1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 3**

1. «Контроль и управлением доступом в операционных системах»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 4851003/20001 Павлов И. А.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Климшин И.И.

<*подпись*>

1. **Цель работы**

Освоение средств контроля и управления доступом пользователей к ресурсам операционной системы, приобретение навыков распределения прав на примере файловой системы NTFS в среде Windows

1. **Описание задачи**
2. Изучить настройку атрибутов безопасности с помощью стандартных средств Windows.
3. Создать утилиту для управления атрибутами безопасности.
4. **Ход работы.**

Для проверки настройки атрибутов безопасности стандартными инструментами, предоставляемыми операционной системой Windows, в первую очередь создадим каталог test\_folder. Отключим наследование в дополнительных параметрах безопасности для данной папки. Изображение выглядит как текст

Автоматически созданное описание

Рис. 1

Проверим, какими правами обладает текущий пользователь:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2

Как мы можем заметить, текущий пользователь обладает доступом только чтению и смене разрешений, так как он владелец файла. Выдадим всем пользователям права на чтение и запись.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3

Теперь проверим изменение прав у текущего пользователя. Изображение выглядит как текст

Автоматически созданное описание

Рис. 4

Права группы “Пользователи” распространились на текущего пользователя.

Для проверки приоритета запрещающих прав над разрешающими разрешим пользователю полный доступ, а группе “Пользователи” запретим права на запись. Изображение выглядит как текст

Автоматически созданное описание

Рис. 5

Действительно запрещающие права имеют больший приоритет, чем разрешающие права.

Для демонстрации суммирования разрешающих прав создадим две разрешающие ACE с различными правами. Изображение выглядит как текст

Автоматически созданное описание

Рис. 6

Права просуммировались.

Также для управленя правами доступа была разработана специальная утилита, которая позволяет создавать, читать, изменять и удалять ACE. Проверим ее работу на файле text.txt. Для начала прочитаем его атрибуты. Изображение выглядит как текст

Автоматически созданное описание

Рис. 7

Как мы можем видеть, программа говорит, что атрибутов у файла нет. Изображение выглядит как текст

Автоматически созданное описание

Рис. 8

Их действительно нет. Создадим атрибут, который даст пользователю права на чтение файла. Изображение выглядит как текст

Автоматически созданное описание

Рис. 9

Попробуем удалить файл. Наша попытка не увенчалась успехом (Рис. 10). Изображение выглядит как текст

Автоматически созданное описание

Рис. 10

Изменим атрибут на запрет полного доступа. Теперь файл нельзя открыть (Рис. 12).

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11

Изображение выглядит как текст

Автоматически созданное описание

Рис. 12

Удалим атрибут. Атрибут действительно удалился. Изображение выглядит как текст

Автоматически созданное описание

Рис. 13

Изображение выглядит как текст

Автоматически созданное описание

Рис. 14

1. **Ответы на контрольные вопросы.**
2. Что такое множество действующих разрешений?

Это множество действий, которые разрешены данному пользователю (группе) по отношению к файлам, каталогам. Вычисляется исходя из ACL.

1. Почему проверка прав доступа к файлу осуществляется только при открытии файла, а не при обращениях к нему?

В компьютере достаточно много процессов, которые оперируют какими-то объектами. Каждому объекту присвоен свой ACL, в который входит большое количество записей ACE. Если бы проверка прав доступа к файлу осуществлялась при каждом обращении к файлу, то это бы скорее всего привело бы к снижению производительности и быстродействия.

1. Можно ли запретить администратору системы доступ к какому-либо файлу? Может ли он обойти это ограничение?

Администратору системы можно запретить доступ к какому-нибудь файлу, но он может просто поменять владельца данного объекта и отменить все запреты, восстановив полный доступ к файлу.

1. Как изменить владельца объекта в среде Windows?

Для того, чтобы изменить владельца файла нужно открыть окно дополнительных параметров безопасности файла и перейти на вкладку владелец.

1. Имеет ли владелец какие-либо права к файлу, если существует ACE, запрещающая полный доступ к этому файлу на имя владельца?

Владелец объекта - обычно пользователь, который его создал обладает правом избирательного управления доступом к объекту, всегда может изменять АСL объекта, чтобы разрешить или запретить другим пользователям доступ к объекту.

1. **Вывод.**

После выполнения лабораторной работы можно сделать вывод о том, что Windows обладает системой настройки атрибутов безопасности, с помощью которой можно гибко настраивать права на файл или каталог для каждого пользователя. Это может быть актуально в системе с большим количеством пользователей.

**ПРИЛОЖЕНИЕ 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <AclAPI.h>

PSID PrintUsers() {

char\* name = "C:\\Users\\Ilya\\Projects of oib\\Laba3\\g.txt";

PSID GetSid = NULL;

PACL Dacl = NULL;

PSECURITY\_DESCRIPTOR Sec = NULL;

PSID sids[20];

LPVOID getace;

PACCESS\_ALLOWED\_ACE ace;

LPSTR uname = NULL;

LPSTR uname2 = NULL;

DWORD size1 = 0;

DWORD size2 = 0;

SID\_NAME\_USE puse;

int user = 0;

if (GetNamedSecurityInfoA(name, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, &GetSid, &Dacl, NULL, &Sec) == ERROR\_SUCCESS) {

int countUsers = Dacl->AceCount;

printf("Введите пользователя:\n");

for (int i = 0; i < countUsers; i++) {

GetAce(Dacl, i, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

PSID gSid = (PSID)&ace->SidStart;

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse) || uname == NULL) {

int err = GetLastError();

if (err == 122) {

uname = (TCHAR\*)malloc(size1 \* sizeof(TCHAR));

uname2 = (TCHAR\*)malloc(size2 \* sizeof(TCHAR));

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse))

{

exit(1);

}

err = 0;

}

}

printf("Пользователь %d: %s\n", i + 1, uname);

}

free(uname);

free(uname2);

scanf("%d", &user);

GetAce(Dacl, user - 1, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

PSID gSid = (PSID)&ace->SidStart;

return gSid;

}

return NULL;

}

DWORD getMask(DWORD\* aMask)

{

int sss = 0;

snova:

printf("Выберите права:\n");

printf("1 - Полный доступ\n");

printf("2 - Удаление\n");

printf("3 - Чтение файла\n");

scanf("%d", &sss);

switch (sss)

{

case(1):

{

\*aMask |= FILE\_ALL\_ACCESS;

break;

}

case(2):

{

\*aMask |= DELETE;

break;

}

case(3):

{

\*aMask |= FILE\_READ\_DATA;

break;

}

default:

printf("Не существует такой команды\n");

goto snova;

}

return \*aMask;

}

void ShowAttrs(ACCESS\_ALLOWED\_ACE\* ace)

{

printf("\n");

if ((ace->Mask & FILE\_ALL\_ACCESS) == FILE\_ALL\_ACCESS)

printf("Полный доступ\n");

if ((ace->Mask & DELETE) == DELETE)

printf("Удаление файла\n");

if ((ace->Mask & WRITE\_DAC) == WRITE\_DAC)

printf("Изменение атрибутов\n");

if ((ace->Mask & FILE\_READ\_DATA) == FILE\_READ\_DATA)

printf("Чтение файла\n");

if ((ace->Mask & FILE\_EXECUTE) == FILE\_EXECUTE)

printf("Запуск файл\n");

return;

}

int ReadACE(char\* fileName) {

PSID GetSid = NULL;

PACL Dacl = NULL;

PSECURITY\_DESCRIPTOR Sec = NULL;

LPVOID getace;

PACCESS\_ALLOWED\_ACE ace;

LPSTR uname = NULL;

LPSTR uname2 = NULL;

DWORD size1 = 0;

DWORD size2 = 0;

SID\_NAME\_USE puse;

int Nace = 0;

if (GetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, &GetSid, &Dacl, NULL, &Sec) == ERROR\_SUCCESS) {

int countAces = Dacl->AceCount;

if (countAces == 0) {

printf("Аттрибутов нет\n");

return 0;

}

else {

for (int i = 0; i < countAces; i++) {

GetAce(Dacl, i, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

PSID gSid = (PSID)&ace->SidStart;

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse) || uname == NULL) {

int err = GetLastError();

if (err == 122) {

uname = (TCHAR\*)malloc(size1 \* sizeof(TCHAR));

uname2 = (TCHAR\*)malloc(size2 \* sizeof(TCHAR));

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse))

{

exit(1);

}

err = 0;

}

}

printf("ACE %d: %s - ", i + 1, uname);

if (ace->Header.AceType == ACCESS\_ALLOWED\_ACE\_TYPE)

{

printf("Разрешить\n");

}

else

{

printf("Запретить\n");

}

}

printf("Выберите ACE\n");

scanf("%d", &Nace);

GetAce(Dacl, Nace - 1, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

ShowAttrs(ace);

free(uname);

free(uname2);

}

}

}

int CreateACE(char\* fileName) {

PSID GetSid = NULL;

PACL Dacl = NULL;

PSECURITY\_DESCRIPTOR Sec = NULL;

int pr = 0;

if (GetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, &GetSid, &Dacl, NULL, &Sec) == ERROR\_SUCCESS) {

int countAces = Dacl->AceCount;

PSID gSid = PrintUsers();

printf("Разрешить или запретить(1/0)\n");

scanf("%d", &pr);

DWORD aMask = 0;

EXPLICIT\_ACCESS addAce;

ZeroMemory(&addAce, sizeof(EXPLICIT\_ACCESS));

addAce.grfAccessPermissions = getMask(&aMask);

if (pr == 1) {

addAce.grfAccessMode = GRANT\_ACCESS;

}

else {

addAce.grfAccessMode = DENY\_ACCESS;

}

addAce.grfInheritance = NO\_INHERITANCE;

addAce.Trustee.TrusteeForm = TRUSTEE\_IS\_SID;

addAce.Trustee.ptstrName = (LPCH)gSid;

PACL NDacl = NULL;

if (SetEntriesInAclA(1, &addAce, Dacl, &NDacl) != ERROR\_SUCCESS) {

int err = GetLastError();

printf("Ошибка\n");

}

else

{

if (SetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, NULL, NDacl, NULL) != ERROR\_SUCCESS) {

int err = GetLastError();

printf("Ошибка");

}

}

if (NDacl != NULL)

LocalFree((HLOCAL)NDacl);

if (Sec != NULL)

LocalFree((HLOCAL)Sec);

}

else

{

return 1;

}

printf("Успешно создан\n\n");

return 0;

}

int ChangeACE(char\* fileName) {

PSID GetSid = NULL;

PACL Dacl = NULL;

PSECURITY\_DESCRIPTOR Sec = NULL;

LPVOID getace;

LPSTR uname = NULL;

LPSTR uname2 = NULL;

DWORD size1 = 0;

DWORD size2 = 0;

SID\_NAME\_USE puse;

PACCESS\_ALLOWED\_ACE ace;

int nace = 0;

if (GetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, &GetSid, &Dacl, NULL, &Sec) == ERROR\_SUCCESS) {

int countAces = Dacl->AceCount;

if (countAces == 0) {

printf("Аттрибутов нет\n");

return 0;

}

else {

for (int i = 0; i < countAces; i++) {

GetAce(Dacl, i, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

PSID gSid = (PSID)&ace->SidStart;

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse) || uname == NULL) {

int err = GetLastError();

if (err == 122) {

uname = (TCHAR\*)malloc(size1 \* sizeof(TCHAR));

uname2 = (TCHAR\*)malloc(size2 \* sizeof(TCHAR));

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse))

{

exit(1);

}

err = 0;

}

}

printf("ACE %d: %s - ", i + 1, uname);

if (ace->Header.AceType == ACCESS\_ALLOWED\_ACE\_TYPE)

{

printf("Разрешить\n");

}

else

{

printf("Запретить\n");

}

}

printf("Выберите ACE\n");

scanf("%d", &nace);

GetAce(Dacl, nace - 1, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

DWORD newMask = 0;

printf("Разрешить или запретить(1/0)\n");

int pr;

scanf("%d", &pr);

if (pr == 1) {

ace->Header.AceType = ACCESS\_ALLOWED\_ACE\_TYPE;

}

else {

ace->Header.AceType = ACCESS\_DENIED\_ACE\_TYPE;

ace->Header.AceFlags = 0;

}

ace->Mask = getMask(&newMask);

if (SetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, NULL, Dacl, NULL) == ERROR\_SUCCESS)

{

printf("Успешно изменено!");

}

}

}

return 0;

}

int DeleteACE(char\* fileName) {

PSID GetSid = NULL;

PACL Dacl = NULL;

PSECURITY\_DESCRIPTOR Sec = NULL;

LPVOID getace;

LPSTR uname = NULL;

LPSTR uname2 = NULL;

DWORD size1 = 0;

DWORD size2 = 0;

SID\_NAME\_USE puse;

PACCESS\_ALLOWED\_ACE ace;

int nace = 0;

if (GetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, &GetSid, &Dacl, NULL, &Sec) == ERROR\_SUCCESS) {

int countAces = Dacl->AceCount;

if (countAces == 0) {

printf("Аттрибутов нет");

return 0;

}

else {

for (int i = 0; i < countAces; i++) {

GetAce(Dacl, i, &getace);

ace = (PACCESS\_ALLOWED\_ACE)getace;

PSID gSid = (PSID)&ace->SidStart;

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse) || uname == NULL) {

int err = GetLastError();

if (err == 122) {

uname = (TCHAR\*)malloc(size1 \* sizeof(TCHAR));

uname2 = (TCHAR\*)malloc(size2 \* sizeof(TCHAR));

if (!LookupAccountSidA(NULL, gSid, uname, &size1, uname2, &size2, &puse))

{

exit(1);

}

err = 0;

}

}

printf("ACE %d: %s - ", i + 1, uname);

if (ace->Header.AceType == ACCESS\_ALLOWED\_ACE\_TYPE)

{

printf("Разрешить\n");

}

else

{

printf("Запретить\n");

}

}

printf("Введите ACE\n");

scanf("%d", &nace);

if (DeleteAce(Dacl, nace - 1) != 0) {

if (SetNamedSecurityInfoA(fileName, SE\_FILE\_OBJECT, DACL\_SECURITY\_INFORMATION, NULL, NULL, Dacl, NULL) == ERROR\_SUCCESS) {

printf("Успешно удален\n");

}

return 0;

}

else {

printf("ACE не удален");

return 0;

}

}

}

return 0;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int comand = 0;

char fileName[MAX\_PACKAGE\_NAME] = { 0 };

printf("Введите команду:\n1 - Создать ACE\n2 - Прочитать ACE\n3 - Перезаписать ACE\n4 - Удалить ACE\n");

scanf("%d", &comand);

switch (comand) {

case 1:

printf("Введите имя файла:\n");

scanf("%s", fileName);

CreateACE(fileName);

break;

case 2:

printf("Введите имя файла:\n");

scanf("%s", fileName);

ReadACE(fileName);

break;

case 3:

printf("Введите имя файла:\n");

scanf("%s", fileName);

ChangeACE(fileName);

break;

case 4:

printf("Введите имя файла:\n");

scanf("%s", fileName);

DeleteACE(fileName);

break;

}

return 0;

}