Используйте видеоролик с демонстрацией работы с маркером доступа процесса.

(Description: Marker  
  
  
Ссылка для закачки ролика  
  
<https://www.sendspace.com/file/hjnngb>

В ролике рассматриваются две программы-

Получения из маркера информации о хозяине (пользователе, запустившем процесс/поток)

И получение привилегий (ролик немного незакончен по техническим причинам).

**Текст первой программы**

Получение информации из маркера доступа

1. Определение источника маркера доступа

#include "stdafx.h"

#include <windows.h>

#include <tlhelp32.h>

#include <stdio.h>

int main ()

{

HANDLE hProcess; // дескриптор процесса

HANDLE hTokenHandle; // дескриптор маркера доступа

TOKEN\_SOURCE ts; // источник маркера доступа

DWORD dwRetLength; // возвращаемая длина буфера

DWORD dwRetCode; // код возврата

// получаем дескриптор процесса

hProcess = GetCurrentProcess();

// открываем маркер доступа процесса

if (!OpenProcessToken(

hProcess, // дескриптор процесса

TOKEN\_QUERY\_SOURCE, // доступ к источнику маркера

&hTokenHandle)) // дескриптор маркера

{

dwRetCode = GetLastError();

printf( "Open process token failed: %u\n", dwRetCode);

return dwRetCode;

}

// получаем источник маркера доступа

if (!GetTokenInformation(

hTokenHandle, // дескриптор маркера доступа

TokenSource, // получаем источник маркера доступа

&ts, // адрес буфера для источника

sizeof(ts), // размер буфера

&dwRetLength)) // требуемый размер буфера в случае неудачи

{

dwRetCode = GetLastError();

printf( "Get token information failed: %u\n", dwRetCode);

return dwRetCode;

}

// распечатываем источник маркера доступа

printf("Source name: ");

for (int i = 0; i < TOKEN\_SOURCE\_LENGTH; ++i)

putchar(ts.SourceName[i]);

putchar('\n');

printf("Source identifier: %X %X\n", ts.SourceIdentifier.HighPart, ts.SourceIdentifier.LowPart);

CloseHandle(hTokenHandle);

getchar();

return 0;

}

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

Текст второй программы

// TokenAccess2.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <windows.h>

#include "stdio.h"

#include "iostream"

int main ()

{

// Get a token handle.

HANDLE hToken;

if (!OpenThreadToken(GetCurrentThread(), TOKEN\_QUERY, FALSE, &hToken))

{

if (!OpenProcessToken(GetCurrentProcess(), TOKEN\_QUERY, &hToken))

return false;

}

// Get the token privilege information.

DWORD dwNeeded = 0;

GetTokenInformation(hToken, TokenPrivileges, NULL, 0, &dwNeeded);

LPBYTE pBuffer = new BYTE[dwNeeded + 1];

GetTokenInformation(hToken, TokenPrivileges, pBuffer, dwNeeded, &dwNeeded);

// Cast to the proper type.

PTOKEN\_PRIVILEGES pTokenPrivileges = reinterpret\_cast<PTOKEN\_PRIVILEGES>(pBuffer);

// Iterate the privileges.

for (DWORD i = 0; i < pTokenPrivileges->PrivilegeCount-15; ++i)

{

// Get and display the privilege name.

DWORD dwSize = 0;

LookupPrivilegeName(NULL, &pTokenPrivileges->Privileges[i].Luid, NULL, &dwSize);

LPWSTR szName =(LPWSTR) new CHAR[dwSize + 1];

LookupPrivilegeName(NULL, &pTokenPrivileges->Privileges[i].Luid, szName, &dwSize);

std::cout<<szName<<std::endl;

// delete[] szName;

// Display the privilege state.

switch(pTokenPrivileges->Privileges[i].Attributes)

{

case SE\_PRIVILEGE\_ENABLED:

std::cout<<szName<<" Enabled"<<std::endl;

break;

case SE\_PRIVILEGE\_ENABLED\_BY\_DEFAULT:

std::cout<<szName<<" Enabled by default"<<std::endl;

break;

case SE\_PRIVILEGE\_REMOVED:

std::cout<<szName<<" Removed"<<std::endl;

break;

case SE\_PRIVILEGE\_USED\_FOR\_ACCESS:

std::cout<<szName<<" Used for access"<<std::endl;

break;

default:

std::cout<<szName<<" Disabled"<<std::endl;

break;

}

// Finish the output for the line.

// std::cout<<")"<<std::endl;

}

// delete[] pBuffer;

CloseHandle(hToken);

getchar();

return 0;

}

Вывод на экран системных номеров привилегий:

