Лабораторна робота № 8

## *Тема*: Захист інформації в операційних системах.

*Мета*: Навчитися розробляти програми захисту інформації в операційних системах.

*Обладнання та програмне забезпечення:* ПК, операційна система Windows, система програмування С++Builder.

### Вказівки для самостійної підготовки

*Під час підготовки необхідно повторити теоретичний матеріал:*

1. Основні завдання забезпечення безпеки.
2. Принципи аутентифікації і керування доступом.
3. Аутентифікація і керування доступом у Windows.
4. Керування доступом.
5. Аудит.
6. Локальна безпека даних.
7. Мережна безпека даних.

###### Теоретичні відомості

1. Основні завдання забезпечення безпеки

Завдання забезпечення безпеки комп'ютерних систем: виконання процедур аутентифікації, авторизації та аудиту, дотримання конфіденційності, доступності та цілісності даних.

**Аутентифікація**

*Аутентифікація* - процес, за допомогою якого система засвідчує, що користувач є тим, за кого себе видає. Під час аутентифікації потрібна інформація, відома обом сторонам (наприклад, пароль). Користувач пред'являє коректне свідчення, дістає позитивну відповідь на вимогу аутентифікації.

**Авторизація**

Після успішної аутентифікації користувачу потрібен доступ до різних ресурсів. *Авторизація*, або *керування доступом* - процес, що переві-ряє, чи має право користувач отримати доступ до запитаних ресурсів.

Авторизацію здійснюють порівнянням інформації про користувача з інформацією про права доступу, пов'язаною із ресурсом. Вона містить тип дії та відомості про користувачів, яким дозволено цю дію виконувати. Якщо користувач має право на запитану дію з ресурсом, йому надають можливість її виконати.

**Аудит**

*Аудитом* - збирання інформації про події у системі, важливі для її безпеки, і збереження цієї інформації у формі, придатній для подальшого аналізу. Подіями можуть бути спроби аутентифікації у системі, спроби отримати доступ до об'єктів тощо. Інформацію зберігають у спеціальному системному журналі. Журнал має вигляд текстового файлу, або спеціальний формат.

**Конфіденційність, цілісність і доступність даних**

*Конфіденційність* - можливість приховання даних від стороннього доступу. Її забезпечують криптографічним захистом даних допомогою шифрування.

*Цілісність* — спроможність захистити дані від вилучення або зміни. Технології підтримки цілісності даних включають цифрові підписи і коди аутентифікації повідомлень.

*Доступність* - гарантія того, що користувач після аутентифікації зможе отримати доступ до запитаного ресурсу, якщо він має на це право. Порушення доступності даних називають *відмовою від обслуговування*, однієї із цілей політики безпеки є запобігання доведенню системи до такої відмови.

2.Принципи аутентифікації і керування доступом

2.1. Основи аутентифікації

**Аутентифікація** розрізняє легітимні та нелегітимні спроби доступу до системи. Дані, які вимагаються від користувачів, зводяться до знання секретної інформації, спільної для користувача і системи (наприклад, пароля).

*До альтернативних свідчень належать:*

* володіння деяким фізичним предметом (смарт-картою);
* біометричні параметри (відбитки пальців тощо).

Розрізняють локальну і мережну аутентифікацію. При **локальній** аутентифікації користувач доводить свою легітимність для використання ресурсів однієї комп'ютерної системи, **мережна** аутентифікація дозволяє користувачу користуватися всіма ресурсами мережі (дані користувача передають для перевірки на спеціальний сервер із будь-якого комп'ютера мережі).

Облікові записи

Для аутентифікації користувача у системі зберігається інформація про нього. Ця інформація називається *обліковим записом,* вона містить:

1. ім'я користувача для входу у систему;
2. ідентифікатор користувача, що є чисельним значенням, унікальним у межах комп'ютера або групи комп'ютерів;
3. інформація про пароль користувача;
4. інформація про обмеження на вхід користувача у систему (термін легітимності облікового запису, періодичність зміни пароля тощо);
5. інформація про групи, до яких належить цей користувач;
6. місце знаходження домашнього каталогу користувача;
7. налаштування сесії користувача (шлях до його командного інтерпретатора тощо).

Інформацію про облікові записи зберігають у *базі даних облікових записів*. Адміністратор системи може змінювати будь-яку інформацію в цій базі.

Групи користувачів

Для зручності адміністрування системи користувачі можуть об'єднуватись у групи. Користувач може одночасно належати до кількох груп. Під час авторизації доступу до об'єктів перевіряють не тільки права самого користувача, але й права груп, до яких він належить.

Інформацію про групи зберігають у базі даних облікових записів. ОС визначає кілька стандартних груп, які створюють під час її установки, зокрема, групу адміністраторів системи і групу звичайних користувачів із обмеженим доступом.

Одноразові паролі

Проблему пересилання пароля мережею розв'язується використанням паролів, дійсні лише один раз під час сесії користувача. Перехоплення такого *одноразового пароля* нічого не дає зловмисникові. Такі паролі у сучасних системах реалізуються за допомогою смарт-карт — електронних пристроїв, у які вбудований мікропроцесор із засобами генерації одноразових паролів.

2.2. Основи керування доступом

Розподіл прав відображають у *матриці доступу,* у якій рядки відповідають суб'єктам авторизації (користувачам, групам користувачів), стовпці - ресурсам (файлам, пристроям тощо).

Використовують два базові підходи для відображення матриці контролю доступу:

*1.списки контролю доступу* (Access Control Lists, ACL) - зберіга-ються стовпці матриці. Для ресурсу задано список суб'єктів, які можуть використовувати цей ресурс.

*2. можливості* (capabilities) - зберігаються рядки матриці. Для кожного суб'єкта задано список ресурсів, які йому дозволено використовувати.

Списки контролю доступу

Для кожного об'єкта задають список, що визначає, які користувачі можуть виконувати операції із цим об'єктом. Для файлових систем об'єктами є файли або каталоги. Елементи списку — це пари (користувач, набір дій), які називають *елементами контролю доступу* (access control elements, АСЕ).

 Проблеми при реалізації списків контролю доступу:

Розмір списку контролю доступу має бути не надто великим, система має ефективно ним керувати.

Права мають бути досить гнучкими, але не дуже складними.

Тому є особливості реалізації списків контролю доступу в різних ОС. Загальною концепцією є задання прав не лише для окремих користувачів, але й для груп користувачів, а також можливість визначити права доступу всіх користувачів системи. У деяких ОС обмежують кількість елементів у списку. У системах лінії Windows ХР можна задавати списки контролю доступу необмеженої кількості елементів (для окремих користувачів, груп, користувачів інших комп'ютерів тощо), і задавати різні комбінації прав.

3. Аутентифікація і керування доступом у Windows

3.1. Загальна архітектура безпеки

Архітектури безпеки Windows ХР містять такі компоненти:

*Процес реєстрації користувачів* (logon process, winlogon) обробляє запити користувачів на реєстрацію у системі та приймає дані від них.

*Менеджер аутентифікації* (Local Security Authority Subsystem, LSASS) проводить аутентифікацію користувача. Компонент також контролює політику аудиту.

*Менеджер облікових записів* (Security Accounts Manager, SAM) підтримує базу даних облікових записів, що містить імена локальних користувачів і груп, а також паролі. Під час перевірки прав користувача SAM взаємодіє із менеджером аутентифікації.

*Довідковий монітор захисту* (Security Reference Monitor, SRM) перевіряє права користувача на доступ до об'єкта і виконує необхідну дію з об'єктом за наявності цих прав.

3.2. Аутентифікація

У Windows кожному користувачу відповідає обліковий запис. Він пов'язаний із *профілем захисту,* який є набором інформації щодо контролю доступу (ім'я користувача, список його груп, пароль тощо). Профілі захисту зберігають у базі даних SAM і використовують для аутентифікації.

Послідовність кроків реєстрації користувача у системі. Процес winlogon очікує введення від користувача. Потік цього процесу виявляє спробу користувача ввійти у систему і пропонує йому ввести ім'я облікового запису та пароль. Для реалізації такого введення winlogon звертається до спеціальної DLL графічної ідентифікації та аутентифікації (GINA).

Дані від користувача передаються процесу winlogon, і аутентифікація із використанням бази даних SAM.

При успішній аутентифікації, створюється об'єкт, що унікально визначає цього користувача в усіх його подальших діях. Цей об'єкт називають *маркером доступу* (access token), він визначає, до яких системних ресурсів мають доступ потоки, створені цим користувачем.

Після успішної аутентифікації користувача LSASS створює процес і приєднує до нього маркер доступу користувача. Цей процес передають підсистемі Win32, що запускає в його адресному просторі застосування, визначене у реєстрі як оболонка системи (за замовчуванням ним є оболонка explorer.exe). Застосування формує візуальне відображення параметрів робочого столу для користувача.

Ідентифікатори безпеки

Із кожним користувачем і групою у Windows ХР пов'язують унікальний *ідентифікатор безпеки* (Security Identifier, SID). Це цілочислове значення. Система безпеки звертається до користувачів і груп тільки за їхнім SID.

Win32 АРІ надає ряд функцій для роботи із SID. Для отримання SID користувача або групи за іменем використовують функцію LookupAccountName():

*BOOL LookupAccountName(LPCTSTR sname, LPCTSTR username, PSID sid, LPDWORD sidsize, LPTSTR dname. LPDWORD dsize, PSID\_NAME\_USE sidtype);*

username — ім'я користувача;

sid — покажчик на буфер пам'яті, у який записується отриманий SID;

sidsize — розмір буфера;

dname - покажчик на буфер пам'яті, у який записується ім'я домена, де зареєстрований користувач, dsize - розмір цього буфера;

sidtype — покажчик на змінну, що задає тип SID (вона може набувати значень SidTypeUser - SID користувача, SidTypeGroup - SID групи).

Маркери доступу

Маркер доступу пов'язують з усіма процесами користувача. Маркер містить таку інформацію:

1. SID користувача, із правами якого виконується процес;
2. список груп, до яких належить цей користувач;
3. список привілеїв, якими володіє користувач;
4. список контролю доступу за замовчуванням, визначає первісні права доступу для об'єктів, створюваних процесами цього користувача.

При спробі процесу відкрити дескриптор об'єкта диспетчер об'єктів звертається до SRM. SRM отримує маркер доступу, пов'язаний із процесом, та використовує його SID і список груп, щоб визначити, чи має процес право доступу до об'єкта.

Для отримання маркера доступу процесу використовують функцію ОреnProcessToken():

*BOOL OpenProcessToken(HANDLE ph, DWORD access, PHANDLE ptoken);*

ph — дескриптор процесу;

access - тип доступу до маркера (TOKEN\_QUERY - читання інформації);

ptoken - змінна, в яку записується дескриптор маркера доступу.

Приклад отримання маркера доступу поточного процесу:

*HANDLE mytoken;*

*OpenProcessToken(GetCurrentProcess(), TOKEN\_QUERY, &mytoken);*

Інформація, що зберігається у маркері доступу, може бути отримана за допомогою функції GetTokenInformation():

*BOOL GetTokenInformation(HANDLE token, TOKEN\_INFORMATION\_CLASS tclass, LPVOID tibuf, DWORD tisize, PDWORD realsize);*

token — дескриптор маркера доступу;

tclass —визначає тип отримуваної інформації (TokenUser — інформація про обліковий запис користувача, TokenGroups — список груп);

tibuf — буфер для розміщення інформації;

tisize — розмір буфера;

realsize —змінна, що містить розмір буфера, необхідного для розміщення інформації .

Приклад отримання SID з маркера доступу:

*GetTokenInformation(mytoken, TokenUser, buf, sizeof(buf), &size);*

*ConvertSidToStringSid(((PTOKEN\_USER)buf)->User. Sid, &ssid);*

*printf("%s\n", ssid);*

4. Керування доступом

Під час створення об'єкта Windows ХР, який використовується більш як одним процесом, йому присвоюють *дескриптор захисту* (security descriptor).

До елементів дескриптора захисту належать:

1. SID власника об'єкта;
2. список контролю доступу (ACL), що визначає права доступу до об'єкта.

Кожний елемент списку контролю доступу (АСЕ) містить такі елементи:

1. тип АСЕ (виділяють дозволяючі і забороняючі АСЕ);
2. ідентифікатор безпеки (SID);
3. набір прав доступу (читання, запис, повний контроль тощо).

Сума прав доступу, наданих окремими АСЕ, формує загальний набір прав доступу, наданих ACL

5. Аудит

5.1. Загальні принципи організації аудиту

Підсистеми аудиту сучасних ОС містять такі компоненти:

1. Засоби визначення *політики аудиту*. Визначивши її, адміністратор системи задає перелік подій, які його цікавлять. Повідомлення поділяються на рівні залежно від важливості.
2. Засоби генерації повідомлень аудиту. Повідомлення генеруються різними компонентами режиму ядра і режиму користувача. Для цього ОС надає відповідну функціональність і системні виклики.

Засоби збереження інформації в журналі аудиту. Їх реалізують централізовано в рамках окремого комп'ютера або мережі, для чого запускають спеціальний фоновий процес, якому передають усю інформацію про повідомлення аудиту

6. Локальна безпека даних

6.1.Принципи шифрування даних на файлових системах

Механізми керування доступом до файлів не можуть запобігти несанкціонованому доступу до інформації у разі викрадення жорсткого диска. Зловмисник може підключити диск до комп'ютера із сумісною версією операційної системи, у якій він є привілейованим користувачем. Можна отримати доступ до всіх файлів на викраденому диску незалежно від того, які для них задані списки контролю доступу. Проблема характерна для переносних комп'ютерів (ноутбуків).

Необхідно організувати конфіденційне зберігання найціннішої інформації. Це реалізують за допомогою шифрування даних на файловій системі.

Шифрування здійснюють на рівні драйвера файлової системи, який перехоплює спроби доступу до файлу і шифрує та дешифрує його «на льоту» у разі, коли користувач надав необхідні дані (наприклад, ключ) для виконання цих операцій.

7. Мережна безпека даних

7.1. Шифрування каналів зв'язку

Види шифрування каналів зв'язку

Рівні шифрування каналів зв'язку в рамках стека протоколів TCP/IP:

1. На канальному рівні *шифрують канали зв'язку*. Для цього пакети автоматично шифрують під час передавання через незахищений фізичний канал і дешифрують після отримання. Ці дії виконує спеціалізоване апаратне забезпечення.
2. На мережному рівні виконують *наскрізне шифрування*, пакети мережного рівня шифрують на хості-джерелі, залишають зашифрованими впродовж усього маршруту і автоматично дешифрують хостом-одержувачем. Передавання таких пакетів мережею не змінюється, тому що їхні заголовки залишаються незашифрованими.
3. На межі між транспортним і прикладним рівнями можна шифрувати дані. Наприклад, застосування може бути модифіковане так, що у разі реалізації прикладного протоколу в ньому використовуватиметься не інтерфейс сокетів, а інтерфейс альтернативної бібліотеки, яка реалізує шифрування даних.

Приклади розробки програм

***Приклад 1*** *.* Програма створення облікового запису користувача

**#include <vcl.h>**

**#include <stdio.h>**

**#include <iostream.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**wchar\_t user\_password[PWLEN]; // пароль користувача**

**wchar\_t user\_comment[256]; // коментарі про користувача**

**USER\_INFO\_1 ui; // інформація про користувача**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input user name: "); // читаємо ім'я користувача**

**wscanf(L"%s", user\_name);**

**ui.usri1\_name = user\_name; // установлюємо ім'я користувача**

**printf("Input user password: ");**

**wscanf(L"%s", user\_password); // читаємо пароль користувача**

**ui.usri1\_password = user\_password; // установлюємо пароль користувача**

**ui.usri1\_priv = USER\_PRIV\_USER; // звичайний користувач**

**ui.usri1\_home\_dir = NULL; // домашнього каталогу немає**

**printf("Input user comment: ");**

**getwchar(); // очищаємо потік**

**\_getws(user\_comment); // читаємо коментарі про користувача**

**ui.usri1\_comment = user\_comment; // установлюємо коментарі**

**ui.usri1\_flags = UF\_SCRIPT; // виконувати скрипт при вході**

**//користувача в систему**

**ui.usri1\_script\_path = NULL; // поки файл зі скриптом не визначаємо**

**// додаємо користувача**

**ret\_status = NetUserAdd(**

**server\_name, // ім'я сервера**

**1, // рівень інформації**

**(LPBYTE)&ui, // адреса інформації про користувача**

**NULL); // індексування в структурі даних немає**

**// перевіряємо на успішне завершення**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net user add failed.\n");**

**printf("Error code: %u\n", ret\_status);**

**return ret\_status;**

**}**

**printf("The user is added.\n");**

**cin.get();**

**return 0;**

**}**

***Приклад 2*** *.*Програма отримання інформації про користувача

**#include <vcl.h>**

**#include <stdio.h>**

**#include <iostream.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаем мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**USER\_INFO\_1 \*ui; // інформація про користувача**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: "); // вводим имя сервера**

**// формируем имя сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input user name: ");**

**wscanf(L"%s", user\_name); // читаємо ім'я користувача**

**// отримуємо інформацію про користувача**

**ret\_status = NetUserGetInfo(**

**server\_name, // ім'я сервера**

**user\_name, // ім'я користувача**

**1, // рівень інформації 1**

**(LPBYTE\*)&ui); // адреса інформації проо користувача**

**// перевіряємо на успішне завершення**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net user get info failed.\n");**

**cin.get();**

**return ret\_status;**

**}**

**wprintf(L"User name: %s\n", ui->usri1\_name);**

**wprintf(L"User commemnt: %s\n", ui->usri1\_comment);**

**NetApiBufferFree(ui); // звільняємо буфер**

**cin.get();**

**return 0;**

**}**

***Приклад 3*** *.* Програма зміни пароля користувача

**#include <vcl.h>**

**#include <iostream.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#include <conio.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t domain\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**wchar\_t old\_password[PWLEN]; // старий пароль користувача**

**wchar\_t new\_password[PWLEN]; // новий пароль користувача**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server or domain name: ");**

**wscanf(L"%s", domain\_name); // читаємо ім'я домена**

**printf("Input user name: ");**

**wscanf(L"%s", user\_name); // читаємо ім'я користувача**

**printf("Input old password: ");**

**wscanf(L"%s", old\_password); // читаємо старий пароль користувача**

**printf("Input new password: ");**

**wscanf(L"%s", new\_password); // читаємо новий пароль користувача**

**// змінюємо пароль користувача**

**ret\_status = NetUserChangePassword(**

**domain\_name, // ім'я домена**

**user\_name, // ім'я користувача**

**old\_password, // старый пароль**

**new\_password); // новый пароль**

**// перевіряємо на успішне завершення**

**if (ret\_status != NERR\_Success)**

**{**

**cout<<("Net user change password failed.\n");**

**cout<<("Net API Status: %d\n", ret\_status);**

**getch();**

**return ret\_status;**

**}**

**cout<<("The password was changed.\n");**

**getch();**

**return 0;**

**}**

***Приклад 4*** *.* Програма видалення облікового запису користувача

**#include <vcl.h>**

**#include <stdio.h>**

**#include <iostream.h>**

**#include <windows.h>**

**#include <lm.h>**

**#include <conio.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**wscanf(L"%s", server\_name); // читаємо ім'я домена**

**printf("Input user name: ");**

**wscanf(L"%s", user\_name); // читаємо ім'я користувача**

**// видаляємо користувача**

**ret\_status = NetUserDel(**

**server\_name, // ім'я домена**

**user\_name); // ім'я користувача**

**// перевіряємо на успішне завершення**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net user del failed.\n");**

**printf("Net API Status: %d\n", ret\_status);**

**getch() ;**

**return ret\_status;**

**}**

**printf("The user is deleted.\n");**

**getch() ;**

**return 0;**

**}**

***Приклад 5*** *.* Програма створення локальної групи користувачів

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я локальної групи**

**wchar\_t comment[MAXCOMMENTSZ]; // коментар про групу**

**LOCALGROUP\_INFO\_1 group\_info; // інформація про групу**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a name for the local group: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я групи**

**group\_info.lgrpi1\_name = group\_name; // установлюємо адресу імені в**

**//структуру**

**printf("Input group comments: ");**

**getwchar(); // очищаємо потік**

**\_getws(comment); // читаємо комментар про локальну групу**

**group\_info.lgrpi1\_comment = comment; // установлюємо комментар**

**// створюємо локальну групу**

**ret\_status = NetLocalGroupAdd(**

**server\_name, // ім'я сервера**

**1, // рівень вхідних даних**

**(LPBYTE)&group\_info, // ім'я групи та комментарі**

**NULL); // індексацію даних не використовуємо**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net local group add failed.\n");**

**return ret\_status;**

**}**

**printf("The group is created.\n");**

**return 0;**

**}**

***Приклад 6*** *.* Програма отримання інформації про локальну групу

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я локальної групи**

**LOCALGROUP\_INFO\_1 \*buf\_ptr = NULL; // адрес буфера для данных**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формируем имя сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a name for the local group: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я групи**

**// отримуємо інформацію про локальну групу**

**ret\_status = NetLocalGroupGetInfo(**

**server\_name, // ім'я сервера**

**group\_name, // ім'я локальної групи**

**1, // отримуємо ім'я групи і комментар**

**(LPBYTE\*)&buf\_ptr); // покажчик на адресу буфера для даних**

**// проверяем завершение функции**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net local group get info failed.\n");**

**NetApiBufferFree(buf\_ptr); // освобождаем буфер для данных**

**return ret\_status;**

**}**

**// выводим на консоль имя локальной группы и комментарий**

**wprintf(L"Local group name: %s\n", buf\_ptr->lgrpi1\_name);**

**wprintf(L"Comment: %s\n", buf\_ptr->lgrpi1\_comment);**

**NetApiBufferFree(buf\_ptr); // звільняємо буфер для даних**

**return 0;**

**}**

***Приклад 7*** *.* Програма додавання членів локальної групи

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я локальної групи**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**LOCALGROUP\_MEMBERS\_INFO\_3 member\_info; // інформація про члена групи**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a local group name: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я групи**

**printf("Input a domain name: ");**

**wscanf(L"%s", user\_name); // вводимо ім'я домена**

**wcscat(user\_name, L"\\"); // приєднуємо символ '\'**

**printf("Input a user name: ");**

**// вводимо ім'я користувача**

**wscanf(L"%s", user\_name + wcslen(user\_name));**

**// установлюємо інформацію про користувача**

**member\_info.lgrmi3\_domainandname = user\_name;**

**// додаємо користувача у локальну групу**

**ret\_status = NetLocalGroupAddMembers(**

**server\_name, // ім'я сервера**

**group\_name, // ім'я групи**

**3, // рівень інформації**

**(LPBYTE)&member\_info, // ім'я облікового запису**

**1); // додаємо одного члена групи**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Ret status: %d\n", ret\_status);**

**printf("Net local group add members failed.\n");**

**return ret\_status;**

**}**

**printf("The member is added.\n");**

**return 0;**

**}**

***Приклад 8*** *.* Програма установки членів локальної групи

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я локальної групи**

**LOCALGROUP\_MEMBERS\_INFO\_3 \*member\_info; // інформація про членів групи**

**DWORD n; // кількість залишених членів локальної групи**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a local group name: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я групи**

**printf("Input number of members: ");**

**scanf("%d", &n);**

**// захоплюємо пам’ять для інформації про користувачів**

**member\_info = new LOCALGROUP\_MEMBERS\_INFO\_3[n];**

**for (DWORD i = 0; i < n; ++i)**

**{**

**// захоплюємо пам’ять під ім'я користувача**

**member\_info[i].lgrmi3\_domainandname = new wchar\_t[256];**

**printf("Input a domain name: ");**

**// вводимо ім'я домена**

**wscanf(L"%s", member\_info[i].lgrmi3\_domainandname);**

**// приєднуємо символ '\'**

**wcscat(member\_info[i].lgrmi3\_domainandname, L"\\");**

**printf("Input a user name: ");**

**// вводимо ім'я користувача**

**wscanf(L"%s", member\_info[i].lgrmi3\_domainandname +**

**wcslen(member\_info[i].lgrmi3\_domainandname));**

**}**

**// вводимо ім'я домена**

**for (int i = 0; i < n; ++i)**

**wprintf(L"%s\n", member\_info[i].lgrmi3\_domainandname);**

**// установлюємо членів в локальній групі**

**ret\_status = NetLocalGroupSetMembers(**

**server\_name, // ім'я сервера**

**group\_name, // ім'я локальної групи**

**3, // рівень інформації**

**(LPBYTE)member\_info, // інформація про SID**

**n); // кількість членів групи, що залишилася**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net local group set members failed.\n");**

**return ret\_status;**

**}**

**printf("The members are set.\n");**

**return 0;**

**}**

***Приклад 9*** *.* Програма видалення члена локальної групи

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я локальної групи**

**wchar\_t user\_name[UNLEN]; // ім'я користувача**

**LOCALGROUP\_MEMBERS\_INFO\_3 member\_info; // інформація про члена групи**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a local group name: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я групи**

**printf("Input a domain name: ");**

**wscanf(L"%s", user\_name); // вводимо ім'я домена**

**wcscat(user\_name, L"\\"); // приєднуємо символ '\'**

**printf("Input a user name: ");**

**// вводимо ім'я користувача**

**wscanf(L"%s", user\_name + wcslen(user\_name));**

**// установлюємо інформацію про користувача**

**member\_info.lgrmi3\_domainandname = user\_name;**

**// видаляємо користувача з локальної групи**

**ret\_status = NetLocalGroupDelMembers(**

**server\_name, // ім'я сервера**

**group\_name, // ім'я групи**

**3, // рівень інформації**

**(LPBYTE)&member\_info, // інформація про SID**

**1); // додаємо одного члена групи**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net local group del members.\n");**

**return ret\_status;**

**}**

**printf("The member is deleted.\n");**

**return 0;**

**}**

***Приклад 10*** *.* Програма видалення локальної групи

**#include <vcl.h>**

**#include <stdio.h>**

**#include <windows.h>**

**#include <lm.h>**

**#pragma comment( lib, "netapi32.lib" ) // підключаємо мережну бібліотеку**

**int main()**

**{**

**wchar\_t server\_name[256] = L"\\\\"; // ім'я сервера**

**wchar\_t group\_name[GNLEN]; // ім'я видаленої локальної групи**

**NET\_API\_STATUS ret\_status; // код повернення з функції**

**printf("Input server name: ");**

**// формуємо ім'я сервера**

**wscanf(L"%s", server\_name + wcslen(server\_name));**

**printf("Input a name of the local group: ");**

**wscanf(L"%s", group\_name); // вводимо ім'я группы**

**// видаляємо локальну групу**

**ret\_status = NetLocalGroupDel(server\_name, group\_name);**

**// перевіряємо код завершення**

**if (ret\_status != NERR\_Success)**

**{**

**printf("Net local group del failed.\n");**

**printf("Error code: %d\n", ret\_status);**

**return ret\_status;**

**}**

**printf("The local group is deleted.\n");**

**return 0;**

**}**

***Завдання для студентів***

* 1. Розробити програми захисту інформації засобами ОС. Для програмної реалізації задач використовувати середовище програмування CodeGear C++ Builder. Програми розробляти у консольному режимі.
  2. Скласти звіт про виконання лабораторної роботи.

Зміст звіту:

* опис функцій для роботи з файловою системою;
* постановка задачі;
* програмний код.

1. При захисті лабораторної роботи підготувати відповіді на контрольні питання.

***Контрольні питання***

1. Охарактеризувати основні завдання забезпечення безпеки.
2. Охарактеризувати поняття: аутентифікація, авторизація, аудит.
3. Охарактеризувати поняття: конфіденційність, цілісність і доступність даних.
4. Охарактеризувати принципи аутентифікації і керування доступом.
5. Охарактеризувати облікові записи.
6. Охарактеризувати групи користувачів.
7. Охарактеризувати одноразові паролі.
8. Визначити списки контролю доступу.
9. Охарактеризувати локальну безпеку даних.
10. Охарактеризуватипринципи роботи EFS.
11. Охарактеризуватипрограмний інтерфейс EFS.
12. Визначити списки мережну безпека даних.

###### ЛІТЕРАТУРА

1. Бондаренко М.Ф., Качко О.Г. Операційні системи. - К.: СМІТ, 2008. - 432с.
2. Бэкон Дж., Харрис Т. Операционные системы. – К.: Издат. группа BHV; СПб.: Питер, 2004. – 800 с.
3. Гордеев А.В., Молчанов А.Ю. Системное программное обеспечение. – СПб.: Питер, 2001. – 736 с.
4. Танненбаум Э. Операционные системы. – СПб.: Питер, 2002. – 1040 с.
5. Таненбаум Э. Современные операционные системы. – СПб.: Питер, 2007. – 1040 с.
6. Шеховцов В.А. Операційні системи. - К.: Видавнича група BHV. 2005. – 576 с.