ЛЕКЦИЯ

Десрипторы безопасности

Дескрипторы безопасности в Windows.

Дескрипторы безопасности используются в Windows для защиты и аудита ресурсов. Дескриптор безопасности содержит владельца, основную группу, дискреционный список контроля доступа и системный список контроля доступа.

Владелец и основная группа.

Поля владельца и основной группы содержат идентификаторы безопасности. Владелец — это принципал безопасности, владеющий объектом. Владелец ресурса располагает полным доступом к объекту, включая возможность добавления и удаления разрешений доступа в дескрипторе безопасности.

Основная группа содержится в дескрипторе безопасности лишь для обеспечения совместимости с подсистемой POSIX. Система Windows не использует эту часть дескриптора безопасности, если не применяются утилиты, которые оперируют сPOSIX. По умолчанию принципал безопасности, создавший объект, записывает в дескриптор безопасности свою основную группу. Основной группой Windows по умолчанию является группа Domain Users.

Основная группа подразумевает членство в группе. При входе пользователя операционная система вставляет SID этой группы в маркер пользователя. Атрибут memberOf не перечисляет основную группу, а лишь включает явно назначенное членство в группах.

Дискреционные и системные списки контроля доступа.

Списки контроля доступа ACL состоят из двух частей. Первая часть списка контроля доступа представляет именованные контрольные флаги. Эти параметры контролируют применение разрешений в списке ACL и правил наследования. Вторая часть списка контроля доступа представляет собственно сам список. Этот список контроля доступа содержит одну или несколько записей управления доступом АСЕ. Флаги управления доступом определяют, каким образом Windows применяет записи управления доступом внутри списка ACL. Изначально Windows использует защищенные и автоматические флаги. Защищенные флаги запрещают модификацию списка контроля доступа путем наследования. Этот флаг является эквивалентом флажка Allowinheritable permissions from parent to propagate to this object (Разрешение наследуемых разрешений доступа). Флаг автоматически разрешает записям управления доступом в списках ACL наследовать разрешения доступа от родительских объектов дочерним.

Записи управления доступом.

Списки управления доступом содержат одну или несколько записей контроля доступа. В Windows записи управления доступом разбиты на два типа: Allow (Разрешить) и Deny (Запретить). Каждый тип АСЕ располагает объектом подтипа и необъектными подтипами. Записи управления доступом Allow и Deny назначают уровень доступа, обеспечиваемый подсистемой авторизации на основе права, запрашиваемого принципалом безопасности. Записи управления доступом к объектам являются исключающими для объектов в AD DS, поскольку они обеспечивают дополнительные поля для наследования объектов. Для большинства остальных ресурсов, как, например, ресурсов файловой системы и реестра, Windows использует необъектные записи управления доступом. Необъектные записи АСЕ обеспечивают наследование контейнеров, то есть объект в контейнере наследует запись контроля доступом контейнера. Этот принцип аналогичен наследованию разрешений доступа файлами от родительских папок. Каждый тип записи управления доступом располагает полем Rights и полем Trustee. Поля с правами обычно заполняются предварительно определенными числами, представляющими действия, которые может выполнять принципал безопасности. Рассмотрим пример с пользователем, запрашивающим чтение или запись файла. В этом случае чтение и запись являются двумя отдельными правами доступа. Поле доверия Trustee представляет идентификатор безопасности, разрешающий или запрещающий указанное право. В качестве примера можно привести пользователя или группу, которой разрешено либо запрещено выполнять действие, указанное в поле Right.

*Маркеры доступа.*

Связующим звеном между SID-идентификатором принципала безопасности и списком ACL является маркер доступа. КогдаWindows выполняет проверку подлинности пользователя с помощью Kerberos, пользователю в процессе входа на локальном компьютере присваивается маркер доступа. Этот маркер включает основной SID пользователя, SID-идентификаторы всех групп, которым принадлежит пользователь, а также привилегии и права пользователя.

**Примечание.** Маркер доступа также может включать в атрибуте SIDHistory дополнительные SID-идентификаторы. ЭтиSID-идентификаторы могут заполняться при перемещении учетных записей пользователей из одного домена в другой.Маркер доступа используется подсистемой безопасности каждый раз при попытке пользователя получить доступ к ресурсу. Когда пользователь пытается получить доступ к локальному ресурсу, этот маркер предоставляется клиентской рабочей станцией всем потокам и приложениям, которые запрашивают данные безопасности перед разрешением доступа к ресурсу. Этот маркер доступа никогда не передается по сети на другие компьютеры. Вместо этого на каждом сервере, где пользователь пытается получить доступ к ресурсу, создается локальный маркер доступа. Например, когда пользователь пытается получить доступ к почтовому ящику на сервере, то на этом сервере создается маркер доступа. В данном случае подсистема безопасности на сервере будет сравнивать SID-идентификаторы в маркере доступа с разрешениями, предоставленными в ACL-списке почтового ящика. Если предоставленные для SID разрешения позволяют доступ, пользователь сможет открыть почтовый ящик.

*Проверка подлинности.*

Для работы процессов подсистемы безопасности, включая использование SID и ACL, нужно обеспечить способ получения пользователями доступа к сети. По сути, пользователи должны иметь возможность указывать свои данные для извлечения маркера доступа из контроллера домена. Этот процесс называется *проверкой подлинности*.

Проверка подлинности выполняется в исходном клиентском входе на компьютер, являющийся членом домена AD DS. Шаги проверки подлинности зависят от операционной системы, с помощью которой клиент входит в сеть.

В случае успешной проверки подлинности пользователю предоставляется доступ в сеть. Если пользователь вошел в домен и все необходимые ему ресурсы находятся в одном лесе, пользователю только один раз будет предложено пройти проверку подлинности. Пока пользователь остается в системе, все разрешения, получаемые им в сети, основаны на начальной проверке подлинности. Хотя учетная запись пользователя проходит проверку подлинности каждый раз при получении пользователем доступа к ресурсам на сервере, где пользователь не проходил проверку подлинности, эта аутентификация прозрачна для пользователя.

Получение привилегий из маркера доступа

#include <windows.h>

#include <stdio.h>

int main ()

{

HANDLE hProcessToken; // дескриптор маркера доступа

PTOKEN\_PRIVILEGES lpTokenPrivileges = NULL;

LPTSTR lpPrivName = NULL;

LPTSTR lpPrivDisplayName = NULL;

// адрес имени привилегии

// адрес имени привилегии для

// отображения

DWORD dwLangld;

DWORD dwLength;

DWORD dwRetCode;

// код возврата

// дескриптор процесса

// чтение информации из маркера доступа

// дескриптор маркера доступа

if (!OpenProcessToken(

GetCurrentProcess(),

TOKEN\_QUERY,

ShProcessToken ))

{

dwRetCode = GetLastError();

perrorC'Set security descriptor owner failed.\n")

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// получаем длину буфера для привилегий

if (!GetTokenlnformation(

hProcessToken,

TokenPrivileges,

lpTokenPrivileges,

0,

&dwLength))

// дескриптор маркера доступа

// получаем привилегии

// адрес буфера

// длина буфера

// требуемая длина

dwRetCode = GetLastError();

if (dwRetCode != ERROR\_INSUFFICIENT\_BUFFER)

{

Perror(“Get token information for length failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// захватываем память для имени привилегии

IpTokenPrivi leges = (TOKEN\_PRIVTLEGES \*) new char [ dwLength ] ;

}

// получаем привилегии маркера доступа

if (!GetTokenInformation(

hProcessToken, // дескриптор маркера доступа

TokenPrivileges, // получаем привилегии

IpTokenPrivileges, // адрес буфера

dwLength, // длина буфера

&dwLength)) // требуемая длина

{

dwRetCode = GetLastError();

perror('Get token information failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// распечатываем привилегии

printf( "User privileges: \n" );

for( unsigned i = 0; i < lpTokenPrivileges->PrivilegeCount; ++i)

{

// печатаем состояние привилегии

if ((IpTokenPrivileges->Privileges[i].Attributes &

SE\_PRIVILEGE\_ENABLED) == SE\_PRIVILEGE\_ENABLED)

printf("SE\_PRIVILEGE\_ENABLED:\n");

if ((IpTokenPrivileges->Privileges[i].Attributes &

SE\_PRIVILEGE\_ENABLED\_BY\_DEFAULT) ==

SE\_PRIVILEGE\_ENABLED\_BY\_DEFAULT)

print f ( "SE\_PRIVILEGE\_ENABLED\_BY\_DEFAULT: \n") ;

if ((IpTokenPrivileges->Privileges[i].Attributes &

SE\_PRIVILEGE\_USED\_FOR\_ACCESS) == SE\_PRIVILEGE\_USED\_FOR\_ACCESS)

printf("SE\_PRIVILEGE\_USED\_FOR\_ACCESS:\n");

if (!IpTokenPrivileges->Privileges[i].Attributes)

printf("The privilege is disabled:\n");

// определяем длину имени привилегии

dwLength = 0;

if (!LookupPrivilegeName(

NULL, // локальная машина

&(IpTokenPrivileges->Privileges[i].Luid), // адрес LUID

lpPrivName, // адрес имени привилегии

&dwLength)) // адрес длины буфера

{

dwRetCode = GetLastError();

if (dwRetCode != ERROR\_INSUFFICIENT\_BUFFER)

{

perror("Lookup privilege name for length failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// захватываем память для имени привилегии

lpPrivName = new char [dwLength + 1] ;

}

// определяем имя привилегии

if (!LookupPrivilegeName(

NULL, // локальная машина

&(IpTokenPrivileges->Privileges[i].Luid), // адрес LUID

lpPrivName, // адрес имени привилегии

&dwLength)) // адрес длины буфера

{

dwRetCode = GetLastError();

perror("Lookup privilege name failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

// определяем длину имени привилегии для отображения

dwLength = 0;

if (ILookupPrivilegeDisplayName (

NULL, // локальная машина

lpPrivName, // имя привилегии

lpPrivDisplayName, // адрес для имени привилегии

&dwLength, // адрес длины буфера

SdwLangld)) // адрес идентификатора языка

{

dwRetCode = GetLastError();

if (dwRetCode != ERROR\_INSUFFICIENT\_BUFFER)

{

perror("Lookup privilege display name for length failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// захватываем память для имени привилегии

lpPrivDisplayName = new char[dwLength + 1];

}

// определяем имя привилегии для отображения

if (ILookupPrivilegeDisplayName (

NULL, // локальная машина

lpPrivName, // имя привилегии

lpPrivDisplayName, // адрес для имени привилегии

&dwLength, // адрес длины буфера

SdwLangld)) // адрес идентификатора языка

{

dwRetCode = GetLastError();

perror("Lookup privilege display name failed.\n");

printf("The last error code: %u\n"/ dwRetCode);

return dwRetCode;

}

// распечатываем имя привилегии и идентификатор языка

printf("\t%s\n", lpPrivDisplayName);

delete[] lpPrivName;

delete[] lpPrivDisplayName;

lpPrivName = NULL;

lpPrivDisplayName = NULL;

}

CloseHandle(hProcessToken);

return 0;

}

Создание дескриптора безопасности

#include <windows.h>

#include <stdio.h>

#include <lm.h>

int main()

{

char chDirName[248] ; // имя каталога

SECURITY\_DESCRIPTOR sd; // дескриптор безопасности каталога

SECURITY\_ATTRIBUTES sa; // атрибуты защиты каталога

DWORD dwErrCode; // код возврата

// инициализируем версию дескриптора безопасности

if (!InitializeSecurityDescriptor(

&sd,

SECURITY\_DESCRIPTOR\_REVISION))

{

dwErrCode = GetLastError();

printf("Initialize security descroptor failed.\n");

printf("Error code: %d\n"/ dwErrCode);

return dwErrCode;

}

// устанавливаем SID владельца объекта

if (!SetSecurityDescriptorOwner(

&sd, // адрес дескриптора безопасности

NULL, //не задаем владельца

SE\_OWNER\_DEFAULTED)) // определить владельца по умолчанию

{

dwErrCode = GetLastError();

perror("Set security descriptor owner failed.\n");

printf("The last error code: %u\n"/ dwErrCode);

return dwErrCode;

}

// устанавливаем SID первичной группы владельца

if (!SetSecurityDescriptorGroup(

&sd, // адрес дескриптора безопасности

NULL, //не задаем первичную группу

SE\_GROUP\_DEFAULTED)) // определить первичную группу по умолчанию

{

dwErrCode = GetLastError();

perror("Set security descriptor group failed.\n");

printf("The last error code: %u\n"/ dwErrCode);

return dwErrCode;

}

// проверяем структуру дескриптора безопасности

if (!IsValidSecurityDescriptor(&sd))

dwErrCode = GetLastError();

perror("Security descriptor is invalid.\n");

printf("The last error code: %u\n"/ dwErrCode);

return dwErrCode;

}

// инициализируем атрибуты безопасности

sa.nLength = sizeof(sa); // устанавливаем длину атрибутов защиты

sa.lpSecurityDescriptor = &sd; // устанавливаем адрес SD

sa.blnheritHandle = FALSE; // дескриптор каталога ненаследуемый

printf("Input a directory name: ");

scanf("%s", chDirName); // вводим имя каталога

// создаем каталог

if (!CreateDirectory(chDirName, &sa))

{

dwErrCode = GetLastError();

perror("Security descriptor is invalid.\n");

printf("The last error code: %u\n", dwErrCode);

return dwErrCode;

}

printf("The directory is created.\n");

return 0;

}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++