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

**ЛАБОРАТОРНАЯ РАБОТА № 12**

1. **«Система контроля доступа»**
2. по дисциплине «Основы информационной безопасности Структуры данных»
3. Выполнил
4. студент гр. 4851003/20001 Федорова А.А.

<*подпись*>

1. Преподаватель

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**

Исследовать атрибуты безопасности и их влияние на доступ к файлу с использованием интерфейса, представленного в OC Windows

Разработать программу, выполняющую создание, редактирование и просмотр атрибутов безопасности для файла.

1. **Ход работы.**

В файловой системе NTFS был создан каталог «test\_folder».

Для созданного каталога были изменены доп.параметры безопасности:

* Снят флаг наследования.
* Произведена установка текущего пользователя как владельца каталога.

Далее было рассмотрено влияние групповых прав на всех членов группы. Для этого группе «Пользователи» были разрешены чтение и запись данных, после чего были зафиксированы разрешения для пользователя «vboxuser».

Помимо этого было доказано, что запрещающие права приоритетнее, чем разрешающие. Для этого группе «Пользователи» была запрещена запись, а пользователю «vboxuser» был дан полный доступ, после чего были зафиксированы разрешения для этого пользователя.

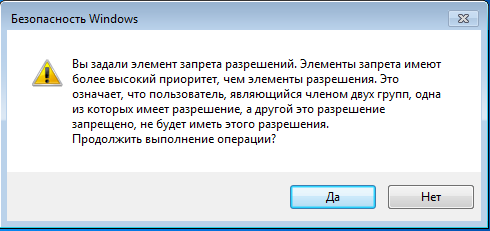


Рисунок 1 – Демонстрация приоритетности запрещающих прав над разрешающими.

Последним проверяемым фактом было суммирование разрешающих прав для двух и более разрешающих ACE. Для этого были созданы две отдельные ACE для группы «Пользователи», первая разрешала удаление, а вторая чтение разрешений, после чего были зафиксированы разрешения для «Пользователи» (Рисунок 2).

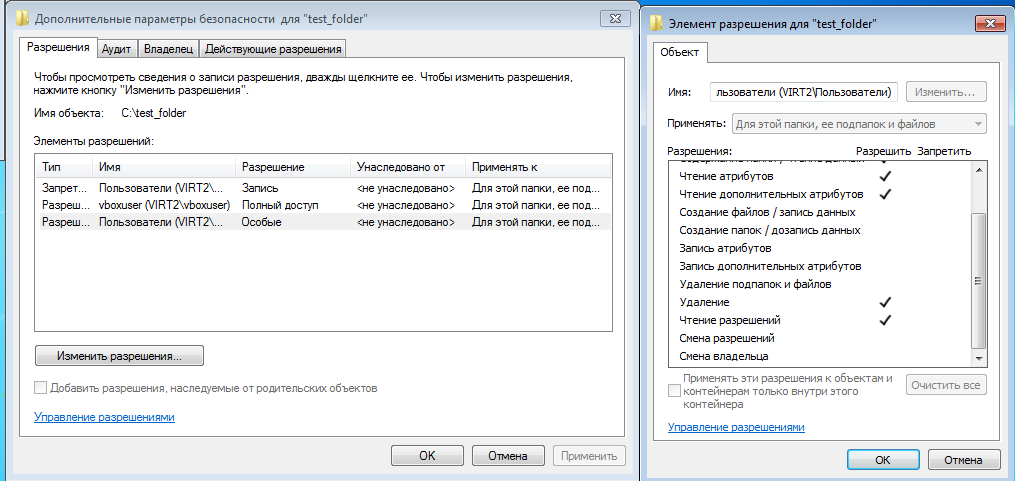


Рисунок 2 – Суммирование прав, заданных разными ACE.

С помощью примеров с MSDN:

https://learn.microsoft.com/ru-ru/windows/win32/secauthz/finding-the-owner-of-a-file-object-in-c--

<https://learn.microsoft.com/ru-ru/windows/win32/secauthz/modifying-the-acls-of-an-object-in-c--?redirectedfrom=MSDN>

была реализована программа, осуществляющая создание, редактирование и просмотр атрибутов безопасности для файла (Приложение 1). В ходе проверки её работы было выяснено, что при создании ACE с её помощью все правила суммирования разрешений, рассмотренные раннее, остаются справедливыми.

С помощью утилиты подтверждена на практике возможность доступа к файлу при наличии соответствующих разрешений и невозможность доступа при их отсутствии.

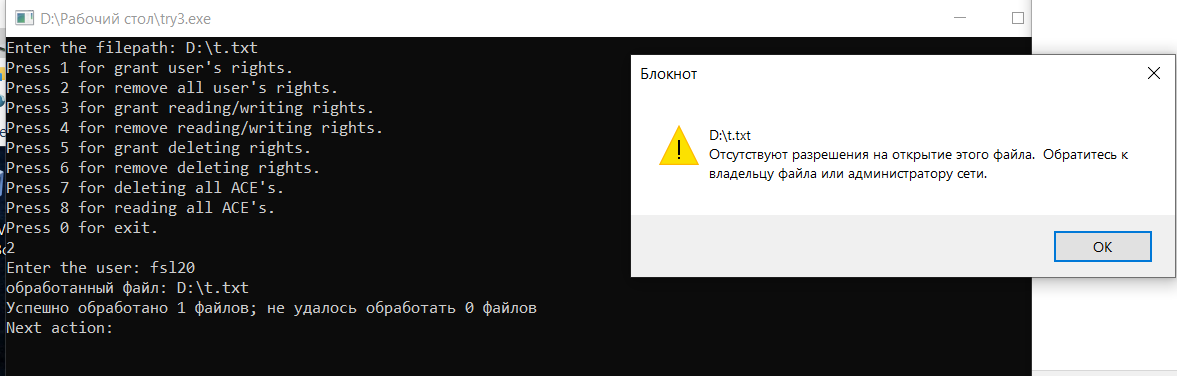


Рисунок 3 – Демонстрация невозможности открыть файл

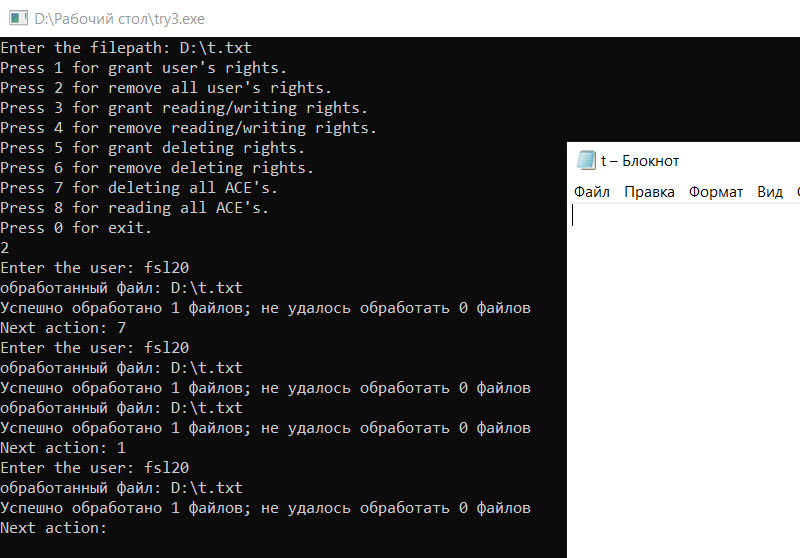


Рисунок 4 – Доступ к файлу разрешен.

1. **Выводы**

В ходе выполнения работы было изучено, как устроена система контроля доступа ОС Windows и особенности этой системы. Эти навыки могут быть полезны, когда системой пользуется сразу несколько человек, и необходимо, чтобы доступом к файлу обладали не все пользователи.

1. **Контрольные вопросы.**
2. Что такое множество действующих разрешений?
3. Множество допустимых прав определяет права пользователя над объектом, какие
4. действия он может над ним выполнять и вычисляется следующим образом:
5. 1) Из всех разрешающих записей (ACE) для данного пользователя составить
6. разрешённое множество прав (может быть пустым);
7. 2) Из всех запрещающих записей (АСЕ) для данного пользователя составить
8. запрещённое множество прав (может быть пустым);
9. 3) Результирующее множество прав определяется путём вычитания множества
10. запрещённых прав из множества разрешённых прав

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.

**ПРИЛОЖЕНИЕ 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define TEXT(quote) \_\_TEXT(quote)

#include <conio.h>

#include <windows.h>

#include <stdio.h>

#include <aclapi.h>

#include <accctrl.h>

LPSTR getName(ACCESS\_ALLOWED\_ACE\* pAce) //based on function from MSDN

{

DWORD dwRtnCode = 0;

BOOL bRtnBool = TRUE;

LPSTR AcctName = NULL;

LPSTR DomainName = NULL;

DWORD dwAcctName = 1, dwDomainName = 1;

SID\_NAME\_USE eUse = SidTypeUnknown;

PSECURITY\_DESCRIPTOR pSD = NULL;

PSID pSid = &pAce->SidStart;

// First call to LookupAccountSid to get the buffer sizes.

bRtnBool = LookupAccountSidA(

NULL, // local computer

pSid,

AcctName,

(LPDWORD)&dwAcctName,

DomainName,

(LPDWORD)&dwDomainName,

&eUse);

// Reallocate memory for the buffers.

AcctName = (LPSTR)GlobalAlloc(

GMEM\_FIXED,

dwAcctName);

// Check GetLastError for GlobalAlloc error condition.

if (AcctName == NULL) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

printf(("GlobalAlloc error = %u\n"), dwErrorCode);

return NULL;

}

DomainName = (LPSTR)GlobalAlloc(

GMEM\_FIXED,

dwDomainName);

// Check GetLastError for GlobalAlloc error condition.

if (DomainName == NULL) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

printf(("GlobalAlloc error = %u\n"), dwErrorCode);

return NULL;

}

// Second call to LookupAccountSid to get the account name.

bRtnBool = LookupAccountSidA(

NULL, // name of local or remote computer

pSid, // security identifier

AcctName, // account name buffer

(LPDWORD)&dwAcctName, // size of account name buffer

DomainName, // domain name

(LPDWORD)&dwDomainName, // size of domain name buffer

&eUse); // SID type

// Check GetLastError for LookupAccountSid error condition.

if (bRtnBool == FALSE) {

DWORD dwErrorCode = 0;

dwErrorCode = GetLastError();

if (dwErrorCode == ERROR\_NONE\_MAPPED)

printf(

("Account owner not found for specified SID.\n"));

else

printf(("Error in LookupAccountSid.\n"));

return NULL;

}

return AcctName;

}

DWORD ReadAceToObjectsSecurityDescriptor(LPSTR pszObjName) {

DWORD dwRes = 0;

SE\_OBJECT\_TYPE ObjectType = SE\_FILE\_OBJECT;

PACL pDacl = NULL;

PSECURITY\_DESCRIPTOR pSD = NULL;

dwRes = GetNamedSecurityInfoA(pszObjName, ObjectType,

DACL\_SECURITY\_INFORMATION,

NULL, NULL, &pDacl, NULL, &pSD);

LPVOID getace;

PACCESS\_ALLOWED\_ACE pAce;

//system("cls");

for (int i = 0; i < (pDacl->AceCount); i++) {

if (!GetAce(pDacl, i, &getace)) {

printf("ERROR in GetAce function\n");

}

pAce = (PACCESS\_ALLOWED\_ACE)getace;

printf("%i. ", i + 1);

LPSTR name = getName(pAce);

printf("%s ", name);

if (pAce->Header.AceType == ACCESS\_ALLOWED\_ACE\_TYPE) printf("is able to\n");

else printf("is not able to\n");

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

printf("- modify the control (security) information\n");

if ((pAce->Mask & WRITE\_OWNER) == WRITE\_OWNER)

printf("- modify the owner SID of the object\n");

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

printf("- delete file\n");

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

printf("- read the information maintained by the object\n");

if ((pAce->Mask & FILE\_WRITE\_DATA) == FILE\_WRITE\_DATA)

printf("- write the information maintained by the object\n");

if ((pAce->Mask & FILE\_GENERIC\_EXECUTE) == FILE\_GENERIC\_EXECUTE)

printf("- execute or alternatively look into the object\n");

printf("\n");

}

return dwRes;

}

DWORD AddAceToObjectsSecurityDescriptor(

LPSTR pszObjName, // name of object

LPSTR pszTrustee, // trustee for new ACE

DWORD dwAccessRights, // access mask for new ACE

ACCESS\_MODE AccessMode // type of ACE

)

{

DWORD dwRes = 0;

PACL pOldDACL = NULL, pNewDACL = NULL;

PSECURITY\_DESCRIPTOR pSD = NULL;

EXPLICIT\_ACCESS\_A ea;

SE\_OBJECT\_TYPE ObjectType = SE\_FILE\_OBJECT; // type of object

TRUSTEE\_FORM TrusteeForm = TRUSTEE\_IS\_NAME; // format of trustee structure

DWORD dwInheritance = NO\_INHERITANCE; // inheritance flags for new ACE

if (NULL == pszObjName)

return ERROR\_INVALID\_PARAMETER;

dwRes = GetNamedSecurityInfoA(pszObjName, ObjectType,

DACL\_SECURITY\_INFORMATION,

NULL, NULL, &pOldDACL, NULL, &pSD);

if (ERROR\_SUCCESS != dwRes) {

printf("GetNamedSecurityInfo Error %u\n", dwRes);

goto Cleanup;

}

ZeroMemory(&ea, sizeof(EXPLICIT\_ACCESS\_A));

ea.grfAccessPermissions = dwAccessRights;

ea.grfAccessMode = AccessMode;

ea.grfInheritance = dwInheritance;

ea.Trustee.TrusteeForm = TrusteeForm;

ea.Trustee.ptstrName = pszTrustee;

dwRes = SetEntriesInAclA(1, &ea, pOldDACL, &pNewDACL);

if (ERROR\_SUCCESS != dwRes) {

printf("SetEntriesInAcl Error %u\n", dwRes);

goto Cleanup;

}

// Attach the new ACL as the object's DACL.

dwRes = SetNamedSecurityInfoA(pszObjName, ObjectType,

DACL\_SECURITY\_INFORMATION,

NULL, NULL, pNewDACL, NULL);

if (ERROR\_SUCCESS != dwRes) {

printf("SetNamedSecurityInfo Error %u\n", dwRes);

goto Cleanup;

}

Cleanup:

if (pSD != NULL)

LocalFree((HLOCAL)pSD);

if (pNewDACL != NULL)

LocalFree((HLOCAL)pNewDACL);

return dwRes;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int cmd = -1;

char fpath[30] = { 0 };

char user[30] = { 0 };

//LPTSTR user =(LPTSTR) (L"fsl20");

printf("Press 1 for grant user's rights.\n");

printf("Press 2 for remove all user's rights.\n");

printf("Press 3 for grant reading/writing rights.\n");

printf("Press 4 for remove reading/writing rights.\n");

printf("Press 5 for grant deleting rights.\n");

printf("Press 6 for remove deleting rights.\n");

printf("Press 7 for deleting all ACE's.\n");

printf("Press 8 for reading all ACE's.\n");

printf("Press 0 for exit.\n");

while (cmd) {

scanf("%d", &cmd);

if (1 <= cmd && cmd <= 7) {

printf("Введите имя файла:\n");

scanf\_s("%s", fpath);

printf("Введите имя пользователя:\n");

scanf\_s("%s", user);

}

if (cmd == 8) {

printf("Введите имя файла:\n");

scanf\_s("%s", fpath);

}

switch (cmd)

{

case 0: break;

case 1:

AddAceToObjectsSecurityDescriptor(fpath, (user), GENERIC\_ALL, SET\_ACCESS);

break;

case 2:

AddAceToObjectsSecurityDescriptor(fpath, (user), GENERIC\_ALL, DENY\_ACCESS);

break;

case 3:

AddAceToObjectsSecurityDescriptor(fpath, user, FILE\_READ\_DATA | FILE\_WRITE\_DATA, SET\_ACCESS);

break;

case 4:

AddAceToObjectsSecurityDescriptor(fpath, (user), FILE\_READ\_DATA | FILE\_WRITE\_DATA, DENY\_ACCESS);

break;

case 5:

AddAceToObjectsSecurityDescriptor(fpath, (user), DELETE, SET\_ACCESS);

break;

case 6:

AddAceToObjectsSecurityDescriptor(fpath, (user), DELETE, DENY\_ACCESS);

break;

case 7:

AddAceToObjectsSecurityDescriptor(fpath, (user), NULL, SET\_ACCESS);

break;

case 8: ReadAceToObjectsSecurityDescriptor(fpath);

break;

default: printf("Incorect command:\n");

break;

}

if (cmd) printf("Next action:\n");

} return 0;

}