FRANK BOLDEWIN'S
WWW.RECONSTRUCTER.ORG

```
push    Z
call    sub_672B3730
add     esp, 0Ch
test    eax, eax
jnz     short loc_672B5428
lea     edx, [esp+110h+LibFileName]
push    edx
call    sub_672B35F0
mov     edi, off_672CA058
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb
not     ecx
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
cmp     eax, /Eh
jnz     loc_672B5455
lea     ecx, [esp+110h+LibFileName]
push    104h
push    ecx
push    2
call    sub_672B3730
add     esp, 0Ch
test    eax, eax
jnz     short loc_672B5428
lea     edx, [esp+110h+LibFi
push    edx
call    sub_672B35F0
mov     edi, off_672CA058
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb
not     ecx
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
```

Hunting rootkits with Windbg v1.1

Frank Boldewin

Scope of this Talk

- In the next few slides the audience learns how to hunt for rootkits with Windbg
- To get a good overview of the different ways how rootkits hide itself from being recognized several techniques from rootkits like Runtime2, Rustock.B, Alipop, Stuxnet as well as TDL3 and TDL4 are introduced
- Of course the techniques used to detect a special rootkit are not limited to the shown cases. ;-)
- Prerequisites are a good understanding about Windows internals and basic Windbg skills

Finding SSDT hooks

- The SSDT is a data array in kernel memory, that stores pointers to the native API functions of Windows, e.g. NtCreateFile
- These functions are handled in NTOSKRNL
- Older rootkits used to hook some distinctive functions to hide its files or registry entries when queried from usermode
- Almost every run-of-the-mill antirootkit tool is able to detect such hooks today

Finding SSDT hooks

- **Viewing the SSDT manually**

```
kd> dds poi(nt!KeServiceDescriptorTable) L poi(nt!KeServiceDescriptorTable+8)
80501030  8059849a nt!NtAcceptConnectPort
80501034  805e5666 nt!NtAccessCheck
80501038  805e8ec4 nt!NtAccessCheckAndAuditAlarm
8050103c  805e5698 nt!NtAccessCheckByType
80501040  805e8efe nt!NtAccessCheckByTypeAndAuditAlarm
80501044  805e56ce nt!NtAccessCheckByTypeResultList
80501048  805e8f42 nt!NtAccessCheckByTypeResultListAndAuditAlarm
8050104c  805e8f86 nt!NtAccessCheckByTypeResultListAndAuditAlarmByHandle
80501050  8060a5da nt!NtAddAtom
80501054  8060b84e nt!NtQueryBootOptions
80501058  805e0a08 nt!NtAdjustGroupsToken
8050105c  805e0660 nt!NtAdjustPrivilegesToken
80501060  805c9684 nt!NtAlertResumeThread
80501064  805c9634 nt!NtAlertThread
80501068  8060ac00 nt!NtAllocateLocallyUniqueId
8050106c  805aa088 nt!NtAllocateUserPhysicalPages
...
```

Pointers to functions

Finding Shadow SSDT hooks

- The Shadow SSDT is another array and stores pointers to functions in the Win32k.sys
- To view its entries we first have to switch to a GUI process context and reload the symbols for the specific module

```
!process 0 0 winlogon.exe
    PROCESS 81ebf6f8  SessionId: .....
.process /p 81ebf6f8
.reload
```

Finding Shadow SSDT hooks

```
kd> dds poi(nt!KeServiceDescriptorTableShadow+10) L poi(nt!KeServiceDescriptorTableShadow+18)
bf997600  bf934ffe win32k!NtGdiAbortDoc
bf997604  bf946a92 win32k!NtGdiAbortPath
bf997608  bf8bf295 win32k!NtGdiAddFontResourceW
bf99760c  bf93e718 win32k!NtGdiAddRemoteFontToDC
bf997610  bf9480a9 win32k!NtGdiAddFontMemResourceEx
bf997614  bf935262 win32k!NtGdiRemoveMergeFont
bf997618  bf935307 win32k!NtGdiAddRemoteMMInstanceToDC
bf99761c  bf839cb5 win32k!NtGdiAlphaBlend
bf997620  bf9479d0 win32k!NtGdiAngleArc
bf997624  bf933a9d win32k!NtGdiAnyLinkedFonts
bf997628  bf947fc8 win32k!NtGdiFontIsLinked
bf99762c  bf90e7e0 win32k!NtGdiArcInternal
...
```

Pointers to functions

Finding Shadow SSDT hooks

- To find SSDT and Shadow SSDT hooks automatically we can use a Windbg script from Lionel d'Hauenens of Laboskopia

```
kd> $$><script\@@init_cmd.wdbg

SysecLabs Windbg Script : Ok :)
('al' for display all commands)
kd> al
  Alias            Value
 -------          -------
 !!display_all_gdt              $$><script\display_all_gdt.wdbg;
 !!display_all_idt              $$><script\display_all_idt.wdbg;
 !!display_all_msrs             $$><script\display_all_msrs.wdbg;
 !!display_current_gdt          $$><script\display_current_gdt.wdbg;
 !!display_current_idt          $$><script\display_current_idt.wdbg;
 !!display_current_msrs         $$><script\display_current_msrs.wdbg;
 !!display_system_call          $$><script\display_system_call.wdbg;
 !!hide_current_process         $$><script\hide_current_process.wdbg;
 !!save_all_reports             $$><script\save_all_reports.wdbg;
 !!search_hidden_process        $$><script\search_hidden_process.wdbg;
 !@check_msr_and_printf_name    $$><script\check_msr_and_printf_name.wdbg;
 !@display_gdt                  $$><script\display_gdt.wdbg;
 !@display_idt                  $$><script\display_idt.wdbg;
 !@display_msrs                 $$><script\display_msrs.wdbg;
 !@display_thread_process_info  $$><script\display_thread_process_info.wdbg;
 !@get_debug_mode               $$><script\get_debug_mode.wdbg;
 !@get_original_ntcall          $$><script\get_original_ntcall.wdbg;
 !@get_original_win32kcall      $$><script\get_original_win32kcall.wdbg;
 !@get_system_version           $$><script\get_system_version.wdbg;
 !@hide_process                 $$><script\hide_process.wdbg;
 !@is_hidden_process            $$><script\is_hidden_process.wdbg;
 !@printf_state_name_of_thread  $$><script\printf_state_name_of_thread.wdbg;
```

Runtime2 Rootkit – Finding SSDT/Shadow SSDT hooks with a Windbg script

```
kd> !!display_system_call

******************
* Current Table *
******************

ServiceDescriptor n°0
---------------------
    ServiceTable              : nt!KiServiceTable (80501030)
    ParamTableBase            : nt!KiArgumentTable (805014a4)
    NumberOfServices          : 0000011c

    Index  Args  Check  System call
    -----  ----  -----  -----------
    0000   0006   OK    nt!NtAcceptConnectPort (8059849a)
    0001   0008   OK    nt!NtAccessCheck (805e5666)
    0002   000B   OK    nt!NtAccessCheckAndAuditAlarm (805e8ec4)
    0003   000B   OK    nt!NtAccessCheckByType (805e5698)
    0004   0010   OK    nt!NtAccessCheckByTypeAndAuditAlarm (805e8efe)
    ....
    ....
    ....
    0041   0002  HOOK-> runtime2+0x24d8 (f74be4d8) ##### Original -> nt!NtDeleteValueKey (80619232)
    0042   000A   OK    nt!NtDeviceIoControlFile (8056d312)
    0043   0001   OK    nt!NtDisplayString (806078aa)
    0044   0007   OK    nt!NtDuplicateObject (805b21f0)
    0045   0006   OK    nt!NtDuplicateToken (805e18a6)
    0046   0002   OK    nt!NtQueryBootOptions (8060b84e)
    0047   0006  HOOK-> runtime2+0x200a (f74be00a) ##### Original -> nt!NtEnumerateKey (80619412)
    0048   0003   OK    nt!NtEnumerateSystemEnvironmentValuesEx (8060b310)
    0049   0006  HOOK-> runtime2+0x21ca (f74be1ca) ##### Original -> nt!NtEnumerateValueKey (8061967c)
    004A   0002   OK    nt!NtExtendSection (805a7c00)
    004B   0006   OK    nt!NtFilterToken (805e1a52)
    004C   0003   OK    nt!NtFindAtom (8060a844)
```

WinDbg Script command from Laboskopia Collection.

http://www.laboskopia.com/download/SysecLabs-Windbg-Script.zip

Rustock.B Rootkit – SYSENTER_EIP hook

- The SYSENTER_EIP (MSR 0x176) usually points to KiFastCallEntry to serve requests from the usermode to access native functions in the SSDT
- This pointer gets hooked by the Rustock.B rootkit
- If Sysenter gets called Rustock checks in its own SDT table if a function is hooked or not. Non hooked native functions have a null pointer. Hooked functions have a pointer to its own handler.
- To avoid easy hook detections the Sysenter_EIP address points to the same module (NTOSKRNL.EXE) as KiFastCallEntry.
- It overwrites a textstring „FATAL_UNHANDLED_HARD_ERROR" with a 5 bytes jump to its real rootkit code.

Rustock.B Rootkit – SYSENTER_EIP hook

```
kd> rdmsr 0x176
msr[176] = 00000000`806b9741
kd> !address 806b9741
  804d7000 - 001f7000
          Usage       KernelSpaceUsageImage
          ImageName   ntkrnlpa.exe

kd> dc 806b9741
806b9741  8576ffe9 4c444e76 485f4445 5f445241  ..v.VNDLED_HARD_
806b9751  4f525245 000a0d52 1c000000 4e000000  ERROR.........N
806b9761  41505f4f 5f534547 49415641 4c42414c  O_PAGES_AVAILABL
806b9771  000a0d45 18000000 50000000 4c5f4e46  E...........PFN_L
806b9781  5f545349 52524f43 0d545055 1c00000a  IST_CORRUPT.....
806b9791  4e000000 5f534944 45544e49 4c414e52  ...NDIS_INTERNAL
806b97a1  5252455f 0a0d524f 24000000 50000000  _ERROR.....$...P
806b97b1  5f454741 4c554146 4e495f54 4e4f4e5f  AGE_FAULT_IN_NON
kd> u 806b9741 L1
nt!_NULL_IMPORT_DESCRIPTOR <PERF> (nt+0x1e2741):
806b9741 e9ff768576      jmp     f6f10e45
kd> u f6f10e45 Le
f6f10e45 60              pushad
f6f10e46 9c              pushfd
f6f10e47 0fa0            push    fs
f6f10e49 1e              push    ds
f6f10e4a 66368b1d834cf1f6 mov     bx,word ptr ss:[0F6F14C83h]
f6f10e52 668ee3          mov     fs,bx
f6f10e55 66368b1dd34cf1f6 mov     bx,word ptr ss:[0F6F14CD3h]
f6f10e5d 668edb          mov     ds,bx
f6f10e60 e84affffff      call    f6f10daf
f6f10e65 1f              pop     ds
f6f10e66 0fa1            pop     fs
f6f10e68 9d              popfd
f6f10e69 61              popad
f6f10e6a ff25234df1f6    jmp     dword ptr ds:[0F6F14D23h]
```

Rustock.B Rootkit – SYSENTER_EIP hook

Another Laboskopia Windbg command shows us the hook automatically

```
kd> !!display_current_msrs
```

Another Laboskopia Script Command

```
###################################
# Model-Specific Registers (MSRs) #
###################################

Processor 00

IA32_P5_MC_ADDR                msr[00000000] = 0
IA32_P5_MC_TYPE                msr[00000001] = 0
IA32_MONITOR_FILTER_LINE_SIZE  msr[00000006] = 0
IA32_TIME_STAMP_COUNTER       *msr[00000010] = 00000124`f733f704
IA32_PLATFORM_ID               msr[00000017] = 0
IA32_APIC_BASE                *msr[0000001B] = 00000000`fee00900
MSR_EBC_HARD_POWERON           msr[0000002A] = 0
MSR_EBC_SOFT_POWERON           msr[0000002B] = 0
MSR_EBC_FREQUENCY_ID           msr[0000002C] = 0
IA32_BIOS_UPDT_TRIG            msr[00000079] = 0
IA32_BIOS_SIGN_ID             *msr[0000008B] = 00000054`00000000
IA32_MTRRCAP                  *msr[000000FE] = 00000000`00000508
IA32_SYSENTER_CS              *msr[00000174] = 00000000`00000008
IA32_SYSENTER_ESP             *msr[00000175] = 00000000`f8ac6000
IA32_SYSENTER_EIP             *msr[00000176] =  ->#  HOOK #<- 00000000`806b9741  nt!_NULL_IMPORT_DESCRIPTOR <PERF> (nt+0x1e2741) (806b9741)
                                                 => Original : nt!KiFastCallEntry (8053c710)
```

Rustock.B Rootkit – Finding hidden registry entries

- To find the hidden registry entries Rustock uses to survive a reboot, we walk the windows hive with the „!reg" command and its parameters
- A hive is a logical group of keys, subkeys, and values in the registry that has a set of supporting files + backup copies
- Hives are stored as files on disk
- Next to standard hives every user has his own hives file

Rustock.B Rootkit – Finding hidden registry entries

- Table of standard hives and their supporting files

Registry hive	Supporting files
HKEY_CURRENT_CONFIG	System, System.alt, System.log, System.sav
HKEY_CURRENT_USER	Ntuser.dat, Ntuser.dat.log
HKEY_LOCAL_MACHINE\SAM	Sam, Sam.log, Sam.sav
HKEY_LOCAL_MACHINE\Security	Security, Security.log, Security.sav
HKEY_LOCAL_MACHINE\Software	Software, Software.log, Software.sav
HKEY_LOCAL_MACHINE\System	System, System.alt, System.log, System.sav
HKEY_USERS\.DEFAULT	Default, Default.log, Default.sav

Rustock.B Rootkit – Finding hidden registry entries

```
kd> !reg hivelist

-------------------------------------------------------------------------------------------------------------
| HiveAddr |Stable Length|Stable Map|Volatile Length|Volatile Map|MappedViews|PinnedViews|U(Cnt)| BaseBlock | FileName
-------------------------------------------------------------------------------------------------------------
| e1cb9560 |       1000  | e1cb95c0 |          0    |  00000000  |        1  |        0  |     0| e1ba7000  | \Microsoft\Windows\UsrClass.dat
| e1aec008 |       b1000 | e1aec068 |       2000    |  e1aec144  |       38  |        0  |     0| e1bfb000  | nstellungen\karlchen\ntuser.dat
| e1771970 |       1000  | e17719d0 |          0    |  00000000  |        1  |        0  |     0| e17f2000  | \Microsoft\Windows\UsrClass.dat
| e17d4ad8 |       37000 | e17d4b38 |       1000    |  e17d4c14  |       14  |        0  |     0| e17eb000  | llungen\LocalService\ntuser.dat
| e177a758 |       1000  | e177a7b8 |          0    |  00000000  |        1  |        0  |     0| e17c7000  | \Microsoft\Windows\UsrClass.dat
| e1737b60 |       37000 | e1737bc0 |       1000    |  e1737c9c  |       14  |        0  |     0| e178d000  | ungen\NetworkService\ntuser.dat
| e15ee540 |       bef000| e1425000 |       4000    |  e15ee67c  |      256  |        0  |     0| e1411000  | emRoot\System32\Config\SOFTWARE
| e15ee9e8 |       3a000 | e15eea48 |          0    |  00000000  |       15  |        0  |     0| e1410000  | temRoot\System32\Config\DEFAULT
| e1606740 |       5000  | e16067a0 |          0    |  00000000  |        2  |        0  |     0| e140a000  | \SystemRoot\System32\Config\SAM
| e160f758 |       a000  | e160f7b8 |       1000    |  e160f894  |        3  |        0  |     0| e1408000  | emRoot\System32\Config\SECURITY
| e12d4160 |       d000  | e12d41c0 |       4000    |  e12d429c  |        0  |        0  |     0| e12d5000  | <NONAME>
| e1035b60 |       2a0000| e1037000 |       1f000   |  e1035c9c  |       92  |       11  |     0| e1036000  | SYSTEM
| e102d008 |       1000  | e102d068 |       1000    |  e102d144  |        0  |        0  |     0| e102e000  | <NONAME>
-------------------------------------------------------------------------------------------------------------

kd> !reg openkeys e1035b60

Index 3:     9193bdfd kcb=e1ad2368 cell=000dfba8 f=00200001 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\PDRELI
             19fb21c2 kcb=e1acfea8 cell=000db700 f=00200002 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\NTLMSSP
Index 4:     c14e9165 kcb=e1bfc440 cell=00283848 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\SERVICEMODELSERVICE 3.0.0.0
             2a52752a kcb=e1d9eea8 cell=0002d020 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\DEVICECLASSES\{9EA331FA-B91B-45F8-9285-BD2BC77AF
             df479d2a kcb=e1b73c20 cell=00226310 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\ENUM\ROOT\LEGACY_HGFS
Index 6:     09d3e7bf kcb=e1ac87e8 cell=00276a38 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\DEVICECLASSES\{A5DCBF10-6530-11D2-901F-00C04FB95
Index 7:     7bd6c2b1 kcb=e1b5add0 cell=00276f20 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\ENUM\USB\VID_0E0F&PID_0003&MI_00\6&29CBCB69&0&0000\LOGCO
Index 8:     d61e0619 kcb=e1bd22d8 cell=80018598 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\USBCCGP\ENUM
             5f24d968 kcb=e1ad0950 cell=00106480 f=00200004 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\WMDMPMSN
             0b7791de kcb=e1a9ed40 cell=80016358 f=00200001 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\DEVICECLASSES\{65E8773E-8F56-11D0-A3B9-00A0C9223
Index b:     86ebdeef kcb=e1722998 cell=0003fa28 f=00200004 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\LSA\KERBEROS
Index d:     626e81fa kcb=e1acdea8 cell=000aff18 f=00200004 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\DMIO
Index e:     dfe04cec kcb=e1ac8008 cell=8001a0e0 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\CONTROL\DEVICECLASSES\{378DE44C-56EF-11D1-BC8C-00A0C9140
....
Index 39a:   3b494cc3 kcb=e1b5ecf8 cell=0026c880 f=00200000 \REGISTRY\MACHINE\SYSTEM\CONTROLSET001\SERVICES\PE386
```

Rustock.B Rootkit – Finding hidden registry entries

```
kd> !reg cellindex e1035b60 0026c880

Map = e1037000 Type = 0 Table = 1 Block = 6c Offset = 880
MapTable     = e103a000
BlockAddress = dc9ad000

pcell:  dc9ad884
```

```
kd> !reg valuelist e1035b60 dc9ad884

Dumping ValueList of Key <pe386> :

[Idx] [ValAddr]   [ValueName]

[   0]      dc7ae09c    Type
[   1]      dc7ae0bc    Start
[   2]      dc9afd3c    ErrorControl
[   3]      dc9ad8dc    ImagePath
[   4]      dc9ad94c    DisplayName
[   5]      dc9563ac    Group
[   6]      dc9c1d4c    ExtParam

 Use '!reg kvalue <ValAddr>' to dump the value
```

Rustock.B Rootkit – Finding the Hidden Registry Entry

```
kd> !reg kvalue dc9ad8dc

Signature: CM_KEY_VALUE_SIGNATURE (kv)
Name       : ImagePath {compressed}
DataLength: 44
Data       : 26c900  [cell index]
Type       : 2
```

```
kd> !reg cellindex e1035b60 26c900

Map = e1037000 Type = 0 Table = 1 Block = 6c Offset = 900
MapTable     = e103a000
BlockAddress = dc9ad000

pcell:  dc9ad904
```

```
kd> dc dc9ad904
dc9ad904  003f005c 005c003f 003a0043 0057005c  \.?.?.\.C.:.\.W.
dc9ad914  004e0049 004f0044 00530057 0073005c  I.N.D.O.W.S.\.s.
dc9ad924  00730079 00650074 0033006d 003a0032  y.s.t.e.m.3.2.:.
dc9ad934  007a006c 00330078 002e0032 00790073  l.z.x.3.2...s.y.
dc9ad944  00000073 ffffffd8 000b6b76 0000002e  s........vk.....
dc9ad954  00280d10 00000001 00730001 70736944  ..(.......s.Disp
dc9ad964  4e79616c 00656d61 00770020 fffffff0  layName. .w.....
dc9ad974  0001686c 00282f38 e465e2d0 fffffbb0  lh..8/(...e.....
```

Rustock.B Rootkit – pIofCallDriver Hook

- Hooks at pIofCallDriver are often used to filter special IRP requests to drivers
- Rustock filters any attempt to directly communicate with NTFS.SYS or FASTFAT.SYS. These files are hidden, can't be copied, nor overwritten or renamed

```
kd> u poi(poi(iofcalldriver+2))
f6f0f89d 56              push    esi
f6f0f89e 57              push    edi
f6f0f89f 8bf9            mov     edi,ecx
f6f0f8a1 8b7708          mov     esi,dword ptr [edi+8]
f6f0f8a4 3b35ab4df1f6    cmp     esi,dword ptr ds:[0F6F14DABh]
f6f0f8aa 7509            jne     f6f0f8b5
f6f0f8ac 52              push    edx
f6f0f8ad 57              push    edi
kd> ln f6f0f89d
kd> !address f6f0f89d
address f6f0f89d not found in any known Kernel Address Range ----
```

Rustock.B Rootkit – IDT hooks

- The Interrupt Descriptor Table (IDT) is a structure which is used when dispatching interrupts
- Interrupts can interrupt an execution of a program to to handle an event
- Interrupts could be a result of a hardware signal or software based using the INT instruction
- The IDT descriptor table can handle 256 entries
- The descriptor to the table can be written with the instruction LIDT and read with SIDT
- Rustock hooks INT 2Eh, which is usually pointing to KiSystemService, a Zw* functions dispatcher and handler for usermode INT 2Eh calls on old hardware not supporting fastcalls via the SYSENTER command

Rustock.B Rootkit – INT 2Eh

- Rustock hooks INT 2Eh to communicate between usermode and kernelmode components
- The „IDT" command shows us the pointer to the handler. KiSystemService is ok, otherwise it's hooked

```
kd> !idt 2e

Dumping IDT:

2e:   806b973c nt!_NULL_IMPORT_DESCRIPTOR <PERF> (nt+0x1e273c)

kd> !address 806b973c
  804d7000 - 001f7000
            Usage       KernelSpaceUsageImage
            ImageName   ntkrnlpa.exe

kd> u 806b973c L1
nt!_NULL_IMPORT_DESCRIPTOR <PERF> (nt+0x1e273c):
806b973c e9d8768576      jmp     f6f10e19
kd> u f6f10e19 L10
f6f10e19 60              pushad
f6f10e1a 9c              pushfd
f6f10e1b 0fa0            push    fs
f6f10e1d 1e              push    ds
f6f10e1e 66368b1d834cf1f6 mov     bx,word ptr ss:[0F6F14C83h]
f6f10e26 668ee3          mov     fs,bx
f6f10e29 66368b1dd34cf1f6 mov     bx,word ptr ss:[0F6F14CD3h]
f6f10e31 668edb          mov     ds,bx
f6f10e34 e876ffffff      call    f6f10daf
f6f10e39 1f              pop     ds
f6f10e3a 0fa1            pop     fs
f6f10e3c 9d              popfd
f6f10e3d 61              popad
f6f10e3e ff257f4cf1f6    jmp     dword ptr ds:[0F6F14C7Fh]
```

Alipop Rootkit – GDT Callgate

- A callgate is a mechanism in Intel x86 arch to change privilege level of the CPU
- The Alipop rootkit installs such a callgate to execute code with the highest privilege (Ring 0) from usermode (Ring 3) without the need to have a driver, e.g. by calling DeviceIOControl
- Callgate usage works by executing "call far ptr <addr>" from usermode code
- Installation of the callgate is done by the bootkit part of Alipop
- Other malware seen in the wild used \Device\PhysicalMemory to install a callgate in the GDT (works only on older windows versions)

ALIPOP Rootkit – GDT Callgate

```
kd> dg 0 3f0
                                                    P Si Gr Pr Lo
Sel     Base     Limit     Type                     l ze an es ng Flags
----  -------- -------- ---------- - -- -- -- -- --------
0000  00ffdf0a 0000dbbb TSS16 Busy 2 Nb By P  Nl 000000c3
0008  00000000 ffffffff Code RE    0 Bg Pg P  Nl 00000c9a
0010  00000000 ffffffff Data RW    0 Bg Pg P  Nl 00000c92
0018  00000000 ffffffff Code RE    3 Bg Pg P  Nl 00000cfa
0020  00000000 ffffffff Data RW    3 Bg Pg P  Nl 00000cf2
0028  80042000 000020ab TSS32 Busy 0 Nb By P  Nl 0000008b
0030  ffdff000 00001fff Data RW    0 Bg Pg P  Nl 00000c92
0038  00000000 00000fff Data RW Ac 3 Bg By P  Nl 000004f3
0040  00000400 0000ffff Data RW    3 Nb By P  Nl 000000f2
0048  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
0050  80549100 00000068 TSS32 Avl  0 Nb By P  Nl 00000089
0058  80549168 00000068 TSS32 Avl  0 Nb By P  Nl 00000089
0060  00022f30 0000ffff Data RW    0 Nb By P  Nl 00000092
0068  000b8000 00003fff Data RW    0 Nb By P  Nl 00000092
0070  ffff7000 000003ff Data RW    0 Nb By P  Nl 00000092
0078  80400000 0000ffff Code RE    0 Nb By P  Nl 0000009a
0080  80400000 0000ffff Data RW    0 Nb By P  Nl 00000092
0088  00000000 00000000 Data RW    0 Nb By P  Nl 00000092
0090  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
0098  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00A0  825d8930 00000068 TSS32 Avl  0 Nb By P  Nl 00000089
00A8  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00B0  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00B8  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00C0  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00C8  00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
 ...
 ...
 ...
03C0  00008003 0000f3c8 <Reserved> 0 Nb By Np Nl 00000000
03C8  00008003 0000f3d0 <Reserved> 0 Nb By Np Nl 00000000
03D0  00008003 0000f3d8 <Reserved> 0 Nb By Np Nl 00000000
03D8  00008003 0000f3e0 <Reserved> 0 Nb By Np Nl 00000000
03E0  800003e8 0003f000 C-GATE32   3 Nb By P  Nl 000000ec
03E8  00000000 ffffffff Code RE    0 Bg Pg P  Nl 00000c9a
03F0  00008003 0000f3f8 <Reserved> 0 Nb By Np Nl 00000000
```

ALIPOP Rootkit – GDT Callgate

```
kd> u 80000000+3f000 L2
8003f000 bbdb0adfff      mov     ebx,0FFDF0ADBh
8003f005 c3              ret
kd> u ffdf0adb L25
ffdf0adb c8000000        enter   0,0
ffdf0adf 31c0            xor     eax,eax
ffdf0ae1 60              pushad
ffdf0ae2 8b5508          mov     edx,dword ptr [ebp+8]
ffdf0ae5 bb00704d80      mov     ebx,offset nt!_imp__VidInitialize <PERF> (nt+0x0) (804d7000)
ffdf0aea 8b4b3c          mov     ecx,dword ptr [ebx+3Ch]
ffdf0aed 8b6c0b78        mov     ebp,dword ptr [ebx+ecx+78h]
ffdf0af1 01dd            add     ebp,ebx
ffdf0af3 8b4d18          mov     ecx,dword ptr [ebp+18h]
ffdf0af6 8b7d20          mov     edi,dword ptr [ebp+20h]
ffdf0af9 01df            add     edi,ebx
ffdf0afb e331            jecxz   ffdf0b2e
ffdf0afd 8b37            mov     esi,dword ptr [edi]
ffdf0aff 01de            add     esi,ebx
ffdf0b01 af              scas    dword ptr es:[edi]
ffdf0b02 52              push    edx
ffdf0b03 ac              lods    byte ptr [esi]
ffdf0b04 29c2            sub     edx,eax
ffdf0b06 c1ca07          ror     edx,7
ffdf0b09 85c0            test    eax,eax
ffdf0b0b 75f6            jne     ffdf0b03
ffdf0b0d 85d2            test    edx,edx
ffdf0b0f 5a              pop     edx
ffdf0b10 e0eb            loopne  ffdf0afd
ffdf0b12 751a            jne     ffdf0b2e
ffdf0b14 f7d1            not     ecx
ffdf0b16 8b5524          mov     edx,dword ptr [ebp+24h]
ffdf0b19 034d18          add     ecx,dword ptr [ebp+18h]
ffdf0b1c 01da            add     edx,ebx
ffdf0b1e 668b044a        mov     ax,word ptr [edx+ecx*2]
ffdf0b22 8b4d1c          mov     ecx,dword ptr [ebp+1Ch]
ffdf0b25 01d9            add     ecx,ebx
ffdf0b27 031c81          add     ebx,dword ptr [ecx+eax*4]
ffdf0b2a 895c241c        mov     dword ptr [esp+1Ch],ebx
ffdf0b2e 61              popad
ffdf0b2f c9              leave
ffdf0b30 c20400          ret     4
kd> dc ffdf0b30 L50
ffdf0b30  7e0004c2 3b008000 5cffdf0b 65005200  ...~...;...\.R.e
ffdf0b40  69006700 74007300 79007200 4d005c00  .g.i.s.t.r.y.\.M
ffdf0b50  63006100 69006800 65006e00 53005c00  .a.c.h.i.n.e.\.S
ffdf0b60  46004f00 57005400 52004100 5c004500  .O.F.T.W.A.R.E.\
ffdf0b70  69004d00 72006300 73006f00 66006f00  .M.i.c.r.o.s.o.f
ffdf0b80  5c007400 69005700 64006e00 77006f00  .t.\.W.i.n.d.o.w
ffdf0b90  5c007300 75004300 72007200 6e006500  .s.\.C.u.r.r.e.n
ffdf0ba0  56007400 72006500 69007300 6e006f00  .t.V.e.r.s.i.o.n
ffdf0bb0  52005c00 6e007500 06000000 c3000800  .\.R.u.n........
ffdf0bc0  71ffdf0b 00005100 3a004300 57005c00  ...q.Q..:.C.\.W
ffdf0bd0  4e004900 4f004400 53005700 61005c00  .I.N.D.O.W.S.\.a
ffdf0be0  69006c00 65002e00 65007800 26000000  .l.i...e.x.e...&
ffdf0bf0  f7002800 5cffdf0b 79005300 74007300  .(....\.S.y.s.t
ffdf0c00  6d006500 6f005200 74006f00 61005c00  .e.m.R.o.o.t.\.a
ffdf0c10  69006c00 65002e00 65007800 00000000  .l.i...e.x.e....
ffdf0c20  0003e800 ff0000ec 000000ff 4d00cf9a  ............M
ffdf0c30  0300905a 04000000 ff000000 b80000ff  Z...............
ffdf0c40  00000000 40000000 00000000 00000000  ........@.......
ffdf0c50  00000000 00000000 00000000 00000000  ................
ffdf0c60  00000000 00000000 e0000000 0e000000  ................
```

TDL3 Rootkit – ATAPI IRP hooks

- The TDL3 rootkit usually infects the ATAPI driver with a small loader for the real rootkit code in the PE resource area of atapi.sys and changes the entrypoint to its loader code
- The real rootkit part is being stored encrypted on disk sectors
- The loader uses low level disk operations to read the sectors, decrypts the mini TDL file system and starts the real rootkit code
- To hide and protect its sectors TDL3 uses IRP hooking in ATAPI.SYS

TDL3 Rootkit – ATAPI IRP hooks

```
kd> !drvobj \driver\atapi 2
Driver object (82180878) is for:
 \Driver\atapi
DriverEntry:   f84e75f7 atapi!GsDriverEntry
DriverStartIo: f84d97c6 atapi!IdePortStartIo
DriverUnload:  f84e3204 atapi!IdePortUnload
AddDevice:     f84e1300 atapi!ChannelAddDevice

Dispatch routines:
[00] IRP_MJ_CREATE                      f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[01] IRP_MJ_CREATE_NAMED_PIPE           f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[02] IRP_MJ_CLOSE                       f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[03] IRP_MJ_READ                        f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[04] IRP_MJ_WRITE                       f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[05] IRP_MJ_QUERY_INFORMATION           f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[06] IRP_MJ_SET_INFORMATION             f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[07] IRP_MJ_QUERY_EA                    f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[08] IRP_MJ_SET_EA                      f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[09] IRP_MJ_FLUSH_BUFFERS               f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0a] IRP_MJ_QUERY_VOLUME_INFORMATION    f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0b] IRP_MJ_SET_VOLUME_INFORMATION      f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0c] IRP_MJ_DIRECTORY_CONTROL           f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0d] IRP_MJ_FILE_SYSTEM_CONTROL         f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0e] IRP_MJ_DEVICE_CONTROL              f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[0f] IRP_MJ_INTERNAL_DEVICE_CONTROL     f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[10] IRP_MJ_SHUTDOWN                    f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[11] IRP_MJ_LOCK_CONTROL                f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[12] IRP_MJ_CLEANUP                     f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[13] IRP_MJ_CREATE_MAILSLOT             f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[14] IRP_MJ_QUERY_SECURITY              f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[15] IRP_MJ_SET_SECURITY                f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[16] IRP_MJ_POWER                       f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[17] IRP_MJ_SYSTEM_CONTROL              f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[18] IRP_MJ_DEVICE_CHANGE               f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[19] IRP_MJ_QUERY_QUOTA                 f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[1a] IRP_MJ_SET_QUOTA                   f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34
[1b] IRP_MJ_PNP                         f84db9f2    atapi!PortPassThroughZeroUnusedBuffers+0x34

kd> ln f84db9f2
(f84db9be)   atapi!PortPassThroughZeroUnusedBuffers+0x34   |  (f84dba06)   atapi!InitHwExtWithIdentify
```

TDL3 Rootkit – ATAPI IRP hooks

```
kd> u f84db9f2 L2
f84db9f2 a10803dfff      mov     eax,dword ptr ds:[FFDF0308h]
f84db9f7 ffa0fc000000    jmp     dword ptr [eax+0FCh]
kd> u poi(poi(ffdf0308)+fc) L10
81c5fe31 55              push    ebp
81c5fe32 8bec            mov     ebp,esp
81c5fe34 8b450c          mov     eax,dword ptr [ebp+0Ch]
81c5fe37 8b4d08          mov     ecx,dword ptr [ebp+8]
81c5fe3a 83ec0c          sub     esp,0Ch
81c5fe3d 53              push    ebx
81c5fe3e 8b5860          mov     ebx,dword ptr [eax+60h]
81c5fe41 a10803dfff      mov     eax,dword ptr ds:[FFDF0308h]
81c5fe46 3b4808          cmp     ecx,dword ptr [eax+8]
81c5fe49 56              push    esi
81c5fe4a 57              push    edi
81c5fe4b ba54020000      mov     edx,254h
81c5fe50 7557            jne     81c5fea9
81c5fe52 8b4318          mov     eax,dword ptr [ebx+18h]
81c5fe55 85c0            test    eax,eax
81c5fe57 7450            je      81c5fea9
```

TDL3 Rootkit – Shared Memory structure (Kernel-/User mode)

- To share information with its usermode components TDL3 uses the structure KUSER_SHARED_DATA
- This structure is accessable from kernel at address 0xFFDF0000 and is mapped to userspace at 0x7FFE0000
- Kernel mode has read/write access to this structure, usermode has only read access
- At KUSER_SHARED_DATA+0308h (SystemCallPad) TDL3 stores a pointer to an own structure
- This structure stores a bunch of things like kernelbase, original ATAPI IRPs, TDL3 FS start, path to its config file …

TDL3 Rootkit – Shared Memory structure (Kernel-/User mode)

```
kd> !kuser
_KUSER_SHARED_DATA at ffdf0000
TickCount:    fa00000 * 0000a906 (0:00:11:16.093)
TimeZone Id: 1
ImageNumber Range: [14c .. 14c]
Crypto Exponent: 0
SystemRoot: 'C:\WINDOWS'
kd> dt _kuser_shared_data ffdf0000
nt!_KUSER_SHARED_DATA
   +0x000 TickCountLow     : 0xa906
   +0x004 TickCountMultiplier : 0xfa00000
   +0x008 InterruptTime    : _KSYSTEM_TIME
   +0x014 SystemTime       : _KSYSTEM_TIME
   +0x020 TimeZoneBias     : _KSYSTEM_TIME
   +0x02c ImageNumberLow   : 0x14c
   +0x02e ImageNumberHigh  : 0x14c
   +0x030 NtSystemRoot     : [260] 0x43
   +0x238 MaxStackTraceDepth : 0
   +0x23c CryptoExponent   : 0
   +0x240 TimeZoneId       : 1
   +0x244 Reserved2        : [8] 0
   +0x264 NtProductType    : 1 ( NtProductWinNt )
   +0x268 ProductTypeIsValid : 0x1 ''
   +0x26c NtMajorVersion   : 5
   +0x270 NtMinorVersion   : 1
   +0x274 ProcessorFeatures : [64]  ""
   +0x2b4 Reserved1        : 0x7ffeffff
   +0x2b8 Reserved3        : 0x80000000
   +0x2bc TimeSlip         : 0
   +0x2c0 AlternativeArchitecture : 0 ( StandardDesign )
   +0x2c8 SystemExpirationDate : _LARGE_INTEGER 0x0
   +0x2d0 SuiteMask        : 0x110
   +0x2d4 KdDebuggerEnabled : 0x1 ''
   +0x2d5 NXSupportPolicy  : 0x2 ''
   +0x2d8 ActiveConsoleId  : 0
   +0x2dc DismountCount    : 0
   +0x2e0 ComPlusPackage   : 0xffffffff
   +0x2e4 LastSystemRITEventTickCount : 0x9d99b
   +0x2e8 NumberOfPhysicalPages : 0x1ff7c
   +0x2ec SafeBootMode     : 0 ''
   +0x2f0 TraceLogging     : 0
   +0x2f8 TestRetInstruction : 0xc3
   +0x300 SystemCall       : 0x7c91eb8b
   +0x304 SystemCallReturn : 0x7c91eb94
   +0x308 SystemCallPad    : [3] 0xffffffff`81cc8550
   +0x320 TickCount        : _KSYSTEM_TIME
   +0x320 TickCountQuad    : 0
   +0x330 Cookie           : 0x5763f1c1
```

TDL3 Rootkit – Shared Memory structure (Kernel-/User mode)

```
kd> dc 81cc8550 Lb0
81cc8550  81c5c000 804d7000 82158040 00000000  .....pM.@.......
81cc8560  7fffa000 00000002 00000025 00000001  ........%.......
81cc8570  f7540c08 00000000 00040001 00000000  ..T.............
81cc8580  81cc8580 81cc8580 00000000 013fffd0  .............?.
81cc8590  013fff84 65313164 30353766 3633632d  ..?.d11ef750-c36
81cc85a0  34342d63 382d3963 2d623563 37653634  c-44c9-8c5b-46e7
81cc85b0  63616662 32313264 00000000 00000000  bfacd212........
81cc85c0  00000000 00000000 00000000 00000000  ................
81cc85d0  00000000 00000000 00000000 f84dc572  ............r.M.
81cc85e0  804f320e f84dc572 804f320e 804f320e  .20.r.M..20..20.
81cc85f0  804f320e 804f320e 804f320e 804f320e  .20..20..20..20.
81cc8600  804f320e 804f320e 804f320e 804f320e  .20..20..20..20.
81cc8610  804f320e f84dc592 f84d87b4 804f320e  .20...M...M..20.
81cc8620  804f320e 804f320e 804f320e 804f320e  .20..20..20..20.
81cc8630  804f320e f84dc5bc f84e3164 804f320e  .20...M.d1N..20.
81cc8640  804f320e 804f320e f84e3130 81c5fe31  .20..20.01N.1...
81cc8650  82180878 82192af0 00000200 00000000  x....*..........
81cc8660  00000007 00a934b8 00000180 00000080  .....4..........
81cc8670  81c5c004 00000000 00a934b9 00000000  .........4......
81cc8680  00000200 81c5c084 00000000 00a934ba  ..............4..
81cc8690  00000000 00000100 81c5c284 00000000  ................
81cc86a0  00a93407 00000168 00000004 f84d2168  .4..h.......h!M.
81cc86b0  00000000 00a93407 0000016c 00000004  .....4..l.......
81cc86c0  f84d216c 00000000 00a93407 000000f8  l!M......4......
81cc86d0  00000004 f84d20f8 00000000 00a93407  ..... M......4..
81cc86e0  00000128 00000004 f84d2128 00000000  (.......(!M.....
81cc86f0  00000000 00000000 00000000 00000000  ................
81cc8700  00000000 00000000 00000000 00000000  ................
81cc8710  00000000 00000000 00000000 00000000  ................
81cc8720  00000000 00000000 00000000 00000000  ................
81cc8730  00000000 00000000 00000000 00000000  ................
81cc8740  00000000 00000000 00000000 00000000  ................
81cc8750  00000000 00000000 00000000 00000000  ................
81cc8760  00000000 00000000 00000000 00000000  ................
81cc8770  00000000 00000000 00000000 00000000  ................
81cc8780  00000000 00000000 00000000 00000000  ................
81cc8790  00000000 00000000 00000000 00000000  ................
81cc87a0  00000000 00690076 00650065 006e0078  ....v.i.e.e.x.n.
81cc87b0  00710071 005c0000 00650044 00690076  q.q...\.D.e.v.i.
81cc87c0  00650063 0049005c 00650064 0049005c  c.e.\.I.d.e.\.I.
81cc87d0  00650064 006f0050 00740072 005c0031  d.e.P.o.r.t.1.\.
81cc87e0  00690076 00650065 006e0078 00710071  v.i.e.e.x.n.q.q.
81cc87f0  00000000 00000000 00000000 00000000  ................
81cc8800  00000000 00000000 00000000 00000000  ................
```

TDL3 Rootkit – TDL mini FS (file system)

```
kd> dc 81c5c000 180
81c5c000  334c4454 00000000 00000000 00000000  TDL3............
81c5c010  00010000 00000010 80000018 00000000  ................
81c5c020  00000000 00000000 00010000 00000001  ................
81c5c030  80000030 00000000 00000000 00000000  0...............
81c5c040  00010000 00000409 00000048 000163e0  ........H....c..
81c5c050  0000038c 00000000 00000000 00000000  ................
81c5c060  00000000 0034038c 00560000 005f0053  ......4...V.S._.
81c5c070  00450056 00530052 004f0049 005f004e  V.E.R.S.I.O.N._.
81c5c080  004e0049 004f0046 00000000 feef04bd  I.N.F.O.........
81c5c090  00010000 00050001 0a280884 00050001  .........(......
81c5c0a0  0a280884 0000003f 00000000 00040004  ..(.?...........
81c5c0b0  00000003 00000007 00000000 00000000  ................
81c5c0c0  000002ec 00530001 00720074 006e0069  ......S.t.r.i.n.
81c5c0d0  00460067 006c0069 00490065 0066006e  g.F.i.l.e.I.n.f.
81c5c0e0  0000006f 000002c8 00300001 00300034  o.........0.4.0.
81c5c0f0  00300039 00420034 00000030 0016004c  9.0.4.B.0...L...
81c5c100  00430001 006d006f 00610070 0079006e  ..C.o.m.p.a.n.y.
81c5c110  0061004e 0065006d 00000000 0069004d  N.a.m.e.....M.i.
81c5c120  00720063 0073006f 0066006f 00200074  c.r.o.s.o.f.t. .
81c5c130  006f0043 00700072 0072006f 00740061  C.o.r.p.o.r.a.t.
81c5c140  006f0069 0000006e 00160054 00460001  i.o.n...T.....F.
81c5c150  006c0069 00440065 00730065 00720063  i.l.e.D.e.s.c.r.
81c5c160  00700069 00690074 006e006f 00000000  i.p.t.i.o.n.....
81c5c170  00440049 002f0045 00540041 00500041  I.D.E./.A.T.A.P.
81c5c180  00200049 006f0050 00740072 00440020  I. .P.o.r.t. .D.
81c5c190  00690072 00650076 00000072 00290072  r.i.v.e.r...r.).
81c5c1a0  00460001 006c0069 00560065 00720065  ..F.i.l.e.V.e.r.
81c5c1b0  00690073 006e006f 00000000 002e0035  s.i.o.n.....5...
81c5c1c0  002e0031 00360032 00300030 0032002e  1...2.6.0.0...2.
81c5c1d0  00380031 00200030 00780028 00730070  1.8.0. .(.x.p.s.
81c5c1e0  005f0070 00700073 005f0032 00740072  p._.s.p.2._.r.t.
81c5c1f0  002e006d 00340030 00380030 00330030  m...0.4.0.8.0.3.
```

TDL3 Rootkit – Traces in the system worker threads

- Drivers requiring delayed processing usually use a work item, using IoQueueWorkItem with a pointer to its callback routine
- When a system worker thread processes the queued item it gets removed and the callback gets invoked
- System worker threads run in the system process context (PID 4)
- TDL3 rootkit is using work items as well
- Whenever work items have been processed or other system threads have been created this leaves traces on the callstack
- As TDL3 does not belong to any known module, the process thread view informs us about this problem

TDL3 Rootkit – Traces in the system worker threads

```
kd> !process 0 0 system
PROCESS 821c8830  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 00af9000  ObjectTable: e1000cc0  HandleCount: 251.
    Image: System

kd> .process /p 821c8830
Implicit process is now 821c8830
.cache forcedecodeuser done
kd> .reload
Connected to Windows XP 2600 x86 compatible target at (Mon Jan 17 17:07:40.941 2011 (GMT+1)), ptr64 FALSE
Loading Kernel Symbols
...............................................................
.....................................................
Loading User Symbols

Loading unloaded module list
.............
kd> !process 821c8830
PROCESS 821c8830  SessionId: none  Cid: 0004    Peb: 00000000  ParentCid: 0000
    DirBase: 00af9000  ObjectTable: e1000cc0  HandleCount: 251.
    Image: System
    VadRoot 821c7298 Vads 4 Clone 0 Private 3. Modified 1316. Locked 0.
    DeviceMap e1004418
    Token                             e10017e8
    ElapsedTime                       00:11:16.093
    UserTime                          00:00:00.000
    KernelTime                        00:00:20.234
    QuotaPoolUsage[PagedPool]         0
    QuotaPoolUsage[NonPagedPool]      0
    Working Set Sizes (now,min,max)  (59, 0, 345) (236KB, 0KB, 1380KB)
    PeakWorkingSetSize                510
    VirtualSize                       1 Mb
    PeakVirtualSize                   2 Mb
    PageFaultCount                    4133
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      7

        THREAD 821c6898  Cid 0004.0030  Teb: 00000000 Win32Thread: 00000000 WAIT: (UserRequest) KernelMode Alertable
            f8af1c68  NotificationEvent
        IRP List:
            820fbe30: (0006,0094) Flags: 00000000  Mdl: 820c5dd8
        Not impersonating
        DeviceMap                 e1004418
        Owning Process            0       Image:         <Unknown>
        Attached Process          821c8830       Image:         System
        Wait Start TickCount      41699          Ticks: 1571 (0:00:00:24.546)
        Context Switch Count      233
        UserTime                  00:00:00.000
        KernelTime                00:00:00.218
        Start Address nt!ExpWorkerThread (0x80533cd0)
        Stack Init f8af2000 Current f8af1bf8 Base f8af2000 Limit f8aef000 Call 0
        Priority 12 BasePriority 12 PriorityDecrement 0 DecrementCount 0
        ChildEBP RetAddr
        f8af1c10 8050017a nt!KiSwapContext+0x2e (FPO: [Uses EBP] [0,0,4])
        f8af1c1c 804f99be nt!KiSwapThread+0x46 (FPO: [0,0,0])
        f8af1c44 81c5d28f nt!KeWaitForSingleObject+0x1c2 (FPO: [5,5,4])
WARNING: Frame IP not in any known module. Following frames may be wrong.
        f8af1c98 8054418b 0x81c5d28f
        f8af1d04 81c5f892 nt!ExAllocatePoolWithTag+0x10b (FPO: [3,18,0])
        f8af1d74 80533dd0 0x81c5f892
```

TDL4 Rootkit – Finding TDL4 with its invalid device object

```
kd> !object \Device\HardDisk0\dr0
Object: 8217cab8  Type: (821a0788) Device
    ObjectHeader: 8217caa0 (old version)
    HandleCount: 0  PointerCount: 3
    Directory Object: e1341458  Name: DR0
kd> !devstack 8217cab8
  !DevObj   !DrvObj            !DevExt   ObjectName
  82169e08  \Driver\PartMgr    82169ec0
> 8217cab8  \Driver\Disk       8217cb70  DR0
Invalid type for DeviceObject 0x8216bd98
kd> dt nt! DEVICE_OBJECT 8216bd98
   +0x000 Type             : 0
   +0x002 Size             : 0x234
   +0x004 ReferenceCount   : 0
   +0x008 DriverObject     : 0x8217d4c8 _DRIVER_OBJECT
   +0x00c NextDevice       : 0x82144040 _DEVICE_OBJECT
   +0x010 AttachedDevice   : 0x8217cab8 _DEVICE_OBJECT
   +0x014 CurrentIrp       : (null)
   +0x018 Timer            : (null)
   +0x01c Flags            : 0x5050
   +0x020 Characteristics  : 0x100
   +0x024 Vpb              : (null)
   +0x028 DeviceExtension  : 0x8216be50
   +0x02c DeviceType       : 7
   +0x030 StackSize        : 1 ''
   +0x034 Queue            : __unnamed
   +0x05c AlignmentRequirement : 1
   +0x060 DeviceQueue      : _KDEVICE_QUEUE
   +0x074 Dpc              : _KDPC
   +0x094 ActiveThreadCount : 0
   +0x098 SecurityDescriptor : 0xe101e198
   +0x09c DeviceLock       : _KEVENT
   +0x0ac SectorSize       : 0
   +0x0ae Spare1           : 1
   +0x0b0 DeviceObjectExtension : 0x8216bfd0 _DEVOBJ_EXTENSION
   +0x0b4 Reserved         : (null)
kd> dt nt! DRIVER_OBJECT 8217d4c8
   +0x000 Type             : 0
   +0x002 Size             : 168
   +0x004 DeviceObject     : 0x8208e540 _DEVICE_OBJECT
   +0x008 Flags            : 4
   +0x00c DriverStart      : 0xf84d2000
   +0x010 DriverSize       : 0x17480
   +0x014 DriverSection    : 0x821d9c28
   +0x018 DriverExtension  : 0x8216c7f0 _DRIVER_EXTENSION
   +0x01c DriverName       : _UNICODE_STRING "\Driver\atapi"
   +0x024 HardwareDatabase : 0x8066e9d8 _UNICODE_STRING "\REGISTRY\MACHINE\HARDWARE\DESCRIPTION\SYSTEM"
   +0x028 FastIoDispatch   : (null)
   +0x02c DriverInit       : 0xf84e75f7     long  atapi!GsDriverEntry+0
   +0x030 DriverStartIo    : 0xf84d97c6     void  atapi!IdePortStartIo+0
   +0x034 DriverUnload     : 0xf84e3204     void  atapi!IdePortUnload+0
   +0x038 MajorFunction    : [28] 0x81cc2446     long  +ffffffff81cc2446
kd> !address 81cc2446
  80fed000 - 01213000
           Usage      KernelSpaceUsageNonPagedPool
```

TDL4 Rootkit – ATAPI DriverStartIO hook

- TDL4 rootkit hooks the ATAPI driver as well, but in a lower level way than its precedessor
- As more and more tools were easily able to dump its files even from usermode via IOCTL_SCSI_PASS_THROUGH_DIRECT calls directly to the port device, TDL4 changed the hook method to DriverStartIO
- This makes it harder to dump the TDL4 files

TDL4 Rootkit – ATAPI DriverStartIO hook

```
kd> !drvobj \driver\atapi 2
Driver object (8216c748) is for:
 \Driver\atapi
DriverEntry:   f84e75f7 atapi!GsDriverEntry
DriverStartIo: 81cc2292
DriverUnload:  f84e3204 atapi!IdePortUnload
AddDevice:     f84e1300 atapi!ChannelAddDevice

Dispatch routines:
[00] IRP_MJ_CREATE                      f84dc572    atapi!IdePortAlwaysStatusSuccessIrp
[01] IRP_MJ_CREATE_NAMED_PIPE           804f320e    nt!IopInvalidDeviceRequest
[02] IRP_MJ_CLOSE                       f84dc572    atapi!IdePortAlwaysStatusSuccessIrp
[03] IRP_MJ_READ                        804f320e    nt!IopInvalidDeviceRequest
[04] IRP_MJ_WRITE                       804f320e    nt!IopInvalidDeviceRequest
[05] IRP_MJ_QUERY_INFORMATION           804f320e    nt!IopInvalidDeviceRequest
[06] IRP_MJ_SET_INFORMATION             804f320e    nt!IopInvalidDeviceRequest
[07] IRP_MJ_QUERY_EA                    804f320e    nt!IopInvalidDeviceRequest
[08] IRP_MJ_SET_EA                      804f320e    nt!IopInvalidDeviceRequest
[09] IRP_MJ_FLUSH_BUFFERS               804f320e    nt!IopInvalidDeviceRequest
[0a] IRP_MJ_QUERY_VOLUME_INFORMATION    804f320e    nt!IopInvalidDeviceRequest
[0b] IRP_MJ_SET_VOLUME_INFORMATION      804f320e    nt!IopInvalidDeviceRequest
[0c] IRP_MJ_DIRECTORY_CONTROL           804f320e    nt!IopInvalidDeviceRequest
[0d] IRP_MJ_FILE_SYSTEM_CONTROL         804f320e    nt!IopInvalidDeviceRequest
[0e] IRP_MJ_DEVICE_CONTROL              f84dc592    atapi!IdePortDispatchDeviceControl
[0f] IRP_MJ_INTERNAL_DEVICE_CONTROL     f84d87b4    atapi!IdePortDispatch
[10] IRP_MJ_SHUTDOWN                    804f320e    nt!IopInvalidDeviceRequest
[11] IRP_MJ_LOCK_CONTROL                804f320e    nt!IopInvalidDeviceRequest
[12] IRP_MJ_CLEANUP                     804f320e    nt!IopInvalidDeviceRequest
[13] IRP_MJ_CREATE_MAILSLOT             804f320e    nt!IopInvalidDeviceRequest
[14] IRP_MJ_QUERY_SECURITY              804f320e    nt!IopInvalidDeviceRequest
[15] IRP_MJ_SET_SECURITY                804f320e    nt!IopInvalidDeviceRequest
[16] IRP_MJ_POWER                       f84dc5bc    atapi!IdePortDispatchPower
[17] IRP_MJ_SYSTEM_CONTROL              f84e3164    atapi!IdePortDispatchSystemControl
[18] IRP_MJ_DEVICE_CHANGE               804f320e    nt!IopInvalidDeviceRequest
[19] IRP_MJ_QUERY_QUOTA                 804f320e    nt!IopInvalidDeviceRequest
[1a] IRP_MJ_SET_QUOTA                   804f320e    nt!IopInvalidDeviceRequest
[1b] IRP_MJ_PNP                         f84e3130    atapi!IdePortDispatchPnp
```

TDL4 Rootkit – Finding the Kernel Callback with a Windbg script

- Rootkits often use kernelcallbacks to get notified when files are loaded, processes or threads are created as well as Registry events occur.
- TDL4 installs a kernelcallback to inject its usermode payload in distinctive windows processes

```
kd> $$><script\callbacks.wdbg

***** Create Process Notify Routine Callbacks ***** (PspCreateProcessNotifyRoutine)
e1542c14  f88bb6ae vmdebug+0x16ae

***** Load Image Notify Routine Callbacks ***** (PspLoadImageNotifyRoutine)
e1ab1c3c  820d98eb

***** Create Thread Notify Routine Callbacks ***** (PspCreateThreadNotifyRoutine)

***** Config Manager/Registry Callbacks ***** (CmpCallBackVector/CallbackListHead)

***** BugCheckCalls Callbacks ***** (KeBugCheckCallbackListHead)
82025108  f83bc5ed NDIS!ndisBugcheckHandler
806dff48  806d57ca hal!HalpBugCheckCallback

***** BugCheckCalls Reason Callbacks ***** (KeBugCheckReasonCallbackListHead)
f8b6fa88  f8b6eac0 mssmbios!SMBiosBugCheckCallbackRoutineSMBios
f8b6fa68  f8b6ea78 mssmbios!SMBiosBugCheckCallbackRoutineRegistry
f8b6faa8  f8b6ea30 mssmbios!SMBiosBugCheckCallbackRoutineACPI
f82b19e8  f82af3e2 VIDEOPRT!pVpBugcheckCallback
82018830  f82d0006 USBPORT!USBPORT_BugCheckSaveData
82018670  f82cff66 USBPORT!USBPORT_BugCheckSaveLog
```

http://www.moonsols.com/wp-content/uploads/downloads/2011/02/callbacks3.txt

TDL4 Rootkit – Dropper dumping after TDL4 infection (before reboot)

```
kd> !pool 81cc18eb
Pool page 81cc18eb region is Nonpaged pool
*81cc0000 : large page allocation, Tag is None, size is 0xf000 bytes
          Pooltag None : call to ExAllocatePool
kd> s -[17]sa 81cc0000 lf000
81cc004d  "!This program cannot be run in D"
81cc5150  "\\?\globalroot%S"
81cc5170  "%s (x64)"
81cc517c  "* (x64)"
81cc533c  "version"
81cc534c  "cfg.ini"
81cc558e  "RtlImageDirectoryEntryToData"
81cc55ae  "RtlImageNtHeader"
81cc55c2  "ExFreePoolWithTag"
81cc55d6  "MmMapLockedPagesSpecifyCache"
81cc560a  "KeDelayExecutionThread"
81cc5624  "KeUnstackDetachProcess"
81cc563e  "ZwClose"
81cc5648  "ZwUnmapViewOfSection"
81cc5660  "KeInsertQueueApc"
81cc5674  "KeInitializeApc"
81cc5686  "FsRtlAllocatePool"
81cc569a  "_snprintf"
81cc56a6  "ZwAllocateVirtualMemory"
81cc56c0  "ZwMapViewOfSection"
81cc56d6  "ZwCreateSection"
81cc56e8  "ZwOpenFile"
81cc56f6  "RtlInitUnicodeString"
81cc570e  "_snwprintf"
81cc571c  "KeStackAttachProcess"
81cc5734  "KeSetEvent"
81cc5742  "PsGetProcessImageFileName"
81cc575e  "ZwOpenProcess"
81cc5624  "KeUnstackDetachProcess"
81cc5c38  "ntoskrnl.exe"
81cc624b  "Invalid partition table"
81cc6263  "Error loading operating system"
81cc6282  "Missing operating system"
81cc62a6  "cfg.ini"
81cc62e6  "bckfg.tmp"
81cc6306  "cmd.dll"
81cc6326  "ldr16"
81cc6346  "ldr32"
81cc6366  "ldr64"
81cc6386  "drv64"
81cc63a6  "cmd64.dll"
81cc7c10  "Fehler beim Laden des Betriebssy"
81cc7c30  "stems"
81cc7c36  "Betriebssystem nicht vorhanden"
81cc7e2b  "Invalid partition table"
81cc7e43  "Error loading operating system"
81cc7e62  "Missing operating system"
kd> .writemem c:\td14.bin 81cc0000 Lf000
Writing f000 bytes...............................
```

TDL4 Rootkit – Dumping injected user mode payload

```
kd> !process 0 1 iexplore.exe
PROCESS 81ccf030  SessionId: 0  Cid: 0208    Peb: 7ffdd000  ParentCid: 0608
    DirBase: 07a802c0  ObjectTable: e1aafeb8  HandleCount: 206.
    Image: IEXPLORE.EXE
    VadRoot 81ef61d8 Vads 140 Clone 0 Private 615. Modified 27. Locked 0.
    DeviceMap e1a1fbc8
    Token                             e1a26040
    ElapsedTime                       00:00:07.796
    UserTime                          00:00:00.078
    KernelTime                        00:00:00.468
    QuotaPoolUsage[PagedPool]         90436
    QuotaPoolUsage[NonPagedPool]      5872
    Working Set Sizes (now,min,max)  (2655, 50, 345) (10620KB, 200KB, 1380KB)
    PeakWorkingSetSize                2655
    VirtualSize                       58 Mb
    PeakVirtualSize                   66 Mb
    PageFaultCount                    4385
    MemoryPriority                    BACKGROUND
    BasePriority                      8
    CommitCharge                      1051

kd> .process /p 81ccf030
Implicit process is now 81ccf030
.cache forcedecodeuser done
kd> .reload
Connected to Windows XP 2600 x86 compatible target at (Tue Jan 18 18:13:15.131 2011 (GMT+1)), ptr64 FALSE
Loading Kernel Symbols
...............................................................
................................................................
Loading User Symbols
...............................................
Loading unloaded module list
.............
kd> !vad 81ef61d8
VAD     level      start      end    commit
820a6db0 ( 5)       250      261        18 Private      EXECUTE_READWRITE
81f86a50 ( 8)       270      270         1 Private      EXECUTE_READWRITE
81e7db18 ( 7)       280      28f         0 Mapped       READWRITE
81f14470 ( 8)       290      2a5         0 Mapped       READONLY
81c70d98 ( 6)       2b0      2ec         0 Mapped       READONLY

kd> dc 250*1000 L90
00250000  00905a4d 00000003 00000004 0000ffff  MZ..............
00250010  000000b8 00000000 00000040 00000000  ........@.......
00250020  00000000 00000000 00000000 00000000  ................
00250030  00000000 00000000 00000000 000000f0  ................
00250040  0eba1f0e cd09b400 4c01b821 685421cd  ........L.!Th
00250050  70207369 72676f72 63206d61 6f6e6e61  is program canno
00250060  65622074 6e757220 206e6920 20534f44  t be run in DOS
00250070  65646f6d 0a0d0d2e 00000024 00000000  mode....$.......
00250080  967bdd5e c515bc1a c515bc1a c515bc1a  ^.{.............
00250090  c580c413 c515bc1b c596c413 c515bc1e  ................
002500a0  c586c413 c515bc14 c514bc1a c515bc8d  ................
002500b0  c548b3d9 c515bc19 c580ee04 c515bc18  ..H.............
002500c0  c59fc413 c515bc17 c584c413 c515bc1b  ................
002500d0  68636952 c515bc1a 00000000 00000000  Rich............
002500e0  00000000 00000000 00000000 00000000  ................
002500f0  00004550 0003014c 4c9b2130 00000000  PE..L...0!.L....
```

TDL4 Rootkit – Finding inline hooks in user mode payload

```
kd> !chkimg -d -v ntdll.dll
Searching for module with expression: ntdll.dll
Will apply relocation fixups to file used for comparison
Will ignore NOP/LOCK errors
Will ignore patched instructions
Image specific ignores will be applied
Comparison image path: c:\Symbols\ntdll.dll\411096D4b7000\ntdll.dll
No range specified

Scanning section:    .text
Size: 501502
Range to scan: 7c911000-7c98b6fe
    7c91deb6-7c91deba  5 bytes - ntdll!ZwProtectVirtualMemory
      [ b8 89 00 00 00:e9 4f 21 14 85 ]
    7c91ea32-7c91ea36  5 bytes - ntdll!NtWriteVirtualMemory (+0xb7c)
      [ b8 15 01 00 00:e9 d3 15 15 85 ]
    7c91eaec-7c91eaf0  5 bytes - ntdll!KiUserExceptionDispatcher (+0xba)
      [ 8b 4c 24 04 8b:e9 1b 15 13 85 ]
Total bytes compared: 225280(44%)
Number of errors: 15
15 errors : ntdll.dll (7c91deb6-7c91eaf0)
kd> !chkimg -d -v mswsock.dll
Searching for module with expression: mswsock.dll
Will apply relocation fixups to file used for comparison
Will ignore NOP/LOCK errors
Will ignore patched instructions
Image specific ignores will be applied
Comparison image path: c:\Symbols\mswsock.dll\4110972440000\mswsock.dll
No range specified

Scanning section:    .text
Size: 168788
Range to scan: 719b1000-719da354
    719b405f-719b4063  5 bytes - mswsock!WSPCloseSocket
      [ 6a 44 68 d8 41:e9 a8 bf 0e 90 ]
    719b4342-719b4346  5 bytes - mswsock!WSPRecv (+0x2e3)
      [ 6a 44 68 b8 44:e9 c5 bc 0c 90 ]
    719b5847-719b584b  5 bytes - mswsock!WSPSend (+0x1505)
      [ 6a 40 68 80 59:e9 c0 a7 0d 90 ]
Total bytes compared: 168788(100%)
Number of errors: 15

Scanning section: SANONTCP
Size: 48458
Range to scan: 719db000-719e6d4a
Total bytes compared: 0(0%)
Number of errors: 0
15 errors : mswsock.dll (719b405f-719b584b)
```

Stuxnet Rootkit – IoRehisterFsRegistrationChange

- Stuxnet mrxnet.sys driver adds a new device object and attaches to the device chain with the objecttype \FileSystem (fastfat, ntfs, cdfs)
- A filesystem registration callback makes it possible to attach to the device chain for each devobj managed by these drvobjs
- This makes it possible to control and intercept IRP requests

Stuxnet Rootkit – IoRegisterFsRegistrationChange

```
kd> $$><script\iocallbacks.wdbg

[*] nt!IopFsNotifyChangeQueueHead
(Driver Info) e100febc  f8494876 sr!SrFsNotification
(Driver Info) e1bd3ee4  f78a69ec mrxnet+0x9ec

[*] nt!IopNotifyShutdownQueueHead
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)

[*] nt!IopCdRomFileSystemQueueHead
(Device Info)
(Device Info)
(Device Info)
(Device Info)
(Device Info)

[*] nt!IopDiskFileSystemQueueHead
(Device Info)
(Device Info)
(Device Info)
(Device Info)

[*] nt!IopTapeFileSystemQueueHead
(Device Info)

[*] nt!IopNetworkFileSystemQueueHead
(Device Info)
(Device Info)
(Device Info)

[*] nt!PnpProfileNotifyList
e100fcdc  80601026 nt!WmipDockUndockEventCallback
e12f548c  f87a16a5 i8042prt!I8xProfileNotificationCallback
```

http://www.moonsols.com/wp-content/uploads/downloads/2011/03/IoCallbacks.txt

FRANK BOLDEWIN'S
WWW.RECONSTRUCTER.ORG

```
push    Z
call    sub_672B3730
add     esp, 0Ch
test    eax, eax
jnz     short loc_672B5428
lea     edx, [esp+110h+LibFileName]
push    edx
call    sub_672B35F0
mov     edi, off_672CA058
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb
not     ecx
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
cmp     eax, 7Eh
jnz     loc_672B5455
lea     ecx, [esp+110h+LibFileName]
push    104h
push    ecx
push    2
call    sub_672B3730
add     esp, 0Ch
test    eax, eax
jnz     short loc_672B5428
lea     edx, [esp+110h+LibFileName]
push    edx
call    sub_672B35F0
mov     edi, off_672CA058
or      ecx, 0FFFFFFFFh
xor     eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb
not     ecx
sub     edi, ecx
mov     esi, edi
mov     ebx, ecx
```

Questions?

Thanks for good discussions and ideas

Michael Hale Ligh

EP_X0FF

Cr4sh

Matthieu Suiche