого рівня. отримані знання будуть необхідні для відновлення алгоритму створення ключа користувача.

Робота с OllyDbg

Розглянемо роботу з цим відладчиком на прикладі версії 1.1 без додаткових плагінів.

Для цього запустимо програму і відкриємо в ній notepad.exe:

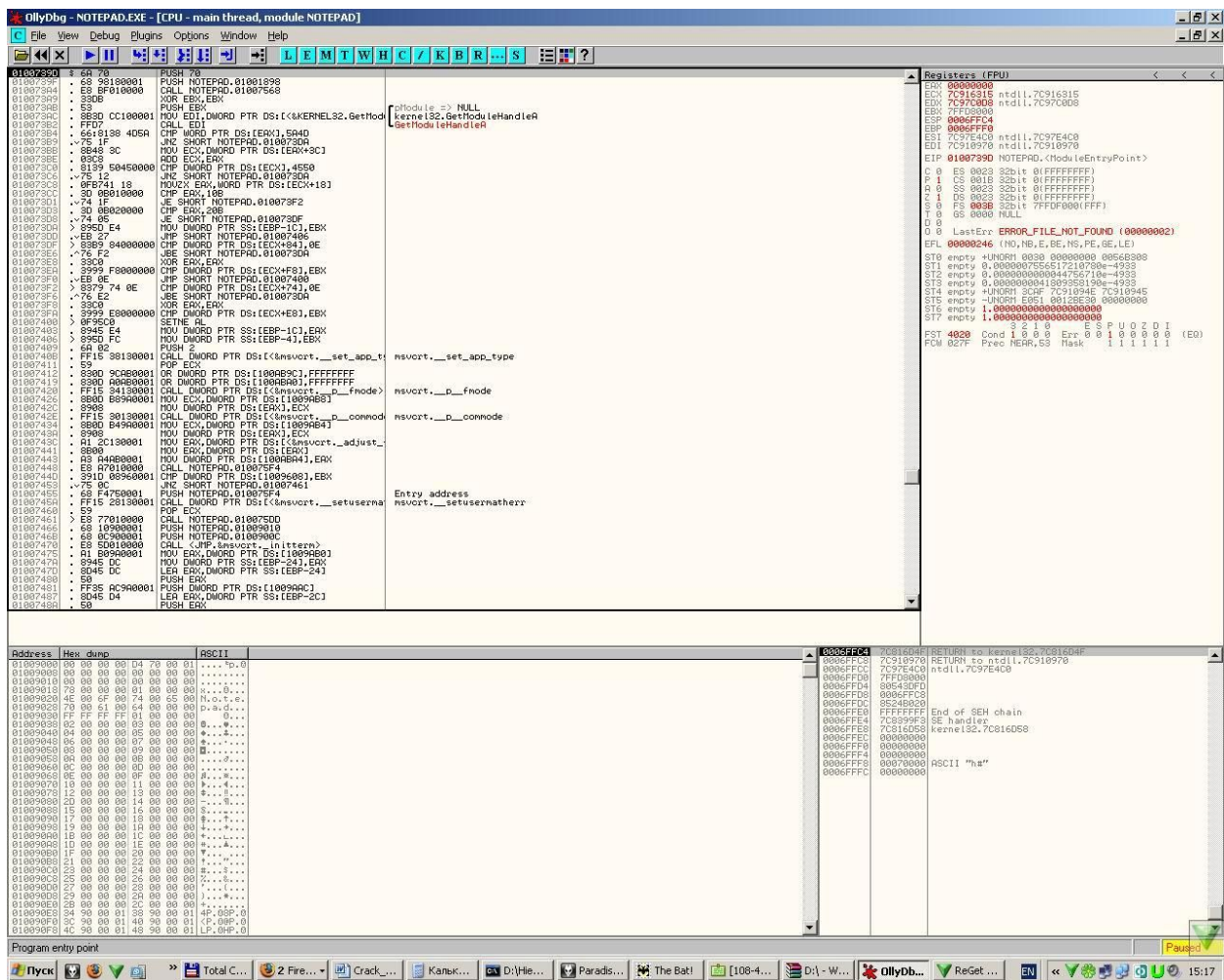


Рис. 14. Загальний вигляд OllyDbg

Зліва перед нами міститься вікно, в яке виводиться асемблерний код програми, праворуч - панель з станом регістрів (по правому клацанню можна вибирати, які додаткові регістри показувати - MMX, 3dNow! І так далі), внизу - дамп стека і оперативної пам'яті.

У вікні з асемблерних лістингом зазначено зсув, hex-код, безпосередньо асемблерні команди, йому відповідна, і розширений коментар. В якості коментаря часто виводиться така корисна інформація, як наприклад, ім'я викликуваної API-функції.

Двічі кликнувши по команді, можна редагувати її та замінювати своєю.

Зверніть увагу на меню, яке з'являється при правому кліку мишею.

Меню Breakpoint дозволяє управляти точками останова: Toggle (F2) ставить брейкпойнтів на вибраній команді, Conditional Breakpoint - останов при виконанні деякого умови, Run To Selection - виконати програму і зупинитися на обраній команді.

Меню Search for містить такі незамінні опції, як All Referenced Text Strings і All Intermodular Calls. Відкривши пункт Search for All Referenced

Text Strings, можна побачити всі дані-рядки, використовувані в програмі, і швидко перейти в те місце, де вони згадуються. Наведемо практичний приклад: програма перевіряє серійний номер і в разі невдачі повідомляє «Your serial number is incorrect». Робимо пошук цього рядка і здійснюємо перехід до того місця, де викликається відповідний MessageBox; напевнодесь недалеко йде перевірка валідності серійного номера, яку можна або пропатчити (наприклад, забити тими ж пор'ами) або спробувати розібратися в алгоритмі і написати кейген. У меню пункт Search for All Intemodular Calls можна знайти список всіх API-функцій, використовуваних у програмі, і поставити на потрібних з них точку зупину. У попередньому прикладі можна було б поставити breakpoint на MessageBoxA і ввести неправильний серійний номер - програма б зупинилася якраз на тому місці, де нам потрібно.

Крім цих пунктів, є ще безліч інших не менш корисних. Пропонуємо студенту ознайомитися з ними самостійно.

Ну і, звичайно, важливим є меню Debug: сподіваємося, що стандартні команди на кшталт Trace Into або Step Over ні в кого проблем не викличуть.

Прочитавши до цього місця, студент вважається досить підготовленим для виконання даної лабораторної роботи.

Вміст завдання

1. Необхідно проаналізувати алгоритм роботи програми і створити блок-схему алгоритму з деталізацій до окремих повідомлень в коді програми.
2. Написати програму генерації паролів для програми що аналізується.
3. Навчитися вносити зміни в код програми для відключення окремих функцій захисту.

Рекомендована література

1. Крис Касперски. Техника и философия хакерских атак. 2-е издание, переработанное и дополненное.
2. В.И. Юров. Assembler. Учебник для ВУЗов. (любое из изданий).
3. Крис Касперски. Образ мышления – дизассемблер IDA.
4. Р.А. Айрапетян. Отладчик SoftIce. Подробный справочник.

Лабораторна робота 5 Автоматизований пошук вразливостей у

вихідних текстах програмного забезпечення, що написані на мові високого рівня.

Мета роботи

Пошук потенційних вразливостей і помилок програмування в вихідних текстах на мові програмування C / C++ з використанням автоматизованого засобу виявлення поширених помилок програмування RATS.

Дослідження виявленої проблеми: визначення типу та категорії помилки, локалізація, розроблення пропозицій щодо усунення. Ранжування виявлених проблемних місць за ступенем серйозності.

Вміст завдання

- Самостійно вибрати файл на мові програмування, підтримуваного утилітою.
- Просканувати даний файл.
- Проаналізувати вихідну інформацію утиліти, визначити характер уразливості, пояснити її значення для безпеки додатку
- Знайти позицію уразливості у вихідному коді і запропонувати варіант її усунення.

Теоретичні відомості

Статичний аналіз коду - це аналіз програмного забезпечення , який виконується над вихідним кодом програм і реалізується без реального виконання досліджуваної програми .

Програмне забезпечення часто містить різноманітні уразливості через помилки в коді програм. Помилки , допущені при розробці програм , в деяких ситуаціях призводять до збою програми , а отже , порушується нормальна робота програми : при цьому часто виникає зміна і псування даних , зупинка програми або навіть системи . Більшість вразливостей пов'язано з неправильною обробкою даних , одержуваних ззовні , або недостатньо суворої їх перевіркою.

Для виявлення вразливостей використовують різні інструментальні засоби , наприклад, статичні аналізатори вихідного коду.

В загальному випадку при статичному аналізі вирішуються такі завдання:

1. дослідження структури програми;
2. виділення набору стандартних системних функцій , використовуваних програмою;
3. виявлення способів і цілей взаємодії програми с зовнішніми пристроями і системними ресурсами;
4. пошук типових вразливостей;
5. пошук руйнують програмних засобів в коді програми.

Сучасні засоби статичного аналізу безпеки програмного забезпечення в

основному досліджують уразливості у вихідних текстах програм.

В даний час існують наступні вільно поширювані засоби пошуку вразливостей: ITS4 , RATS , FLA WFINDER , LCLINT .

Всі ці засоби мають схожу функціональність , виявляючи уразливості в програмному забезпеченні , написаному на мовах C , C + + та інших , на основі бази даних відомих помилок програмування , що призводять до вразливостей.

Класифікація вразливостей захисту

Коли вимога коректної роботи програми на всіх можливих вхідних даних порушується, стає можливою поява так званих вразливостей захисту (security vulnerability). Уразливості захисту можуть приводити до того, що одна програма може використовуватися для подолання обмежень захисту всієї системи в цілому.

Класифікація вразливостей захисту в залежності від програмних помилок:

1. Переповнення буфера (buffer overflow). Ця вразливість виникає через відсутність контролю за виходом за межі масиву в пам'яті під час виконання програми. Коли занадто великий пакет даних переповнює буфер обмеженого розміру, вміст сторонніх елементів пам'яті перезаписується, і відбувається збій і аварійний вихід з програми. За місцем розташування буфера в пам'яті процесу розрізняють переповнення буфера в стеку (stack buffer overflow), купи(heap buffer overflow) і області статичних даних (bss buffer overflow) .

2. Уразливості "зіпсованого введення" (tainted input vulnerability) .

Уразливості "зіпсованого введення" можуть виникати у випадках , коли вводяться користувачем дані без достатнього контролю передаються інтерпретатору деякої зовнішньої мови (зазвичай це мова Unix shell або SQL) . У цьому випадку користувач може таким чином задати вхідні дані , що запущений інтерпретатор виконає зовсім не ту команду , яка передбачалася авторами вразливої програми.

3. Помилки форматних рядків (format string vulnerability) . Даний тип вразливостей захисту є підкласом уразливості " зіпсованого введення " . Він виникає через недостатній контроль параметрів при використанні функцій форматного введення-виведення printf , fprintf , scanf , і т. д. стандартної бібліотеки мови Cі . Ці функції приймають в якості одного з параметрів символічний рядок , заданий формат введення або виведення подальших аргументів функції . Якщо користувач сам може задати вид форматування , то вразливість може виникнути в результаті невдалого застосування функцій форматування рядків.

4. Уразливості як наслідок помилок синхронізації (race conditions) .

Проблеми, пов'язані з багатозадачністю , призводять до ситуації, під назвою "стан гонки ": програма, не розрахована на виконання в багатозадачному середовищі , може вважати , що , наприклад , використовувані нею при

роботі файли не може змінити інша програма . Як наслідок , зловмисник , вчасно підмінюючий вміст цих робочих файлів, може нав'язати програмі виконання певних дій .

Звичайно , крім перерахованих існують і інші класи уразливостей захисту . Статичні аналізатори вказують на ті місця в програмі , в яких можливо знаходиться помилка . Ці підозрілі фрагменти коду можуть як містити помилку , так і виявитися абсолютно безпечними. Частина тих огріх , які допускають програмісти , не критична і загрожує тільки нестабільністю програми . Інші, навпаки , дозволяють впроваджувати шелл -код і виконувати довільні команди на віддаленому сервері. Особливий ризик в коді представляють команди , що дозволяють виконати buffer overflow та інші схожі типи атак. Таких команд дуже багато , у випадку з C / C + + це функції для роботи з рядками

(xstrcpy () , strcat () , gets () , sprintf () , printf () , snprintf () , syslog ()) , системні команди (access () , chown () , chgrp () , chmod () , tmpfile () , tmpnam () , tempnam () , mktemp ()) ,

а також команди системних викликів (exec () , system () , popen ()) .

Вручну досліджувати весь код (особливо , якщо він складається з декількох тисяч рядків) досить складно. А значить, можна прогледіти передачу якої-небудь функції неперевічених параметрів . Значно полегшити завдання можуть спеціальні засоби для аудиту , в тому числі , відома утиліта **RATS** (Rough Auditing Tool for Security) від компанії Fortify . Вона не тільки успішно впорається з обробкою коду , написаного на C / C + + , але зможе обробити ще й скрипти на Perl , PHP і Python. У базі утиліти знаходиться велика добірка з детальним описом проблемних місць в коді. За допомогою аналізатора вона обробить зведений їй вихідний код і спробує виявити баги , після чого повідомити інформацію про знайдені недоліки . RATS працює через командний рядок , як під Windows , так і * nix -системами .

Потенційно небезпечні функції

Таблиця. 1 Потенційно небезпечні функції в популярних мовах програмування

Мова програмування	Потенційно небезпечні функції
C/C++	<ul style="list-style-type: none"> - gets(), - Strcpy(),strcat(), - printf(), sprint(),vsprintf(), fsprintf(), - memmove(), memcpy(),wmemmove(), wmemcpy() - scanf(), sscanf(), fscanf(), vscanf(),vsscanf, - wcsncpy(), wcscat(), wcsncat() wmemset(),wcsncpy() - system(), popen(), execlp(), execvp(), ShellExecute(), ShellExecuteEx(), _wsystem()
Perl	System(), exec(), fork(), readpipe(),open(), , eval()

Python	exec, eval, os.system, os.popen, execfile, input, compile
Java	Class.forName(), Class.newInstance(), Runtime.exec()
PHP	include(), require(), readfile(), show_source(), highlight(), import_request_variables(), extract(), parse_str(), eval(), assert(), passthru(), exec(), system(), shell_exec(), proc_open(), mysql_query(), fopen(), echo

Використання утиліти RATS

Для установки RATS необхідно попередньо встановити бібліотеку аналізатора XML файлів, яка називається Expat. Вона доступна на сайті <http://expat.sourceforge.net/>. Встановивши Expat, необхідно в системній змінній PATH вказати шлях до бібліотек Expat, наприклад,

set PATH =% path%; C:\Program Files\Expat 2.1.0\Bin

або використати Control Panel (Панель управління) -> System (Система)-> Advanced (Додатково) / Environment Variables (Переменные среды) в Windows 8.

RATS є утилітою командного рядка, що не вимагає інсталяції. Каталог, в якому знаходиться утиліта, так само необхідно прописати в системній змінній PATH.

Параметри виклику утиліти:

rats.exe [-adhilrwxR] [--help] [--database|--db] name1 name2 ... namen

ключі утиліти

- a <fun> виводиться повідомлення про кожного входження функції <fun> в аналізованому файлі
- d <filename> або -db або -database вказує бібліотеку вразливостей, відмінної від використовуваної за замовчуванням.
- h або -help довідкова інформація
- l <language> або -language <language> зазначення мови програмування, для якого виконується перевірка
- r або -references вихідна інформація містить опис викликів об'єктів, які не є функціями, однак можуть бути джерелом уразливості
- w <1,2,3> або -warning <1,2,3> установка рівня детальності повідомлень. За замовчуванням рівень = 2
- x не використовувати вбудовану базу даних вразливостей
- R або -no -recursion не аналізувати файли, розташовані в підкаталогах аналізованого каталогу
- xml висновок інформації у форматі XML.
- html виведення інформації у форматі HTML.
- noheader не включати в вихідну інформацію дані про використовуваних

базах вразливостей

- **nofooter** не включати в вихідну інформацію допоміжні дані про час , витрачений на перевірку

- **resultonly** комбінація ключів - **noheader** , - **nofooter** , - **quiet**

- **columns** виводить номер рядка файлу , в якій виявлена уразливість

- **context** виводить рядок коду , в якій виявлена уразливість

База сигнатур утиліти представляє собою файли в форматі XML, для тих мов програмування, які можуть аналізуватися RATS. У дистрибутиві це файли rats-c.xml, rats-openssl.xml, rats-perl.xml, rats-php.xml, rats-python.xml, rats-ruby.xml.

Кожна сигнатура представлена у вигляді:

<Vulnerability>

<Name> ім'я_вразливої_функції </Name>

= <Info>

<Description> Опис_вразливості</Description>

<Severity> Рівень_ризик</Severity>

</Info>

= <InputProblem>

<Arg> Номер_аргументу </Arg>

<Severity> Рівень_ризик</Severity>

</InputProblem>

</Vulnerability>

Наприклад:

<Vulnerability>

<Name>_texeclpe</Name>

= <Info>

<Description>Many program execution commands under Windows will search the path for a program if you do not explicitly specify a full path to the file. This can allow trojans to be executed instead. Also, be sure to specify a file extension, since otherwise multiple extensions will be tried by the operating system, providing another opportunity for trojans.</Description>

<Severity>High</Severity>

</Info>

= <InputProblem>

<Arg>1</Arg>

<Severity>High</Severity>

</InputProblem>

</Vulnerability>

Така база може бути доповнена програмістом, у разі, якщо стане відомою нова уразливість.

Приклад команди перевірки файлу

D:\RATS\rats.exe --noheader -r --context crack_me.cpp > test.out

Лабораторна робота № 6 Механізми безпеки баз даних

Мета роботи: Набуття навичок налаштування системи безпеки MS SQL Server (створення користувача, розмежування прав доступу).

Теоретичні відомості.

Захист баз даних в Microsoft SQL Server

Забезпечення безпеки – важливий компонент більшості додатків баз даних, тому не потрібно думати, що в процесі розробки цим можна зайнятись в останню чергу. Кожен елемент додатку баз даних варто розробляти з урахуванням вимог забезпечення безпеки.

Керування доступом до екземплярів SQL Server

Встановлюючи з'єднання з екземпляром SQL Server, ви маєте пред'явити коректну інформацію для перевірки автентичності. Ядро бази даних виконує двохфазний процес автентифікації. Спочатку перевіряється, чи співставлене дане ім'я користувачькому запису, яка має дозвіл на підключення до екземпляра SQL Server. Далі ядро бази перевіряє, чи має даний обліковий запис дозвіл на доступ до тієї бази, до якої намагається він підключитися.

Вибір режиму автентифікації

При виборі режиму перевірки автентичності рекомендується слідувати наведеним рекомендаціям:

- **Режим перевірки автентичності Windows.** В цьому режимі SQL Server при перевірці автентичності користувача, що запитує доступ до екземпляра SQL Server, розраховує на операційну систему. Користувачу не потрібно надавати які-небудь облікові дані в рядку підключення, оскільки він вже автентифікований в ОС.
- **Комбінований режим перевірки автентичності (SQL Server і Windows).** В цьому режимі користувач може підключитись до SQL Server з використанням режиму перевірки автентичності або Windows, або SQL Server. В останньому випадку SQL Server перевіряє облікові дані користувача на відповідність дійсним іменам входу SQL Server. При використанні режиму перевірки автентичності SQL Server користувач має вказати в рядку з'єднання ім'я користувача і пароль для доступу до SQL Server.

Режим перевірки автентичності можна налаштувати через SQL Server Management Studio, виконавши такі дії:

1. В меню Start (Пуск) виберіть All Programs, Microsoft SQL Server 2008, SQL Server Management Studio (Всі програми, Microsoft SQL Server 2008, Середовище SQL Server Management Studio).
2. В діалоговому вікні Connect To Server (З'єднання з сервером) натисніть кнопку Connect (З'єднати).
3. В панелі Object Explorer (оглядач об'єктів) натисніть правою кнопкою мишки на значку екземпляра SQL Server і виберіть з контекстного меню пункт Properties (Властивості).
4. В панелі Select A Page (Вибір сторінки) виділіть значок Security (Безпека).
5. В секції Server Authentication (Серверна перевірка автентичності) виберіть потрібний режим перевірки автентичності.

Важливо. Для того, щоб зміна режиму перевірки автентичності вступила в дію потрібно перевантажити екземпляр SQL Server.

З'єднання з екземпляром SQL Server

Надання доступу користувачам і групам Windows

Можна дозволити користувачам ОС встановлювати з'єднання з сервером SQL Server шляхом створення імені входу для користувача чи групи Windows. За умовчанням, доступ до SQL Server наданий тільки членам локальної групи адміністраторів Windows та обліковому записові служби, що запускає служби SQL.

Доступ до екземпляра SQL Server можна надати, створивши ім'я входу або шляхом безпосереднього введення команд SQL, або через інтерфейс SQL Server Management Studio. Наступний код надає доступ до екземпляра SQL Server користувачу домена Windows ADWORKS\jluca:

```
CREATE LOGIN [ADWORKS\jluca] FROM WINDOWS;
```

При підключенні до SQL Server з використанням імені входу Windows, SQL Server довіряє ОС щодо перевірки автентичності і тільки перевіряє чи має користувач Windows відповідне ім'я входу, яке визначено в цьому екземплярі сервера SQL Server, або чи належить це ім'я входу групі Windows з відповідним іменем входу в цей екземпляр SQL Server.

З'єднання, що використовує ім'я входу Windows, називається довірчим.

Надання доступу іменам входу SQL Server

В режимі перевірки автентичності Windows та SQL Server також можна створити імена входу SQL Server і керувати ними. При створенні імені входу SQL Server необхідно задати для цього імені входу пароль. Користувачі

мають вказувати пароль при з'єднанні з екземпляром SQL Server. При створенні імені входу SQL Server можна задати для нього ім'я бази даних і мову за умовчанням.

Доступ до екземпляра SQL Server можна надати, створивши ім'я входу SQL Server або шляхом безпосереднього введення команд SQL, або через інтерфейс SQL Server Management Studio. В наступному прикладі створюємо ім'я входу SQL Server "Mary" і назначаємо для користувача Mary базу даних Adventure Works якості бази даних за умовчанням.

```
CREATE LOGIN Mary  
WITH PASSWORD = '34TY$$543',  
DEFAULT_DATABASE = AdventureWorks;
```

В процесі інсталяції SQL Server створюється одне ім'я входу SQL Server - sa. Ім'я входу sa створюється у будь-якому випадку, навіть якщо ви вибрали в процесі інсталяції режим перевірки автентичності Windows.

Права доступу до екземпляра SQL Server

Для розмежування доступу SQL Server надаються серверні ролі:

- **bulkadmin** – може виконувати вираз BULK INSERT
- **dbcreator** – може створювати, змінювати, видаляти і відновлювати бази даних
- **diskadmin** – може управляти файлами на диску
- **processadmin** – може завершати процеси
- **securityadmin** – може керувати іменами входу і назначати дозвіл
- **serveradmin** – може змінювати параметри сервера і завершати роботу сервера
- **setupadmin** – може керувати зв'язаними серверами і виконувати системні збережені процедури
- **sysadmin** – може виконувати на сервері будь-які дії

Додавання імені входу до серверної ролі. Додати ім'я входу до існуючої серверної ролі можна за допомогою системної збереженої функції sp_addsrvrolemember. Наступний приклад додає Mary до системної ролі sysadmin:

```
EXECUTE sp_addsrvrolemember "Mary", "sysadmin";
```

Надання доступу до баз даних

Всі імена входу, за виключенням серверної ролі sysadmin, мають бути співставленні користувачам баз даних, які, в свою чергу, співставленні бази даних, до якої їм потрібен доступ. Члени ролі sysadmin співставленні користувачу dbo у всіх базах сервера.

Додавання користувача бази даних

Додати користувача бази даних можна за допомогою інструкції CREATE USER. Наступний приклад коду T-SQL створює ім'я входу Peter і співставленого з ним користувача в базі даних Adventure Works:

- Створюємо ім'я входу Peter

```
CREATE LOGIN Peter WITH PASSWORD='Tyu87IOR0';
```

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Додаємо користувача бази даних Peter, який співставлений імені входу Peter в БД AdventureWorks.

```
CREATE USER Peter FOR LOGIN Peter;
```

Керування користувачами бази даних

Перевірити чи має поточне ім'я входу доступ до бази даних можна за допомогою наступної інструкції:

```
SELECT HAS_DBACCESS("AdventureWorks");
```

Якщо необхідно тимчасово вимкнути доступ користувачу до бази даних, можна відізвати дозвіл CONNECT для цього користувача. Наступний приклад відзиває дозвіл CONNECT для користувача Peter:

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Відзиваємо дозвіл connect для Peter on the database AdventureWorks.

```
REVOKE CONNECT TO Peter;
```

Видалити користувача в базі даних можна за допомогою інструкції DROP USER.

Надання дозволу на базу даних

Створивши користувачів бази даних, необхідно керувати дозволами для цих користувачів. Це можна здійснити шляхом додавання користувачів в ролі бази даних або надавши самим користувачам відповідний дозвіл.

Створення ролі бази даних

Ролі бази даних можна використовувати для призначення дозволів бази даних в групі користувачів. В SQL Server за умовчанням створюється набір

ролей бази даних:

- **db_accessadmin** – може керувати доступом до бази даних
- **db_backupoperator** – може виконувати резервне копіювання бази даних
- **db_datareader** – може читати дані з таблиць всіх користувачів
- **db_datawriter** – може добавляти, видаляти і оновлювати дані в таблицях всіх користувачів
- **db_ddladmin** – може виконувати будь-які команди DDL в базі даних
- **db_denydatareader** – не може читати будь-які дані в таблицях користувачів
- **db_denydatawriter** – не може добавляти, видаляти і оновлювати будь-які дані в таблицях користувачів
- **db_owner** – може виконувати всі дії стосовно налаштування конфігурації та обслуговування
- **db_securityadmin** – може змінювати членство в ролях бази даних і керувати дозволами
- **public** – Особлива роль бази даних. Всі користувачі належать до ролі public. Користувачів з групи public неможна видалити.

Можна додавати ролі бази даних, якщо потрібно згрупувати користувачів у відповідності до вимог визначених дозволів. Наступний приклад коду створює роль бази даних з іменем Auditors і додає користувача Peter в цю нову роль.

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Створюємо роль Auditors в базі даних AdventureWorks.

```
CREATE ROLE Auditors;  
GO
```

- Додаємо користувача Peter до ролі Auditors

```
EXECUTE sp_addrolemember "Auditors", "Peter";
```

Надання повноважень на доступ до бази даних

В якості альтернативи використання фіксованих ролей бази даних можна надавати дозволи користувачам і ролям бази даних. Дозволами можна керувати за допомогою інструкцій GRANT, DENY і REVOKE. В

наступному прикладі ми надаємо дозвіл BACKUP DATABASE користувачу Peter:

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Надаємо дозвіл користувачу бази даних Peter на резервне копіювання бази даних AdventureWorks.

```
GRANT BACKUP DATABASE TO Peter;
```

Коли ви використовуєте інструкцію DENY для видалення дозволу для користувача, цей користувач не може успадкувати той же самий дозвіл, якщо являється членом ролі бази даних, яка має цей дозвіл. Але якщо для видалення дозволу ви скористаетесь інструкцією REVOKE, то користувач зможе успадкувати той самий дозвіл, якщо являється членом ролі бази даних, якій цей дозвіл вже надано. Використовуйте інструкцію REVOKE тільки для видалення дозволу, який було надано раніше.

Керування доступом до схем

В SQL Server починаючи з версії 2005 реалізована концепція ANSI для схем. Схеми – це контейнери об'єктів, які дозволяють згрупувати об'єкти баз даних. Схеми мають великий вплив на те, як користувачі посилаються на об'єкти баз даних. В SQL Server об'єкт баз даних називається іменем, яке складається з чотирьох компонентів наступної структури:

```
<Server>.<Database>.<Schema>.<Object>.
```

Схеми бази даних можна створювати за допомогою інструкції CREATE SCHEMA. Створивши схему, можна створювати об'єкти бази даних і назначати дозволи в межах однієї транзакції, яка визивається інструкцією CREATE SCHEMA.

В наступному прикладі створюється схема з іменем Accounting, назначаючи користувача Peter власником схеми і створюється таблиця з іменем Invoices. В цьому прикладі також надається дозвіл select ролі бази даних public.

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Створюємо схему Accounting з власником Peter.

```
CREATE SCHEMA Accounting AUTHORIZATION Peter;  
GO
```

- Створюємо таблицю Invoices в схемі Accounting.

```
CREATE TABLE Accounting.Invoices (
InvoiceID int,
InvoiceDate smalldatetime,
ClientID int);
GO
```

- Надаємо дозвіл SELECT на нову таблицю ролі public.

```
GRANT SELECT ON Accounting.Invoices TO public;
GO
```

- Додаємо рядок даних в нову таблицю. Зверніть увагу на двохкомпонентне ім'я, яке ми використовуємо для звертання до таблиці в поточній базі даних.

```
INSERT INTO Accounting.Invoices
VALUES (101,getdate(),102);
```

Видалити схему можна за допомогою інструкції DROP SCHEMA. В SQL Server не допускається видалення схеми, якщо в схемі є об'єкти. Інформацію стосовно схем можна отримати, виконавши запит до представлення каталога sys.schemas. Наступний приклад виконує запит до представлення каталога sys.schemas, щоб отримати інформацію щодо схеми:

```
SELECT *
FROM sys.schemas;
```

Керування доступом до таблиць і стовпців

Таблиця і стовпці зберігають дані, які добувають і створюють додатки. Керуванням доступом до цих даних здійснюється через ієрархію дозволів SQL Server. Керувати цією ієрархією дозволів можна за допомогою інструкцій GRANT, DENY та REVOKE.

GRANT. Дозволяє ролі або користувачу виконувати операції, що визначені в момент надання дозволу.

DENY. Забороняє користувачу або ролі визначені дозволи і попереджає наслідування цих дозволів від інших ролей..

REVOKE. Відкликає раніше заборонені або надання дозволи.

Зміна прав доступу до таблиці

Доступ до таблиці керується дійсними дозволами, які надані користувачу на дану таблицю. Доступом користувача до таблиці можна керувати за

допомогою керуванням дозволами до таблиці. Дозволи, якими можна керувати:

- **ALTER** - дозволяє змінювати властивості таблиці
- **CONTROL** – надає дозволи, що аналогічні володінню
- **DELETE** - дозволяє видаляти рядки з таблиці
- **INSERT** - дозволяє додавати рядки в таблицю
- **REFERENCES** – дозволяє посилалися на таблицю з зовнішнього ключа
- **SELECT** – дозволяє здійснювати вибірку рядків з таблиці
- **TAKE OWNERSHIP** – дозволяє присвоювання схеми або таблиці
- **UPDATE** – дозволяє змінювати рядки в таблиці
- **VIEW DEFINITION** - дозволяє доступ к метаданным таблицы

Доступ користувачам бази даних і ролям можна надавати за допомогою інструкції GRANT. В наступному прикладі дозвіл SELECT, INSERT і UPDATE на таблицю Sales.Customer надаються користувачу Bill (код для управління доступом до таблиць в цьому і наступних розділах містить в файлах прикладів під іменем ManagingAccessToTables.sql.).

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Надаємо користувачу Sara деякі дозволи на таблицю Sales.Customer table.

```
GRANT SELECT,INSERT,UPDATE  
ON Sales.Customer TO Bill;
```

Обмеження доступу до таблиці

Якщо потрібно не допустити користувача до таблиці, то можна зітнутися з двома ситуаціями. Якщо ви до цього надавали користувачу дозвіл на цю таблицю, то для видалення раніше наданих дозволів потрібно скористуватися інструкцією REVOKE. Наприклад:

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Відкликаємо дозвіл SELECT на таблицю Sales.Customer у Bill.

```
REVOKE SELECT  
ON Sales.Customer TO Bill;
```


Хоча користувач може зберегти відізований дозвіл, якщо належить до ролі, якій це дозвіл надано. В цьому випадку необхідно використовувати інструкцію DENY, щоб заборонити доступ цьому користувачу. Наприклад:

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Забороняємо користувачу Bill дозвіл DELETE на таблицю Sales.Customer незалежно від того, які дозволи цей користувач міг успадкувати від ролі.

```
DENY DELETE  
ON Sales.Customer TO Bill;
```

Надання доступу до окремих стовпчиків

В SQL Server є можливість надати або відключити доступ до окремих стовпчиків, замість того, щоб працювати зі всією таблицею. Ця можливість надає гнучкість у прикритті доступу, наприклад, до конфіденційних даних в деяких стовпцях. Дозволи, якими можна керувати для стовпчиків у таблиці:

- **SELECT** - дозволяє виконати вибірку з стовпців
- **UPDATE** - дозволяє змінювати дані у стовпчику

Доступ до окремих стовпців можна надати за допомогою інструкції GRANT. В наступному прикладі дозволи SELECT та UPDATE надаються користувачу Bill на стовпці Demographics та Modified Date таблиці Sales.Individual.

- Змінюємо контекст з'єднання на базу даних AdventureWorks.

```
USE AdventureWorks;  
GO
```

- Надаємо дозвіл SELECT та UPDATE користувачу Bill на конкретні стовпці таблиці Sales.Individual

```
GRANT SELECT,UPDATE (  
Demographics, ModifiedDate) ON Sales.Individual TO Bill;
```

Криптографічний захист таблиць БД

Для шифрування змісту колонок таблиці БД необхідно виконати наступні дії:

1. Впевнитися, що для екземпляра SQL Server створено мастер-ключ. Мастер-ключ є вершиною ієрархії методів криптографічного захисту. Він створюється при інсталяції екземпляра сервера.

```
USE master;  
GO  
SELECT *  
FROM sys.symmetric_keys  
WHERE name = '##MS_ServiceMasterKey##';  
GO
```

2. Створити мастер-ключ бази даних encrypt_test:

```
USE encrypt_test;  
GO  
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Password123';  
GO
```

3. Створити сертифікат. Сертифікат підписується SQL Server.

```
USE encrypt_test;  
GO  
CREATE CERTIFICATE Certificate1  
WITH SUBJECT = 'Protect Data';  
GO
```

4. Створення симетричного ключа:

```
USE encrypt_test;  
GO  
CREATE SYMMETRIC KEY SymmetricKey1  
WITH ALGORITHM = AES_128  
ENCRYPTION BY CERTIFICATE Certificate1;  
GO
```

5. Зміна схеми даних. Для зберігання зашифрованої інформації даних тип колонки має бути varbinary, тому в таблицю додається колонка такого типу з іменем Credit_card_number_encrypt:

```
USE encrypt_test;  
GO  
ALTER TABLE Customer_data  
ADD Credit_card_number_encrypt varbinary(MAX) NULL  
GO
```

6. Шифрування колонки таблиці. Для шифрування використовується команда EncryptByKey. Перед шифруванням необхідно відкрити симетричний ключ, а по закінченню закрити.

```
USE encrypt_test;
GO
-- Opens the symmetric key for use
OPEN SYMMETRIC KEY SymmetricKey1
DECRYPTION BY CERTIFICATE Certificate1;
GO
UPDATE Customer_data
SET Credit_card_number_encrypt = EncryptByKey
(Key_GUID('SymmetricKey1'),Credit_card_number)
FROM dbo.Customer_data;
GO
-- Closes the symmetric key
CLOSE SYMMETRIC KEY SymmetricKey1;
GO
```

7. Видалити незашифровану колонку:

```
USE encrypt_test;
GO
ALTER TABLE Customer_data
DROP COLUMN Credit_card_number;
GO
```

8. Читання зашифрованих даних:

```
USE encrypt_test;
GO
OPEN SYMMETRIC KEY SymmetricKey1
DECRYPTION BY CERTIFICATE Certificate1;
GO
-- Читання розшифрованих даних
SELECT
Customer_id, Credit_card_number_encrypt AS 'Encrypted Credit Card Number',
CONVERT(varchar, DecryptByKey(Credit_card_number_encrypt))
AS 'Decrypted Credit Card Number'
FROM dbo.Customer_data;

CLOSE SYMMETRIC KEY SymmetricKey1;
GO
```

9. Для додавання даних в зашифровану колонку використовуються команди:

```

USE encrypt_test;
GO
OPEN SYMMETRIC KEY SymmetricKey1
DECRYPTION BY CERTIFICATE Certificate1;
INSERT INTO dbo.Customer_data (Customer_id, Customer_Name,
Credit_card_number_encrypt)
VALUES (25665, 'mssqltips4', EncryptByKey( Key_GUID('SymmetricKey1'),
CONVERT(varchar,'4545-58478-1245') ) );
GO

```

10. Надання повноважень для доступу до зашифрованих даних. Для доступу до зашифрованих даних користувач повинний мати доступ до сертифіката і симетричного ключа. Надати такі повноваження користувачу test можна командою:

```

GRANT VIEW DEFINITION ON SYMMETRIC KEY::SymmetricKey1 TO test;
GO
GRANT VIEW DEFINITION ON Certificate::Certificate1 TO test;
GO

```

Завдання для лабораторної роботи

1. Напишіть команди Transact SQL для наступних дій:
 1. Створити новий обліковий запис;
 2. Назначити користувачу роль БД (будь-яку);
 3. Надати користувачу *Bill* повноваження на доступ к збереженої процедури (будь-якої);
 4. Відкликати всі надані користувачу *John* привілеї.
2. Поясніть призначення наступних команд Transact SQL:
 - a) GRANT SELECT, INSERT
ON SUPPLIES
TO J_Smith
WITH GRANT OPTION AS Economists
 - b) EXEC[UTE] sp_addlogin 'king_of_the_db',
'a2h7d0f7dg84mdf94',
'PROJECTS',
'Ukrainian',
'master',
'NULL'
 - c) REVOKE ALL TO 'M_Ivanenko'
CASCADE

3. Напишіть послідовність команд, яка шифрує вміст однієї з колонок таблиці в створеній БД.

Питання по самоконтролю по лабораторній роботі

1. Які етапи автентифікації проходять користувачі для роботи з MS SQL Server?
2. Як можна встановити довірче з'єднання?
3. Коли варто використовувати змішаний режим автентифікації?
4. Який термін використовується фактично при доступі об'єкта до БД? (login чи user)
5. Яке призначення ролі сервера і ролі БД?
6. Що значить параметр CASCADE?
7. Коли виникає конфлікт доступу?
8. Призначення сертифікату в системі криптографічного захисту MS SQL Server?

Лабораторна робота 7. Побудова моделі порушника в АС класу 3

Мета роботи

Навчитись аналізувати середовища функціонування інформаційної системи на наявність загроз та потенційних порушників.

Вміст завдання

1. Задати в інтерфейсі Microsoft SDL Threat Modeling Tool діаграму інформаційних потоків в мережі факультету.
2. Згенерувати перелік загроз.
3. Навести модель порушника, модель загроз.
4. Проаналізувати виявлені загрози, виконати ранжирування загроз згідно відповідних ризиків.

Теоретичні відомості

Модель порушника

В загальному виді модель порушника може бути наведена описово. Спочатку треба задатись категоріями користувачів, які потенційно можуть мати доступ до ресурсів інформаційної системи. Наприклад, типовий перелік категорій користувачів:

- 1.Адміністратори:
 - 1.1.Адміністратор безпеки;
 - 1.2. Адміністратор мережі;

- 1.3. Адміністратор криптографічної підсистеми;
- 1.4. Системний адміністратор;
- 1.5.Адміністратор баз даних.
- 2.Користувачі.
- 3.Оператори.
- 4.Керівники (якщо їх права відрізняються від прав користувачів).
- 5.Супровідники, розробники, постачальники.
- 6.Технічний персонал

7.Зовнішні користувачі.

Далі формується модель порушника у виді таблиці:

Табл.6.1. Модель порушника

Категорія порушника	Кваліфікація	Права в системі	Засоби, якими володіє порушник
Наводяться категорії потенційних порушників з числа 1-7.	Припускається високою	Опис прав користувачів відповідної категорії	Доступні порушнику засоби, які можуть бути використані для здійснення порушень

На основі переліку загроз, притаманних даним інформаційній системі формується модель загроз у виді таблиці:

Табл.6.2. Модель загроз

Загроза	Джерело загрози	Мета	Порушник
Формулюється вид загрози відповідно класифікації загроз в даній інформаційній системі	Наявність вразливості чи іншого фактора наявності загрози	Ресурси інформаційної системи, які є метою навмисного впливу / вражаються від ненавмисного впливу	Порушник, який може виконати відповідний вплив на систему

Класифікація загроз може бути проведена відповідно будь-яких методик класифікації (наприклад, STRIDE).

Підходи до моделювання атак

Існує принаймні 3 загальні підходи до моделювання атак.

1.Зосереджений на атакуючому

Проводиться аналіз мети атакуючого, як він може досягти цієї мети.

Цей підхід звичайно починається з розгляду точок доступу, через які може діяти атакуючий, та ресурсів, які є його метою.

2.Зосереджений на програмному забезпеченні

Software-центричне моделювання загроз (також «системоцентричне», чи «архітектуроцентричне») починає з аналізу конструкції системи, та розгляду типів атак, які можуть бути застосовані відносно кожного з елементів системи. Цей підхід використовується в моделюванні загроз в Microsoft's Security Development Lifecycle.

Зосереджений на ресурсах, щодо яких здійснюються загрози

Починається з аналізу доступу до сховищ інформації, яка підлягає захисту.

При аналізі вразливостей системи слід проводити її тестування. Тестові спроби вторгень виконують на модельних зразках захищуваних даних із використанням техніки «білої скриньки» (вихідні коди програмного забезпечення, щодо якого шукаються вразливості, відомі, також відомі вхідні дані тестів та реакція на них) чи «чорної скриньки» (відомі лише вхідні дані та реакція програмного забезпечення на них). Значна кількість вразливостей операційних систем, систем управління базами даних, веб-серверів та іншого системного забезпечення є відомою і навіть описана у відповідних RFC.

Приблизний перелік етапів, які слід здійснити при аналізі загроз:

1.Визначити вимоги щодо прикладного ПЗ:

- Ідентифікувати бізнес-цілі;
- Ідентифікувати ролі користувачів, які будуть взаємодіяти з програмним забезпеченням;
- Ідентифікувати дані, якими оперує ПЗ;
- Ідентифікувати різні випадки використання функцій ПЗ при операціях з даними під керуванням даного ПЗ.

2.Розглянути модель архітектури:

- Моделі компонентів інформаційної системи;
- Модель взаємодії компонентів інформаційної системи;
- Модель взаємодії із зовнішнім середовищем;
- Модель взаємодії ролей користувачів із компонентами та сховищами даних для кожного випадку використання.

3. Ідентифікувати будь-які загрози для конфіденційності, цілісності, доступності даних та ПЗ, засновані на матриці доступу (правилах розмежування доступу);

4. Визначити ризики та їхні наслідки. Визначити процедури по запобіганню ризикам;
5. Неперервно відновлювати модель, базуючись на поточному стані засобів та заходів захисту в інформаційній системі.

Microsoft SDL Threat Modeling Tool дозволяє визначити притаманні системі загрози відповідно класифікації STRIDE. Діаграма інформаційних потоків розподіленої системи задається за допомогою графічних елементів (рис.6.1). Інформація про притаманні загрози формується у виді звіту, в якому загрози класифіковані по 6 класам STRIDE (рис.6.2).

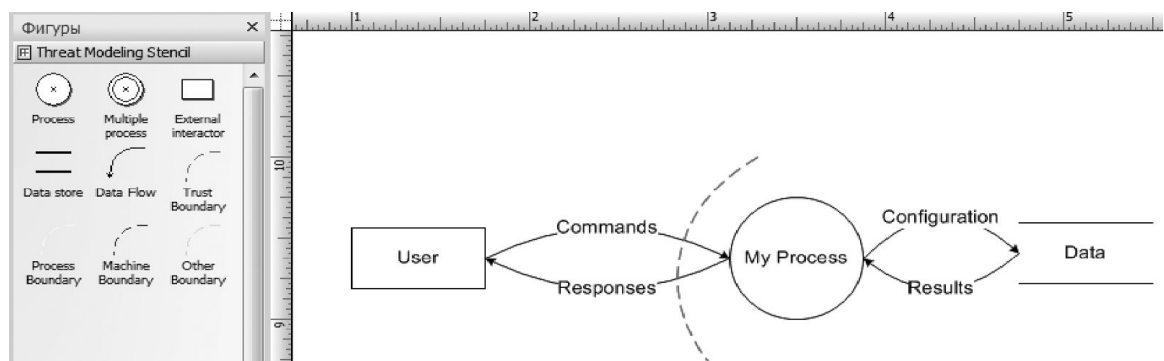


Рис. 6.1. Елементи для побудови інформаційних потоків у системі

	ID	Element Name	Element Type	Element Diagram References	Threat Type	Bug ID	Completion
▶	3	Commands (User to ...	DataFlow	Context	Tampering		
	4	Commands (User to ...	DataFlow	Context	InformationDisclosure		
	5	Commands (User to ...	DataFlow	Context	DenialOfService		
	9	Configuration (My Pr...	DataFlow	Context	Tampering		
	10	Configuration (My Pr...	DataFlow	Context	InformationDisclosure		
	11	Configuration (My Pr...	DataFlow	Context	DenialOfService		
	6	Responses (My Proce...	DataFlow	Context	Tampering		
	7	Responses (My Proce...	DataFlow	Context	InformationDisclosure		
	8	Responses (My Proce...	DataFlow	Context	DenialOfService		
	12	Results (Data to My P...	DataFlow	Context	Tampering		
	13	Results (Data to My P...	DataFlow	Context	InformationDisclosure		
	14	Results (Data to My P...	DataFlow	Context	DenialOfService		
	21	Data	DataStore	Context	Tampering		
	22	Data	DataStore	Context	Repudiation		
	23	Data	DataStore	Context	InformationDisclosure		
	24	Data	DataStore	Context	DenialOfService		
	1	User	Interactor	Context	Spoofing		
	2	User	Interactor	Context	Repudiation		
	15	My Process	Process	Context	Spoofing		
	16	My Process	Process	Context	Tampering		
	17	My Process	Process	Context	Repudiation		
	18	My Process	Process	Context	InformationDisclosure		
	19	My Process	Process	Context	DenialOfService		
	20	My Process	Process	Context	ElevationOfPrivilege		

Рис.6.2. Загрози, притаманні системі та їх класифікація

Лабораторна робота 8. Механізми захисту операційних систем

Мета роботи

Ознайомитись із вбудованими засобами захисту в операційних системах Windows та Linux.

Вміст завдання

Частина а)

Для ОС Windows XP SP2 (чи вище) здійснити такі дії:

1.Задати політику безпеки у вигляді таблиці:

Табл.1

	Папка1	Файл1	Диск1	...
Корист1 Група Адміністратори	Права	Права	Права	
Корист2 Група Користувачі	Права	Права	Права	
Корист3 Група Гості	Права	Права	Права	

2.Створити заданих користувачів(суб'єктів) та об'єкти.

3.Стосовно об'єктів задати списки контролю доступу відповідно до політики безпеки (табл.1). Навести ACL.

4.Дослідити правила наслідування прав

- для суб'єктів, що належать до груп із різними правами,
- дослідити наслідування правил доступу для файлів, що вкладені до папки із заданими правами,
- дослідити, чи зберігаються задані права на файл, якщо скопіювати його до іншої директорії,

5. Перевірити ситуацію по дії дозволів:

- Користувачів не можуть працювати із каталогом чи файлом, якщо вони не мають явного дозволу на це, чи не відносяться до групи, яка має відповідний дозвіл;
- Дозволи мають накопичувальний ефект, за виключенням права No Access, яке відмінює усі інші існуючі дозволи. Наприклад, якщо група Engineering має дозвіл Change для якогось файла, а група Finance має для цього файла лише дозвіл Read і Петров входить до обох груп, то в Петрова буде дозвіл Change. Однак, якщо дозвіл для групи Finance зміниться на No Access, то Петров не зможе використовувати цей файл, незважаючи на те, що він входить до групи, яка має на це дозвіл.
- Дозвіл Full Control відрізняється від Change тим, що дає право на зміну дозволів (Change Permission) та вступ у володіння файлом (Take Ownership).

- 6.Задати аудит дій (загальний аудит та аудит стосовно об'єктів). Навести SACL.
- 7.Здійснити спроби доступу а)згідно правил політики безпеки б)всупереч правилам створеними користувачами до створених суб'єктів.
- 8.Пересвідчитись, що журнал реєструє відповідні події. Проаналізувати, де система зберігає дані реєстрації та які можливості надаються по їх перегляду\впорядкуванню.
- 9.Проаналізувати можливості ОС з криптографічного захисту.
- 10.Проаналізувати можливості ОС по керуванню паролями.

Результати дій оформити у вигляді звіту, до якого включити табл.1 та скріншоти, що ілюструють відповідні дії. Навести висновки та відповіді на поставлені запитання

Вказати, які із розглянутих засобів можна віднести до підсистеми ідентифікації-автентифікації, до підсистеми аудиту, розмежування доступу, криптографічної підсистеми).

Частина б)

1.Для ОС Linux Debian виконайте послідовність дій та поясніть одержані на кожному кроці результати. Для довідки використовуйте команду `man` ім'я_команди

1. `adduser` ім'я_користувача
2. `> file`
3. `ls -l`
4. `chmod права file`
5. `ls -l`
6. `vim &`
7. `adduser` інший_користувач
8. `su` інший_користувач
9. `cat file`
10. `less file`
11. `more file`
12. `chown` інший_користувач: інший_користувач `file`
13. `chmod 777 file`
14. `echo "test " >> file`
15. `mv file /home/інший_користувач`
16. `cp file file2`
17. `touch file`
18. `ps auxx | grep test1`
19. `killall -u` ім'я_користувача
20. `kill` номер_процеса
21. Перейдіть до користувача ім'я_користувача.

22. /etc/init.d/ssh restart
23. /etc/init.d/ssh stop
24. Перейдіть до користувача інший_користувач та виконайте те ж саме.
25. Перейдіть до користувача ім'я_користувача.
26. /var/run/bind\$ b=12
27. /var/run/bind\$ echo \$b
28. Перейдіть до користувача інший_користувач
29. echo \$b
30. b=3
31. echo \$b
32. Поверніться до користувача ім'я_користувача та виконайте echo \$b

2. Знайдіть файли зі встановленим suid, sgid (оберіть з них декілька найбільш відомих).

Поясніть, чому відповідні команди не можуть функціонувати без цього біта.

3. Створіть файл. Встановіть для нього відповідні флаги. Коли таке встановлення неможливе?

4. Відновіть політику розмежування доступу для користувачів mark, ed, herb, lynda, wheel по прикладу файла /etc/sudoers (див. приклад нижче).

5. Чи може користувач lynda запускати інтерпретатор команд?

Чи вимагається від користувачів вводити пароль?

6. Використовуючи visudo, ознайомтесь з можливостями редактора. Попередньо створіть декількох користувачів, задайте для них політику безпеки та реалізуйте її за допомогою visudo для файла /etc/sudoers.

Приклад файла /etc/sudoers:

```
# Псевдонимы для факультетов физики и вычислительной техники
Host_Alias CS = tigger, anchor, piper, moet, sigi
Host_Alias PHYSICS = eprince, pprince, icarus

# Набор команд
Cmdn Alias      DUMP = /sbin/dump, /sbin/restore
Cmdn Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmdn Alias      SHELLS = /bin/sh, /bin/tcsh, /bin/csh, /bin/bash, /bin/ash,

# Права доступа
mark, ed        PHYSICS = ALL
herb            CS = /usr/sbin/tcpdump : PHYSICS = (operator) DUMP
lynda           ALL = (ALL) ALL, !SHELLS
%wheel          ALL, !PHYSICS = NOPASSWD: PRINTING
```

Теоретичні відомості

Захист даних в Linux

Кожен файл в Linux належить власнику та групі. Власник файла має особливий привілей, недоступний для інших користувачів системи: він може змінювати права доступу до файлу. Зокрема, власник файла може задати права таким чином, що ніхто, крім нього, не зможе звертатись до файлу.

Власником файла завжди є одна людина, тоді як до групи власників може входити декілька користувачів. Традиційно інформація про групи зберігалась в /etc/group, але тепер частіше її зберігають на мережевому сервері NIS чи LDAP.

Власник файла повинен визначити, які операції можуть виконувати над файлом члени групи.

Виявити ідентифікатори власників можна за допомогою команди

ls -l ім'я_файлу

Приклад:

\$ ls -l /fruit/orange/file

Файлом володіє користувач orange, а група, якій він належить, називається fruit.

ОС Linux відстежує не імена власників та груп, а їх ідентифікатори. У загальному випадку ідентифікатори користувачів (UID) та відповідні ним імена зберігаються в файлі /etc/passwd, а ідентифікатори та назви груп знаходяться в файлі /etc/group. Текстові еквіваленти ідентифікаторів визначаються виключно для зручності користувачів. Щоб команда типу ls могла вивести інформацію про приналежність файлу у зручному для користувача виді, вона повинна переглянути базу ідентифікаторів та знайти в ній потрібні імена.

Процес не може явно змінити жоден зі своїх ідентифікаторів, але є особлива ситуація, коли нові ефективні ідентифікатори встановлюються непрямо. Річ у тому, що існують два спеціальних біта, які встановлюються в масці прав доступу до файла:

Setuid (Set User ID- біт зміни ідентифікатора користувача) та Setgid (Set Group ID- біт зміни ідентифікатора групи). Якщо запускається виконуваний файл, в якого встановлений один із цих бітів, то ефективними ідентифікаторами створюваного процесу стають ідентифікатори власника файла, а не ідентифікатори користувача, який запустив програму. Зміна повноважень дійсна тільки на час роботи програми.

Біт «setuid» дозволяє рядовим користувачам запускати програми, які виконують жорстко регламентовані адміністративні дії. Наприклад, команда passwd, за допомогою якої користувач змінює свій пароль, звертається до файлу /etc/shadow (чи /etc/passwd), який належить суперкористувачу, для цього в неї встановлений біт «setuid». Вона модифікує файл строго визначеним образом та завершується. Але навіть така дія може стати причиною зловживань, тому, перед тим, як внести зміни, команда passwd перевіряє, чи знає користувач свій поточний пароль.

Суперкористувач (root)

Визначальна характеристика облікового запису суперкористувача - UID, що дорівнює нулю. Linux не забороняє змінювати ім'я цього облікового запису, чи створювати інший запис з нульовим ідентифікатором, це може призвести до непередбачуваних наслідків.

Традиційна система UNIX дозволяє суперкористувачу (тобто, будь-якому паровесу, ефективний ідентифікатор користувача якого дорівнює нулю) виконувати над файлом чи процесом будь-яку припустиму операцію. Крім того, деякі системні виклики (звертання до ядра) дозволено здійснити лише суперкористувачу.

Приклади операцій, доступних лише суперкористувачу:

- зміна кореневого каталога процесу за допомогою команди `chroot`;
- створення файлів пристроїв;
- встановлення системного годинника;
- збільшення лімітів використання ресурсів та підвищення пріоритетів процесів;
- задання мережевого імені комп'ютера;
- конфігурування мережевих інтерфейсів;
- відкриття привілейованих мережевих портів (з номерами менше 1024);
- останов системи.

Оскільки суперкористувач є таким самим членом системи, як і інші, користувачі, можна увійти до системи безпосередньо під іменем `root`. Однак, це є невдалим рішенням з точки зору спостережуваності системи: не буде зроблено ніяких записів про те, які операції виконав суперкористувач; сценарій реєстрації суперкористувача не передбачає збору додаткової ідентифікуючої інформації. Коли під іменем `root` в систему можуть входити декілька користувачів, не існує способу визначити, хто з них та коли це робив.

Внаслідок цього в більшості систем реєстрації реєстрація під іменем `root` може бути заборонена на терміналах та по мережі, тобто всюди, окрім системної консолі. Краще одержати доступ до облікового запису `root` за допомогою команди `su`.

Знаючи пароль якогось користувача, можна зареєструватись в системі під його іменем, вводячи `su ім'я_користувача`. У відповідь буде видано запит на ввід пароля. Інший варіант: спочатку стати суперкористувачем, скористувавшись командою `su`, а потім за допомогою цієї ж команди перейти в інший обліковий запис – в цьому разі вводити пароль не вимагається.

Утиліта `sudo`

Ця утиліта в якості аргумента приймає командний рядок, який підлягає виконанню від імені користувача `root` (або іншого вповноваженого користувача). Утиліта звертається до файла `/etc/sudoers`, де міститься список

користувачів, які мають дозвіл на її виконання, та перелік команд, які вони можуть вводити на конкретному комп'ютері. Якщо запитувана команда дозволена, утиліта `sudo` запрошує користувача ввести його власний пароль, та виконує команду від імені суперкористувача.

Для модифікації файлу `etc/sudoers` призначена спеціальна команда `visudo`, яка перевіряє, чи не редагується цей файл кимось стороннім, потім відкриває його в редакторі, а перед інсталяцією файлу виконує синтаксичний контроль. Останній етап особливо важливий, оскільки помилка у файлі `/etc/sudoers` може не дозволити повторно визвати утиліту `sudo` для виправлення файлу.

Псевдокористувачі

Тільки користувач `root` має для ядра Linux особливий статус. Є, однак ще декілька псевдокористувачських облікових записів, які використовуються для системних цілей. Паролі цих псевдокористувачів в файлі `/etc/shadow` зазвичай замінюють зірочкою, щоб не можна було увійти до системи під службовим іменем:

Власник системних команд. В деяких UNIX-системах користувач `bin` є власником більшості системних команд, а також каталогів, в яких вони зберігаються. Призначення окремого користувача для цих цілей часто вважається надлишковим і небезпечним, тому в сучасних системах (в тому числі і Linux) відповідна роль в основному виконується користувачем `root`. Але `bin` є стандартною, тому від неї не можна повністю позбавитись.

Власник непривілейованих системних команд. Файли та процеси, які є частиною ОС, але не повинні належати суперкористувачу, іноді передаються у володіння користувачу `daemon`. Як і обліковий запис `bin`, запис `daemon` майже не задіяна у більшості дистрибутивів Linux.

Загальний користувач мережевої файлової системи (NFS). Мережева файлова система NFS (Network File System) використовує обліковий запис `nobody` для представлення суперкористувачів в інших системах, де є загальнодоступні файли. Щоб позбавити суперкористувачів їх виключних прав, які є зайвими у даному контексті, NFS повинна на час сеансу віддаленого доступу замінити нульовий ідентифікатор чимось іншим. Для цієї мети і слугує обліковий запис `nobody`.

Аспекти безпеки

Файл `/etc/shadow` в принципі не потребує обслуговування. Хоча, наприклад, існує сценарій Perl, який дозволяє перевіряти цей файл на наявність порожніх (нульових) паролей:

```
$ sudo perl -F: -ane 'print if not $F[1];' /etc/shadow
```

Цей сценарій виконує перевірку, відсилає результати адміністратору і може запускатись за допомогою демона `cron`. Щоб спростити собі виконання перевірки правомірності внесених в облікові записи змін, можна написати сценарій, який буде щоденно порівнювати файл `/etc/passwd` з його

версією за попередній день (за допомогою команди diff) та повідомляти про виявлені розбіжності.

Єдина відмінна риса користувача root полягає в тому, що його ідентифікатор дорівнює нулю. Оскільки в файлі /etc/passwd може бути декілька записів з таким ідентифікатором, то існує і декілька способів входу в систему в якості суперкористувача.

Один із способів, який хакери, одержавши доступ до інтерпретатора команд суперкористувача, широко застосовують для відкриття «таємного ходу» - редагування файлу /etc/passwd шляхом вводу в нього нових облікових записів з ідентифікатором користувача, що дорівнює 0. Оскільки такі команди як who та w, працюють з реєстраційним іменем, яке зберігається в файлі /var/run/utmp, а не з ідентифікатором власника реєстраційного інтерпретатора, вони неможливі виявити хакера, який виглядає як рядовий користувач, хоча насправді зареєстрований в системі в якості суперкористувача. Щоб запобігти цьому, можна використовувати наступний міні-сценарій:

```
$ perl -F: -ane 'print if not $F[2];' /etc/passwd
```

Цей сценарій відображає будь-які записи файлу passwd, в яких ідентифікатор користувача не вказаний або рівний нулю. Сценарій можна адаптувати для пошука записів з підозрілими ідентифікаторами груп або ідентифікаторами користувачів, які співпадають з ідентифікаторами заданих співробітників (наприклад, керівників).

Використання утиліти sudo

Оскільки повноваження суперкористувача розділити не можна, принаймні довільним чином, складно надати комусь право виконувати конкретну операцію (наприклад, створення резервних копій), не надаючи можливість вільно працювати у системі. Якщо ж обліковий запис root доступний деякій групі адміністраторів, то постає питання, хто саме з цієї групи під обліковим записом суперкористувача виконав ті чи інші дії.

Щоб вирішити цю проблему використовується утиліта sudo. Ця утиліта в якості аргумента приймає командний рядок, який підлягає виконанню від імені суперкористувача (чи іншого вповноваженого користувача).

Утиліта звертається до файлу /etc/sudoers, де міститься список користувачів, які мають дозвіл на його виконання, і перелік команд, які вони можуть вводити на конкретному комп'ютері. Якщо команда, яка запитується, дозволена, утиліта sudo запрошує користувача ввести його власний пароль та виконує команду від імені суперкористувача. Далі sudo дозволяє, не вводячи пароль, виконувати інші команди, але тільки до тих пір, поки не вичерпається часовий ліміт (зазвичай 5 хвилин, але його величину можна змінювати).

Утиліта sudo веде журнал, де реєструються виконані команди,

комп'ютери, на яких вони виконувались, та користувачі, які їх викликали, а також каталоги, з яких запускались команди, та час їх виклику. Ця інформація може направлятись в систему Syslog або зберігатись в будь-якому журнальному файлі за бажанням користувача.

Для модифікації файла `/etc/sudoers` призначена спеціальна команда `visudo`, яка перевіряє, чи не редагується файл кимось стороннім, потім відкриває його в редакторі, а перед інсталляцією файлу виконує синтаксичний контроль. Останній етап особливо важливий, оскільки помилка у файлі `/etc/sudoers` може не дозволити повторно визвати утиліту `sudo` для виправлення файлу.

Утиліта `sudo` має недоліки. Найбільший з них полягає в тому, що будь-яка вразливість системи захисту того чи іншого привілейованого користувача еквівалентна порушенню безпеки самого облікового запису `root`. Щоб протистояти цьому, можна попередити тих, хто має право виконувати утиліту `sudo`, про необхідність максимально захищати свої облікові записи. Можна регулярно запускати утиліту `John the Ripper`, перевіряючи стійкість паролів привілейованих користувачів.

Інший недолік – можливість тимчасово вийти в інтерпретатор команд із дозволеної програми, або виконати команду `sudo sh` або `sudo su`, якщо вони дозволені.

Флаги Процес не може явно змінити жоден зі своїх ідентифікаторів, але є особлива ситуація, коли нові діючі ідентифікатори встановлюються непрямо. Існують два спеціальних біта, які встановлюються в масці прав доступу до файлу: `setuid` (Set User ID – біт зміни ідентифікатора користувача) та `setgid` (Set Group ID- біт зміни ідентифікатора групи). Якщо запускається виконуваний файл, в якого встановлений один із цих бітів, то діючими ідентифікаторами створюваного процесу стають ідентифікатори власника файлу, а не ідентифікатори користувача, який запустив програму. Зміна повноважень дійсна тільки на час роботи програми. Біт `setuid` дозволяє рядовим користувачам запускати програми, які виконують жорстко регламентовані адміністративні дії. Наприклад, команда `passwd`, за допомогою якої

Користувач змінює свій пароль, звертається до файлу `/etc/shadow` (чи `/etc/passwd`), які належать суперкористувачу, внаслідок чого в неї встановлений біт `setuid`. Вона модифікує файл строго визначеним чином та завершується. Щоб унеможливити несанкціоноване втручання в цей процес, перед виконанням команда `passwd` перевіряє, чи знає користувач свій поточний пароль.

Якщо біт `setgid` встановлений для каталога, то створювані в ньому файли будуть приймати ідентифікатор групи каталога, а не групи, в яку входить власник файлу. Це спрощує користувачам, які належать до однієї

групи, спільний доступ до каталогів. Слід також враховувати, що біт `setgid` можна встановлювати для файлів, які не є виконуваними. Таким чином запитується спеціальний режим блокування файлів при їх відкритті.

Встановити біт `suid` або `sgid` можна, вказавши за допомогою `chmod` права доступа в числовому виді:

```
chmod 4755 файл
```

або в мнемонічному виді:

```
chmod u+s файл
```

В деяких системах UNIX доступна лише одна з цих двох форм.

Біт установки ефективного групового ідентифікатора (`sgid`) при запуску файла на виконання діє схожим до SUID чином: як і у випадку біта SUID, встановлений SGID викликає присвоєння процесу, виконуваний код якого знаходиться в запущеному файлі, ефективного ідентифікатора ідентифікатора групи, який дорівнює ідентифікатору групи цього файла.

В виводі команди `ls` файли зі встановленими бітами SUID и SGID відрізняються тим, що в полі, де зазвичай стоїть "x" (біт виконання), розташований символ "s":

```
-r-xr-sr-x 1 roottty 10040 Nov 4 2002 /usr/sbin/wall
```

```
-r-sr-sr-x 1 rootsys 22168 Nov 4 2002 /usr/bin/passwd
```

Це значить, що присутні обидва біта – і біт запуску і біт `suid` (чи `sgid`, відповідно). Якщо спробувати встановити біт `suid` или `sgid` на файл, для якого у відповідному праві доступа (власника чи групи) не буде біта запуску, то система не дасть цього зробити. Оскільки для каталога біти `suid` та `sgid` мають інше значення, то біти `suid` / `sgid` та біти права пошуку в каталозі можуть бути встановлені окремо. В виводі `ls` в правах доступа до каталога при відсутності права пошука та наявності бітів `suid` / `sgid` літера S в виводі прав доступа буде заголовною:

```
dr-Sr-xr-x 2 root other 512 May 10 01:48 enum
```

```
-rw-r--r-- 1 root other 0 May 10 01:47 q
```

Зверніть увагу на права доступа до каталогу `enum`.

Знайти всі файли, в яких встановлений біт `suid`, можна за допомогою команди

```
find / -perm -u+s
```

а файли з встановленим бітом `sgid` – за допомогою команди

```
find / -perm -g+s
```

Щоб не допустити зламу системи, слід уважно відноситись до файлів, права доступа до яких дозволяють запускати їх від чужого імені. Поява таких

файлів в системі може полегшити спробу НСД. Програма passwd, наприклад, написана таким чином, що користувач, який її запускає, не зможе за її допомогою зробити нічого, крім зміни власного пароля.

Захист даних в ОС Windows

Розглянемо засоби захисту, які запропоновано в операційних системах Windows. Підхід до захисту в ОС цього типу базується на однакових принципах, наведемо відомості по «родоначальнику» цих операційних систем - Windows NT/2000.

Windows NT/2000 — це єдине сімейство операційних систем Microsoft, при розробці якого з самого початку було поставлено завдання створити операційну систему, що відповідає вимогам рівня захищеності C2 за «Критеріями оцінювання захищених комп'ютерних систем» (TCSEC) Міністерства оборони США, що відомі як «Оранжева книга».

Захищеність операційної системи по класу C2 вимагає відповідність наступним вимогам:

1. Обов'язкова ідентифікація й автентифікація всіх користувачів операційної системи.
2. Розмежувальний контроль доступу — надання користувачам можливості захисту даних, що їм належать.
3. Системний аудит — здатність системи вести докладний аудит усіх дій, що виконують користувачі й сама операційна система.
4. Захист об'єктів від повторного використання — здатність системи запобігти доступу користувача до ресурсів, з якими до цього працював інший користувач (наприклад, забезпечення неможливості повторного використання звільненої пам'яті або читання інформації з файлів, що були вилучені).

2 грудня 1999 року уряд США оголосив, що операційні системи Windows NT 4.0 Workstation і Windows NT 4.0 Server успішно пройшли сертифікацію по класу C2. Для Windows 2000 подібна процедура сертифікації відбулась в 2002 році, але вже не за «Критеріями оцінювання захищених комп'ютерних систем» (TCSEC) Міністерства оборони США, а за міжнародним стандартом ISO 15408, що введено в дію з 1999 року.

Розглянемо детальніше можливості засобів захисту операційної системи Windows NT. Система підтримує 4 типи суб'єктів та 13 типів об'єктів доступу.

Автентифікація та ідентифікація відбуваються за допомогою процесу WinLogon, що перевіряє істинність користувача за допомогою інших модулів підсистеми авторизації (LSA, MSV) та запускає в разі успіху процес UserInit.exe. Останній виконується з повноваженнями даного користувача і створює для останнього робоче середовище — підключає відповідний користувачеві ключ реєстру, настройки з user profile, та запускає програмну оболонку (explorer.exe). Архітектура підсистеми авторизації досить гнучка і

дозволяє використовувати будь-які способи перевірки істинності. Проте в стандартній конфігурації використовується лише проста парольна автентифікація. Образи паролів зберігаються в спеціальному розділі реєстру, при цьому використовуються два типи хеш-функцій: за алгоритмом MD4 (NT-hash) та менш стійка з використанням DES (LM-hash), остання для сумісності з клієнтами попередньої серверної ОС Microsoft — Lan Manager. Важливою особливістю є те, що авторизація користувача може відбуватися як локально, так і делегуватися контролерові домену NT.

За допомогою утиліти User Manager Windows NT забезпечує широкі можливості по керуванню обліковими записами користувачів. Так, для кожного користувача може бути задано ряд атрибутів, таких як належність до груп, місцезнаходження user profile, робочі години, повноваження на доступ по комутованих лініях і т.д. Крім того, може бути задана політика керування обліковими записами, що регламентує:

- 1) мінімальний та максимальний терміни життя паролю;
- 2) мінімальну довжину паролю;
- 3) унікальність паролю як вимога не належати до заданої кількості востаннє використаних;
- 4) кількість невдалих спроб автентифікації, після яких обліковий запис блокується, та відрізок часу, протягом якого вони мають відбутися;
- 5) тривалість блокування облікового запису.

Також ця утиліта дозволяє призначати користувачам привілеї (права на всю систему, а не на конкретний об'єкт, наприклад входити в систему локально або змінювати системний час) та задавати політику аудиту.

Windows NT реалізує дискреційну модель розмежування доступу. Керування доступом здійснюється в Windows NT за допомогою спеціального модулю, що носить назву reference monitor та реалізується викликом функції SeAccessCheck ядра ОС при будь-якій спробі суб'єкта отримати доступ. При цьому використовуються дві структури даних — маркер доступу суб'єкта, що є носієм його повноважень, та дескриптор захисту об'єкта, що містить ідентифікатори власника об'єкта та його первинної групи, список контролю доступу (ACL) та список аудиту (SACL). Матриця доступу в даній ОС, таким чином, зберігається у вигляді множини списків контролю доступу об'єктів. Останні, на відміну від ОС Unix, мають нефіксовану довжину і можуть містити довільну кількість елементів контролю доступу (Access Control Entry, ACE). Кожен з ACE містить ідентифікатор суб'єкта та список методів (прав), за якими йому дозволено або заборонено доступ до даного об'єкта. ACE, що забороняють доступ, мають більший пріоритет. У випадку відсутності ACE, що визначає потрібні права, у доступі буде відмовлено. Основними правами є R-читання, W-запис, X-виконання, D-видалення, P-зміна прав доступу до даного об'єкта, O-право стати власником об'єкту.

Задати права доступу до файлових та принтерних об'єктів можна за допомогою Windows NT Explorer. Для цього необхідно виділити об'єкт, за допомогою правої кнопки миші викликати меню, в якому вибрати пункт "Properties" (в російській локалізації "Свойства"), далі у діалоговому вікні вибрати вкладку "Security" ("Безопасность"). У вікні, що відкривається, для кожного суб'єкта доступу (користувача, псевдокористувача або групи) можна вибрати так звані "відображувані" права. Відображувані права фактично є попередньо сформованими наборами елементарних прав. Якщо є необхідність, можна керувати безпосередньо елементарними правами доступу, яких для деяких видів об'єктів є більше 20. Для цього слід натиснути кнопку "Advanced..." ("Дополнительно..."). У вікні, що відкривається, можна переглядати і змінювати власника об'єкта, керувати наслідуванням прав доступу до підкаталогів і файлів, викликати додаткове вікно для перегляду і заміни всіх прав доступу для кожного суб'єкта, а також задавати параметри аудита цього об'єкта (див. далі).

Реєстрація подій у Windows NT здійснюється шляхом виклику спеціальних функцій ядра ОС, що додають записи у файли *.evt директорії \winnt\system32\config. Усього є три журнали – системний, прикладного ПО та безпеки. Реєстрацію подій, таким чином, може здійснювати будь-який компонент робочого середовища, проте сама ОС забезпечує її шляхом виклику цих функцій з модулю reference monitor. Для цього використовується список аудиту, що носить дещо неправильну назву "системний список контролю доступу" (SACL) та визначає для кожного об'єкта, що саме буде реєструватися при спробах отримати доступ тим чи іншим суб'єктом. На додаток до цього, можна фільтрувати записи реєстрації шляхом визначення так званої "політики аудиту".

Перегляд журналів реєстрації за звичай здійснюється утилітою Event Viewer, що надає досить широкі можливості задання фільтрів відображення записів, зокрема за часом реєстрації, типом, категорією та джерелом події. У Windows 2000 адміністратори мають доступ до перегляду журналів реєстрації через Control Panel (в російській локалізації "Панель управления"), що доступна через стартове меню, а також через пункт меню "Administrative tasks" (в російській локалізації "Администрирование"). Списки аудиту об'єктів можна задати за допомогою Windows NT Explorer: "Properties" → "Security" → "Advanced..." → "Auditing" (в російській локалізації "Свойства" → "Безопасность" → "Дополнительно..." → "Параметры аудита"). Політику аудиту задають окремо – за допомогою User Manager в Windows NT 4.0 або "Administrative tasks" → "Local security policy" ("Администрирование" → "Локальная политика безопасности") в Windows 2000.