TsSysKit.sys分析报告

By bruceli

该驱动为qq管家穿透驱动，提供文件、注册表穿透，和解锁文件、驱动加载、句柄、进程等操作，导出接口给其他驱动使用，设备名[\\Device\\TSSysKit](file:///\\Device\\TSSysKit)，符号名[\\DosDevices\\TSSysKit](file:///\\DosDevices\\TSSysKit)。加密手段：Rabbit算法、MD5算法。

[一、驱动入口DriverEntry 2](#_Toc432244632)

[1.1 检测加载者是否为Ntos 2](#_Toc432244633)

[1.2 执行删除任务 DoDeleteJob 3](#_Toc432244634)

[二、驱动接口Interface 5](#_Toc432244635)

[2.1 DeviceExtension接口 5](#_Toc432244636)

[2.2 与WRK代码异同点 6](#_Toc432244637)

[2.3 重置/保存注册表对象ObjectInitialzer例程 6](#_Toc432244638)

[2.4 NtCreateKey 7](#_Toc432244639)

[2.5 NtOpenKey 9](#_Toc432244640)

[2.6 NtQueryValueKey 10](#_Toc432244641)

[2.7 NtSetValueKeyEx 11](#_Toc432244642)

[2.8 NtDeleteValueKey 12](#_Toc432244643)

[2.9 NtDeleteKey 13](#_Toc432244644)

[2.10 IopCreateFile 14](#_Toc432244645)

[三、控制码 17](#_Toc432244646)

[3.1 TSSysKit x86 IoControlCode对应表 17](#_Toc432244647)

[0x221C00解锁文件 21](#_Toc432244648)

[0x222004普通结束进程 21](#_Toc432244649)

[0x22242C穿透创建服务加载驱动 22](#_Toc432244650)

[3.2 TSSysKit x64 IoControlCode对应表 25](#_Toc432244651)

[四、默认派遣例程 30](#_Toc432244652)

[4.1 根据进程id结束进程 30](#_Toc432244653)

[4.2 获取当前进程进程名 30](#_Toc432244654)

[4.3 由进程ID获取进程设备名 31](#_Toc432244655)

[4.4 设备名转DOS路径 32](#_Toc432244656)

[4.5 得到EPROCESS对应ImageDosPath 33](#_Toc432244657)

[4.6 随机化程序名机制 34](#_Toc432244658)

[4.7 根据进程文件名获取进程信息 35](#_Toc432244659)

[4.8 两种方式调用内核函数 35](#_Toc432244660)

[4.9 获取对象类型 36](#_Toc432244661)

[4.10 基础库功能——检测腾讯程序合法性 36](#_Toc432244662)

[4.11 解锁文件 38](#_Toc432244663)

[五、获取ObjectInitializer 44](#_Toc432244664)

[5.1 获取注册表OBJECT\_TYPE，匹配对象类型 44](#_Toc432244665)

[5.2获取ParseProcedure 46](#_Toc432244666)

[5.3 获取GetCellRoutine偏移，Hook GetCellRoutine 47](#_Toc432244667)

[5.4 Hook和UnHook GetCellRoutine 48](#_Toc432244668)

[5.5 创建系统线程获取 Cm\*函数 49](#_Toc432244669)

[5.6 匹配结构 54](#_Toc432244670)

[5.8 获取DeviceObject对象类型 73](#_Toc432244671)

# 一、驱动入口DriverEntry

* 获取系统版本\_1
* 获取EPROCESS的ObjectTypeIndex (win7以前 0x1C win8 0x1F Win8.1 0x1E)
* 创建[\\Device\\TSSysKit](file:///\\Device\\TSSysKit)设备和[\\DosDevices\\TSSysKit](file:///\\DosDevices\\TSSysKit)符号链接
* 设置DeviceExtension为通信接口（Interface函数指针）
* 注册IRP\_MJ\_CREATE、IRP\_MJ\_CLOSE、IRP\_MJ\_DEVICE\_CONTROL派遣例程为DefaultDispatch
* 获取CmpKeyObject、DeviceObject的OBJECT\_TYPE\_INITIALIZER

GetCmpKeyObjectInitializerFunc GetDeviceObjectInitializerFunc

* 获取ZwQueryVirtualMemory、IoVolumeDeviceToDosName、PsGetProcessSectionBaseAddress地址和NtQueryVirtualMemory的index
* 检测加载者是否为Ntos (CheckIopLoadDriver)
* 执行开机删除操作(DoDeleteJob) 删除ShutdownRecord.ini指定的文件



## 1.1 检测加载者是否为Ntos

顺带保存IopLoadDriver

int CheckIopLoadDriver()

{

unsigned int IopLoadDriverNext; // esi@2

PRTL\_PROCESS\_MODULES modules; // ebx@5

int IopLoadDriver; // esi@8

PVOID Base; // eax@9

PVOID Callers[4]; // [sp+8h] [bp-18h]@1

BOOL ret; // [sp+18h] [bp-8h]@1

ULONG SystemInformationLength; // [sp+1Ch] [bp-4h]@1

Callers[0] = 0;

Callers[1] = 0;

Callers[2] = 0;

Callers[3] = 0;

ret = 0;

SystemInformationLength = 0;

::IopLoadDriver = 0;

if ( RtlWalkFrameChain(Callers, 4u, 0) == 4 )

// [0]=RtlWalkFrameChain next

// [1]=DriverEntry

// [2]=IopLoadDriver

// [3]=IopInitializeSystemDrivers

{

IopLoadDriverNext = Callers[3];

if ( MmIsAddressValid(Callers[3]) )

{

if ( IopLoadDriverNext >= MmUserProbeAddress && \*(IopLoadDriverNext - 5) == 0xE8u )// call \*\*\*

{

SystemInformationLength = sizeof(SYSTEM\_MODULE\_INFORMATION) + 3 \* sizeof(RTL\_PROCESS\_MODULE\_INFORMATION);

modules = ExAllocatePoolWithTag(0, SystemInformationLength, '!KIT');

if ( modules )

{

memset(modules, 0, SystemInformationLength);

if ( ZwQuerySystemInformation(SystemModuleInformation,modules,SystemInformationLength,&SystemInformationLength) >= 0 )

{

if ( modules->NumberOfModules )

{

IopLoadDriver = \*(IopLoadDriverNext - 4) + IopLoadDriverNext;

if ( MmIsAddressValid(IopLoadDriver) )

{

Base = modules->Modules[0].ImageBase;

if ( IopLoadDriver >= Base && IopLoadDriver <= (Base + modules->Modules[0].ImageSize) )

{ // 检测IopLoadDriver是否在ntos中

::IopLoadDriver = IopLoadDriver;

ret = 1;

}

}

}

}

ExFreePool(modules);

}

}

}

}

return ret;

}

## 1.2 执行删除任务 DoDeleteJob

如果腾讯安装目录存在ShutdownRecord.ini文件则删除该文件，解析文件列表中指定的文件并逐个删除，其中解析ini的api、文件操作都是自己实现的。删除方式采用NtSetInformationFile 置FileDispositionInformation。

ShutdownRecord.ini格式：

[DELETEFILECOUNT]

Count=3

[DELETEFILELIST]

0=0.txt

1=1.txt

2=2.txt

enum

{

WIN2000=1,

WINXP=2,

WINXPSP3=3,

WINVISTA=4,

WIN7=5,

WIN8=7,

WIN8\_1=8,

WIN10=9,

UNKNOWN=10,

};

# 二、驱动接口Interface

## 2.1 DeviceExtension接口

DeviceObject->DeviceExtension(=Interface) 通过制定序号返回对应函数指针，穿透函数内部在ObOpenObjectByName前后会保存和恢复注册表Objectinitializer：

FARPROC Interface(intindex)

{//注意下面的函数都是自己实现的穿透函数

switch(index)

{

case 1:

return NtOpenKey;

case 2:

return NtQueryValueKey;

case 3:

return NtSetValueKeyEx;

case 4:

return NtDeleteValueKey;

case 5:

return NtDeleteKey;

case 20:

return IopCreateFile;

case 21:

return NtReadFile;

case 22:

return NtWriteFile;

case 23:

return NtSetInformationFile;

case 24:

return NtQueryInformationFile;

case 25:

return NtQueryDirectoryFile;

}

}

函数原型：

0：NTSTATUS \_\_stdcall NtCreateKey(PHANDLE KeyHandle, ACCESS\_MASK DesiredAccess, POBJECT\_ATTRIBUTES ObjectAttributes, ULONG TitleIndex, PUNICODE\_STRING Class, ULONG CreateOptions, PULONG Disposition)

1：NTSTATUS \_\_stdcall NtOpenKey(PHANDLE KeyHandle, ACCESS\_MASK DesiredAccess, POBJECT\_ATTRIBUTES ObjectAttributes)

2：NTSTATUS \_\_stdcall NtQueryValueKey(HANDLE KeyHandle, PUNICODE\_STRING ValueName, KEY\_VALUE\_INFORMATION\_CLASS KeyValueInformationClass, PVOID KeyValueInformation, ULONG Length, PULONG ResultLength)

3：NTSTATUS \_\_stdcall NtSetValueKeyEx(HANDLE Handle, PUNICODE\_STRING ValueName, ULONG Type, PVOID Data, ULONG DataSize)

和NtSetValueKey的用法类似，只是没有TitleIndex这个参数

4：NTSTATUS \_\_stdcall NtDeleteValueKey(HANDLE KeyHandle, PUNICODE\_STRING ValueName)

5：NTSTATUS \_\_stdcall NtDeleteKey(HANDLE KeyHandle)

20：NTSTATUS \_\_stdcall IopCreateFile(PHANDLE FileHandle, ACCESS\_MASK DesiredAccess, POBJECT\_ATTRIBUTES ObjectAttributes, PIO\_STATUS\_BLOCK IoStatusBlock, PLARGE\_INTEGER AllocationSize, ULONG FileAttributes, ULONG ShareAccess, ULONG Disposition, ULONG CreateOptions, PVOID EaBuffer, ULONG EaLength, CREATE\_FILE\_TYPE CreateFileType, PVOID ExtraCreateParameters, ULONG Options, ULONG InternalFlags, PVOID DeviceObject)

21：NTSTATUS \_\_stdcall NtReadFile(HANDLE FileHandle, HANDLE Event, PIO\_APC\_ROUTINE ApcRoutine, PVOID ApcContext, PIO\_STATUS\_BLOCK IoStatusBlock, PVOID Buffer, ULONG Length, PLARGE\_INTEGER ByteOffset, PULONG Key)

22：NTSTATUS \_\_stdcall NtWriteFile(HANDLE FileHandle, HANDLE Event, PIO\_APC\_ROUTINE ApcRoutine, PVOID ApcContext, PIO\_STATUS\_BLOCK IoStatusBlock, PVOID Buffer, ULONG Length, PLARGE\_INTEGER ByteOffset, PULONG Key)

23：NTSTATUS \_\_stdcall NtSetInformationFile(HANDLE FileHandle, PIO\_STATUS\_BLOCK IoStatusBlock, PVOID FileInformation, ULONG Length, FILE\_INFORMATION\_CLASS FileInformationClass, BOOL DelCurrentFile)

24：NTSTATUS \_\_stdcall NtQueryInformationFile(HANDLE FileHandle, PIO\_STATUS\_BLOCK IoStatusBlock, PVOID FileInformation, ULONG Length, FILE\_INFORMATION\_CLASS FileInformationClass, BOOL DelCurrentFile)

25：NTSTATUS \_\_stdcall NtQueryDirectoryFile(HANDLE FileHandle, HANDLE Event, PIO\_APC\_ROUTINE ApcRoutine, PVOID ApcContext, PIO\_STATUS\_BLOCK IoStatusBlock, PVOID FileInformation, ULONG Length, FILE\_INFORMATION\_CLASS FileInformationClass, BOOLEAN ReturnSingleEntry, PUNICODE\_STRING FileName, BOOLEAN RestartScan)

### 2.2 与WRK代码异同点

注册表穿透操作是使用Cm\*函数实现Nt\*函数，而文件穿透操作则基本和WRK代码一致，Iop\*函数最大程度的实现内联。

这些实现中和WRK主要区别在于：去掉了AccessMode=UserMode分支、和ObReferenceObjectByName调用前后重置ObjectInitializer

NtReadFile实现额外处理了IoCallDriver返回文件锁定（STATUS\_FILE\_LOCK\_CONFLICT）的处理，此时通过创建文件映射实现读取

## 2.3 重置/保存注册表对象ObjectInitialzer例程

enum

{

EClose,

EDelete,

EParse,

ESecurtiy,

EQueryName,

EOpen,

};

BOOLEAN ReplaceObjectinitializer(int Type,FARPROC\* OutFunc,BOOLEAN ResetOrRestore)

{//重置/保存注册表对象ObjectInitialzer例程，用于ObOpenObjectByName执行前后

FARPROC\* Initailier = NULL;

if(Type <0 || Type >= ECmMax)

{

if(!RegObjectInitialzer[Type] || !CmpKeyObjectType || !OutFunc)

return 0;

}

if(VersionIndex >= WIN2000 && VersionIndex <= WINXPSP3)

{

switch(Type)

{

case EClose:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.CloseProcedure;

break;

case EDelete:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.DeleteProcedure;

break;

case EParse:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.ParseProcedure;

break;

case ESecurity:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.SecurityProcedure;

break;

case EQueryName:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.QueryNameProcedure;

break;

case EOpen:

Initailier = &((POBJECT\_TYPE\_XP)CmpKeyObjectType)->TypeInfo.OpenProcedure;

break;

}

}

else if(VersionIndex >= WINVISTA && VersionIndex <= WIN10)

{

switch(Type)

{

case EClose:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.CloseProcedure;

break;

case EDelete:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.DeleteProcedure;

break;

case EParse:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.ParseProcedure;

break;

case ESecurity:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.SecurityProcedure;

break;

case EQueryName:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.QueryNameProcedure;

break;

case EOpen:

Initailier = &((POBJECT\_TYPE\_WIN7)CmpKeyObjectType)->TypeInfo.OpenProcedure;

break;

}

}

if(ResetOrRestore)

{//Get

if(\*Initailier != RegObjectInitialzer[Type])

{

\*OutFunc = \*Initailier;

\*Initailier = RegObjectInitialzer[Type];

return TRUE;

}

}

else

{//Set

if(\*OutFunc != \*Initailier)

{

\*Initailier = \*OutFunc;

return TRUE;

}

}

return FALSE;

}

## 2.4 NtCreateKey

NTSTATUS \_\_stdcall NtCreateKey(PHANDLE KeyHandle,ACCESS\_MASK DesiredAccess,POBJECT\_ATTRIBUTES ObjectAttributes,ULONG TitleIndex,PUNICODE\_STRING Class,ULONG CreateOptions,PULONG Disposition)

{//

NTSTATUS status;

KPROCESSOR\_MODE mode;

CM\_PARSE\_CONTEXT ParseContext;

PCM\_KEY\_BODY KeyBody = NULL;

HANDLE Handle = 0;

UNICODE\_STRING CapturedObjectName = {0};

FARPROC SavedInitializer = NULL;

BOOL NeedRestore = FALSE;

RtlZeroMemory(&ParseContext,sizeof(ParseContext));

if (ARGUMENT\_PRESENT(Class))

{

ParseContext.Class = \*Class;

}

if ((CreateOptions & (REG\_LEGAL\_OPTION | REG\_OPTION\_PREDEF\_HANDLE)) != CreateOptions)

{

return STATUS\_INVALID\_PARAMETER;

}

ParseContext.TitleIndex = 1;

ParseContext.CreateOptions = CreateOptions;

ParseContext.Disposition = 0L;

ParseContext.CreateLink = FALSE;

ParseContext.PredefinedHandle = NULL;

ParseContext.CreateOperation = TRUE;

ParseContext.OriginatingPoint = NULL;

if(ReplaceObjectinitializer(EParse,&SavedInitializer,1))//还原为系统默认值

NeedRestore = TRUE;

status = ObOpenObjectByName(ObjectAttributes,CmpKeyObjectType,mode,NULL,DesiredAccess,(PVOID)&ParseContext,&Handle);

if(NeedRestore)

ReplaceObjectinitializer(EParse,&SavedInitializer,0);

if (status==STATUS\_PREDEFINED\_HANDLE)

{

if(VersionIndex < WINVISTA)

{

status = ObReferenceObjectByHandle(Handle,0,CmpKeyObjectType,KernelMode,(PVOID \*)(&KeyBody),NULL);

if (NT\_SUCCESS(status))

{

HANDLE TempHandle;

TempHandle = (HANDLE)LongToHandle(KeyBody->Type);

ObDereferenceObject(KeyBody);

ZwClose(Handle);

\*KeyHandle = TempHandle;

status = STATUS\_SUCCESS;

}

}

else

{

TempHandle = (HANDLE)LongToHandle(KeyBody->Type);

ObDereferenceObject((PVOID)KeyBody);

NtClose(Handle);

\*KeyHandle = ParseContext.OriginatingPoint;

status = STATUS\_SUCCESS;

}

}

if (ARGUMENT\_PRESENT(Disposition))

{

\*Disposition = ParseContext.Disposition;

}

return status;

}

## 2.5 NtOpenKey

NTSTATUS \_\_stdcall NtOpenKey(PHANDLE KeyHandle, ACCESS\_MASK DesiredAccess, POBJECT\_ATTRIBUTES ObjectAttributes)

{

CM\_PARSE\_CONTEXT ParseContext;

PVOID Context;

HANDLE Handle =0;

NTSTATUS status = STATUS\_SUCCESS;

PCM\_KEY\_BODY KeyBody;

FARPROC SavedInitializer = NULL;

BOOL NeedRestore = FALSE;

RtlZeroMemory(&ParseContext,sizeof(CM\_PARSE\_CONTEXT));

Context = VersionIndex!=WIN2000?&ParseContext:NULL;

ParseContext.CreateOperation = FALSE;

if(ReplaceObjectinitializer(EParse,&SavedInitializer,1))//还原为系统默认值

NeedRestore = TRUE;

status = ObOpenObjectByName(ObjectAttributes,CmpKeyObjectType,KernelMode,NULL,DesiredAccess,(PVOID)&Context,&Handle);

if(NeedRestore)

ReplaceObjectinitializer(EParse,&SavedInitializer,0);

if (status==STATUS\_PREDEFINED\_HANDLE)

{

status = ObReferenceObjectByHandle(Handle,0,CmpKeyObjectType,KernelMode,(PVOID \*)(&KeyBody),NULL);

if (NT\_SUCCESS(status))

{

\*KeyHandle = (HANDLE)LongToHandle(KeyBody->Type);

ObDereferenceObject((PVOID)KeyBody);

if(\*KeyHandle)

{

status = STATUS\_SUCCESS;

}

else

{

status = STATUS\_OBJECT\_NAME\_NOT\_FOUND;

}

}

ZwClose(Handle);

}

else if (NT\_SUCCESS(status))

{

\*KeyHandle = Handle;

}

return status;

}

## 2.6 NtQueryValueKey

NTSTATUS \_\_stdcall NtQueryValueKey(HANDLE KeyHandle,PUNICODE\_STRING ValueName,KEY\_VALUE\_INFORMATION\_CLASS KeyValueInformationClass,

PVOID KeyValueInformation,ULONG Length,PULONG ResultLength)

{

NTSTATUS status;

PCM\_KEY\_BODY KeyBody = NULL;

KPROCESSOR\_MODE mode;

UNICODE\_STRING LocalValueName = {0};

if(!ValueName || (KeyValueInformationClass != KeyValueBasicInformation && KeyValueInformationClass != KeyValueFullInformation

&& KeyValueInformationClass != KeyValuePartialInformation && KeyValueInformationClass != KeyValueFullInformationAlign64 &&

KeyValueInformationClass != KeyValuePartialInformationAlign64))

return STATUS\_INVALID\_PARAMETER;

if(CmMatchData[VersionIndex][ECmQueryValueKey].InitFlag && CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr)

{

mode = KeGetPreviousMode();

status = ObReferenceObjectByHandle(KeyHandle,KEY\_QUERY\_VALUE,CmpKeyObjectType,mode,(PVOID \*)(&KeyBody),NULL);

LocalValueName = \*ValueName;

if (NT\_SUCCESS(status))

{

switch(VersionIndex)

{

case WIN2000:

case WINXP:

case WINXPSP3:

CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr(KeyBody->KeyControlBlock,LocalValueName,

KeyValueInformationClass,KeyValueInformation,Length,ResultLength);

break;

case WINVISTA:

case WIN7:

case WIN7\_1:

CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr(KeyBody,KeyValueInformationClass,KeyValueInformation

Length,ResultLength,LocalValueName);

break;

case WIN8:

CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr(KeyBody,LocalValueName,KeyValueInformationClass,

KeyValueInformation,Length,ResultLength);

break;

case WIN8\_1:

case WIN10:

CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr(KeyBody,KeyValueInformationClass,KeyValueInformation,

Length,ResultLength,LocalValueName);

break;

default:

status = STATUS\_NOT\_SUPPORTED;

break;

}

}

}

else

{

status = STATUS\_NOT\_SUPPORTED;

}

if(KeyBody)

ObDereferenceObject(KeyBody);

return status;

}

## 2.7 NtSetValueKeyEx

NTSTATUS \_\_stdcall NtSetValueKeyEx(HANDLE KeyHandle,PUNICODE\_STRING ValueName,ULONG Type,PVOID Data,ULONG DataSize)

{

NTSTATUS status;

PCM\_KEY\_BODY KeyBody = NULL;

KPROCESSOR\_MODE mode;

UNICODE\_STRING LocalValueName = {0};

PWSTR CapturedName=NULL;

OBJECT\_HANDLE\_INFORMATION HandleInformation = {0};

if(CmMatchData[VersionIndex][ECmQueryValueKey].InitFlag && CmMatchData[VersionIndex][ECmQueryValueKey].FuncAddr)

{

mode = KeGetPreviousMode();

status = ObReferenceObjectByHandle(KeyHandle,KEY\_SET\_VALUE,CmpKeyObjectType,mode,(PVOID \*)(&KeyBody),&HandleInformation);

LocalValueName = \*ValueName;

if (NT\_SUCCESS(status))

{

switch(VersionIndex)

{

switch(VersionIndex)

{

case WIN2000:

case WINXP:

case WINXPSP3:

CmMatchData[VersionIndex][ECmSetValueKey].FuncAddr(KeyBody->KeyControlBlock,&LocalValueName,Type,Data,DataSize);

break;

case WINVISTA:

case WIN7:

case WIN7\_1:

case WIN8:

case WIN8\_1:

CmMatchData[VersionIndex][ECmSetValueKey].FuncAddr(KeyBody,&LocalValueName,Type,Data,DataSize,0,

HandleInformation.HandleAttributes & 4);

break;

case WIN10:

CmMatchData[VersionIndex][ECmSetValueKey].FuncAddr(KeyBody,&LocalValueName,Type,Data,DataSize,0,

HandleInformation.HandleAttributes & 4);

break;

default:

status = STATUS\_NOT\_SUPPORTED;

break;

}

}

}

else

{

status = STATUS\_NOT\_SUPPORTED;

}

if(KeyBody)

ObDereferenceObject(KeyBody);

return status;

}

## 2.8 NtDeleteValueKey

NTSTATUS \_\_stdcall NtDeleteValueKey(HANDLE KeyHandle,PUNICODE\_STRING ValueName)

{

NTSTATUS status;

PCM\_KEY\_BODY KeyBody = NULL;

KPROCESSOR\_MODE mode;

UNICODE\_STRING LocalValueName = {0};

PWSTR CapturedName=NULL;

OBJECT\_HANDLE\_INFORMATION HandleInformation = {0};

if(CmMatchData[VersionIndex][ECmDeleteValueKey].InitFlag && CmMatchData[VersionIndex][ECmDeleteValueKey].FuncAddr)

{

mode = KeGetPreviousMode();

status = ObReferenceObjectByHandle(KeyHandle,KEY\_SET\_VALUE,CmpKeyObjectType,mode,(PVOID \*)(&KeyBody),&HandleInformation);

LocalValueName = \*ValueName;

if (NT\_SUCCESS(status))

{

switch(VersionIndex)

{

case WIN2000:

case WINXP:

case WINXPSP3:

CmMatchData[VersionIndex][ECmDeleteValueKey].FuncAddr(KeyBody->KeyControlBlock,LocalValueName);

break;

case WINVISTA:

case WIN7:

case WIN7\_1:

CmMatchData[VersionIndex][ECmDeleteValueKey].FuncAddr(KeyBody,KeyHandle,HandleInformation.HandleAttributes & 4,LocalValueName);

break;

case WIN8:

CmMatchData[VersionIndex][ECmDeleteValueKey].FuncAddr(KeyBody,LocalValueName,KeyHandle,HandleInformation.HandleAttributes & 4);

break;

case WIN8\_1:

case WIN10:

CmMatchData[VersionIndex][ECmDeleteValueKey].FuncAddr(KeyBody,&LocalValueName,Type,Data,DataSize,0,

HandleInformation.HandleAttributes & 4);

break;

default:

status = STATUS\_NOT\_SUPPORTED;

break;

}

}

}

else

{

status = STATUS\_NOT\_SUPPORTED;

}

if(KeyBody)

ObDereferenceObject(KeyBody);

return status;

}

## 2.9 NtDeleteKey

NTSTATUS \_\_stdcall NtDeleteKey(HANDLE KeyHandle)

{

NTSTATUS status;

PCM\_KEY\_BODY KeyBody = NULL;

KPROCESSOR\_MODE mode;

UNICODE\_STRING LocalValueName = {0};

PWSTR CapturedName=NULL;

OBJECT\_HANDLE\_INFORMATION HandleInformation = {0};

if(CmMatchData[VersionIndex][ECmDeleteKey].InitFlag && CmMatchData[VersionIndex][ECmDeleteKey].FuncAddr)

{

mode = KeGetPreviousMode();

status = ObReferenceObjectByHandle(KeyHandle,KEY\_SET\_VALUE,CmpKeyObjectType,mode,(PVOID \*)(&KeyBody),&HandleInformation);

LocalValueName = \*ValueName;

if (NT\_SUCCESS(status))

{

switch(VersionIndex)

{

case WIN2000:

case WINXP:

case WINXPSP3:

case WINVISTA:

case WIN7:

case WIN7\_1:

case WIN8:

case WIN8\_1:

case WIN10:

CmMatchData[VersionIndex][ECmDeleteKey].FuncAddr(KeyBody);

break;

default:

status = STATUS\_NOT\_SUPPORTED;

break;

}

}

}

else

{

status = STATUS\_NOT\_SUPPORTED;

}

if(KeyBody)

ObDereferenceObject(KeyBody);

return status;

}

## 2.10 IopCreateFile

代码基本和WRK一致，区别在于2点：

OPEN\_PACKET结构不同

ObOpenObjectByName调用之前会将IoFileObjectType的ObjectInitailier重置为系统初始值

// WIN2000

// WINXP

// WINXPSP3

#pragma pack(push)

#pragma pack(8)

typedef struct \_OPEN\_PACKET\_XP

{

CSHORT Type;

CSHORT Size;

PFILE\_OBJECT FileObject;

NTSTATUS FinalStatus;

ULONG\_PTR Information;

ULONG ParseCheck;

PFILE\_OBJECT RelatedFileObject;

LARGE\_INTEGER AllocationSize;

ULONG CreateOptions;

USHORT FileAttributes;

USHORT ShareAccess;

PVOID EaBuffer;

ULONG EaLength;

ULONG Options;

ULONG Disposition;

PFILE\_BASIC\_INFORMATION BasicInformation;

PFILE\_NETWORK\_OPEN\_INFORMATION NetworkInformation;

CREATE\_FILE\_TYPE CreateFileType;

PVOID ExtraCreateParameters;

BOOLEAN Override;

BOOLEAN QueryOnly;

BOOLEAN DeleteOnly;

BOOLEAN FullAttributes;

PDUMMY\_FILE\_OBJECT LocalFileObject;

BOOLEAN TraversedMountPoint;

ULONG InternalFlags;

PDEVICE\_OBJECT TopDeviceObjectHint;

} OPEN\_PACKET\_XP, \*POPEN\_PACKET\_XP;

#pragma pack(pop)

// WINVISTA

// WIN7

// WIN7\_1

#pragma pack(push)

#pragma pack(8)

typedef struct \_OPEN\_PACKET\_WIN7

{

CSHORT Type;

CSHORT Size;

PFILE\_OBJECT FileObject;

NTSTATUS FinalStatus;

ULONG\_PTR Information;

ULONG ParseCheck;

PFILE\_OBJECT RelatedFileObject;

POBJECT\_ATTRIBUTES OriginalAttributes;

LARGE\_INTEGER AllocationSize;

ULONG CreateOptions;

USHORT FileAttributes;

USHORT ShareAccess;

PVOID EaBuffer;

ULONG EaLength;

ULONG Options;

ULONG Disposition;

PFILE\_BASIC\_INFORMATION BasicInformation;

PFILE\_NETWORK\_OPEN\_INFORMATION NetworkInformation;

CREATE\_FILE\_TYPE CreateFileType;

PVOID MailslotOrPipeParameters;

BOOLEAN Override;

BOOLEAN QueryOnly;

BOOLEAN DeleteOnly;

BOOLEAN FullAttributes;

PDUMMY\_FILE\_OBJECT LocalFileObject;

ULONG InternalFlags;

IO\_DRIVER\_CREATE\_CONTEXT DriverCreateContext;

} OPEN\_PACKET\_WIN7, \*POPEN\_PACKET\_WIN7;

#pragma pack(pop)

// WIN8

// WIN8\_1

// WIN10

#pragma pack(push)

#pragma pack(8)

typedef struct \_OPEN\_PACKET\_WIN8

{

CSHORT Type;

CSHORT Size;

PFILE\_OBJECT FileObject;

NTSTATUS FinalStatus;

ULONG\_PTR Information;

ULONG ParseCheck;

union

{

PFILE\_OBJECT RelatedFileObject;

PDEVICE\_OBJECT ReferencedDeviceObject;

};

POBJECT\_ATTRIBUTES OriginalAttributes;

LARGE\_INTEGER AllocationSize;

ULONG CreateOptions;

USHORT FileAttributes;

USHORT ShareAccess;

PVOID EaBuffer;

ULONG EaLength;

ULONG Options;

ULONG Disposition;

PFILE\_BASIC\_INFORMATION BasicInformation;

PFILE\_NETWORK\_OPEN\_INFORMATION NetworkInformation;

CREATE\_FILE\_TYPE CreateFileType;

PVOID MailslotOrPipeParameters;

BOOLEAN Override;

BOOLEAN QueryOnly;

BOOLEAN DeleteOnly;

BOOLEAN FullAttributes;

PDUMMY\_FILE\_OBJECT LocalFileObject;

ULONG InternalFlags;

KPROCESSOR\_MODE AccessMode;

IO\_DRIVER\_CREATE\_CONTEXT DriverCreateContext;

} OPEN\_PACKET\_WIN8, \*POPEN\_PACKET\_WIN8;

#pragma pack(pop)

# 三、控制码

IRP\_MJ\_DEVICE\_CONTROL => 调用DeviceIoControl派遣(DeviceIoControlDispatch)

能力：

解锁文件

驱动加载

句柄操作

进程打开、结束

注册表穿透操作

## 3.1 TSSysKit x86 IoControlCode对应表

0x221C00

解锁文件

buffer= sizeof=0x804

+00 NTSTATUS status out

+04 WCHAR FileName[1024] in

0x221C04

0x221C08

0x221C0C

0x221C10

0x221C14

尚未实现

0x222004

普通结束进程

buffer= sizeof=4

+00 DWORD ProcessId

0x222008

穿透NtDeleteKey

buffer= sizeof=8

+00 NTSTATUS status out

+04 HANDLE KeyHandle in

0x22200C

穿透NtDeleteValueKey 成员含义见NtDeleteValueKey

buffer= sizeof=0xC

+00 NTSTATUS status out

+04 HANDLE KeyHandle in

+08 PUNICODE\_STRING ValueName in

0x222010

通过进程id或进程对象名(只能选一)穿透打开进程NtOpenProcess 成员含义见NtOpenProcess

buffer= sizeof=0x18

+00 NTSTATUS status out

+04 HANDLE ProcessHandle out

+08 ACCESS\_MASK DesiredAccess in

+0C POBJECT\_ATTRIBUTES ObjectAttributes in

+10 PCLIENT\_ID ClientId

0x222404

普通关闭句柄NtClose

buffer= sizeof=8

+00 NTSTATUS status out

+04 HANDLE Handle in

0x222408

穿透创建注册表项NtCreateKey 成员含义见NtCreateKey

buffer= sizeof=0x20

+00 HANDLE KeyHandle out

+04 NTSTATUS status out 注意status不是第一成员了！！

+08 ULONG Disposition out

+0C ACCESS\_MASK DesiredAccess in

+10 POBJECT\_ATTRIBUTES ObjectAttributes in

+14 ULONG TitleIndex in

+18 PUNICODE\_STRING Class in

+1C ULONG CreateOptions in

0x22240C

穿透打开注册表项NtOpenKey 成员含义见NtOpenKey

buffer= sizeof=0x10

+00 HANDLE KeyHandle out

+04 NTSTATUS status out 注意status不是第一成员了！！

+08 ACCESS\_MASK DesiredAccess in

+0C POBJECT\_ATTRIBUTES ObjectAttributes in

0x222410

同0x222008 NtDeleteKey

buffer= sizeof=8

+00 NTSTATUS status

+04 HANDLE KeyHandle

0x222414

穿透删除注册表项NtDeleteValueKey 成员含义见NtDeleteValueKey

buffer= sizeof=0xC

+00 NTSTATUS status out

+04 HANDLE KeyHandle in

+08 PUNICODE\_STRING ValueName in

0x222418

穿透设置注册表项NtSetValueKeyEx 成员含义见NtSetValueKey

buffer= sizeof=0x1C

+00 NTSTATUS status out

+04 HANDLE KeyHandle in

+08 PUNICODE\_STRING ValueName in

+0C ULONG TitleIndex 未使用

+10 ULONG Type in

+14 PVOID Data in

+18 ULONG DataSize in

0x22241C

穿透设置注册表项NtQueryValueKey 成员含义见NtQueryValueKey

buffer= sizeof=0x1C

+00 DWORD ResultLength out

+04 NTSTATUS status out 注意status不是第一成员了！！

+08 HANDLE KeyHandle in

+0C PUNICODE\_STRING ValueName

+10 ULONG Type out KeyValueInformation->DataLength

+14 PVOID Data out KeyValueInformation->Data

+18 ULONG DataLength out KeyValueInformation->DataLength

0x222420

穿透枚举注册表项NtEnumerateKey 成员含义见NtEnumerateKey

buffer= sizeof=0x1C

+00 NTSTATUS status out

+04 DWORD ResultLength out

+08 HANDLE KeyHandle in

+0C ULONG Index in

+10 KEY\_INFORMATION\_CLASS KeyInformationClass in

+14 PVOID KeyInformation in

+18 ULONG Length

0x222424

NtEnumerateValueKey 成员含义见NtEnumerateValueKey

buffer= sizeof=0x1C

+00 NTSTATUS status out

+04 ULONG ResultLength out

+08 HANDLE KeyHandle in

+0C ULONG Index in

+10 KEY\_VALUE\_INFORMATION\_CLASS KeyValueInformationClass in

+14 PVOID KeyValueInformation out

+18 ULONG Length in

0x222428

TsSysKit驱动是否初始化

\*(DWORD\*)buffer => SomeFlag

\*(DWORD\*)buffer <= IsTsSysKitInit

0x22242C

穿透创建服务加载驱动

buffer= sizeof=0x91C

+000 WCHAR ImagePath[260] 驱动文件路径

+208 DWORD Type 驱动注册表Type项

+20C DWORD Start 驱动注册表项Start类型

+210 DWORD flag (决定是否设置注册表Tag和Group信息)

+214 ？？？

+468 DWORD Tag 驱动注册表Tag项

+46C WCHAR DisplayName[300] 驱动注册表项DisplayName

+6C4 WCHAR ServiceName[300] 驱动服务名

0x222430

获取操作系统版本

buffer=RTL\_OSVERSIONINFOEXW sizeof=0x11C

0x222800

0x222804

2个初始化TSSysKit的通道

0x224008

+00 DWORD Tag=0x20120502

+04 DWORD =0

0x22400C

穿透创建文件

Buffer= sizeof=0x30 具体参数含义见NtCreateFile

+00 NTSTATUS status out

+04 PHANDLE FileHandle

+08 ACCESS\_MASK DesiredAccess

+0C POBJECT\_ATTRIBUTES ObjectAttributes

+10 PIO\_STATUS\_BLOCK IoStatusBloc

+14 PLARGE\_INTEGER AllocationSize

+18 ULONG FileAttributes

+1C ULONG ShareAccess

+20 ULONG CreateDisposition

+24 ULONG CreateOptions

+28 PVOID EaBuffer

+2C ULONG EaLength

0x224010

穿透打开文件

Buffer= sizeof=0x1C 具体参数含义见NtOpenFile

+00 NTSTATUS status out

+04 PHANDLE FileHandle

+08 PHANDLE FileHandle

+0C POBJECT\_ATTRIBUTES ObjectAttributes

+10 PIO\_STATUS\_BLOCK IoStatusBlock

+14 ULONG ShareAccess

+18 ULONG OpenOptions

0x224014

穿透读取文件

Buffer= sizeof=0x28 具体参数含义见NtReadFile

+00 NTSTATUS status out

+04 HANDLE FileHandle in

+08 HANDLE Event in

+0C PIO\_APC\_ROUTINE ApcRoutine in

+10 PVOID ApcContext in

+14 PIO\_STATUS\_BLOCK IoStatusBlock in

+18 PVOID Buffer out

+1C ULONG Length in

+20 PLARGE\_INTEGER ByteOffset

+24 PULONG Key

0x224018

穿透写入文件

Buffer= sizeof=0x28 具体参数含义见NtWriteFile

+00 NTSTATUS status out

+04 HANDLE FileHandle in

+08 HANDLE Event in

+0C PIO\_APC\_ROUTINE ApcRoutine in

+10 PVOID ApcContext in

+14 PIO\_STATUS\_BLOCK IoStatusBlock in

+18 PVOID Buffer out

+1C ULONG Length in

+20 PLARGE\_INTEGER ByteOffset

+24 PULONG Key

0x22401C

普通关闭句柄NtClose

buffer= sizeof=8

+00 NTSTATUS status out

+04 HANDLE Handle in

0x224020

穿透设置文件

Buffer= sizeof=0x1C 具体参数含义见NtSetInformationFile

+00 NTSTATUS status;

+04 HANDLE FileHandle in

+08 PIO\_STATUS\_BLOCK IoStatus out

+0C PVOID FileInformation in

+10 ULONG Length in

+14 FILE\_INFORMATION\_CLASS FileInformationClass in

+18 BOOL DelCurrentFile in

0x224024

穿透查询文件

Buffer= sizeof=0x1C 具体参数含义见NtQueryInformationFile

+00 NTSTATUS status out

+04 HANDLE FileHandle in

+08 PIO\_STATUS\_BLOCK IoStatus out

+0C PVOID FileInformation in

+10 ULONG Length in

+14 FILE\_INFORMATION\_CLASS FileInformationClass in

+18 BOOL DelCurrentFile in

0x224028

尚未实现

0x22402C

穿透查询目录

Buffer= sizeof=0x30 具体参数含义见NtQueryDirectoryFile

+00 NTSTATUS status out

+04 HANDLE FileHandle in

+08 HANDLE Event in

+0C PIO\_APC\_ROUTINE ApcRoutine 未使用

+10 PVOID ApcContext 未使用

+14 PIO\_STATUS\_BLOCK IoStatus out

+18 PVOID FileInformation in

+1C ULONG Length in

+20 FILE\_INFORMATION\_CLASS FileInformationClass in

+24 BOOLEAN ReturnSingleEntry in

+28 PUNICODE\_STRING FileName in

+2C BOOLEAN RestartScan in

0x228404

穿透查询文件属性

Buffer= sizeof=0xC 具体参数含义见NtQueryAttributesFile

+0 NTSTATUS status out

+4 POBJECT\_ATTRIBUTES ObjectAttributes in 路径前缀匹配\??\c:

+8 FILE\_NETWORK\_OPEN\_INFORMATION networkInformation out

### 0x221C00解锁文件

见3.13 解锁文件

### 0x222004普通结束进程

BOOLEAN TerminateProcessById(HANDLE ProcessId)

{

BOOLEAN Result = FALSE;

PEPROCESS Process = NULL;

HANDLE ProcessHandle = NULL;

if(NT\_SUCCESS(PsLookupProcessByProcessId(ProcessId,&Process)) &&

NT\_SUCCESS(ObOpenObjectByPointer(Process,0,NULL,PROCESS\_ALL\_ACCESS,NULL,KernelMode,&ProcessHandle)) &&

NT\_SUCCESS(ZwTerminateProcess(ProcessHandle,0)))

{

Result = TRUE;

}

if(Process)

{

ObDereferenceObject(Process);

Process = NULL;

}

if(ProcessHandle)

ZwClose(ProcessHandle);

return Result;

}

### 0x22242C穿透创建服务加载驱动

struct LOADDRIVERSTRUCT

{

WCHAR ImagePath[260]; //驱动文件路径

DWORD Type; //驱动注册表Type项

DWORD Start; //驱动注册表项Start类型

WCHAR Group[300];//驱动注册表Group名

DWORD Tag; //驱动注册表Tag项

WCHAR DisplayName[300]; //驱动注册表项DisplayName

WCHAR ServiceName[300]; //驱动服务名

};

#define MakeUnicodeString(X) {sizeof(X),sizeof(X)+2,X}

UNICODE\_STRING UImagePath=MakeUnicodeString(L"ImagePath");

UNICODE\_STRING UType=MakeUnicodeString(L"Type");

UNICODE\_STRING UStart=MakeUnicodeString(L"Start");

UNICODE\_STRING UGroup=MakeUnicodeString(L"Group");

UNICODE\_STRING UDisplayName=MakeUnicodeString(L"DisplayName");

UNICODE\_STRING UErrorControl=MakeUnicodeString(L"ErrorControl");

UNICODE\_STRING UTag=MakeUnicodeString(L"Tag");

UNICODE\_STRING UZwLoadDriver=MakeUnicodeString(L"ZwLoadDriver");

struct LOADDRIVERPARAM

{

WORK\_QUEUE\_ITEM WorkItem;

KEVENT Event;

ULONG mem1;

PUNICODE\_STRING DriverServiceName;

NTSTATUS Status;

};

void LoadDriverWorker(LOADDRIVERPARAM\* WorkItem)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL,outstatus;

HANDLE KeyHandle = NULL;

OBJECT\_ATTRIBUTES Oa;

if(WorkItem)

{

InitializeObjectAttributes(&Oa,WorkItem->DriverServiceName,OBJ\_CASE\_INSENSITIVE|OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = NtOpenKey(&KeyHandle,KEY\_READ,&Oa);//穿透

if(NT\_SUCCESS(status))

{//xp win7的IopLoadDriver 为不同的调用方式

if(VersionInfo < WINVISTA)

{//NTSTATUS \_\_stdcall IopLoadDriver(HANDLE KeyHandle, BOOLEAN CheckForSafeBoot, BOOLEAN IsFilter, NTSTATUS \*DriverEntryStatus)

IopLoadDriver(KeyHandle,HANDLE\_FLAG\_INHERIT,FALSE,&outstatus);

if(status == STATUS\_FAILED\_DRIVER\_ENTRY)

status = outstatus;

else if(status == STATUS\_DRIVER\_FAILED\_PRIOR\_UNLOAD)

status = STATUS\_OBJECT\_NAME\_NOT\_FOUND;

}

else

{//NTSTATUS \_\_userpurge IopLoadDriver<eax>(HANDLE KeyHandle<ecx>, BOOLEAN CheckForSafeBoot, BOOLEAN IsFilter, NTSTATUS \*DriverEntryStatus) 第一参用ecx传值

status = IopLoadDriver(KeyHandle,HANDLE\_FLAG\_INHERIT,FALSE,&outstatus);////事先获取的函数指针，见1.1

}

}

}

WorkItem->Status = status;

KeSetEvent(WorkItem->Event,0,FALSE);

}

NTSTATUS LoadDriverEx(PUNICODE\_STRING DriverServiceName)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL;

LOADDRIVERPARAM LoadDriver;

if(IopLoadDriver)//事先获取的函数指针，见1.1

{

LoadDriver.DriverServiceName = DriverServiceName;

LoadDriver.mem1 = 0;

KeInitializeEvent(&LoadDriver.Event,NotificationEvent,FALSE);

ExInitializeWorkItem(&LoadDriver,LoadDriverWorker,&LoadDriver);

ExQueueWorkItem(&LoadDriver.WorkItem,DelayedWorkQueue);

KeWaitForSingleObject(&LoadDriver.Event,UserRequest,KernelMode,FALSE,NULL);

return LoadDriver.Status;

}

else

{

FARPROC ZwLoadDriver = MmGetSystemRoutineAddress(&UZwLoadDriver);

if(ZwLoadDriver)

return ZwLoadDriver(DriverServiceName);

}

return status;

}

NTSTATUS CreateServiceAndLoadDriver(DWORD InLen,LOADDRIVERSTRUCT\* Data)

{//InLen = IrpSp->Parameters.DeviceIoControl.InputBufferLength

NTSTATUS status = STATUS\_UNSUCCESSFUL;

UNICODE\_STRING DriverServicePath;

UNICODE\_STRING ServiceName;

OBJECT\_ATTRIBUTES Oa;

WCHAR\* Buf = NULL;

HANDLE KeyHandle = NULL;

const int BufLen = 520;

ULONG ErrorControl = SERVICE\_ERROR\_NORMAL;

ULONG Disposition = REG\_OPENED\_EXISTING\_KEY;

if(!SeSinglePrivilegeCheck(SE\_LOAD\_DRIVER\_PRIVILEGE,UserMode))

return STATUS\_PRIVILEGE\_NOT\_HELD;

if(VersionInfo < WINXP)

return STATUS\_NOT\_SUPPORTED;

if((Data->Type & SERVICE\_DRIVER) && Data->ServiceName && Data->ImagePath && Data->DisplayName)

{

RtlInitUnicodeString(&ServiceName,Data->ServiceName);

Buf = (WCHAR\*)ExAllocatePool(NonPagedPool,BufLen);

RtlZeroMemory(Buf,BufLen);

wcscpy(Buf,L"\\Registry\\Machine\\System\\CurrentControlSet\\Services\\");

DriverServicePath.Length = 2\*wcslen(Buf);

DriverServicePath.MaximumLength = BufLen;

DriverServicePath.Buffer = Buf;

status = RtlAppendUnicodeStringToString(&DriverServicePath, &ServiceName);

if(NT\_SUCCESS(status))

{

InitializeObjectAttributes(&Oa,&DriverServicePath,OBJ\_CASE\_INSENSITIVE,NULL,NULL);

status = ZwCreateKey(&KeyHandle,KEY\_READ|KEY\_SET\_VALUE, &Oa, 0,NULL, 0, &Disposition);//穿透

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UImagePath,REG\_SZ,Data->ImagePath,2\*wcslen(Data->ImagePath)+2);//穿透

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UType,REG\_DWORD,Data->Type,sizeof(DWORD));

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UStart,REG\_DWORD,Data->Start,sizeof(DWORD));

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UDisplayName,REG\_SZ,Data->DisplayName,2\*wcslen(Data->DisplayName)+2);

}

if(NT\_SUCCESS(status))

{//此处q管源码有bug

status = ZwSetValueKeyEx(KeyHandle,&UGroup,REG\_SZ,Data->Group,2\*wcslen(Data->Group)+2);

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UTag,REG\_DWORD,Data->Tag,sizeof(DWORD));

}

if(NT\_SUCCESS(status))

{

status = ZwSetValueKeyEx(KeyHandle,&UErrorControl,REG\_DWORD,ErrorControl,sizeof(DWORD));

}

if(NT\_SUCCESS(status))

{

ZwFlushKey(KeyHandle);

status = LoadDriverEx(&DriverServicePath);

}

}

if(KeyHandle)

{

ZwClose(KeyHandle);

KeyHandle = NULL;

}

if(Buf)

ExFreePool(Buf);

}

## 3.2 TSSysKit x64 IoControlCode对应表

0x22200C

穿透创建文件

Buffer= sizeof=0x50 具体参数含义见NtCreateFile

+00 NTSTATUS status out

+08 PHANDLE FileHandle

+10 ACCESS\_MASK DesiredAccess

+18 POBJECT\_ATTRIBUTES ObjectAttributes

+20 PIO\_STATUS\_BLOCK IoStatusBloc

+28 PLARGE\_INTEGER AllocationSize

+30 ULONG FileAttributes

+34 ULONG ShareAccess

+38 ULONG CreateDisposition

+3C ULONG CreateOptions

+40 PVOID EaBuffer

+48 ULONG EaLength

0x222010

穿透打开文件

Buffer= sizeof=0x30 具体参数含义见NtOpenFile

+00 NTSTATUS status out

+08 PHANDLE FileHandle

+10 PHANDLE FileHandle

+18 POBJECT\_ATTRIBUTES ObjectAttributes

+20 PIO\_STATUS\_BLOCK IoStatusBlock

+28 ULONG ShareAccess

+2C ULONG OpenOptions

0x222014

穿透读取文件

Buffer= sizeof=0x50 具体参数含义见NtReadFile

+00 NTSTATUS status out

+08 HANDLE FileHandle in

+10 HANDLE Event in

+18 PIO\_APC\_ROUTINE ApcRoutine in

+20 PVOID ApcContext in

+28 PIO\_STATUS\_BLOCK IoStatusBlock in

+30 PVOID Buffer out

+38 ULONG Length in

+40 PLARGE\_INTEGER ByteOffset

+48 PULONG Key

0x222018

穿透写入文件

Buffer= sizeof=0x50 具体参数含义见NtWriteFile

+00 NTSTATUS status out

+08 HANDLE FileHandle in

+10 HANDLE Event in

+18 PIO\_APC\_ROUTINE ApcRoutine in

+20 PVOID ApcContext in

+28 PIO\_STATUS\_BLOCK IoStatusBlock in

+30 PVOID Buffer out

+38 ULONG Length in

+40 PLARGE\_INTEGER ByteOffset

+48 PULONG Key

0x22201C

普通关闭句柄NtClose

buffer= sizeof=0x10

+00 NTSTATUS status out

+04 HANDLE Handle in

0x222020

穿透设置文件

Buffer= sizeof=0x30 具体参数含义见NtSetInformationFile

+00 NTSTATUS status;

+08 HANDLE FileHandle in

+10 PIO\_STATUS\_BLOCK IoStatus out

+18 PVOID FileInformation in

+20 ULONG Length in

+24 FILE\_INFORMATION\_CLASS FileInformationClass in

+28 BOOL DelCurrentFile in

0x222024

穿透查询文件

Buffer= sizeof=0x30 具体参数含义见NtQueryInformationFile

+00 NTSTATUS status out

+08 HANDLE FileHandle in

+10 PIO\_STATUS\_BLOCK IoStatus out

+18 PVOID FileInformation in

+20 ULONG Length in

+24 FILE\_INFORMATION\_CLASS FileInformationClass in

+28 BOOL DelCurrentFile in

0x222028

尚未实现

0x22202C

穿透查询目录

Buffer= sizeof=0x58 具体参数含义见NtQueryDirectoryFile

+00 NTSTATUS status out

+08 HANDLE FileHandle in

+10 HANDLE Event in

+18 PIO\_APC\_ROUTINE ApcRoutine 未使用

+20 PVOID ApcContext 未使用

+28 PIO\_STATUS\_BLOCK IoStatus out

+30 PVOID FileInformation in

+38 ULONG Length in

+3C FILE\_INFORMATION\_CLASS FileInformationClass in

+40 BOOLEAN ReturnSingleEntry in

+48 PUNICODE\_STRING FileName in

+50 BOOLEAN RestartScan in

0x222030

获取内部版本号

buffer= sizeof=8

+0 <= 0x20110929i64

0x222034

穿透查询文件属性

Buffer= sizeof=0x18 具体参数含义见NtQueryAttributesFile

+00 NTSTATUS status out

+08 POBJECT\_ATTRIBUTES ObjectAttributes in 路径前缀匹配\??\c:

+10 FILE\_NETWORK\_OPEN\_INFORMATION networkInformation out

0x222038

解锁文件

buffer= sizeof=0x804

+00 NTSTATUS status out

+04 WCHAR FileName[1024] in

0x222144

普通关闭句柄NtClose

buffer= sizeof=0x10

+00 NTSTATUS status out

+08 HANDLE Handle in

0x222148

穿透创建注册表项NtCreateKey 成员含义见NtCreateKey

buffer= sizeof=0x20

+00 HANDLE KeyHandle out

+04 NTSTATUS status out 注意status不是第一成员了！！

+08 ULONG Disposition out

+0C ACCESS\_MASK DesiredAccess in

+10 POBJECT\_ATTRIBUTES ObjectAttributes in

+14 ULONG TitleIndex in

+18 PUNICODE\_STRING Class in

+1C ULONG CreateOptions in

0x22214C

穿透打开注册表项NtOpenKey 成员含义见NtOpenKey

buffer= sizeof=0x20

+00 HANDLE KeyHandle out

+08 NTSTATUS status out 注意status不是第一成员了！！

+10 ACCESS\_MASK DesiredAccess in

+18 POBJECT\_ATTRIBUTES ObjectAttributes in

0x222150

穿透删除注册表项NtDeleteKey 成员含义见NtDeleteKey

buffer= sizeof=0x10

+00 NTSTATUS status

+08 HANDLE KeyHandle

0x222154

穿透删除注册表项NtDeleteValueKey 成员含义见NtDeleteValueKey

buffer= sizeof=0x18

+00 NTSTATUS status out

+08 HANDLE KeyHandle in

+10 PUNICODE\_STRING ValueName in

0x222158

穿透设置注册表项NtSetValueKeyEx 成员含义见NtSetValueKey

buffer= sizeof=0x30

+00 NTSTATUS status out

+08 HANDLE KeyHandle in

+10 PUNICODE\_STRING ValueName in

+18 ULONG TitleIndex 未使用

+1C ULONG Type in

+20 PVOID Data in

+28 ULONG DataSize in

0x22215C

穿透设置注册表项NtQueryValueKey 成员含义见NtQueryValueKey

buffer= sizeof=0x30

+00 DWORD ResultLength out

+04 NTSTATUS status out 注意status不是第一成员了！！

+08 HANDLE KeyHandle in

+10 PUNICODE\_STRING ValueName

+18 ULONG Type out KeyValueInformation->DataLength

+20 PVOID Data out KeyValueInformation->Data

+28 ULONG DataLength out KeyValueInformation->DataLength

0x222160

穿透枚举注册表项NtEnumerateKey 成员含义见NtEnumerateKey

buffer= sizeof=0x28

+00 NTSTATUS status out

+04 ULONG ResultLength out

+08 HANDLE KeyHandle in

+10 ULONG Index in

+14 KEY\_INFORMATION\_CLASS KeyValueInformationClass in

+18 PVOID KeyInformation out

+20 ULONG Length in

0x222164

穿透枚举注册表项NtEnumerateValueKey 成员含义见NtEnumerateValueKey

buffer= sizeof=0x28

+00 NTSTATUS status out

+04 ULONG ResultLength out

+08 HANDLE KeyHandle in

+10 ULONG Index in

+14 KEY\_VALUE\_INFORMATION\_CLASS KeyValueInformationClass in

+18 PVOID KeyValueInformation out

+20 ULONG Length in

0x222284

通过进程id或进程对象名(只能选一)穿透打开进程NtOpenProcess 成员含义见NtOpenProcess

buffer= sizeof=0x30

+00 NTSTATUS status out

+08 HANDLE ProcessHandle out

+10 ACCESS\_MASK DesiredAccess in

+18 POBJECT\_ATTRIBUTES ObjectAttributes in

+20 PCLIENT\_ID ClientId

0x222288

buffer= sizeof=4

+00 DWORD ProcessId

0x222430

获取操作系统版本

buffer=RTL\_OSVERSIONINFOEXW sizeof=0x11C

0x222800

0x222804

2个初始化TSSysKit的通道

0x222808

获取CPUID信息

buffer= sizeof=0x38

+00 ULONG Fn0000\_0000\_EBX EBX EDX ECX 组成"GenuineIntel"或"AuthenticAMD"

+04 ULONG Fn0000\_0000\_EDX

+08 ULONG Fn0000\_0000\_ECX

+10 ULONG CpuType 0:未知 1:INTEL 2:AMD

+14 BOOLEAN VMBit 是否支持vm Intel Virutalization Technology / AMD Secure Virtual Machine

+18 ULONGLONG MSR3A Intel IA32\_FEATURE\_CONTROL MSR(0x3A) / AMD Read NX support

+20 ULONGLONG NXsupport Intel CR4 / AMD Msr(0xC0000080)

+28 ULONGLONG VMXONBit Intel MSR(0x3A) activate VMXON outside of SMX mode / AMD Msr(0xC0010114)

+30 ULONG NRIP Fn8000\_000A\_EDX&4

# 四、默认派遣例程

* IRP\_MJ\_CREATE

检查当前所属进程是否有腾讯标记 (CheckDriverLoaderValid)

重置驱动注册表项信息 (ResetRegServiceInfo)

随机化EPROCESS的ImageFileName(RandomImageNameToHide)

## 4.1 根据进程id结束进程

void TerminateProcessById(HANDLE ProcessId)

{

PEPROCESS Process = NULL;

HANDLE ProcessHandle = NULL;

NTSTATUS status = PsLookupProcessByProcessId(ProcessId,&Process);

if(NT\_SUCCESS(status))

{

status = ObOpenObjectByPointer(Process,0,NULL,PROCESS\_ALL\_ACCESS,0,NULL,&ProcessHandle);

if(NT\_SUCCESS(status))

{

status = ZwTerminateProcess(ProcessHandle,0);

}

}

if(Process)

{

ObDereferenceObject(Process);

Process = NULL;

}

if(ProcessHandle)

ZwClose(ProcessHandle);

检测PE格式合法性

bool CheckNtImageValid(LPVOID ImageAddress)

{

if(ImageAddress && MmIsAddressValid(ImageAddress))

{

PIMAGE\_DOS\_HEADER DosHeader = (IMAGE\_DOS\_HEADER)ImageAddress;

if(MmIsAddressValid(&DosHeader->e\_lfanew) && DosHeader->e\_magic == 'ZM')

{

PIMAGE\_NT\_HEADERS NtHeader = (PIMAGE\_NT\_HEADERS)((BYTE\*)DosHeader + DosHeader->e\_lfanew);

if(NtHeader && MmIsAddressValid(NtHeader) && NtHeader->Signature == 'EP')

return true;

}

}

}

## 4.2 获取当前进程进程名

typedef ULONG DWORD;

typedef struct \_MEMORY\_BASIC\_INFORMATION

{

PVOID BaseAddress;

PVOID AllocationBase;

DWORD AllocationProtect;

SIZE\_T RegionSize;

DWORD State;

DWORD Protect;

DWORD Type;

} MEMORY\_BASIC\_INFORMATION, \*PMEMORY\_BASIC\_INFORMATION;

typedef struct \_MEMORY\_SECTION\_NAME

{

UNICODE\_STRING SectionFileName;

WCHAR NameBuffer[0];

} MEMORY\_SECTION\_NAME, \*PMEMORY\_SECTION\_NAME;

extern "C" PVOID \_\_stdcall PsGetProcessSectionBaseAddress(PEPROCESS Process);

extern "C" NTSTATUS \_\_stdcall ZwQueryVirtualMemory(HANDLE ProcessHandle,PVOID BaseAddress,MEMORY\_INFORMATION\_CLASS MemoryInformationClass,

PVOID MemoryInformation,SIZE\_T MemoryInformationLength,PSIZE\_T ReturnLength);

#define MEM\_IMAGE 0x1000000

bool GetCurrentProcessName(PVOID Buffer,SIZE\_T Length)

{

NTSTATUS status;

if(!Buffer || !Length)

return;

UNICODE\_STRING UIoVolumeDeviceToDosName;

PVOID ImageBase = PsGetProcessSectionBaseAddress(IoGetCurrentProcess());

PVOID SectionName = ExAllocatePool(NonPagedPool,Length + sizeof(MEMORY\_SECTION\_NAME));

if(!SectionName)

return;

if(ImageBase)

{

MEMORY\_BASIC\_INFORMATION BasicInfo;

status = ZwQueryVirtualMemory(NtCurrentProcess(),ImageBase,MemoryBasicInformation,&BasicInfo,sizeof(BasicInfo),NULL);

if(NT\_SUCCESS(status) && BasicInfo.Type == MEM\_IMAGE)

{

status = ZwQueryVirtualMemory(NtCurrentProcess(),ImageBase,MemorySectionName,SectionName,Length + sizeof(MEMORY\_SECTION\_NAME),NULL);

if(NT\_SUCCESS(status))

{

wcsncpy((WCHAR\*)Buffer,((PMEMORY\_SECTION\_NAME)SectionName)->SectionFileName.Buffer,Length);

return true;

}

}

}

return false;

}

## 4.3 由进程ID获取进程设备名

bool GetProcessNameById(HANDLE ProcessId,PVOID Buffer,SIZE\_T Length)

{

NTSTATUS status;

if(ProcessId == (HANDLE)4)

{

wcsncpy((WCHAR\*)Buffer,L"System",Length);

}

else if(ProcessId == PsGetCurrentProcessId())

{

GetCurrentProcessName(Buffer,Length);

}

else

{

PEPROCESS Process = NULL;

if(NT\_SUCCESS(PsLookupProcessByProcessId(ProcessId,&Process)))

{

KAPC\_STATE KApc;

KeStackAttachProcess(Process,&KApc);

GetCurrentProcessName(Buffer,Length);

KeUnstackDetachProcess(&KApc);

ObDereferenceObject(&Process);

}

}

return true;

}

## 4.4 设备名转DOS路径

NTSTATUS GetDeviceDosName(WCHAR\* DeviceName,WCHAR\* DosName,DWORD Len)

{

NTSTATUS status;

//检查设备路径

if(!DeviceName || !DosName)

return STATUS\_INVALID\_PARAMETER;

if(wcsnicmp(DeviceName, L"\\Device\\", 8u))

return STATUS\_INVALID\_PARAMETER\_1;

WCHAR\* ptr = wcsstr(DeviceName + 8,L"\\");

if(!ptr)

return STATUS\_UNSUCCESSFUL;

int len = ptr - DeviceName;

PVOID Buffer = ExAllocatePool(NonPagedPool,2\*len+2);

if(!Buffer)

return;

wcsncpy((WCHAR\*)Buffer,DeviceName,len);

//根据设备名获取设备对象

PDEVICE\_OBJECT DeviceObject;

UNICODE\_STRING UDeviceName;

PFILE\_OBJECT FileObject;

RtlInitUnicodeString(&UDeviceName,(WCHAR\*)Buffer);

//GetDeviceObjectByName

status = IoGetDeviceObjectPointer(&UDeviceName,0,&FileObject,&DeviceObject);

if(NT\_SUCCESS(status))

{

if(DeviceObject->Type == FILE\_DEVICE\_DISK)

{

UNICODE\_STRING RootDeviceDosName;

status = IoVolumeDeviceToDosName(DeviceObject,&RootDeviceDosName);

if(NT\_SUCCESS(status))

{

wcsncpy(DosName,RootDeviceDosName.Buffer,RootDeviceDosName.Length);

ExFreePool(RootDeviceDosName.Buffer);

int len2 = wcslen((WCHAR\*)Buffer);//拼接全路径

wcsncat(DosName,DeviceName+len2,Len);

}

}

else if(DeviceObject->Type == FILE\_DEVICE\_NETWORK\_FILE\_SYSTEM)

{

wcsncpy(DosName,L"\\",Len);

int len2 = wcslen((WCHAR\*)Buffer);

wcsncat(DosName,DeviceName+len2,Len);

}

else

{

status = STATUS\_DEVICE\_DATA\_ERROR;

}

ObReferenceObject(FileObject);

ObReferenceObject(DeviceObject);

}

ExFreePool(Buffer);

}

## 4.5 得到EPROCESS对应ImageDosPath

void GetProcessDosPathByObject(PEPROCESS Process,LPVOID Buffer,ULONG Len)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL;

HANDLE ProcessHandle = NULL;

HANDLE FileHandle = NULL;

const int FileBufSize = 4096;

PUNICODE\_STRING pFilePath = ExAllocatePoolWithTag(NonPagedPool,FileBufSize);

if(!pFilePath)

return;

status = ObOpenObjectByPointer(Process,OBJ\_KERNEL\_HANDLE,NULL,0,NULL,KernelMode,&ProcessHandle);

if(NT\_SUCCESS(status))

{

status = NtQueryInformationProcess(ProcessHandle,ProcessImageFileName,pFilePath,FileBufSize,NULL);

if(NT\_SUCCESS(status) && MmIsAddressValid(pFilePath->Buffer))

{

OBJECT\_ATTRIBUTES oa;

IO\_STATUS\_BLOCK IoStatusBlock;

InitializeObjectAttributes(&oa,pFilePath,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = IoCreateFile(&FileHandle,GENERIC\_READ | SYNCHRONIZE,&oa,&IoStatusBlock,NULL,FILE\_ATTRIBUTE\_NORMAL,

FILE\_SHARE\_READ,FILE\_OPEN,FILE\_NON\_DIRECTORY\_FILE | FILE\_SYNCHRONOUS\_IO\_NONALERT,NULL,0,CreateFileTypeNone,

NULL,IO\_NO\_PARAMETER\_CHECKING);

if(NT\_SUCCESS(status))

{// GetProcessDosPathByHandle

OBJECT\_HANDLE\_INFORMATION HandleInformation;

PFILE\_OBJECT FileObject = NULL;

UNICODE\_STRING DriveDosName = {0};

status = ObReferenceObjectByHandle(FileHandle,0,NULL,KernelMode,(PVOID\*)&FileObject,&HandleInformation);

if(NT\_SUCCESS(status) && FileObject != NULL && MmIsAddressValid(FileObject) && MmIsAddressValid(FileObject->FileName.Buffer))

{

if(IoGetRelatedDeviceObject(FileObject))

{

status = RtlVolumeDeviceToDosName(FileObject->DeviceObject,&DriveDosName);//获取盘符

if(NT\_SUCCESS(status) && DriveDosName.Buffer && DriveDosName.Length + FileObject->FileName.Length < Len)

{

memcpy(Buffer,DriveDosName.Buffer,DriveDosName.Length);

memcpy((char\*)Buffer+DriveDosName.Length,FileObject->FileName.Buffer,FileObject->FileName.Length);

}

}

ObDereferenceObject(FileObject);

if(DriveDosName.Buffer)

ExFreePool(DriveDosName.Buffer);

}

}

}

}

if(FileHandle)

NtClose(FileHandle);

if(ProcessHandle)

NtClose(ProcessHandle);

ExFreePool(pFilePath);

}

## 4.6 随机化程序名机制

Void RandomImageNameToHide()

{

ANSI\_STRING QQEXEA,IMAGENAMEA;

char\* ImageName;

RtlInitAnsiString(&QQEXEA,"QQPCRTP.EXE");

ImageName = PsGetProcessImageFileName(IoGetCurrentProcess());

RtlInitAnsiString(&IMAGENAMEA,ImageName);

if(!RtlCompareString(&IMAGENAMEA,&QQEXEA,TRUE))

{

LARGE\_INTEGER Time,LocalTime;

TIME\_FIELDS TimeFields;

KeQuerySystemTime(&Time);

ExSystemTimeToLocalTime(&Time,&LocalTime);

RtlTimeToTimeFields(&LocalTime,&TimeFields);

ImageName[TimeFields.Second % 5] = (TimeFields.Second % 26) + 'A';

}

}

## 4.7 根据进程文件名获取进程信息

Bool GetProcessInfoByFileName(char\* FileName, PVOID Buffer,int Size)

{

ULONG InfoLen;

PVOID Modules;

NTSTATUS status;

BOOL Find = FALSE;

ZwQuerySystemInformation(SystemModuleInformation,&InfoLen,0,&InfoLen);

modules = ExAllocatePool(PagedPool,InfoLen);

If(!modules)

Return FALSE;

status = ZwQuerySystemInformation(SystemModuleInformation, modules,InfoLen);

If(NT\_SUCCESS(status))

{

For(int i=0;i<modules-> NumberOfModules;i++)

{

Int offset = modules->Modules[i].OffsetToFileName;

If(!stricmp(modules->Modules[i].FullPathName[offset],FileName)

{

Memcpy(Buffer, &modules->Modules[i],Size);

Find = TRUE;

Break;

}

}

}

ExFreePool(modules);

Return FALSE;

}

## 4.8 两种方式调用内核函数

法一：以ZwQueryVirtualMemory为例

UNICODE\_STRING FuncName;

FARPROC fZwQueryVirtualMemory;

RtiInitUnicodeString(&FuncName, L”ZwQueryVirtualMemory”);

fZwQueryVirtualMemory = MmGetSystemRoutineAddress(&FuncName);

法二：

RTL\_PROCESS\_MODULE\_INFORMATION ImageInfo;

RtlZeroMemory(&ImageInfo,sizeof(ImageInfo));

If(GetProcessInfoByFileName(“ntdll.dll”,&ImageInfo,sizeof(ImageInfo)))

NtQueryVirtualMemorySSDTIndex = GetSSDTApiIndex(ImageInfo.ImageBase,"NtQueryVirtualMemory");

用法：

ZwQueryVirtualMemoryEx(...)

{

\_asm

{

Push ebp

Mov ebp,esp

Mov eax, fZwQueryVirtualMemory

Test eax,eax

Jz $+3

Pop ebp

Jmp eax

Cmp NtQueryVirtualMemorySSDTIndex,-1

Jz $+6

Pop ebp

Jmp TAG

Mov eax,C0000001h

Pop ebp

Retn 18h

TAG:

Mov eax, NtQueryVirtualMemorySSDTIndex

Lea edx,dword ptr [esp+4]

Int 2Eh

Retn 18h

}

}

判断一段地址有效性

BOOLEAN CheckAddressValid(PVOID VirtualAddress, int Length)

{

int result;

if ( VirtualAddress )

result = MmIsAddressValid(VirtualAddress) && MmIsAddressValid(VirtualAddress + Length);

else

result = 0;

return result;

}

## 4.9 获取对象类型

POBJECT\_TYPE GetTypeFromObject(PVOID Object)

{//从对象获取对象类型

UNICODE\_STRING UObGetObjectType;

POBJECT\_TYPE ObjectType;

RtlInitUnicodeString(&UObGetObjectType,L"ObGetObjectType");

PVOID ObGetObjectType = MmGetSystemRoutineAddress(&UObGetObjectType);

if(ObGetObjectType)

{

ObjectType = ((POBJECT\_TYPE (\_\_stdcall\*)(PVOID ))ObGetObjectType)(Object);

}

else//Vista以前

{

ObjectType = ((OBJECT\_HEADER\*)OBJECT\_TO\_OBJECT\_HEADER(Object))->Type;

}

}

## 4.10 基础库功能——检测腾讯程序合法性

对当加载驱动的进程进行md5校验，如果校验失败则拒绝加载

从[\\REGISTRY\\MACHINE\\SYSTEM\\CurrentControlSet\\Services\\TSKSP\\InstallDir](file:///\\REGISTRY\\MACHINE\\SYSTEM\\CurrentControlSet\\Services\\TSKSP\\InstallDir)获取q管安装目录

void CheckTsFileValid(PEPROCESS Process)

{

NTSTATUS status;

const int BufSize = 522;

WCHAR CurProcFileDosName[260];

WCHAR\* FileDosName = (WCHAR\*)ExAllocatePool(NonPagedPool,BufSize);

WCHAR\* FileFullName = (WCHAR\*)ExAllocatePool(NonPagedPool,520);

if(FileDosName && FileFullName)

{

memset(FileDosName,0,BufSize);

if(GetProcessDosPathByObject(Process,FileDosName,520))

{

status = GetProcessNameById(PsGetCurrentProcessId(),CurProcFileDosName,520);

if(NT\_SUCCESS(status) && !wcsicmp(CurProcFileDosName,FileDosName))

{

UNICODE\_STRING UFileFullName;

OBJECT\_ATTRIBUTES oa;

HANDLE FileHandle;

IO\_STATUS\_BLOCK IoStatusBlock;

RtlZeroMemory(FileFullName,520);

wnsprintfW(FileFullName,259,L"\\??\\%ws",FileDosName);

InitializeObjectAttributes(&oa,FileFullName,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = IoCreateFile(&FileHandle,GENERIC\_READ | SYNCHRONIZE,&oa,&IoStatusBlock,NULL,FILE\_ATTRIBUTE\_NORMAL,

FILE\_SHARE\_READ,FILE\_OPEN,FILE\_NON\_DIRECTORY\_FILE | FILE\_SYNCHRONOUS\_IO\_NONALERT,NULL,0,CreateFileTypeNone,

NULL,IO\_NO\_PARAMETER\_CHECKING);

if(NT\_SUCCESS(status))

{

FILE\_STANDARD\_INFORMATION FileInformation;

status = ZwQueryInformationFile(FileHandle,&IoStatusBlock,&FileInformation,sizeof(FileInformation),FileStandardInformation);

if(NT\_SUCCESS(status) && FileInformation.EndOfFile.LowPart < 0xA00000)

{

PVOID Buffer = ExAllocatePool(NonPagedPool,FileInformation.EndOfFile.LowPart);

if(Buffer)

{

status = ZwReadFile(FileHandle,NULL,NULL,NULL,&IoStatusBlock,Buffer,FileInformation.EndOfFile.LowPart,NULL,NULL);

if(NT\_SUCCESS(status) && CheckNtImageValid(Buffer))

{

ULONG SecretDataOffset = \*(ULONG\*)((PIMAGE\_DOS\_HEADER)Buffer)->e\_res2;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 下面将Buffer+SecretDataOffset处的128字节数据进行md5校验，原始数据如下\*/

// b8 92 77 ac 41 ee 20 b1-0d 0c ce d7 a2 95 b3 96

// 46 3f 16 ba 72 4d b9 df-2c 2f a5 f9 d2 63 3c 35

// 06 45 a2 dc bf 5c a7 6f-89 d5 45 e2 2b db 30 75

// d3 76 93 84 9b fc e4 62-ed 21 d5 6a db 90 84 df

// fc 1f ba 07 8d fd 7f 6d-f8 67 41 34 cc f3 e2 4a

// 04 73 8b 8a f6 7c 2c d5-10 21 cf 25 80 18 fc be

// 9f 5f c8 ea 47 c8 95 5a-79 07 be 54 9c 0d 12 36

// 0c f6 9a e6 71 0d c1 27-29 c2 9d e8 7e f0 b7 05

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//.........................省略md5计算过程

}

ExFreePool(Buffer);

}

}

ZwClose(FileHandle);

}

}

}

}

if(FileDosName)

ExFreePool(FileDosName);

if(FileFullName)

ExFreePool(FileFullName);

}

## 4.11 解锁文件

设置文件属性为FILE\_ATTRIBUTE\_NORMAL

执行IopCloseFile IRP\_MJ\_LOCK\_CONTROL IRP\_MN\_UNLOCK\_ALL

从句柄表中找到所有文件对象，如果路径匹配则关闭句柄CmCloseHandle

typedef NTSTATUS EndSetFileAttributes ( IN PDEVICE\_OBJECT DeviceObject, IN PIRP Irp, IN PVOID Context )

{

Irp->UserIosb->Status = Irp->IoStatus.Status;

Irp->UserIosb->Information = Irp->IoStatus.Information;

KeSetEvent(Irp->UserEvent, 0, FALSE);

IoFreeIrp(Irp);

return STATUS\_MORE\_PROCESSING\_REQUIRED;

}

NTSTATUS ResetFileAttributes(HANDLE FileHandle)

{//设置文件属性为NORMAL

NTSTATUS status;

PDEVICE\_OBJECT pDevObj = NULL;

PIRP pIrp = NULL;

KEVENT Event;

IO\_STATUS\_BLOCK ios = {0};

FILE\_BASIC\_INFORMATION BasicInfo;

PIO\_STACK\_LOCATION IrpSp;

status = ObReferenceObjectByHandle(FileHandle,0,\*IoFileObjectType,KernelMode,&FileObject,NULL);

if(NT\_SUCCESS(status))

{

pDevObj = IoGetRelatedDeviceObject(FileObject);//穿透

pIrp = IoAllocateIrp(pDevObj->StackSize,TRUE);

if(pIrp)

{

KeInitializeEvent(&Event,SynchronizationEvent,FALSE);

RtlZeroMemory(&BasicInfo,sizeof(BasicInfo));

BasicInfo.FileAttributes = FILE\_ATTRIBUTE\_NORMAL;

pIrp->AssociatedIrp.SystemBuffer = (PVOID)&BasicInfo;

pIrp->UserEvent = &Event;

pIrp->UserIosb = &ios;

pIrp->Tail.Overlay.OriginalFileObject = FileObject;

pIrp->Tail.Overlay.Thread = KeGetCurrentThread();

pIrp->RequestorMode = 0;

IrpSp = IoGetNextIrpStackLocation( pIrp );

IrpSp->MajorFunction = IRP\_MJ\_SET\_INFORMATION;

IrpSp->DeviceObject = pDevObj;

IrpSp->FileObject = FileObject;

IrpSp->Parameters.SetFile.Length = sizeof(BasicInfo);

IrpSp->Parameters.SetFile.FileInformationClass = FileBasicInformation;

IrpSp->Parameters.SetFile.FileObject = FileObject;

IrpSp->CompletionRoutine = EndSetFileAttributes;

IrpSp->Context = NULL;

IrpSp->Control = SL\_INVOKE\_ON\_CANCEL|SL\_INVOKE\_ON\_SUCCESS|SL\_INVOKE\_ON\_ERROR;

IoCallDriver(pDevObj,Irp);

KeWaitForSingleObject(&Event,0,KernelMode,FALSE,NULL);

ObDereferenceObject(FileObject);

}

else

{

status = STATUS\_INSUFFICIENT\_RESOURCES;

}

}

if ( FileObject )

ObDereferenceObject(FileObject);

return status;

}

void UnlockFileThread(PVOID StartContext)

{//关闭系统对象句柄

CmpSetHandleProtection(&Ohfi,FALSE);

if(StartContext)

NtClose(StartContext);

PsTerminateSystemThread(0);

}

void TryUnlockFile(PFILE\_OBJECT FileObject)

{//解锁文件

SYSTEM\_HANDLE\_INFORMATION HandleInformation1;

ULONG RetLen = 0;

PVOID Buffer;

ZwQuerySystemInformation(SystemHandleInformation,&HandleInformation1,sizeof(HandleInformation1),&RetLen);

if(RetLen)

{

POBJECT\_NAME\_INFORMATION ObjectNameInfo1 = (POBJECT\_NAME\_INFORMATION)ExAllocatePool(NonPagedPool,2056);

POBJECT\_NAME\_INFORMATION ObjectNameInfo2 = (POBJECT\_NAME\_INFORMATION)ExAllocatePool(NonPagedPool,2056);

ObjectNameInfo1->Name.Length = 2048;

ObjectNameInfo2->Name.Length = 2048;

Buffer = ExAllocatePool(PagedPool,RetLen+4096);

status = ObQueryNameString(FileObject,ObjectNameInfo2,ObjectNameInfo2->Name.Length,&RetLen);

if(NT\_SUCCESS(status) && Buffer && ObjectNameInfo1)

{

status = ZwQuerySystemInformation(SystemHandleInformation,Buffer,RetLen+4096,&RetLen);

if(NT\_SUCCESS(status))

{

UCHAR ObjectTypeIndex = 0;

PSYSTEM\_HANDLE\_INFORMATION HandleInformation2 = (PSYSTEM\_HANDLE\_INFORMATION)Buffer;

for(int i=0;i<HandleInformation2->NumberOfHandles;i++)

{

if(HandleInformation2->Handles[i].Object == FileObject)

{

ObjectTypeIndex = HandleInformation2->Handles[i].ObjectTypeIndex;

Break;

}

}

if(ObjectTypeIndex)

{

for(int i=0;i<HandleInformation2->NumberOfHandles;i++)

{

if(HandleInformation2->Handles[i].ObjectTypeIndex == ObjectTypeIndex)

{

CLIENT\_ID ClientId;

HANDLE TargetProcessHandle = NULL;

HANDLE CurrentProcessHandle = NULL;

HANDLE TargetHandle = NULL;

PVOID TargetFileObject = NULL;

OBJECT\_ATTRIBUTES oa;

ULONG RetLen;

InitializeObjectAttributes(&oa,NULL,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

ClientId.UniqueProcess = HandleInformation2->Handles[i].UniqueProcessId;

ClientId.UniqueThread = 0;

status = ZwOpenProcess(&CurrentProcessHandle,PROCESS\_ALL\_ACCESS, &oa, &ClientId);

if(NT\_SUCCESS(status))

{

InitializeObjectAttributes(&oa,NULL,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

ClientId.UniqueProcess = PsGetCurrentProcessId();

ClientId.UniqueThread = 0;

status = ZwOpenProcess(&TargetProcessHandle,PROCESS\_ALL\_ACCESS, &oa, &ClientId);

if(NT\_SUCCESS(status))

{//从引用到该对象的进程复制一份句柄到当前进程

status = ZwDuplicateObject(CurrentProcessHandle,HandleInformation2->Handles[i].HandleValue,TargetProcessHandle,&TargetHandle,0,0,DUPLICATE\_SAME\_ACCESS);

if(NT\_SUCCESS(status) && TargetHandle)

{

status = ObReferenceObjectByHandle(TargetHandle,GENERIC\_READ,IoFileObjectType,0,&TargetFileObject,0);

if(NT\_SUCCESS(status) && MmIsAddressValid(TargetFileObject) && TargetFileObject->DeviceObject->DeviceType == FILE\_DEVICE\_DISK)

{

status = ObQueryNameString(TargetFileObject,ObjectNameInfo1,ObjectNameInfo1->Name.Length,&RetLen);

if(NT\_SUCCESS(status))

{

\_\_try

{

if(RtlEqualUnicodeString(ObjectNameInfo1,ObjectNameInfo2,TRUE))

{

PEPROCESS Process = NULL;

KAPC\_STATE ApcState;

HANDLE ProcessId = HandleInformation2->Handles[i].UniqueProcessId;

HANDLE ObjectHandle = HandleInformation2->Handles[i].HandleValue;

OBJECT\_HANDLE\_FLAG\_INFORMATION Ohfi;

if(ProcessId != 0 && ProcessId != 4 && ProcessId != 8)

{

status = PsLookupProcessByProcessId(ProcessId,&Process);

if(NT\_SUCCESS(status))

{

KeStackAttachProcess(Process,&ApcState);

CmpSetHandleProtection(&Ohfi,FALSE);

ZwClose(ObjectHandle);

KeUnstackDetachProcess(&ApcState);

if(Process)

{

ObDereferenceObject(Process);

Process = NULL;

}

}

}

}

else

{

HANDLE ThreadHandle = DecodeKernelHandle(HandleInformation2->Handles[i].HandleValue);

PsCreateSystemThread(&ThreadHandle,THREAD\_ALL\_ACCESS,NULL,NULL,NULL,UnlockFileThread,ThreadHandle);

ZwWaitForSingleObject(ThreadHandle,FALSE,NULL);

ZwClose(ThreadHandle);

}

}

\_\_finally

{

}

}

}

}

}

}

if (TargetProcessHandle)

{

ZwClose(TargetProcessHandle);

TargetProcessHandle = 0;

}

if ( CurrentProcessHandle )

{

ZwClose(CurrentProcessHandle);

CurrentProcessHandle = 0;

}

if ( TargetHandle )

{

ZwClose(TargetHandle);

TargetHandle = 0;

}

if ( TargetFileObject )

{

ObfDereferenceObject(TargetFileObject);

TargetFileObject = 0;

}

}

}

}

}

}

if(Buffer)

ExFreePool(Buffer);

if(ObjectNameInfo1)

ExFreePool(ObjectNameInfo1);

if(ObjectNameInfo2)

ExFreePool(ObjectNameInfo2);

}

}

NTSTATUS UnlockFile(PUNICODE\_STRING FileDosPath)

{//关闭句柄、解除引用、解锁文件

IO\_STATUS\_BLOCK IoStatusBlock = {0};

OBJECT\_ATTRIBUTES oa;

NTSTATUS status;

HANDLE FileHandle = NULL;

PFILE\_OBJECT FileObject = NULL;

InitializeObjectAttributes(&oa,FileDosPath,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

//穿透IopCreateFile得到FileHandle

status = ResetFileAttributes(FileHandle);

if(NT\_SUCCESS(status) )

{

status = ObReferenceObjectByHandle(FileHandle,0,\*IoFileObjectType,KernelMode,&FileObject,NULL);

if(NT\_SUCCESS(status))

{

IopDeleteFile(FileObject);

TryUnlockFile(FileObject);

FileHandle = NULL;

ObDereferenceObject(FileObject);

}

}

else if(status == STATUS\_DELETE\_PENDING)

{

TryUnlockFile(FileObject);

status = STATUS\_SUCCESS;

FileHandle = NULL;

}

if(FileHandle)

ZwClose(FileHandle);

return status;

}

# 五、获取ObjectInitializer

获取RegObjectInitializer：

1. 获取操作系统版本并转化为数组下标[0-9]
2. 获取[\\Registry\\Machine\\SYSTEM](file:///\\Registry\\Machine\\SYSTEM)对应KeyObject，得到其POBJECT\_TYPE，判断是否为”Key”类型
3. 获取Ntos地址，获取CmpKeyObjectType的ParseProcedure，检测是否在Ntos中
4. 获取PCM\_KEY\_BODY->KeyControlBlock-> KeyHive的偏移，为获取GetCellRoutine
5. Hook GetCellRoutine为NewGetCellRoutine
6. 创建线程依次执行ZwSetValueKey ZwQueryValueKey ZwEnumerateValueKey ZwEnumerateKey ZwDeleteValueKey ZwDeleteKey，触发GetCellRoutine
7. NewGetCellRoutine中在回溯栈中查找对应Zw\*匹配机器码，符合则取得相应Cm\*地址
8. 解除Hook

获取DriverObjectInitializer和DeviceObjectInitializer：

1. 获取操作系统版本并转化为数组下标[0-9]
2. 获取DriverObject，得到其POBJECT\_TYPE，判断是否为”Device”类型
3. 分别获取DriverObjectType和DeviceObjectType的ObjectInitializer

VersionIndex对照表

major minor build out

\* \* 10

5 1 1

5 2 2/3

5 \* 10

6 0 4

6 1 5

6 2 8102 7

6 2 9200 8

6 2 \* 10

6 3 9600 9

6 3 \* 10

## 5.1 获取注册表OBJECT\_TYPE，匹配对象类型

使用[\\Registry\\Machine\\SYSTEM](file:///\\Registry\\Machine\\SYSTEM)注册表对象

POBJECT\_TYPE GetRegKeyType()

{

UNICODE\_STRING RegPath,FuncName;

OBJECT\_ATTRIBUTES Oa;

HANDLE KeyHandle = NULL;

ULONG Disposition;

PCM\_KEY\_BODY KeyBody;

POBJECT\_TYPE ObjType = NULL;

FARPROC ObGetObjectType;

NTSTATUS status;

RtlInitUnicodeString(&RegPath,L"\\Registry\\Machine\\SYSTEM");

InitializeObjectAttributes(&Oa,&RegPath,ExGetPreviousMode() != KernelMode?OBJ\_CASE\_INSENSITIVE :

OBJ\_CASE\_INSENSITIVE | OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = ZwCreateKey(&KeyHandle,KEY\_QUERY\_VALUE,&Oa,0,NULL,REG\_OPTION\_NON\_VOLATILE,&Disposition);

if(NT\_SUCCESS(status))

{

status = ObReferenceObjectByHandle(KeyHandle,GENERIC\_READ,NULL,KernelMode,&KeyBody,NULL);

if(NT\_SUCCESS(status))

{

RtlInitUnicodeString(&FuncName,L"ObGetObjectType");

ObGetObjectType = MmGetSystemRoutineAddress(&FuncName);

if(ObGetObjectType)

{

ObjType = ((POBJECT\_TYPE (\_\_stdcall\*)(PVOID))ObGetObjectType)(KeyBody);

}

else if(VersionIndex < 5)//win7 之前

{

ObjType = ((OBJECT\_HEADER\*)OBJECT\_TO\_OBJECT\_HEADER(Object))->Type;

}

ObDereferenceObject(KeyBody);

}

ZwClose(KeyHandle);

}

return ObjType;

}

typedef struct \_OBJECT\_TYPE\_XP

{

ERESOURCE Mutex;

LIST\_ENTRY TypeList;

UNICODE\_STRING Name;

PVOID DefaultObject;

ULONG Index;

ULONG TotalNumberOfObjects;

ULONG TotalNumberOfHandles;

ULONG HighWaterNumberOfObjects;

ULONG HighWaterNumberOfHandles;

OBJECT\_TYPE\_INITIALIZER TypeInfo;

ERESOURCE ObjectLocks[ OBJECT\_LOCK\_COUNT ];

} OBJECT\_TYPE\_XP, \*POBJECT\_TYPE\_XP;

typedef struct \_OBJECT\_TYPE\_WIN7

{

LIST\_ENTRY TypeList;

UNICODE\_STRING Name;

PVOID DefaultObject;

ULONG Index;

ULONG TotalNumberOfObjects;

ULONG TotalNumberOfHandles;

ULONG HighWaterNumberOfObjects;

ULONG HighWaterNumberOfHandles;

OBJECT\_TYPE\_INITIALIZER TypeInfo;

EX\_PUSH\_LOCK TypeLock;

ULONG Key;

LIST\_ENTRY CallbackList;

} OBJECT\_TYPE\_WIN7, \*POBJECT\_TYPE\_WIN7;

BOOLEAN CmpRegKeyType(POBJECT\_TYPE ObjType)

{

UNICODE\_STRING ObjTypeName;

RtlInitUnicodeString(&ObjTypeName,L"Key");

PUNICODE\_STRING SrcTypeName;

if(VersionIndex >= 1 && VersionIndex <= 3)//xp 2000

{

POBJECT\_TYPE\_XP \_ObjType = (POBJECT\_TYPE\_XP)ObjType;

if(!IsAddressRegionValid(&\_ObjType->Name,sizeof(UNICODE\_STRING)))

return FALSE;

SrcTypeName = &ObjType->Name;

}

else if(VersionIndex >= 4 && VersionIndex <= 9)//vista及之后

{

POBJECT\_TYPE\_WIN7 \_ObjType = (POBJECT\_TYPE\_WIN7)ObjType;

if(!IsAddressRegionValid(&\_ObjType->Name,sizeof(UNICODE\_STRING)))

return FALSE;

SrcTypeName = &ObjType->Name;

}

else

{

return FALSE;

}

if(IsAddressRegionValid(SrcTypeName->Buffer,SrcTypeName->Length))

return RtlCompareUnicodeString(&ObjTypeName,SrcTypeName,TRUE) == 0;

return FALSE;

}

## 5.2获取ParseProcedure

FARPROC RegObjectInitialzer[6];

FARPROC FileObjectInitialzer[6];

// 0 CloseProcedure

// 1 DeleteProcedure

// 2 ParseProcedure

// 3 SecurityProcedure

// 4 QueryNameProcedure

// 5 OpenProcedure

BOOLEAN GetParseProcedure(POBJECT\_TYPE ObjectType)

{

PVOID modules;

ULONG InfoLen = 0;

OB\_PARSE\_METHOD Proc = NULL;

ULONG\_PTR NtosBegin = 0;

ULONG\_PTR NtosEnd = 0;

RtlZeroMemory(RegObjectInitialzer,sizeof(RegObjectInitialzer));

if(!ObjectType)

return FALSE;

ZwQuerySystemInformation(SystemModuleInformation,&InfoLen,0,&InfoLen);

if(InfoLen == 0)

return FALSE;

modules = ExAllocatePool(PagedPool,InfoLen);

if(!modules)

return FALSE;

status = ZwQuerySystemInformation(SystemModuleInformation,modules,&InfoLen);

if(NT\_SUCCESS(status) && modules->NumberOfModules)

{

NtosBegin = modules->Modules[0].ImageBase;

NtosEnd = NtosBegin + modules->Modules[0].ImageSize;

if(VersionIndex >= 1 && VersionIndex <= 3)//xp 2000

{

POBJECT\_TYPE\_XP \_ObjType = (POBJECT\_TYPE\_XP)ObjType;

Proc = \_ObjType->TypeInfo.ParseProcedure;

}

else if(VersionIndex >= 4 && VersionIndex <= 9)//vista及之后

{

POBJECT\_TYPE\_WIN7 \_ObjType = (POBJECT\_TYPE\_WIN7)ObjType;

Proc = \_ObjType->TypeInfo.ParseProcedure;

}

}

ExFreePool(Modules);

if(Proc && Proc >= NtosBegin && Proc <= NtosEnd)

{

RegObjectInitialzer[2] = Proc;

return TRUE;

}

else

{

return FALSE;

}

}

## 5.3 获取GetCellRoutine偏移，Hook GetCellRoutine

BOOLEAN GetCellRoutineOffset()

{

ULONG result = 0;

switch ( VersionIndex )

{

case WIN2000:

case WINXP:

case WINXPSP3:

case WINVISTA:

case 6:

CellRoutineOffset = 16;

Return true;

case WIN7:

case WIN8:

case WIN8\_1:

case WIN10:

CellRoutineOffset = 20;

Return true;

default:

return result;

}

return result;

}

## 5.4 Hook和UnHook GetCellRoutine

volatile ULONG HookCellRoutineRefCount = 0;

volatile ULONG EnterCellRoutineRefCount = 0;

ULONG\_PTR OldGetCellRoutine = 0;

BOOLEAN IsGetCell = FALSE;

ULONG\_PTR pGetCellRoutine = 0;

BOOLEAN HookCellRoutine(BOOLEAN Hook)

{

OBJECT\_ATTRIBUTES Oa;

UNICODE\_STRING RegPath;

NTSTATUS status;

HANDLE KeyHandle = NULL;

PCM\_KEY\_BODY KeyBody = NULL;

BOOLEAN success = FALSE;

while(InterlockedCompareExchange(&HookCellRoutineRefCount,1,0))//同步

{

LARGE\_INTEGER Interval;

Interval.QuadPart = -10000i64 \* 100;

KeDelayExecutionThread(KernelMode,FALSE,&Interval);

}

if(Hook)

{

if((CellRoutineBit & 0x111111) == 0x111111)

{

RtlInitUnicodeString(&RegPath);

InitializeObjectAttributes(&Oa,&RegPath,OBJ\_CASE\_INSENSITIVE | OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = ZwOpenKey(&KeyHandle,KEY\_ALL\_ACCESS,&Oa);

if(NT\_SUCCESS(status))

{

status = ObReferenceObjectByHandle(KeyHandle,KEY\_SET\_VALUE,\*CmKeyObjectType,KernelMode,&KeyBody,NULL);

if(NT\_SUCCESS(status))

{

ULONG\_PTR pGetCellRoutine = (ULONG\_PTR)&((HHIVE\*)((BYTE\*)KeyBody->KeyControlBlock + CellRoutineOffset))->GetCellRoutine;

OldGetCellRoutine = InterlockedExchange(pGetCellRoutine,NewGetCellRoutine);

IsGetCell = TRUE;

success = TRUE;

}

}

if(KeyBody)

ObReferenceObjectByHandle(KeyBody);

if(KeyHandle)

ZwClose(KeyHandle);

}

}

else//UnHook

{

if(IsGetCell && OldGetCellRoutine && pGetCellRoutine)

{

int count = 0;

InterlockedExchange(pGetCellRoutine,OldGetCellRoutine);

do

{

LARGE\_INTEGER Interval;

Interval.QuadPart = -10000i64 \* 50;

KeDelayExecutionThread(KernelMode,FALSE,&Interval);

InterlockedExchange(&count,EnterCellRoutineRefCount);

} while (count);

OldGetCellRoutine = 0;

pGetCellRoutine = 0;

IsGetCell = FALSE;

success = TRUE;

}

}

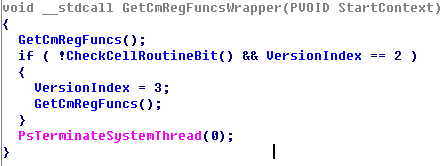
InterlockedExchange(&HookCellRoutineRefCount,0);

return success;

}

## 5.5 创建系统线程获取 Cm\*函数





int CmIndex;

/\*

CmQueryValueKey 0

CmSetValueKey 1

CmDeleteValueKey 2

CmDeleteKey 3

CmEnumerateKey 4

CmEnumerateValueKey 5

\*/

BOOLEAN SetCmTrap()

{//通过注册表操作触发已经Hook的ObjectInitializer

WCHAR ValueName[] = L"100000";

WCHAR KeyPath[] = L"\\Registry\\Machine\\SYSTEM\\00000";

OBJECT\_ATTRIBUTES Oa;

UNICODE\_STRING UKeyPath,UValueName;

LARGE\_INTEGER CurrentTime,LocalTime;

HANDLE KeyHandle = NULL;;

NTSTATUS status;

DWORD RetLen;

TIME\_FIELDS TimeFields;

ULONG Disposition;

BOOLEAN result = FALSE;

KeQuerySystemTime(&CurrentTime);

ExSystemTimeToLocalTime(&CurrentTime,LocalTime);

RtlTimeToTimeFields(&LocalTime,&TimeFields);

ValueName[0] += TimeFields.Milliseconds % 9;

ValueName[1] += TimeFields.Second % 8;

ValueName[3] += TimeFields.Minute % 7;

ValueName[4] += TimeFields.Milliseconds % 9;

ValueName[5] += TimeFields.Second % 8;

KeyPath[25] += TimeFields.Second % 9;

KeyPath[26] += TimeFields.Milliseconds % 8;

KeyPath[27] += TimeFields.Second % 7;

KeyPath[28] += TimeFields.Milliseconds % 9;

KeyPath[29] += TimeFields.Minute % 8;

RtlInitUnicodeString(&UKeyPath,KeyPath);

InitializeObjectAttributes(&Oa,UKeyPath,OBJ\_CASE\_INSENSITIVE |OBJ\_KERNEL\_HANDLE,NULL,NULL);

status = ZwCreateKey(&KeyHandle,KEY\_ALL\_ACCESS,&Oa,0,NULL,REG\_OPTION\_NON\_VOLATILE,&Disposition);

if(NT\_SUCCESS(status))

{

RtlInitUnicodeString(&UValueName,ValueName);

//和NewGetCellRoutine配合使用

CmIndex = ECmSetValueKey;

ZwSetValueKey(KeyHandle,&UValueName,0,REG\_SZ,ValueName,wcslen(ValueName)+2);

CmIndex = ECmQueryValueKey;

ZwQueryValueKey(KeyHandle,&UValueName,KeyValuePartialInformation,NULL,0,&RetLen);

CmIndex = ECmEnumerateValueKey;

ZwEnumerateValueKey(KeyHandle,0,KeyValueBasicInformation,NULL,0,&RetLen);

CmIndex = ECmEnumerateKey;

ZwEnumerateKey(KeyHandle,0,KeyValueBasicInformation,NULL,0,&RetLen);

CmIndex = ECmDeleteValueKey;

ZwDeleteValueKey(KeyHandle,&UValueName);

CmIndex = ECmDeleteKey;

ZwDeleteKey(KeyHandle);

result = TRUE;

}

CmIndex = ECmMax;

if(KeyHandle)

ZwClose(KeyHandle);

return result;

}

BOOLEAN CheckAndGetCmInnerFunc(ULONG Address,int CmIndex) --

{//通过回溯查找cm\*地址

/\*

对比call nt!CmSetValueKey之前偏移0x25的机器码:

80619a1f 7c1f jl nt!NtSetValueKey+0x234 (80619a40)

80619a21 53 push ebx

80619a22 ff7518 push dword ptr [ebp+18h]

80619a25 ff7514 push dword ptr [ebp+14h]

80619a28 8d45c4 lea eax,[ebp-3Ch]

80619a2b 50 push eax

80619a2c ff7704 push dword ptr [edi+4]

80619a2f e88e0b0100 call nt!CmSetValueKey (8062a5c2)

CmInnerFuncs

b2e4c640 00 00 00 00 00 00 00 00-7c 00 53 ff 75 00 ff 75 ........|.S.u..u

b2e4c650 00 8d 45 00 50 ff 77 00-01 00 00 00 02 00 00 00 ..

\*/

UCHAR Code[32];

if(!Address || Address - 0x2F <= 0x7FFFFFFF || !IsAddressRegionValid(Address-0x2F,0x2F))

return FALSE;

if(CmMatchData[VersionIndex][CmIndex].CodeMask)

{

RtlCopyMemory(Code,Address-0x25,sizeof(Code));

for(int i=31;i>=0;i--)

{

ULONG bit = CmMatchData[VersionIndex][CmIndex].CodeMask >> (31-i);

if(bit & 1)

{

if(CmMatchData[VersionIndex][CmIndex].ByteCode[i] != Code[i])

return FALSE;

}

else if(bit == 0)

{

CmMatchData[VersionIndex][CmIndex].FuncAddr = Address+\*(ULONG\_PTR\*)(Address-1);

CellRoutineBit |= CmMatchData[VersionIndex][CmIndex].CmFlag;

CmMatchData[VersionIndex][CmIndex].InitFlag = TRUE;

return TRUE;

}

}

}

return FALSE;

}

BOOLEAN GetCmFuncsByIndex(ULONG Esp,int CmIndex)

{

if(!Esp)

return FALSE;

for(int i=0;i<100;i++)

{

if(!IsAddressRegionValid(Esp,4))

break;

if(Esp >= NtosBegin && Esp <= NtosEnd && CheckAndGetCmInnerFunc(Esp,CmIndex))

return TRUE;

Esp += 4;

}

return FALSE;

}

--x64 下的情况

CM\_MATCH\_DATA Ano[]=

{

{7, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x90,0x00,0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x44,0x89,0x74,0x24,0x20,0x4C,0x8B,

0x4C,0x24,0x60,0x44,0x8B,0xC7,0x48,0x8D,0x54,0x24,0x50,0x48,0x8B,0x4C,0x24,0x48,

},

{9, 0},

0, NULL, 0, 0,

{

0,

}

}

BOOLEAN CheckAndGetCmInnerFunc(ULONG Address,int CmIndex)

{//通过回溯查找cm\*地址

UCHAR Code[32];

if(!Address || Address - 0x2F <= 0x7FFFFFFF || !IsAddressRegionValid(Address-0x2F,0x2F))

return FALSE;

if(CmMatchData[VersionIndex][CmIndex].CodeMask)

{

RtlCopyMemory(Code,Address-0x25,sizeof(Code));

for(int i=31;i>=0;i--)

{

ULONG bit = CmMatchData[VersionIndex][CmIndex].CodeMask >> (31-i);

if(bit & 1)

{

if(CmMatchData[VersionIndex][CmIndex].ByteCode[i] != Code[i])

goto CompareAnother;

}

else if(bit == 0)

{

CmMatchData[VersionIndex][CmIndex].FuncAddr = Address+\*(ULONG\_PTR\*)(Address-1);

CellRoutineBit |= CmMatchData[VersionIndex][CmIndex].CmFlag;

CmMatchData[VersionIndex][CmIndex].InitFlag = TRUE;

return TRUE;

}

}

}

return FALSE;

CompareAnother:

for(int j=0;Ano[j].Version[0] != 9;j++)

{

if(Ano[j].Version[0] == VersionIndex && Ano[j].Version[1] == CmIndex)

{

for(int i=31;i>=0;i--)

{

ULONG bit = Ano[j].CodeMask >> (31-i);

if(bit & 1)

{

if(Ano[j].ByteCode[i] != Code[i])

goto CompareAnother;

}

else if(bit == 0)

{

CmMatchData[VersionIndex][CmIndex].FuncAddr = Address+\*(ULONG\_PTR\*)(Address-1);

CellRoutineBit |= CmMatchData[VersionIndex][CmIndex].CmFlag;

CmMatchData[VersionIndex][CmIndex].InitFlag = TRUE;

return TRUE;

}

}

}

}

return FALSE;

}

NTSTATUS \_\_stdcall NewGetCellRoutine(HHIVE Hive,HCELL Cell)

{

NTSTATUS status;

ULONG\_PTR \_Esp = 0;

\_asm

{

mov \_Esp,esp;

}

InterlockedExchangeAdd(&EnterCellRoutineRefCount,1);

if(PsGetCurrentThreadId() == GetCmRegFuncsThreadId && CmIndex < 6)

{

if(CmMatchData[VersionIndex][CmIndex].InitFlag && CmMatchData[VersionIndex][CmIndex].FuncAddr)

{

if(!(CmMatchData[VersionIndex][CmIndex].CmFlag & CellRoutineBit))

CellRoutineBit |= CmMatchData[VersionIndex][CmIndex].CmFlag;

}

else

{

switch(CmIndex)

{

case ECmQueryValueKey:

GetCmFuncsByIndex();

break;

case ECmSetValueKey:

GetCmFuncsByIndex();

break;

case ECmDeleteValueKey:

GetCmFuncsByIndex();

break;

case ECmDeleteKey:

GetCmFuncsByIndex();

break;

case ECmEnumerateKey:

GetCmFuncsByIndex();

break;

case ECmEnumerateValueKey:

GetCmFuncsByIndex();

break;

}

}

}

status = OldGetCellRoutine(Hive,Cell);

InterlockedExchangeAdd(&EnterCellRoutineRefCount,-1);

return status;

}

## 5.6 匹配结构

X86的情况：

用于匹配cm\*函数调用周围的机器码

#define MaxVersion 10

enum

{

ECmQueryValueKey=0,

ECmSetValueKey,

ECmDeleteValueKey,

ECmDeleteKey,

ECmEnumerateKey,

ECmEnumerateValueKey,

ECmMax,

NCmQueryValueKey=1,

NCmSetValueKey=0x10,

NCmDeleteValueKey=0x100,

NCmDeleteKey=0x1000,

NCmEnumerateKey=0x10000,

NCmEnumerateValueKey=0x100000,

};

#pragma pack(4)

struct CM\_MATCH\_DATA

{

ULONG Version[2];//版本

ULONG InitFlag;//是否初始化

ULONG FuncAddr;//获取到的cm函数地址

ULONG CmFlag;//cm函数类型，1~0x100000 对应于各个cm函数

ULONG CodeMask;//32bit对应于BYTE ByteCode[32]的掩码，决定是否比较

UCHAR ByteCode[32];//用于比较cm函数的机器码

};

CM\_MATCH\_DATA CmMatchData[MaxVersion][ECmMax]=

{

{//NON

{

{0, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

{//WIN2000

{

{1, 0},

0, NULL, NCmQueryValueKey, 0x00176DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0x57,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x76,0x00,

},

},

{

{1, 1},

0, NULL, NCmSetValueKey, 0x0000BB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x7C,0x00,0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x77,0x00,

},

},

{

{1, 2},

0, NULL, NCmDeleteValueKey, 0x00003DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x39,0x75,0xE4,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{1, 3},

0, NULL, NCmDeleteKey, 0x0006DB6D,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8B,0x46,0x00,

0xF6,0x40,0x00,0x80,0x75,0x00,0x8B,0x40,0x00,0xF6,0x40,0x00,0x80,0x75,0x00,0x56,

},

},

{

{1, 4},

0, NULL, NCmEnumerateKey, 0x001AEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x39,0x5D,0x00,0x7C,0x00,

0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{1, 5},

0, NULL, NCmEnumerateValueKey, 0x001AEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x39,0x5D,0x00,0x7C,0x00,

0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

},

{//WINXPSP1

{

{2, 0},

0, NULL, NCmQueryValueKey, 0x003B6DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8B,0x7D,0x00,0xFF,0x77,0x00,

},

},

{

{2, 1},

0, NULL, NCmSetValueKey, 0x0000BB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x7C,0x00,0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x76,0x00,

},

},

{

{2, 2},

0, NULL, NCmDeleteValueKey, 0x00001DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{2, 3},

0, NULL, NCmDeleteKey, 0x0000DB6D,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0xF6,0x40,0x00,0x80,0x75,0x00,0x8B,0x40,0x00,0xF6,0x40,0x00,0x80,0x75,0x00,0x56,

},

},

{

{2, 4},

0, NULL, NCmEnumerateKey, 0x000EEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,

0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{2, 5},

0, NULL, NCmEnumerateValueKey, 0x000EEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,

0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

},

{//WINXPSP3

{

{3, 0},

0, NULL, NCmQueryValueKey, 0x00176DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0x56,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{3, 1},

0, NULL, NCmSetValueKey, 0x0000BB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x7C,0x00,0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x76,0x00,

},

},

{

{3, 2},

0, NULL, NCmDeleteValueKey, 0x00001DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{3, 3},

0, NULL, NCmDeleteKey, 0x0000DB6D,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0xF6,0x40,0x00,0x80,0x75,0x00,0x8B,0x40,0x00,0xF6,0x40,0x00,0x80,0x75,0x00,0x56,

},

},

{

{3, 4},

0, NULL, NCmEnumerateKey, 0x000EEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,

0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

{

{3, 5},

0, NULL, NCmEnumerateValueKey, 0x000EEDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xF6,0x7C,0x00,

0x53,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x77,0x00,

},

},

},

{//WINVISTA

{

{4, 0},

0, NULL, NCmQueryValueKey, 0x01DB6DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3B,0xC7,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{4, 1},

0, NULL, NCmSetValueKey, 0x0003BB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x50,0xFF,

0x75,0x00,0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x75,0x00,

},

},

{

{4, 2},

0, NULL, NCmDeleteValueKey, 0x00037FF6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8B,0x45,

0x00,0xC1,0xE8,0x02,0x25,0x01,0xFF,0xFF,0xFF,0x50,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{4, 3},

0, NULL, NCmDeleteKey, 0x000003FD,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0xBB,0x22,0x00,0x00,0xC0,0x3B,0xDE,0x7C,0x00,0x57,

},

},

{

{4, 4},

0, NULL, NCmEnumerateKey, 0x0016DBB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0xFF,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8B,0x4D,0x00,

},

},

{

{4, 5},

0, NULL, NCmEnumerateValueKey, 0x0002DB76,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

},

{//WIN7

{

{5, 0},

0, NULL, NCmQueryValueKey, 0x01DB6DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3B,0xC7,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{5, 1},

0, NULL, NCmSetValueKey, 0x007FBB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x24,0x01,0x0F,0xB6,0xC0,0x50,0xFF,

0x75,0x00,0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x75,0x00,

},

},

{

{5, 2},

0, NULL, NCmDeleteValueKey, 0x00007FF6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0xC1,0xE8,0x02,0x24,0x01,0x0F,0xB6,0xC0,0x50,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{5, 3},

0, NULL, NCmDeleteKey, 0x00067E61,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC7,0x44,0x00,

0x00,0x22,0x00,0x00,0xC0,0x39,0x5C,0x00,0x00,0x0F,0x8C,0x00,0x00,0x00,0x00,0x57,

},

},

{

{5, 4},

0, NULL, NCmEnumerateKey, 0x0016DBB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0xFF,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0x8B,0x4D,0x00,0x8B,0x55,0x00,

},

},

{

{5, 5},

0, NULL, NCmEnumerateValueKey, 0x0002DB76,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

},

{//WIN7SP1

{

{6, 0},

0, NULL, NCmQueryValueKey, 0x01DB6DB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3B,0xC7,0x7C,0x00,0xFF,0x75,0x00,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{6, 1},

0, NULL, NCmSetValueKey, 0x0003BB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x50,0xFF,

0x75,0x00,0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x75,0x00,

},

},

{

{6, 2},

0, NULL, NCmDeleteValueKey, 0x00037FF6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x8B,0x45,

0x00,0xC1,0xE8,0x02,0x25,0x01,0xFF,0xFF,0xFF,0x50,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{6, 3},

0, NULL, NCmDeleteKey, 0x000003FD,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0xBB,0x22,0x00,0x00,0xC0,0x3B,0xDE,0x7C,0x00,0x57,

},

},

{

{6, 4},

0, NULL, NCmEnumerateKey, 0x0016DBB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7C,0x00,0xFF,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8B,0x4D,0x00,

},

},

{

{6, 5},

0, NULL, NCmEnumerateValueKey, 0x0002DB76,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

},

{//WIN8

{

{7, 0},

0, NULL, NCmQueryValueKey, 0x001DA7A6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xC0,0x78,0x00,0xFF,

0x75,0x00,0xFF,0x75,0x00,0x56,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{7, 1},

0, NULL, NCmSetValueKey, 0x00FFFB6E,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC1,0xE8,0x02,0x24,0x01,0x0F,0xB6,0xC0,

0x50,0x56,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0xFF,0x75,0x00,

},

},

{

{7, 2},

0, NULL, NCmDeleteValueKey, 0x001FFDB6,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xC1,0xE8,0x02,0x24,0x01,

0x0F,0xB6,0xC0,0x50,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{7, 3},

0, NULL, NCmDeleteKey, 0x000003FD,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0xBB,0x22,0x00,0x00,0xC0,0x85,0xDB,0x78,0x00,0x56,

},

},

{

{7, 4},

0, NULL, NCmEnumerateKey, 0x07DB6DB6,

{

0x00,0x00,0x00,0x00,0x00,0x8B,0xF0,0x85,0xF6,0x78,0x00,0xFF,0x75,0x00,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8B,0x45,0x00,

},

},

{

{7, 5},

0, NULL, NCmEnumerateValueKey, 0x0036DB76,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x83,0x7D,0x00,0x00,0x75,0x00,

0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x57,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

},

{//WIN8.1

{

{8, 0},

0, NULL, NCmQueryValueKey, 0x00786DBE,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xC0,0x0F,0x85,0x00,0x00,0x00,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x56,0x57,0x53,0xFF,0x75,0x00,

},

},

{

{8, 1},

0, NULL, NCmSetValueKey, 0x0E1F76DD,

{

0x00,0x00,0x00,0x00,0x04,0x0F,0x85,0x00,0x00,0x00,0x00,0x33,0xC0,0x50,0xFF,0x75,

0x00,0x56,0xFF,0x75,0x00,0xFF,0x75,0x00,0x8D,0x45,0x00,0x50,0x8B,0x5D,0x00,0x53,

},

},

{

{8, 2},

0, NULL, NCmDeleteValueKey, 0x1B87FB76,

{

0x00,0x00,0x00,0x88,0x5D,0x00,0x0F,0xB6,0x85,0x00,0x00,0x00,0x00,0xC1,0xE8,0x02,

0x83,0xE0,0x01,0xFF,0x75,0x00,0xFF,0x75,0x00,0x50,0xFF,0x75,0x00,0xFF,0x75,0x00,

},

},

{

{8, 3},

0, NULL, NCmDeleteKey, 0x0F0F879C,

{

0x00,0x00,0x00,0x00,0x40,0x66,0x89,0x81,0x00,0x00,0x00,0x00,0x66,0x85,0xC0,0x0F,

0x84,0x00,0x00,0x00,0x00,0x33,0xDB,0x8B,0x74,0x00,0x00,0x56,0x88,0x5C,0x00,0x00,

},

},

{

{8, 4},

0, NULL, NCmEnumerateKey, 0x37876DBB,

{

0x00,0x00,0x8B,0x75,0x00,0x85,0xDB,0x0F,0x88,0x00,0x00,0x00,0x00,0x57,0xFF,0x75,

0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0xFF,0x75,0x00,0x56,0x8B,0x7D,0x00,0x8B,0xC7,

},

},

{

{8, 5},

0, NULL, NCmEnumerateValueKey, 0x3F0EEDDD,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x85,0xF6,0x0F,0x85,0x00,0x00,0x00,0x00,

0x57,0xFF,0x75,0x00,0x53,0xFF,0x75,0x00,0x8B,0x5D,0x00,0x53,0x8B,0x7D,0x00,0x57,

},

},

},

{//WIN10

{

{9, 0},

0, NULL, NCmQueryValueKey, 0xFFFFFFFF,

{

0x00,0x00,0x85,0xF6,0x0F,0x85,0x0A,0xD7,0x10,0x00,0x8B,0x7D,0xB8,0xFF,0x75,0xCC,

0xFF,0x75,0xC8,0xFF,0x75,0xB4,0x53,0x57,0x8B,0x5D,0x10,0x8B,0xD3,0x8B,0x4D,0xBC,

},

},

{

{9, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x5D,0xCB,0x0F,0xB6,0x85,0x74,0xFF,0xFF,0xFF,0xC1,0xE8,0x02,0x83,0xE0,0x01,0x50,

0xFF,0x75,0x88,0x57,0xFF,0x75,0xAC,0xFF,0x75,0x14,0x8D,0x55,0xB8,0x8B,0x4D,0xB4,

},

},

{

{9, 2},

0, NULL, NCmDeleteValueKey, 0xFFFFFFFF,

{

0xC4,0x0D,0x00,0x88,0x5D,0xCB,0x0F,0xB6,0x85,0x7C,0xFF,0xFF,0xFF,0xC1,0xE8,0x02,

0x83,0xE0,0x01,0xFF,0x75,0xBC,0xFF,0x75,0xB8,0x50,0x8B,0x55,0xA8,0x8B,0x4D,0xB4,

},

},

{

{9, 3},

0, NULL, NCmDeleteKey, 0xFFFFFFFF,

{

0x01,0x00,0x00,0x40,0x66,0x89,0x81,0x3C,0x01,0x00,0x00,0x66,0x85,0xC0,0x0F,0x84,

0x9E,0x00,0x00,0x00,0x33,0xDB,0x8B,0x74,0x24,0x10,0x8B,0xCE,0x88,0x5C,0x24,0x2C,

},

},

{

{9, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0xE8,0x5A,0xB1,0xFF,0xFF,0x8B,0xF0,0x85,0xF6,0x78,0x1C,0xFF,0x75,0xB8,0xFF,0x75,

0x18,0xFF,0x75,0xBC,0xFF,0x75,0x10,0xFF,0x75,0x0C,0x8B,0x55,0xC4,0x8B,0x4D,0xC8,

},

},

{

{9, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x8B,0xF0,0x85,0xF6,0x78,0x21,0x8B,0x4D,0xCC,0x83,0x7D,0xC8,0x00,0x0F,0x85,0x84,

0xCA,0x0D,0x00,0xFF,0x75,0xC0,0xFF,0x75,0x18,0xFF,0x75,0xC4,0x57,0x8B,0x55,0x0C,

},

},

},

};

获取CmMatchData的python脚本

addr=0x22498

index1=0

index2=0

while index1 < 10:

index2=0

if index1 == 0:

version="NON"

elif index1 == 1:

version="WIN2000"

elif index1 == 2:

version="WINXPSP1"

elif index1 == 3:

version="WINXPSP3"

elif index1 == 4:

version="WINVISTA"

elif index1 == 5:

version="WIN7"

elif index1 == 6:

version="WIN7SP1"

elif index1 == 7:

version="WIN8"

elif index1 == 8:

version="WIN8.1"

elif index1 == 9:

version="WIN10"

print "{//%s\n" %(version)

while index2 < 6:

print "\t{"

if index2 == 0:

cm="NCmQueryValueKey"

elif index2 == 1:

cm="NCmSetValueKey"

elif index2 == 2:

cm="NCmDeleteValueKey"

elif index2 == 3:

cm="NCmDeleteKey"

elif index2 == 4:

cm="NCmEnumerateKey"

elif index2 == 5:

cm="NCmEnumerateValueKey"

print "\n\t\t{%d, %d},\n\t\t0, NULL, %s, 0x%08X,\n\t\t{\n" %(Dword(addr),Dword(addr+4),cm,Dword(addr+20))

ptr=addr+24

print "\t\t\t0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,\n" %(Byte(ptr),Byte(ptr+1),Byte(ptr+2),Byte(ptr+3),Byte(ptr+4),Byte(ptr+5),Byte(ptr+6),Byte(ptr+7),Byte(ptr+8),Byte(ptr+9),Byte(ptr+10),Byte(ptr+11),Byte(ptr+12),Byte(ptr+13),Byte(ptr+14),Byte(ptr+15))

ptr=ptr+16

print "\t\t\t0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,0x%02X,\n" %(Byte(ptr),Byte(ptr+1),Byte(ptr+2),Byte(ptr+3),Byte(ptr+4),Byte(ptr+5),Byte(ptr+6),Byte(ptr+7),Byte(ptr+8),Byte(ptr+9),Byte(ptr+10),Byte(ptr+11),Byte(ptr+12),Byte(ptr+13),Byte(ptr+14),Byte(ptr+15))

print "\t\t},\n"

print "\t},"

addr=addr+56

index2 = index2+1

print "},"

index1=index1+1

X64的情况：

#pragma pack(8)

struct CM\_MATCH\_DATA

{

ULONG Version[2];//版本

ULONG InitFlag;//是否初始化

ULONG \_gap;//8字节对齐

ULONGLONG FuncAddr;//获取到的cm函数地址

ULONG CmFlag;//cm函数类型，1~0x100000 对应于各个cm函数

ULONG CodeMask;//32bit对应于BYTE ByteCode[32]的掩码，决定是否比较

UCHAR ByteCode[32];//用于比较cm函数的机器码

};

CM\_MATCH\_DATA CmMatchData[MaxVersion][ECmMax]=

{

{//NON

{

{0, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{0, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

{//WIN2000

{

{1, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{1, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{1, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{1, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{1, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{1, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

{//WINXPSP1

{

{2, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{2, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{2, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{2, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{2, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{2, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

{//WINXPSP3

{

{3, 0},

0, NULL, NCmQueryValueKey, 0xFFFFFFFF,

{

0x44,0x89,0x64,0x24,0x20,0x4D,0x8B,0xCE,0x44,0x8B,0xC3,0x48,0x8D,0x94,0x24,0x30,

0x01,0x00,0x00,0x4C,0x8B,0x64,0x24,0x48,0x4C,0x89,0x64,0x24,0x40,0x49,0x8B,0xCC,

},

},

{

{3, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x01,0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x44,0x89,0x74,0x24,0x20,0x4D,0x8B,0xCD,

0x44,0x8B,0x84,0x24,0x58,0x01,0x00,0x00,0x48,0x8D,0x54,0x24,0x48,0x48,0x8B,0xCE,

},

},

{

{3, 2},

0, NULL, NCmDeleteValueKey, 0xFFFFFFFF,

{

0x28,0x44,0x24,0x40,0x66,0x0F,0x7F,0x84,0x24,0xC0,0x00,0x00,0x00,0x4C,0x8B,0x84,

0x24,0x10,0x01,0x00,0x00,0x48,0x8D,0x94,0x24,0xC0,0x00,0x00,0x00,0x48,0x8B,0xCE,

},

},

{

{3, 3},

0, NULL, NCmDeleteKey, 0xFFFFFFFF,

{

0x49,0x3B,0xCD,0x74,0x0A,0xF6,0x41,0x04,0x80,0x0F,0x85,0xE2,0x44,0x0B,0x00,0x41,

0x3A,0xF5,0x0F,0x85,0x93,0x44,0x0B,0x00,0x41,0x3B,0xFD,0x7C,0x0F,0x48,0x8B,0xCB,

},

},

{

{3, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0x4C,0x89,0x74,0x24,0x20,0x45,0x8B,0xCC,0x44,0x8B,0x84,0x24,0x58,0x01,0x00,0x00,

0x4C,0x8B,0x64,0x24,0x50,0x49,0x8B,0xD4,0x48,0x8B,0x74,0x24,0x40,0x48,0x8B,0xCE,

},

},

{

{3, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x00,0x00,0x89,0x44,0x24,0x20,0x4C,0x8B,0x8C,0x24,0x38,0x01,0x00,0x00,0x44,0x8B,

0xC6,0x8B,0x94,0x24,0x28,0x01,0x00,0x00,0x4C,0x8B,0x64,0x24,0x50,0x49,0x8B,0xCC,

},

},

},

{//WINVISTA

{

{4, 0},

0, NULL, NCmQueryValueKey, 0xFFFFFFFF,

{

0x4C,0x89,0x6C,0x24,0x28,0x44,0x89,0x64,0x24,0x20,0x4D,0x8B,0xCE,0x44,0x8B,0xC7,

0x48,0x8D,0x94,0x24,0x40,0x01,0x00,0x00,0x4C,0x8B,0x64,0x24,0x60,0x49,0x8B,0xCC,

},

},

{

{4, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x48,0x8B,0x84,0x24,0x20,0x01,0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x44,0x89,0x6C,

0x24,0x20,0x4D,0x8B,0xCC,0x44,0x8B,0xC6,0x48,0x8D,0x54,0x24,0x48,0x48,0x8B,0xCF,

},

},

{

{4, 2},

0, NULL, NCmDeleteValueKey, 0xFFFFFFFF,

{

0x28,0x44,0x24,0x40,0x66,0x0F,0x7F,0x84,0x24,0xB0,0x00,0x00,0x00,0x4C,0x8B,0x84,

0x24,0x00,0x01,0x00,0x00,0x48,0x8D,0x94,0x24,0xB0,0x00,0x00,0x00,0x48,0x8B,0xCF,

},

},

{

{4, 3},

0, NULL, NCmDeleteKey, 0xFFE7F3FF,

{

0x49,0x3B,0xCD,0x74,0x0A,0xF6,0x41,0x04,0x80,0x0F,0x85,0x58,0x38,0x0E,0x00,0x41,

0x3A,0xF5,0x0F,0x85,0x06,0x38,0x0E,0x00,0x41,0x3B,0xDD,0x7C,0x1D,0x48,0x8B,0xCF,

},

},

{

{4, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0x30,0x89,0x74,0x24,0x28,0x4C,0x89,0x74,0x24,0x20,0x45,0x8B,0xCD,0x45,0x8B,0xC7,

0x48,0x8B,0x7C,0x24,0x50,0x48,0x8B,0xD7,0x48,0x8B,0x74,0x24,0x40,0x48,0x8B,0xCE,

},

},

{

{4, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x24,0x28,0x44,0x89,0x64,0x24,0x20,0x4C,0x8B,0xCE,0x45,0x8B,0xC7,0x44,0x8B,0xAC,

0x24,0x68,0x01,0x00,0x00,0x41,0x8B,0xD5,0x48,0x8B,0x74,0x24,0x50,0x48,0x8B,0xCE,

},

},

},

{//WIN7

{

{5, 0},

0, NULL, NCmQueryValueKey, 0xFFFFFFFF,

{

0x4C,0x89,0x6C,0x24,0x28,0x44,0x89,0x64,0x24,0x20,0x4D,0x8B,0xCE,0x44,0x8B,0xC7,

0x48,0x8D,0x94,0x24,0x40,0x01,0x00,0x00,0x4C,0x8B,0x64,0x24,0x60,0x49,0x8B,0xCC,

},

},

{

{5, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x48,0x8B,0x84,0x24,0x30,0x01,0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x44,0x89,0x6C,

0x24,0x20,0x4D,0x8B,0xCC,0x44,0x8B,0xC6,0x48,0x8D,0x54,0x24,0x48,0x48,0x8B,0xCF,

},

},

{

{5, 2},

0, NULL, NCmDeleteValueKey, 0xFFFFFFFF,

{

0x28,0x44,0x24,0x40,0x66,0x0F,0x7F,0x84,0x24,0xB0,0x00,0x00,0x00,0x4C,0x8B,0x84,

0x24,0x00,0x01,0x00,0x00,0x48,0x8D,0x94,0x24,0xB0,0x00,0x00,0x00,0x48,0x8B,0xCF,

},

},

{

{5, 3},

0, NULL, NCmDeleteKey, 0xFFE7F3FF,

{

0x49,0x3B,0xCD,0x74,0x0A,0xF6,0x41,0x04,0x80,0x0F,0x85,0x7A,0x55,0x0D,0x00,0x41,

0x3A,0xF5,0x0F,0x85,0x28,0x55,0x0D,0x00,0x41,0x3B,0xDD,0x7C,0x1D,0x48,0x8B,0xCF,

},

},

{

{5, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0x44,0x89,0x64,0x24,0x28,0x48,0x89,0x7C,0x24,0x20,0x45,0x8B,0xCE,0x45,0x8B,0xC7,

0x48,0x8B,0x7C,0x24,0x50,0x48,0x8B,0xD7,0x48,0x8B,0x74,0x24,0x40,0x48,0x8B,0xCE,

},

},

{

{5, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x24,0x28,0x44,0x89,0x64,0x24,0x20,0x4C,0x8B,0xCE,0x45,0x8B,0xC7,0x44,0x8B,0xAC,

0x24,0x68,0x01,0x00,0x00,0x41,0x8B,0xD5,0x48,0x8B,0x74,0x24,0x50,0x48,0x8B,0xCE,

},

},

},

{//WIN7SP1

{

{6, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x48,0x8D,0x94,0x24,0x30,0x01,0x00,0x00,0x4C,0x8B,0x64,0x24,0x48,0x4C,0x89,0x64,

0x24,0x40,0x49,0x8B,0xCC,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{6, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x4D,0x8B,0xCD,0x44,0x8B,0x84,0x24,0x58,0x01,0x00,0x00,0x48,0x8D,0x54,0x24,0x48,

0x48,0x8B,0xCE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{6, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x4C,0x8B,0x84,0x24,0x10,0x01,0x00,0x00,0x48,0x8D,0x94,0x24,0xC0,0x00,0x00,0x00,

0x48,0x8B,0xCE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{6, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x41,0x3A,0xF5,0x0F,0x85,0x4B,0x37,0x0D,0x00,0x41,0x3B,0xFD,0x7C,0x0F,0x48,0x8B,

0xCB,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{6, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{6, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

{//WIN8

{

{7, 0},

0, NULL, NCmQueryValueKey, 0xDFF7FFFF,

{

0x4C,0x89,0x7C,0x24,0x28,0x44,0x89,0x74,0x24,0x20,0x4D,0x8B,0xCC,0x44,0x8B,0xC7,

0x48,0x8D,0x94,0x24,0x70,0x01,0x00,0x00,0x4C,0x8B,0x74,0x24,0x50,0x49,0x8B,0xCE,

},

},

{

{7, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x89,0x5C,0x24,0x20,0x4C,0x8B,0x4C,0x24,0x60,

0x45,0x8B,0xC5,0x48,0x8D,0x54,0x24,0x50,0x48,0x8B,0x7C,0x24,0x48,0x48,0x8B,0xCF,

},

},

{

{7, 2},

0, NULL, NCmDeleteValueKey, 0xFFFF7FFE,

{

0xE1,0x01,0x0F,0x28,0x44,0x24,0x40,0x66,0x0F,0x7F,0x84,0x24,0xE0,0x00,0x00,0x00,

0x4C,0x8B,0xC6,0x48,0x8D,0x94,0x24,0xE0,0x00,0x00,0x00,0x48,0x8B,0x4C,0x24,0x50,

},

},

{

{7, 3},

0, NULL, NCmDeleteKey, 0x009887FF,

{

0xE4,0x01,0x00,0x00,0xFF,0xC0,0x66,0x89,0x81,0xE4,0x01,0x00,0x00,0x66,0x85,0xC0,

0x0F,0x84,0xB8,0x00,0x00,0x00,0x48,0x8B,0x7D,0xA7,0x45,0x8A,0xFD,0x48,0x8B,0xCF,

},

},

{

{7, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0x30,0x89,0x74,0x24,0x28,0x48,0x89,0x7C,0x24,0x20,0x45,0x8B,0xCC,0x45,0x8B,0xC6,

0x48,0x8B,0x7C,0x24,0x50,0x48,0x8B,0xD7,0x48,0x8B,0x74,0x24,0x48,0x48,0x8B,0xCE,

},

},

{

{7, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x00,0x4C,0x89,0x64,0x24,0x28,0x89,0x74,0x24,0x20,0x4D,0x8B,0xCE,0x45,0x8B,0xC7,

0x44,0x8B,0x74,0x24,0x50,0x41,0x8B,0xD6,0x48,0x8B,0x74,0x24,0x58,0x48,0x8B,0xCE,

},

},

},

{//WIN8.1

{

{8, 0},

0, NULL, NCmQueryValueKey, 0xFFFFFFFF,

{

0x01,0x00,0x00,0x4C,0x89,0x6C,0x24,0x28,0x44,0x89,0x7C,0x24,0x20,0x4D,0x8B,0xCC,

0x44,0x8B,0x44,0x24,0x48,0x48,0x8D,0x94,0x24,0x60,0x01,0x00,0x00,0x49,0x8B,0xCE,

},

},

{

{8, 1},

0, NULL, NCmSetValueKey, 0xFFFFFFFF,

{

0xB8,0x00,0x00,0x00,0x48,0x89,0x44,0x24,0x28,0x44,0x89,0x74,0x24,0x20,0x4C,0x8B,

0x4C,0x24,0x68,0x44,0x8B,0xC7,0x48,0x8D,0x54,0x24,0x50,0x48,0x8B,0x4C,0x24,0x60,

},

},

{

{8, 2},

0, NULL, NCmDeleteValueKey, 0xFFFFFFFF,

{

0xE1,0x01,0x0F,0x28,0x44,0x24,0x40,0x66,0x0F,0x7F,0x84,0x24,0xE0,0x00,0x00,0x00,

0x4C,0x8B,0xC6,0x48,0x8D,0x94,0x24,0xE0,0x00,0x00,0x00,0x48,0x8B,0x4C,0x24,0x50,

},

},

{

{8, 3},

0, NULL, NCmDeleteKey, 0xFFFFFFFF,

{

0xE4,0x01,0x00,0x00,0xFF,0xC0,0x66,0x89,0x81,0xE4,0x01,0x00,0x00,0x66,0x85,0xC0,

0x0F,0x84,0xBC,0x00,0x00,0x00,0x48,0x8B,0x7D,0xB7,0x45,0x8A,0xFD,0x48,0x8B,0xCF,

},

},

{

{8, 4},

0, NULL, NCmEnumerateKey, 0xFFFFFFFF,

{

0x8B,0x84,0x24,0xA0,0x01,0x00,0x00,0x89,0x44,0x24,0x28,0x4C,0x89,0x74,0x24,0x20,

0x45,0x8B,0xCF,0x45,0x8B,0xC5,0x48,0x8B,0x54,0x24,0x58,0x48,0x8B,0x4C,0x24,0x48,

},

},

{

{8, 5},

0, NULL, NCmEnumerateValueKey, 0xFFFFFFFF,

{

0x4C,0x24,0x58,0x48,0x39,0x5C,0x24,0x60,0x0F,0x85,0x7E,0xC0,0x19,0x00,0x4C,0x89,

0x6C,0x24,0x28,0x89,0x44,0x24,0x20,0x4D,0x8B,0xCE,0x44,0x8B,0xC6,0x41,0x8B,0xD7,

},

},

},

{//WIN10

{

{9, 0},

0, NULL, NCmQueryValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{9, 1},

0, NULL, NCmSetValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{9, 2},

0, NULL, NCmDeleteValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{9, 3},

0, NULL, NCmDeleteKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{9, 4},

0, NULL, NCmEnumerateKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

{

{9, 5},

0, NULL, NCmEnumerateValueKey, 0x00000000,

{

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

},

},

},

},

## 5.8 获取DeviceObject对象类型

POBJECT\_TYPE GetDeviceObjectType()

{

UNICODE\_STRING UAcpi;

UNICODE\_STRING UFilePath;

NTSTATUS status;

PDRIVER\_OBJECT pDrvObj = NULL;

PDEVICE\_OBJECT pDevObj = NULL;

HANDLE FileHandle = NULL;

POBJECT\_TYPE ObjectType = NULL;

RtlInitUnicodeString(&UAcpi,L"\\Driver\\ACPI");

status = ObReferenceObjectByName(&UAcpi,OBJ\_CASE\_INSENSITIVE|OBJ\_KERNEL\_HANDLE,NULL,0,\*IoDriverObjectType,KernelMode,NULL,&pDrvObj);

if(NT\_SUCCESS(status) && pDrvObj && pDrvObj->DeviceObject)

{

pDevObj = pDrvObj->DeviceObject;

}

if(pDevObj)

{

ObjectType = (POBJECT\_TYPE)((PUCHAR)pDevObj-16);

}

if(pDrvObj)

{

ObDereferenceObject(pDrvObj);

pDrvObj = NULL;

}

if(FileHandle)

{

ZwClose(FileHandle);

FileHandle = NULL;

}

return ObjectType;

}