**ЛЕКЦИЯ. СИСТЕМА БЕЗОПАСНОСТИ ОС**

**Требования к безопасности**

Требования к безопасности компьютерных систем определяются на основе международных стандартов по оценке защищенности. В настоящее время основным таким стандартом является ISO/IEC 15408 Common *Criteria* for *Information Technology Security Evaluation* (*Общие критерии* оценки безопасности информационных технологий), сокращенно Common *Criteria* (Общие критерии)1. В этом стандарте определены семь уровней безопасности – от EAL1 (минимальная *оценка безопасности*) до EAL7, причем требования к высшим уровням EAL5–EAL7 устанавливаются каждой страной индивидуально. Большинство современных операционных систем (в том числе семейство клиентских и серверных систем Microsoft: от *Windows* 2000 до *Windows* 7 и *Windows Server* 2008) сертифицировано на уровень EAL4+ (плюс означает Flaw Remediation – исправление ошибок, постоянный выпуск обновлений)2.

Основными требованиями к безопасности являются следующие.

**1. Обязательная идентификация и аутентификация.**

До выполнения любых действий *пользователь* должен представиться системе (*идентификация*) и подтвердить, что он является тем, кем представился (*аутентификация*). Обычно реализуется посредством ввода уникального имени пользователя и пароля.

В *Windows* за идентификацию и *аутентификацю* пользователей отвечают процессы Winlogon.exe и Lsass.exe.

**2. Управляемый доступ к объектам.**

*Пользователь*-владелец объекта должен иметь возможность предоставлять *доступ* к объекту определенным пользователям и/или группам пользователей.

Безопасный *доступ* реализуется в *Windows* компонентом *Security* Reference Monitor (*SRM*, *монитор* контроля безопасности) исполнительной системы Ntoskrnl.exe.

**3. Аудит.**

Система должна уметь отслеживать и записывать все события, связанные с доступом к объектам.

В *Windows аудит* поддерживается *SRM* и Lsass.exe.

**4. Защита при повторном использовании объектов.**

Если область памяти выделялась какому-либо пользователю, а затем была освобождена, то при последующем выделении этой области все данные в ней (даже зашифрованные) должны быть стерты.

В *Windows* освобожденная *память* очищается системным потоком обнуления страниц, работающим во время простоя системы (с нулевым приоритетом).

Далее в лекции будет рассмотрена организация управляемого доступа к объектам в *SRM*, а также *права* и привилегии пользователей.

**Организация управляемого доступа к объектам**

**Принцип организации доступа**

Принцип организации управляемого безопасного доступа к объектам выглядит следующим образом. У каждого пользователя в системе имеется свой маркер доступа (access token), в котором указан уникальный идентификатор пользователя. Процессы, создаваемые пользователем, наследуют его маркер.

С другой стороны, все объекты в системе имеют структуру данных, которая называется дескриптор защиты (security descriptor). В эту структуру

входит список идентификаторов пользователей, которые могут (или не могут) получить доступ к объекту, а также вид доступа (только чтение, чтение и запись, полный доступ и т.д.).

При попытке доступа процесса к объекту идентификатор из маркера доступа процесса сравнивается с идентификаторами, содержащимися в дескрипторе защиты объекта, и на основании результатов сравнения доступ разрешается или запрещается.

Рассмотрим структуры данных и функции, отвечающие за реализацию этого принципа в ядре Windows.

**Идентификаторы защиты**

Для однозначного определения пользователя в системе используются идентификаторы защиты (SID – Security Identifier). Кроме пользователей, SID имеется у групп пользователей, компьютеров, доменов и членов доменов.

SID генерируется системой случайным образом так, что вероятность совпадения SID у разных пользователей близка к нулю.

В WRK структура SID описывается в файле public\sdk\inc\ntseapi.h.

SID состоит из следующих частей:

- номер версии – поле Revision (1 байт);

- код агента идентификатора (identifier authority) – поле IdentifierAuthority (6 байт);

- коды субагентов (subauthority values) – поле SubAuthority (от 1 до 15 кодов по 4 байта каждый).

Количество кодов субагентов хранится в поле SubAuthorityCount.

Последний код субагента называется относительным идентификатором (relative identifier, RID), поскольку все учетные записи пользователей на компьютере могут иметь одинаковые коды, кроме RID. RID, который равен 500, обозначает локального администратора.

Существует множество предопределенных SID (см., например, статью

Microsoft – http://support.microsoft.com/kb/243330).

**Маркер доступа**

Идентификаторы безопасности пользователей хранятся в маркерах доступа (access token). Во время входа пользователя в систему процесс Lsass.exe создает для него маркер доступа, который назначается первому пользовательскому процессу UserInit.exe, остальные процессы, запущенные пользователем, наследуют этот маркер. Маркер доступа процесса хранится в поле Token структуры EPROCESS (см. лекцию "Процессы и потоки").

Маркер доступа представлен структурой TOKEN, описанной в файле base\ntos\se\tokenp.h и имеющей следующие основные поля:

- TokenId – идентификатор маркера;

- UserAndGroups – SID учетной записи пользователя и групп, в которые данная учетная запись входит.

При проверке возможности доступа пользователя к определенному ресурсу, например, файлу на диске NTFS, система проверяет, входит ли SID учетной записи в список доступа файла.

- Privileges – список привилегий.

- DefaultDacl – список управления избирательным доступом по умолчанию (DACL, Discretionary Access-Control List). При создании процессом объектов, из маркера доступа процесса извлекается данное поле и помещается в атрибуты безопасности вновь созданного

объекта.

**Дескриптор защиты**

Объекты, к которым могут получать доступ процессы, имеют специальный атрибут – дескриптор защиты (security descriptor), содержащий информацию обо всех пользователях, которым разрешен или запрещен доступ к объекту.

Структура данных SECURITY\_DESCRIPTOR, представляющая дескриптор защиты, включает следующие основные поля:

- Owner – SID владельца;

- Dacl – список управления избирательным доступом;

- Sacl – системный список управления доступом.

Списки управления доступом (ACL, Access-Control List) в системе представлены заголовком (ACL Header) и последовательностью элементов списка (ACE, Access-Control Entry) .

Заголовок списка описывается структурой ACL (файл public\sdk\inc\ntseapi.h), в которой хранятся количество элементов списка ACL (поле AceCount) и общий размер списка без заголовка (поле AclSize).

Элементы ACE имеют заголовок (ACE Header), описываемый структурой ACE\_HEADER , маску доступа и идентификатор

безопасности SID. В заголовке указывается тип ACE (поле AceType);

Основными

являются ACCESS\_ALLOWED\_ACE\_TYPE (доступ разрешен) и ACCESS\_DENIED\_ACE\_TYPE (доступ запрещен).

Маска доступа (Access Mask) описывает разнообразные виды доступа к объектам.

В маске выделяются стандартные права

доступа (Standard Access Rights), применимые к большинству объектов, и специфичные для объектов права доступа (Object-Specific Access Rights).

Стандартными являются следующие права доступа:

- DELETE – право на удаление объекта;

- READ\_CONTROL – право на просмотр информации о дескрипторе защиты объекта;

- SYNCHRONIZE – право на использование объекта для синхронизации;

- WRITE\_DAC – право на изменение списка DACL;

- WRITE\_OWNER – право на смену владельца объекта.

Списки управления доступом бывают двух видов: DACL и SACL. Список управления избирательным доступом (DACL, Discretionary Access-Control List) определяет пользователей, которые могут получать доступ к объекту, а также указывает тип доступа.

В системном списке управления доступом (SACL, System Access-control List) перечислены пользователи и операции, которые должны учитываться в журнале аудита безопасности (security audit log).

За проверку возможности доступа процесса к объекту отвечает функция SeAccessCheck (файл base\ntos\se\accessck.c). На вход функции поступают следующие параметры:

- дескриптор защиты объекта (SecurityDescriptor);

- маркер доступа процесса (элемент структуры SubjectSecurityContext);

- маска запрашиваемого доступа (DesiredAccess);

- маска ранее предоставленного доступа (PreviouslyGrantedAccess);

- режим работы процессора (AccessMode);

Функция возвращает TRUE, если процессу возможно предоставить доступ к объекту, а также маску предоставленного доступа (GrantedAccess).

Если доступ запрещен, функция возвращает FALSE.

**Права и привилегии**

Кроме операций с объектами система должна контролировать множество других действий пользователей, например, вход в систему, включение/выключение компьютера, изменение системного времени, *загрузка* драйверов и т.д.

Для управления такими действиями, не связанными с доступом к конкретным объектам, система использует два механизма – *права* учетных записей и привилегии.

Право учетной записи (*account right*) – разрешение или запрет на определенный вид входа в систему.

Различают следующие виды входа:

- интерактивный (локальный) вход;

- вход из сети;

- вход через службу удаленных рабочих столов (ранее называлось – "через службу терминалов");

- вход в качестве службы;

- вход в качестве пакетного задания.

Проверка прав учетных записей осуществляется не в ядре, а в процессах Winlogon.exe и Lsass.exe.

*Привилегия* (privilege) – разрешение или запрет определенных действий в системе, например, включение/выключение компьютера

или*загрузка* драйверов. Привилегии хранятся в *поле Privileges* структуры маркера доступа *TOKEN* (см. выше).

*Список* всех привилегий в системе можно посмотреть в оснастке *MMC* "Локальная *политика безопасности*" (*Local Security Policy*), раздел "Локальные политики" – "Назначение прав пользователей" (*Local* Policies – *User* Rights *Assignment*)

Вызывается оснастка через

Панель управления – *Администрирование*. (*Control Panel* – *Administrative Tools*).

**Программа по теме «Система Безопасности ОС»**

**Создание дескриптора безопасности**

Дескриптор безопасности создается для ресурса, например, каталога. Дескриптор безопасности связывает ресурс с пользователем или процессом. В последующем при обращении к ресурсу проверяется, содержится ли данный процесс (пользователь) в дескрипторе безопасности. Пользователи идентифицируются с помощью SID (system identifier). SID – это число, уникально идентифицирующее пользователя. Когда пользователь запускает процесс, процесс наследует SID данного пользователя. Если процесс пытается работать с каталогом или файлом, то его SID должен содержаться в дескрипторе безопасности этого файла или ресурса. Теперь смотрите программу. О ней я рассказываю на видеоролике.

// DescriptorSafetyWin.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

//Создание дескриптора безопасности

#include <windows.h>

#include <stdio.h>

#include <lm.h>

int main()

{

//char chDirName[248] ; // имя каталога

TCHAR chDirName[]=TEXT("c:/work/ovg5");

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.bInheritHandle = 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);

getchar();

return dwErrCode;

}

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

getchar();

return 0;

}