Лекция 8. Безопасность

Архитектура ОС Windows

27 ноября 2014 г.

Маркер доступа

Определения

Контекст безопасности: (*security context*) — текущие действующие атрибуты или правила безопасности.

Маркер доступа: (*access token*) — описание информации безопасности для сеанса.

Привилегия: (*privilege*) — право пользователя выполнять системную операцию.

Домен Windows: (*Windows domain*) — набор компьютеров и связанных групп безопасности, управляемых как единое целое.

Виды маркеров доступа

Определения

Олицетворение: (*impersonation*) — механизм, позволяющий процессу или потоку использовать маркер доступа другого пользователя.

Основной маркер: (*primary token*) — присваивается процессу при запуске, представляет его информацию о безопасности по умолчанию.

Маркер олицетворения: (*impersonation token*) — дополнительный маркер для потока, позволяющий ему временно заимствовать профиль защиты другого пользователя.

Вход в систему

LogonUser()

```
BOOL LogonUser(
  _In_      LPTSTR  lpszUsername,
  _In_opt_  LPTSTR  lpszDomain,
  _In_opt_  LPTSTR  lpszPassword,
  _In_      DWORD   dwLogonType,
  _In_      DWORD   dwLogonProvider,
  _Out_     PHANDLE phToken
);
```

LOGON32_LOGON_BATCH
LOGON32_LOGON_INTERACTIVE
LOGON32_LOGON_NETWORK
LOGON32_LOGON_SERVICE
...

Таблица 1: значения dwLogonType

LOGON32_PROVIDER_DEFAULT
...

Таблица 2: значения dwLogonProvider

Использование маркеров доступа

CreateProcessAsUser()

```
BOOL WINAPI CreateProcessAsUser(
  _In_opt_      HANDLE      hToken,
  /* остальные параметры CreateProcess() */
);
```

ImpersonateLoggedOnUser(), RevertToSelf()

```
BOOL WINAPI ImpersonateLoggedOnUser(
  _In_          HANDLE      hToken
);

BOOL WINAPI RevertToSelf(void);
```

Базы данных безопасности

Определения

База данных локальных политик безопасности: (*LSASS policy database*) — содержит настройки политик безопасности локальной системы (`HKLM\SECURITY`):

- доверенные домены для аутентификации попыток входа;
- субъекты, имеющие доступ к системе, тип (интерактивно, сетевой вход, служба);
- назначенные привилегии субъектам;
- проводимые виды аудита безопасности.

Базы данных безопасности (окончание)

Определения

База данных диспетчера учётных записей: (*SAM database*) — содержит данные локальных пользователей и групп вместе с паролями и другими атрибутами (`HKLM\SAM`).

Активный каталог: (*Active directory*) — хранит информацию о пользователях, группах и компьютерах домена, вместе с паролями и привилегиями.

Компоненты безопасности

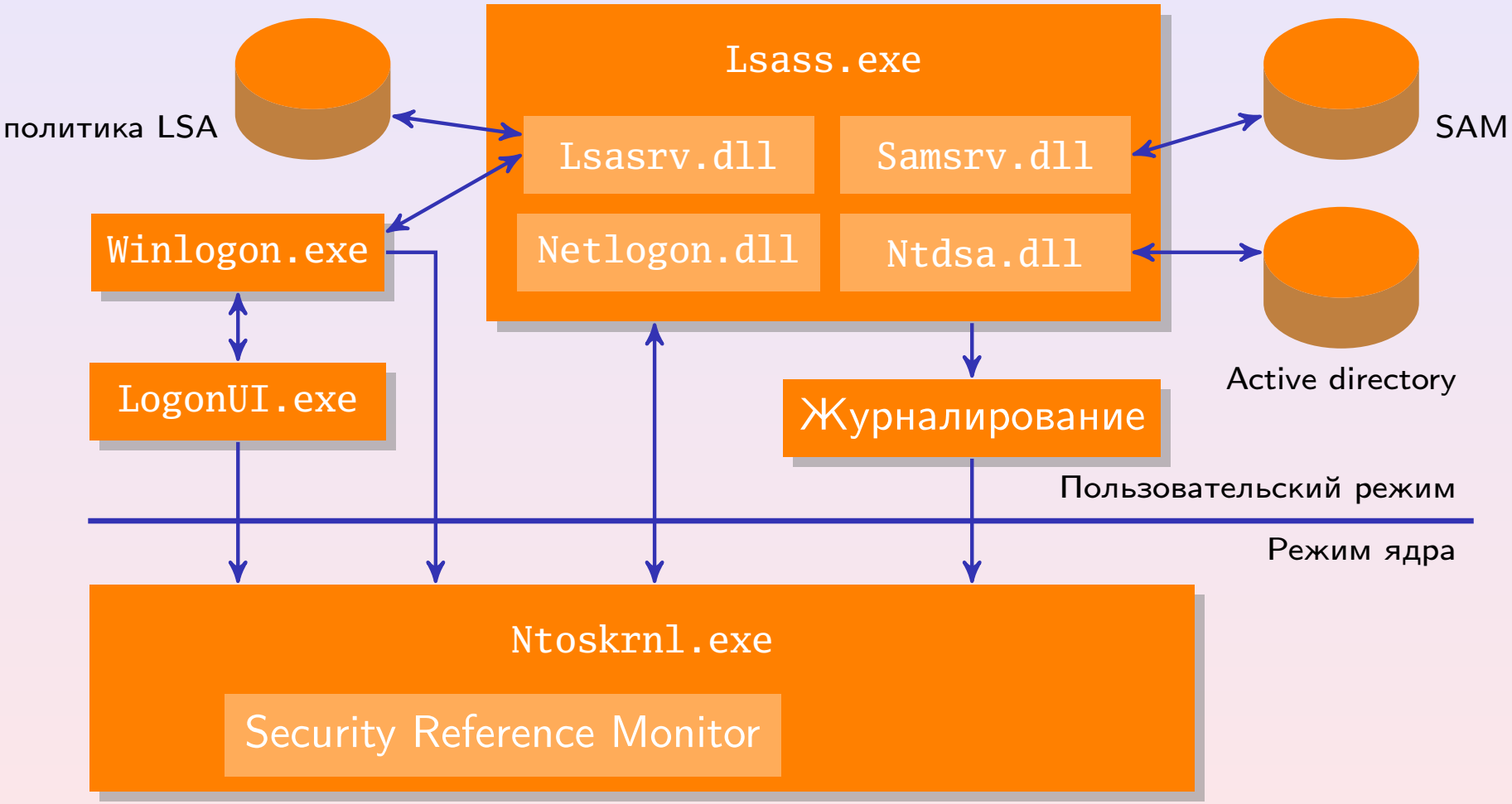

Рис. 1: компоненты безопасности Windows

Монитор состояния защиты

Функции монитора состояния защиты (Security Reference Monitor, SRM)

- Определение структуры маркера доступа для представления контекста зашиты;
- Проверка прав доступа к объектам;
- Манипулирование привилегиями;
- Генерация сообщений аудита безопасности.

Подсистема LSASS

Функции подсистемы проверки подлинности локальной системы безопасности

- Политика безопасности локальной системы:
  - какие пользователи имеют право входа в систему,
  - политики паролей,
  - привилегии пользователей и групп,
  - настройки аудита безопасности,
  - ...
- Аутентификация пользователей;
- Журналирование сообщений аудита безопасности.

Подсистема LSASS (окончание)

Файл	Название
`Lsasrv.dll`	Local Security Authority service
`Samsrv.dll`	Security Accoutns Manager
`Ntdsa.dll`	Active Directory
`Netlogon.dll`	Network Logon service

Таблица 3: библиотеки, загружаемые LSASS («службы»)

Определение (`Netlogon.dll`)

Служба сетевого входа: устанавливает безопасное соединение с контроллером домена, по которому отправляются запросы системы безопасности:

- интерактивный вход, если КД на Windows NT 4;
- подтверждение аутентификации, если (NT) LAN Manager.

Также используется для входов Active Directory.

Управление локальными политиками безопасности

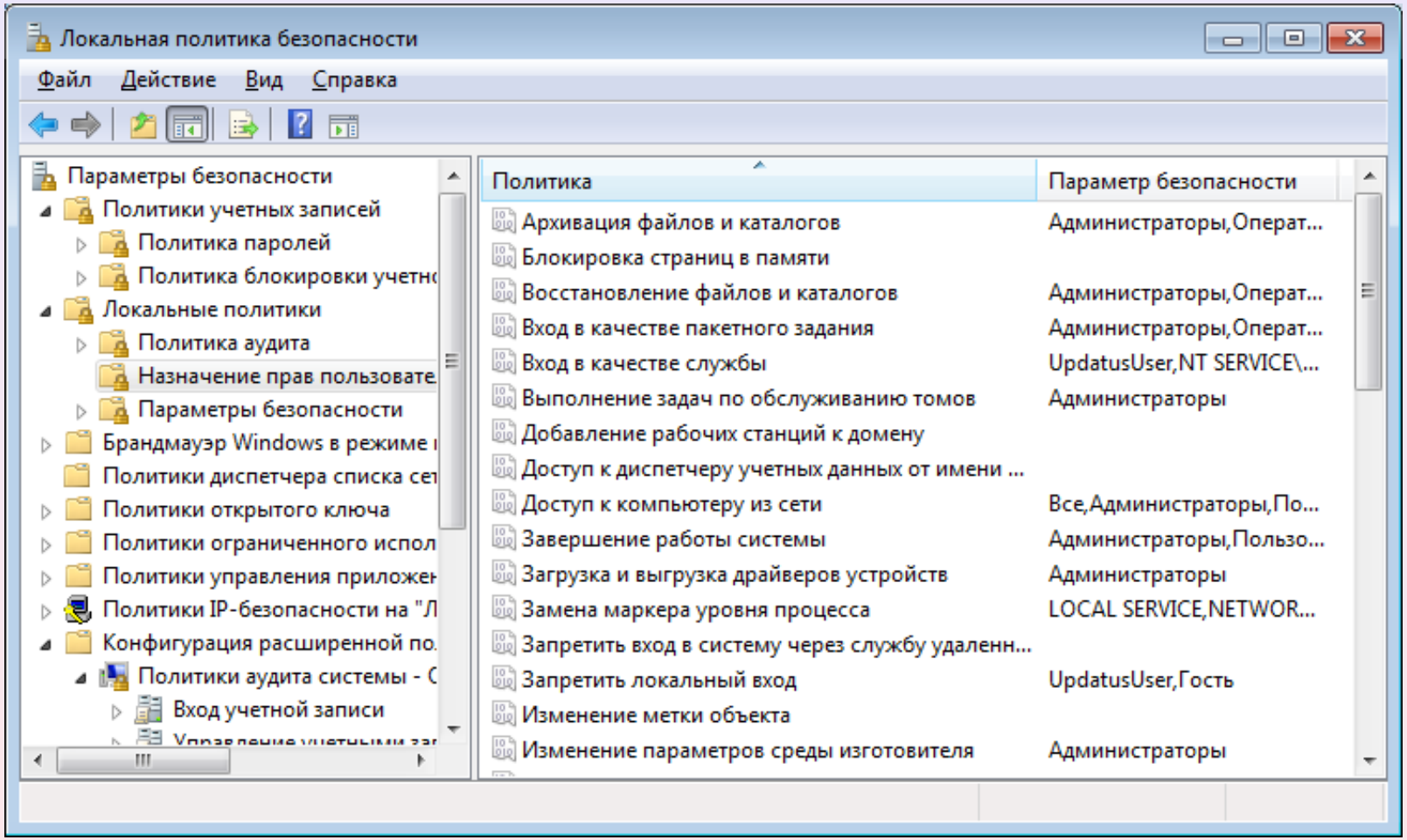

Рис. 2: окно управления локальными политиками безопасности

Компоненты аутентификации

Определения

Пакеты аутентификации: (*Authentication packages*) — библиотеки (DLL), отвечают за проверку имени и пароля пользователя

Поставщики учётных данных: (*Credential providers, CP*) — используются для получения учётных данных (имя/пароль, PIN смарт-карты, биометрические данные, . . .) — серверы COM внутри процесса.

Интерактивный диспетчер входа в систему: (*Interactive logon manager*, `Winlogon.exe`) — отвечает на SAS и управление интерактивными сеансами (создаёт I процесс пользователя, . . .)

Пользовательский интерфейс входа в систему: (*Logon user interface*, `LogonUI.exe`) — предоставляет пользовательский интерфейс для аутентификации, использует поставщиков учётных данных.

Другие компоненты безопасности

Определения

Драйвер устройства безопасности ядра: (*Kernel security device driver*, `Ksecdd.sys`) — библиотека, реализующая соединения ALPC, используется компонентами безопасности ядра (EFS, ...) для связи с LSASS.

AppLocker: (`AppIdSvc.dll`) — позволяет контролировать администраторам, какие приложения, библиотеки и сценарии могут быть использованы субъектами.

Идентификаторы защиты (SID)

Объекты идентификации

- пользователи;
- группы (локальные или доменные);
- локальные компьютеры;
- домены;
- члены доменов.

Идентификатор защиты (SID)

Определения

Идентификатор безопасности: (*Security identifier, SID*) — структура переменной длины для идентификации сущности.

Агент: сторона, выдавшая SID.

Субагент: попечитель, уполномоченный агентом.

Виды агентов

- локальная система;
- домен под управлением Windows.

Структура идентификатора защиты

Структура

- версия структуры SID;
- код агента идентификатора (48 бит);
- код субагента (выбирается случайно) или код относительного идентификатора (RID) (32 бит, повтор n раз).

Пример SID

Пример

S-1-5-21-1463437245-1224812800-863842198-1128

Состав

- версия структуры SID = 1;
- код агента идентификатора = 5 (центр безопасности Windows);
- коды субагентов (4 раза);
- RID = 1128.

Правила назначения SID

Назначение SID компьютеру домена

Локальный компьютер получает SID с тем же номером версии, кодом агента идентификатора, такими же кодами субагентов, что и у SID домена.

Виды агентов

- компьютеру (Windows Setup);
- локальным учётным записям (Windows) — на основе SID компьютера с добавлением RID — с 1000 и каждый раз +1;
- доменам — аналогично (`dcpromo.exe`);
- учётным записям доменов — аналогично на основе SID доменов.

Определение принадлежности SID

RID	Пользователь
500	Администратор
501	Гость

Таблица 4: RID основных пользователей

SID	Имя	Описание
S-1-0-0	Null	Пустая группа
S-1-1-0	World	Все пользователи
S-1-2-0	Local	Пользователи, регистрируемые на локально (физически) подключаемых терминалах

Таблица 5: общеизвестные SID

Списки управления доступом

Определения

Элемент управления доступом: (*Access control entry, ACE*) — структура, содержащая набор прав доступа к охраняемому объекту и идентификатор безопасности (SID) субъекта, к которому эти права относятся.

Список управления доступом: (*Access control list, ACL*) — список прав доступа для охраняемого объекта (структур ACE).

Список управления избирательным доступом: (*Discretionary access control list, DACL*) — список управления доступом (ACL), определяющий права доступа субъектов к текущему объекту.

Системный список управления доступом: (*System access control list, SACL*) — список управления доступом (ACL), определяющий операции заданных субъектов над текущим объектом, которые должны регистрироваться в журнале аудита безопасности.

Дескриптор защиты

Определение

Дескриптор защиты: (*security descriptor*) — структура данных для описания защитной информации для охраняемого объекта. Включает:

- SID владельца;
- SID первичной группы;
- DACL (необязательно);
- SACL (необязательно);

Правила DACL

- DACL не задан (== `NULL`) $\Rightarrow$ все пользователи имеют полный доступ;
- DACL пуст (не содержит ACE) $\Rightarrow$ ни один пользователь не получает прав.

Пример работы с правами учётных записей

Пример (simple_sec.cpp)

```
#define WINVER 0x0600

#include <Windows.h>
#include <Ntsecapi.h>
#include <Ntdef.h>
#include <Sddl.h>
#include <tchar.h>

#include <iostream>
#include <iterator>
#include <algorithm>
#include <cstdlib>
```

Пример (продолжение)

```
using namespace std;

const size_t g_cuSize = 65536;
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
LSA_UNICODE_STRING get(LPWSTR lpwszStr)
{
  static CHAR s_szBuffer[g_cuSize];
  cin.getline(s_szBuffer, g_cuSize);
  MultiByteToWideChar(
    CP_OEMCP, 0, s_szBuffer, -1, lpwszStr, g_cuSize);
  USHORT ushLength = wcslen(lpwszStr);
  LSA_UNICODE_STRING lsa_string;
  lsa_string.Buffer = lpwszStr;
  lsa_string.Length = ushLength * sizeof (WCHAR);
  lsa_string.MaximumLength = (ushLength + 1) * sizeof (WCHAR);
  return lsa_string;
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
void print(LPCWSTR lpcwszStr, int nSize = -1)
{
  static CHAR s_szBuffer[g_cuSize];
  nSize = WideCharToMultiByte(
    CP_OEMCP, 0, lpcwszStr, nSize, s_szBuffer, g_cuSize,
    NULL, NULL);
  copy(
    s_szBuffer, s_szBuffer + nSize,
    ostream_iterator <CHAR> (cout));
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
void print(const LSA_UNICODE_STRING &rcStr)
{
  const int cnSize = rcStr.Length / sizeof (WCHAR);
  print(rcStr.Buffer, cnSize);
}

const char *getSidTypeStr(SID_NAME_USE nSidNameUse)
{
  switch (nSidNameUse)
  {
    case SidTypeUser:
      return "user SID";
      //
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
    case SidTypeGroup:
      return "group SID";
      //
    case SidTypeDomain:
      return "domain SID";
      //
    case SidTypeAlias:
      return "alias SID";
      //
    case SidTypeWellKnownGroup:
      return "SID for a well-known group";
      //
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
    case SidTypeDeletedAccount:
      return "SID for a deleted account";
      //
    case SidTypeInvalid:
      return "SID that is not valid";
      //
    case SidTypeUnknown:
      return "SID of unknown type";
      //
    case SidTypeComputer:
      return "SID for a computer";
  }    // switch (nSidNameUse)
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
  //
  return "???";
}    // getSidTypeStr()

int main()
{
  BOOL bResult = TRUE;
  DWORD dwSidSize = 0, dwDomainSize = 0;
  PSID pSid = 0;
  LPWSTR lpwszDomainName = 0;
  LPWSTR lpwszSid = 0;
  PLSA_UNICODE_STRING pLsaUserRights = 0;
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
//
// Чтение имени системы
//
cout << "System: " << flush;
static WCHAR s_wszSysName[g_cuSize];
LSA_UNICODE_STRING lsaSysName = get(s_wszSysName);
cout << "Querying \"";
print(lsaSysName);
cout << "\"" << endl;
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
//
// Получение дескриптора объекта политики
//
LSA_HANDLE lsahPolicy;
LSA_OBJECT_ATTRIBUTES lsaObjectAttributes = { 0 };
NTSTATUS ntsResult = LsaOpenPolicy(
  &lsaSysName,              // Имя целевой системы
  &lsaObjectAttributes,     // Атрибуты объекта
  POLICY_LOOKUP_NAMES,      // Разрешения доступа
  &lsahPolicy);             // Возвращаемый дескриптор политики
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
if (ntsResult != STATUS_SUCCESS)
{
  cerr <<
    "LsaOpenPolicy returned " <<
    LsaNtStatusToWinError(ntsResult) << endl;
  return -1;
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
//
// Чтение имени учётной записи
//
static WCHAR s_wszName[g_cuSize];
cout << "Name: " << flush;
LSA_UNICODE_STRING lsaName = get(s_wszName);
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
//
// Получение имён доменов
//
PLSA_REFERENCED_DOMAIN_LIST pDomains = 0;
PLSA_TRANSLATED_SID pSids = 0;
ntsResult = LsaLookupNames(
  lsahPolicy,                  // Дескриптор политики
  1,                           // Количество имён
  &lsaName,                    // Массив имён
  &pDomains,                   // Возвращаемые домены
  &pSids);                     // Возвращаемые идентификаторы
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
if (ntsResult != STATUS_SUCCESS)
{
  cerr <<
    "LsaLookupNames returned " <<
    LsaNtStatusToWinError(ntsResult) << endl;
  goto l_cleanup;
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
cout << "Domains:" << endl;
for (ULONG i = 0; i < pDomains[0].Entries; ++ i)
{
  cout << "  ";
  print(pDomains[0].Domains[i].Name);
  cout << " — ";
  ConvertSidToStringSidW(
    pDomains[0].Domains[i].Sid, &lpwszSid);
  print(lpwszSid);
  cout << endl;
  LocalFree(lpwszSid);
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
//
// Получение SID учётной записи
//
SID_NAME_USE nSidNameUse;
bResult = LookupAccountNameW(
  s_wszSysName,                  // Имя целевой системы
  s_wszName,                     // Имя учётной записи
  NULL,                          // Адрес SID
  &dwSidSize,                    // Адрес размера SID
  NULL,                          // Адрес имени домена
  &dwDomainSize,                 // Адрес размера имени домена
  &nSidNameUse);                 // Адрес типа учётной записи
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
if (!bResult &&
  GetLastError() != ERROR_INSUFFICIENT_BUFFER)
{
  cerr <<
    "LookupAccountNameW 1 returned " <<
    GetLastError() << endl;
  goto l_cleanup;
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
pSid = (PSID) malloc(dwSidSize);
lpwszDomainName =
  (LPWSTR) malloc(dwDomainSize * sizeof (WCHAR));
bResult = LookupAccountNameW(
  s_wszSysName,                  // Имя целевой системы
  s_wszName,                     // Имя учётной записи
  pSid,                          // Адрес SID
  &dwSidSize,                    // Адрес размера SID
  lpwszDomainName,               // Адрес имени домена
  &dwDomainSize,                 // Адрес размера имени домена
  &nSidNameUse);                 // Адрес типа учётной записи
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
if (!bResult)
{
  cerr <<
    "LookupAccountNameW 2 returned " <<
    GetLastError() << endl;
  goto l_cleanup;
}
ConvertSidToStringSidW(pSid, &lpwszSid);
cout << "Account: ";
print(lsaName);
cout << " — ";
print(lpwszSid);
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
cout <<
  "\nType: " << getSidTypeStr(nSidNameUse) << endl;
LocalFree(lpwszSid);
//
// Получение прав учётной записи
//
ULONG ulNumRights;
ntsResult = LsaEnumerateAccountRights(
  lsahPolicy,                  // Дескриптор политики
  pSid,                        // SID учётной записи
  &pLsaUserRights,             // Адрес адреса массива прав
  &ulNumRights);               // Адрес количества прав
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
if (ntsResult != STATUS_SUCCESS)
{
  cerr <<
    "LsaEnumerateAccountRights returned " <<
    LsaNtStatusToWinError(ntsResult) << endl;
  goto l_cleanup;
}
```

Пример работы с правами (продолжение)

Пример (simple_sec.cpp, продолжение)

```
cout << "Account rights:" << endl;
for (ULONG i = 0; i < ulNumRights; ++ i)
{
  cout << "   ";
  print(pLsaUserRights[i]);
  cout << endl;
}
```

Пример работы с правами (окончание)

Пример (simple_sec.cpp, окончание)

```
  //
l_cleanup:
  //
  // Очистка
  //
  free(lpwszDomainName);
  free(pSid);
  LsaFreeMemory(pLsaUserRights);
  LsaFreeMemory(pSids);
  LsaFreeMemory(pDomains);
  LsaClose(lsahPolicy);
}    // main()
```

Вывод программы проверки прав учётной записи

Пример

```
System:
Querying ""
Name: stu003
Domains:
  stu003-home - S-1-5-21-1075481764-988665211-876742088
Account: stu003 - S-1-5-21-1075481764-988665211-876742088-1000
Type: user SID
LsaEnumerateAccountRights returned 2
```

Вывод программы проверки прав (окончание)

Пример

```
System:
Querying ""
Name: Администраторы
Domains:
  BUILTIN - S-1-5-32
Account: Администраторы - S-1-5-32-544
Type: alias SID
Account rights:
  SeSecurityPrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeSystemtimePrivilege
  ...
```

Пример работы с привилегиями

Пример (privileges.cpp)

```
bool shutdown(bool bShutdown)
{
  //
  // Получение маркера текущего процесса
  //
  HANDLE hToken;
  BOOL bResult = OpenProcessToken(
    GetCurrentProcess(),                    // Дескриптор процесса
    TOKEN_ADJUST_PRIVILEGES |
    TOKEN_QUERY,                            // Права доступа к маркеру
    &hToken);                               // Адрес маркера
  if (!bResult)
    return false;
```

Пример работы с привилегиями (продолжение)

Пример (privileges.cpp, продолжение)

```
//
// Получение LUID привилегии отключения
//
static TOKEN_PRIVILEGES s_Privileges;
bResult = LookupPrivilegeValue(
  NULL,                                 // Имя системы
  SE_SHUTDOWN_NAME,                     // Имя привилегии
  &s_Privileges.Privileges[0].Luid); // Адрес LUID
if (!bResult)
  return false;
```

Пример работы с привилегиями (продолжение)

Пример (privileges.cpp, продолжение)

```
//
// Установка привилегии для процесса
//
s_Privileges.PrivilegeCount = 1;      // Одной привилерии
s_Privileges.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
bResult = AdjustTokenPrivileges(
  hToken,                             // Маркер
  FALSE,                              // Не отключать всё
  &s_Privileges,                      // Новое состояние
  0,                                  // Размер буфера состояния
  (PTOKEN_PRIVILEGES) NULL,           // Буфер предыдущего состояния
  NULL);                              // Адрес размера буфера
```

Пример работы с привилегиями (окончание)

Пример (privileges.cpp, окончание)

```
  if (!bResult)
    return false;
  //
  // Отключение системы и вынуждение закрытия всех приложений
  //
  UINT uFlags = (bShutdown ? EWX_POWEROFF : EWX_REBOOT);
  bResult = ExitWindowsEx(uFlags, 0);
  if (!bResult && bShutdown)
    bResult = ExitWindowsEx(EWX_SHUTDOWN, 0);
  //
  return bResult;
}    // shutdown()
```

Уровни целостности

SID	Ур.	Имя	Описание
S-1-16-0x0000	0	Untrusted	Процессы, запускаемые группой Anonymous
S-1-16-0x1000	1	Low	Internet Explorer в защищённом режиме
S-1-16-0x2000	2	Medium	Обычные приложения при включённом UAC
S-1-16-0x3000	3	High	Обычные приложения при отключённом UAC, административные приложения при включённом UAC, запущенные с запросом повышения прав
S-1-16-0x4000	4	System	Службы, приложения системного уровня (Wininit, Winlogon, Smss, ...)

Таблица 6: SID некоторых уровней целостности

Выбор уровня целостности для процесса

Правила выбора уровня целостности дочернего процесса

- Явное понижение уровня (`DuplicateTokenEx()`, `SetTokenInformation()`);
- Родительский процесс имеет уровень Medium или выше, файл образа дочернего процесса имеет уровень целостности $\Rightarrow$ выбирается наименьший;
- Иначе уровень наследуется.

Правила выбора уровня целостности объекта

- Явное задание;
- При отсутствии считается Medium;
- При создании без явного указания при уровне процесса от Medium выбирается Medium, иначе — как у процесса.

Мандатные политики

Название	Присутствие по умолчанию	Описание
Отказ в записи (no-write-up)	Неявно на всех объектах	Ограничение записи от процессов с меньшим уровнем
Отказ в чтении (no-read-up)	Только на объектах процессов	Ограничение чтения от процессов с меньшим уровнем
Отказ в исполнении (no-execute-up)	Только на файлах, реализующих классы COM	Ограничение исполнения от процессов с меньшим уровнем

Таблица 7: мандатные политики объекта (в ACE)

Создание ограниченного маркера

Группы

- Встроенные администраторы;
- Администраторы домена;
- Операторы архивирования;
- Криптографические операторы;
- ...

Привилегии

- SeBackupPrivilege;
- SeCreateTokenPrivilege;
- SeImpersonatePrivilege;
- SeLoadDriverPrivilege;
- ...

Отфильтрованный административный маркер

Правила создания копии административного маркера

- Уровень целостности устанавливается в Medium;
- Все упомянутые SID помечаются как имеющие силу только в отказе (deny-only);
- Удаляются все привилегии кроме некоторых (отключение, изменение часового пояса, . . .)

Проверки доступа к объекту

Флаги маркера доступа по умолчанию

- TOKEN_MANDATORY_NO_WRITE_UP;
- TOKEN_MANDATORY_NEW_PROCESS_MIN.

Последовательность проверки прав доступа к объекту

1. Проверка мандатной целостности;
2. Проверка разграничения доступа.

Изоляция привилегий пользовательского интерфейса

WM_NULL	WM_GETTEXTLENGTH	WM_RENDERFORMAT
WM_MOVE	WM_GETHOTKEY	WM_DRAWCLIPBOARD
WM_SIZE	WM_GETICON	WM_CHANGECBCHAIN
WM_GETTEXT	WM_THEMECHANGED	

Таблица 8: сообщения, не блокируемые от процессов с низшим уровнем целостности

Добавление сообщений исключения

ChangeWindowMessageFilterEx()

```
BOOL WINAPI ChangeWindowMessageFilterEx(
  _In_          HWND                hWnd,
  _In_          UINT                uMessage,
  _In_          DWORD               dwAction,
  _Inout_opt_   PCHANGEFILTERSTRUCT pChangeFilterStruct
);
```

MSGFLT_ALLOW	MSGFLT_DISALLOW	MSGFLT_RESET

Таблица 9: значения dwAction

Пример манифеста приложения

Пример

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel level="asInvoker" uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

Значения настроек манифеста

Значения `level`

`asInvoker`: как у родителя;

`highestAvailable`: наиболее высокие возможные привилегии;

`requireAdministrator`: с маркером полных административных прав.

Результат `uiAccess = "true"`

- Приложение должно иметь цифровую подпись;
- Приложение должно быть установлено только в безопасном месте (`%ProgramFiles%`, `%SystemRoot%`, ...);
- При запуске от обычного пользователя получает средний уровень целостности (между 0x2000 и 0x3000) и высокий (0x3000) от администратора;
- Запрос повышения прав доступа не выводится.

Виртуализация

Случаи отключения виртуализации

- 64-битные приложения;
- Службы;
- Исполняемые файлы с манифестом, совместимым с UAC (`requestedExecutionLevel`);
- Работа с правами администратора;
- У администратора нет прав доступа записи;
- Операция в режиме ядра;
- Операция в режиме олицетворения.

Запрос на повышение прав доступа

Алгоритм вывода запроса

1. Запуск образа, требующего административных прав, активизирует службу информации приложений (Application Information Service, AIS, `Appinfo.dll` в `Svchost.exe`);
2. AIS запускает `Consent.exe`;
3. `Consent.exe` делает снимок экрана, накладывает затемнение, переключается на безопасный рабочий стол, выводит изображение в качестве фона, выводит диалоговое окно (зависящее от наличия цифровых подписей Microsoft, . . . , обычных/административных прав);
4. При отказе возвращается ошибка отказа в доступе;
5. При согласии создаётся процесс при помощи `CreateProcessAsUser()`, родителем устанавливается процесс, инициировавший запуск.

Пример запуска программы

Пример

```
SHELLEXECUTEINFO cShellExInfo =
{
  sizeof (SHELLEXECUTEINFO),     // cbSize
  SEE_MASK_FLAG_NOASYNC |
  SEE_MASK_FLAG_NO_UI |
  SEE_MASK_NOCLOSEPROCESS,       // fMask
  m_hWndPrompt,                  // hWnd
  _T("runas"),                   // lpVerb
  lpctszFilePath,                // lpFile
  lpctszParameters,              // lpParameters
  lpctszCurrentDir,              // lpDirectory
  // ...
```

Пример запуска программы (окончание)

Пример (окончание)

```
    SW_SHOWNORMAL,                  // nShow
    NULL                            // hInstApp
};
//
BOOL bSuccess = ShellExecuteEx(&cShellExInfo);
```