



```
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, 0xFFFFFFFFh
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+LibFi1M]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

Rustock.C

When a myth comes true

Frank Boldewin

Hack.Lu 2008



Agenda

- The family's history
- How the myth started rolling
- The hunt for answers
- The loader
- The beast itself
 - Protection layers
 - Inside the rootkit
 - The botnet user mode code
- Lessons learned



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

The family's history

- Rustock aka Spambot is able to send spam emails and always used top notch rootkit techniques to hide its tracks
- First version (Rustock.A) appeared in Nov 2005, followed by Rustock.B in July 2006
- Code maintained probably only by one Russian guy, who is known as "pe386" or "ntldr" in the underground
- From a reverse engineers point of view, this malware family was always a challenging task and with every evolution step also the degree of analyzing difficulty increased



```
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, 0xFFFFFFFFh
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+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

How the myth started rolling



```
push    Z
call    sub_672B3730
add    eax, eax
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
cal    sub_672B3730
mov    edi, offset 672CA058
or     eax, eax
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne
not    edx
sub    edx, edx
mov    esi, edi
mov    mov
cmp    eax, /bD
jnz    loc_672B3730
lea    ecx, [esp+110h+LibFileName]
push   ecx
push   push
push   call
add    add
test   test
jnz    jnz
lea    lea
push   push
call   call
add    add
test   test
jnz    jnz
lea    lea
push   push
call   call
add    add
test   test
jnz    jnz
lea    lea
push   push
repne repne
0010:85D491EB 00 02 00 00 00 50 00 00-00 FC 31 00 00 00 FC 09 00 . . . P . . . i . . .
0010:85D491FB 00 52 53 44 53 38 7E AB-BB 70 A5 88 4A 91 F3 09 .RSD$8" . . p. J. .
0010:85D4920B 40 58 2C 6E 7E 03 00 00-00 5A 3A 5C 4E 65 77 50 @X,n~ . . . Z:\NewP
0010:85D4921B 72 6F 6A 65 63 74 73 5C-73 70 61 6D 62 6F 74 5C rojects\spambot\
0010:85D4922B 72 75 73 74 6F 63 6B 2E-63 5C 64 72 69 76 65 72 rustock.c\driver\
0010:85D4923B 5C 61 73 6D 5F 5C 64 72-69 76 65 72 2E 70 64 62 \asm_\driver.pdb'
```

How the myth started rolling

- In Oct 2007 some people reported that a new Rustock version was seen in the wild
- Unfortunately nobody was able to prove this assertion, because of lack of a sample
- After some weeks without success in hunting, most people in the AV-industry claimed it to be myth...
- At least for 8 months. However in May 2008 the AV-company Dr. Web released a small article, giving a few details about the inner workings of Rustock.c as well as a snapshot showing a .pdb string



```
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, 0xFFFFFFFFh
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+LibFileName]
push    edx
call    sub_672B35F0
mov     edi, off_672CA058
or      ecx, 0xFFFFFFFFh
xor     eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb   |
not    ecx
sub    edi, ecx
mov     esi, edi
mov     ebx, ecx
```

The hunt for answers



```
push    Z
call    sub_672B3730
add    eax, eax
test   jnz    short loc_672B5428
lea     edx, [esp+110h+LibFileName]
push   edx
call   sub_672B25D2
mov    edi, offset 672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea     edx, [esp+114h+LibFileName]
repne scasb
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

The hunt for answers

- After some further days a few samples of Rustock.C made the rounds and everyone in the industry started analyzing it
- Unfortunately these samples crashed with a BSOD on every box, right after starting the driver (We will see later why)
- Further an unanswered question was its way of infection as well as...
- Where is the dropper code?
- With help of BFK's huge malware DB it was easy to answer the question for the dropper and its infection way
- Recorded traffic revealed that Rustock.C spread through the Iframe-Cash network aka Russian Business Network



```
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, 0xFFFFFFFFh
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+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

The loader



Loader code protector properties

- Spaghetti-code with polymorphic jumps, e.g.
 - MOV EDI, offset_18030 / ADD EDI, 0F2F25958h / JMP EDI
 - MOV ECX, 0E3242A4h / JMP DWORD PTR [ECX-0E30C17Ch]
 - MOV EBX, 0Ch / XCHG EBX, [ESP+EBX] / RETN 10h
- RC4 encrypted
- aPLib packed
- Unpacked code still spaghetti code structure combined with deliberately unoptimized code, e.g.
 - MOV EAX,1234 -> XOR EAX,EAX / OR EAX,1200 / ADD EAX,34
- Strings like registry paths or IP and port infos are runtime assembled to prevent easy detection
- TDI based Kernel mode socket implementation is used for communication
- No extra antidebug, antidump, antivm ...



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

Loaders inner workings

- Grabs several OS and PCI infos from victims system
 - OS infos are queried from registry
 - PCI infos like PCI to Host Bridge and PCI to ISA Bridge are queried through low level IO port access (CF8/CFC)
- Gathered infos are encrypted with TEA and then send to a fake HTTPS server at 208.66.194.215
- Server crypts the real Rustock.C driver with the victim specific data and sends it back on the same channel
- Loader starts the crypted driver and ends



```
push    Z
call    sub_672B3730
add    P00000000
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
call   sub_672B25C0
mov    edi, offset 872CA058
or
xor
lea
repne scasb
not
sub
mov
mov
cmp
jnz
lea
push
push
push
call
add
test
jnz
lea
push
call
sub_672B3730
add
test
jnz
lea
push
call
sub_672B25C0
or
ecx, 0
xor
eax, e
lea
edx, D
repne scasb
not
ecx
sub
edi, ecx
mov
esi, edi
mov
ebx, ecx
```

Send data illustrated

■ Unencrypted

31 DC 84 9B 25 05 00 00	86 80 90 71	86 80 10 71	1...Z....a...a
64bit TimeStampCounter (RDTSC)	7190 = Device 8086 = Vendor	7110 = Device 8086 = Vendor	
C7 78 9B 47 05 00 00 00-01 00 00 00 28 0A 00 00	Install Date = 28.01.08 18:15:35	5.1 = CurrentVersion 2600 = CurrentBuildNumber	.x.G.....C...
35 00 35 00 33 00 37 00-35 00 2D 00 38 00 36 00 34 00 30 00 2D 00 33 00 32 00-32 00 38 00 35 00 30 00 30 00 36 00 2D 00 32 00 33 00-34 00 35 00 36 00 00 00 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 00 00 00	ProductId	5.5.3.7.5.-6.4: 0.-3.2.2.8.5.0: 6.-2.3.4.5.6: ::.....:.....:.....

■ Encrypted

```
00000000 3b e9 f8 b0 a9 4f 01 d1 58 b9 55 4b 62 18 e8 f5 ;....o.. x.Ukb...
00000010 ab 94 7e ec f3 0d ca 1e 21 65 ee ef 45 35 df ce ..~..... !e..E5..
00000020 c8 81 31 e0 77 68 4f d9 d8 a6 24 35 1d 30 63 65 ..1.who. ..$5.0ce
00000030 da 04 4f 0f 18 6f ec 58 42 ab 3f c3 22 9e b6 9c ..0..o.X B.?."...
00000040 43 ce 79 73 2a b3 e1 27 75 81 11 34 b1 df f8 af C.y$*..' u..4....
00000050 bd a1 38 ac c1 b3 9e 79 56 5f e2 35 e7 12 87 a1 ..8....y V_.5....
00000060 9d cf 3a 3c 47 f9 04 b8 a7 84 46 34 22 68 52 99 ..:<G... .F4"hR.
```

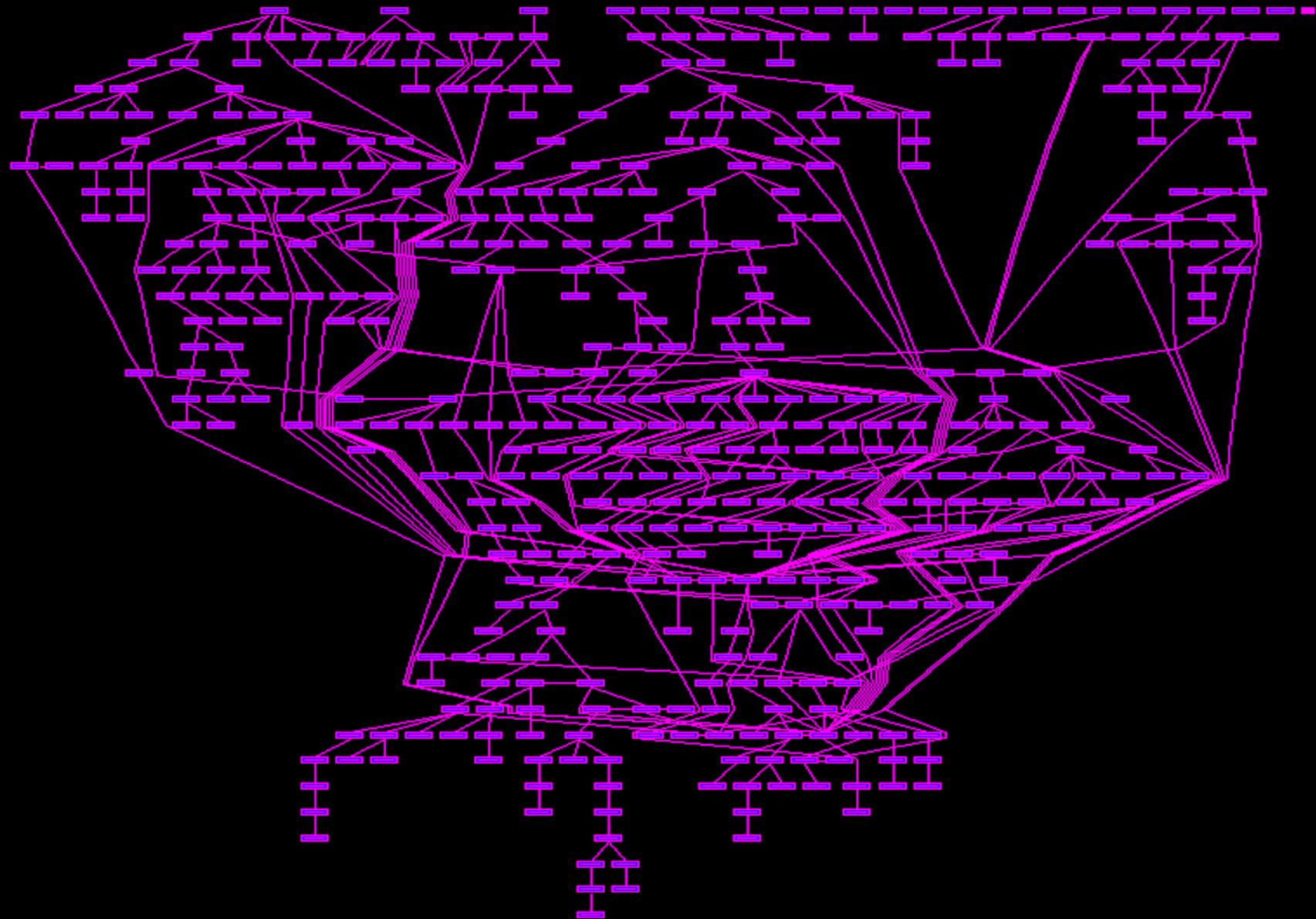


```
push    Z
call    sub_672B3730
add    esp, 0Ch
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, ecx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

The beast itself



Rustock.C Spaghetti-Code





```
push    Z
call    sub_672B3730
add    eax, eax
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push
call
mov    edx, offset loc_672B5428
or     ecx, 0F00010200
xor    eax, eax
lea    edx, [e00010206
repne scasb
not    ecx, 000010208
sub    edi, edi
mov    esi, edi
mov    ebx, edi
cmp    eax, /E00010210
jnz    loc_672B5428
lea    ecx, [e00010214
push   104h
push   ecx
push   2
call   sub_672B5428
add    esp, 000010222
test   eax, eax
jnz    short loc_10222
lea    edx, [e00010224
push   edx
call   sub_672B5428
mov    edi, offset loc_10222
or     ecx, 0F0001022C
xor    eax, eax
lea    edx, [e0001022F
repne scasb
not    ecx, 00010232
sub    edi, edi
mov    esi, edi
mov    ebx, edi
```

■ Easy polymorphic decrypter (Anti AV-signature measure)

```
pusha
mov    ecx, 287h
xor    ebx, ebx
xor    edx, edx
add    ebx, 39E96BF0h ; COD
adc    edx, 0
dec    ecx
jnz    short loc_1020A
mov    esi, offset loc_10233
mov    edi, esi
mov    ecx, 0D6EBh
; COD
mov    eax, ebx
shr    eax, 3
add    edx, eax
xchg   ebx, edx
lodsd
sub    eax, ebx
stosd
dec    ecx
jnz    short loc_10222
popa
; DAT
jmp    JumpToSecondLayer
```



Protection layer 2

- Searches the NTOSKRNL base and stores it
- Builds a checksum over its own buffer and encrypts NTOSKRNL image base value with this DWORD
- When trying to find NtQuerySystemInformation the checksum gets recalculated and decrypts the stored NTOSKRNL image base value. If someone changed the code in the meantime, a wrong image base value leads to BSOD
- Imports are found by using 32-bit hash values, instead of function names
- Allocates memory with ExAllocateMemoryPoolWithTag and copies the majority of its code into this area and directly jumps to layer 3



Protection layer 3

- Overwrites DRx registers
 - DR0-3 (hardware breakpoint detection)
 - DR7 (kernel debugger detection)
- 2nd code checksum trick (modified code leads to BSOD)
- Overwrites whole IDT table with fake handler, for the time of unpacking, to disturb kernel debuggers, which hook INT1 (single stepping + hardware breakpoints) and INT3 (software breakpoints)

```
FakeInterruptHandler:
    push    ebp
    mov     ebp, esp
    sub    esp, 4
    iret
```
- Software BP checks (0xCC)
- Query 8 bytes of PCI information from system (like the loader did)
- Adds 1 dword pre-stored in the buffer and uses these 12 bytes as RC4 decryption key over all 5 PE-sections
- After every PE-section decryption the buffer gets aPLib decompressed



Protection layer 3

- If the 8 bytes of PCI information are different from original ones, decryption fails and system crashes
- Brute forcing the key depends on the machine power and some luck while enumerating through the PCI vendor/device table
- To generate a more random key, 111 empty rounds after RC4init is used
- Imports rebuilding and auto section relocation are also handled in this stage
- Before jumping to the unpacked rootkit code the IDT gets restored to its original state



```
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, 0xFFFFFFFFh
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+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

Inside the rootkit



Inside the rootkit

- Unpacked code still spaghetti code structure combined with deliberately unoptimized code
- Checks the presents of kernel debuggers
 - WinDbg (KdDebuggerEnabled)
 - String-scans in memory for NTICE + Syser traces
- Registers a callback routine with KeRegisterBugCheckCallback, which cleans its memory when KeBugCheck happens
- Code checksum routine
- Software breakpoint checks (0xCC)



Inside the rootkit

- Botnet usermode code, stored in the last PE section, gets injected into winlogon.exe or services.exe under VISTA
- Driver infector
 - Infects a random Microsoft driver listed in HKLM\SYSTEM\CurrentControlSet\Control\Safeboot\Minimal registry path
 - Rustock looks for version information strings inside the binaries before infection (scans for "Microsoft Windows")
- Disinfection is time based, before it infects another MS driver, but can be forced when trying to change an infected binary



Inside the rootkit

- NTOSKRNL hook at _KiFastCallEntry, a very smart way to control all Nt/Zw variants of native functions
 - The hook is protecting usermode botnet component to hide its threads and from being read, written, erased or terminated and to have a communication channel through INT 2Eh, between both rings
 - The following native functions are being hooked:
 - ZwQuerySystemInformation
 - ZwReadVirtualMemory
 - ZwWriteVirtualMemory
 - ZwProtectVirtualMemory
 - ZwCreateThread
 - ZwTerminateThread
 - ZwOpenThread
 - ZwDuplicateObject
 - ZwDelayExecution
 - ZwSetEvent
 - ZwSetInformationThread
 - ZwResumeThread
 - ZwTerminateProcess
 - ZwCreateUserProcess (only on VISTA)
 - ZwCreateThreadEx (only on VISTA)



Inside the rootkit

- NTFS.SYS hooks to fake file size and to notice read/writes on infected driver
 - _NtfsFsdWrite
 - _NtfsFsdRead
 - _NtfsFsdSetInformation
 - _NtfsFastQueryFSInfo
 - _NtfsFsdClose
 - _NtfsFsdCreate
 - _NtfsFsdDispatchWait
 - _NtfsFsdDirectoryControl
- In case of FAT32 the hooks are placed on FASTFAT.SYS



```
push    Z
call    sub_672B3730
add    eax, eax
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
cal    sub_672B25F8
mov    edi, offset_672CA058
or     edx, FFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb
not    ecx
sub    edi, edi
mov    esi, edi
mov    ebx, eax
cmp    eax, loc_672B5455
jnz    loc_672B5455
lea    edx, [esp+110h+LibFileName]
push   edx
push   ecx
push   edx
call   sub_672B3730
add    esp, 2
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+114h+LibFileName]
push   edx
push   ecx
push   edx
call   sub_672B3730
add    esp, 2
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+114h+LibFileName]
push   edx
repne scasb
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

Inside the rootkit

- To prevent local sniffing, also some hooks are placed on IP-based drivers
 - TCPIP.SYS
 - _ARPSendData
 - _TCPDispatch
 - _TCPDispatchInternalDeviceControl
 - ARPClose
 - FreeARPIInterface
 - _ARPRegister
- WANARP.SYS
 - _WANSendPackets



```
push    Z
call    sub_672B3730
add    ECX, ECX
test    eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
```

Inside the rootkit

Inside the rootkit

- Two different types of hooks are used (indirect call + push/ret)

```

or      ecx,    0008:F6A5F508 15607FA6F6 ADC    EAX, HAL!KfReleaseSpinLock
xor      eax,    0008:F6A5F50D FF5708 PUSH   DWORD PTR [EBP+08]
lea      edx,    0008:F6A5F510 8D45FC LEA    EAX, [EBP-04]
repne scasb  0008:F6A5F513 50 PUSH   CALL   [8659BEDC]
not      ecx,    0008:F6A5F514 FF15DCBE5986 CMP   DWORD PTR [EBP-04], 00000103
sub      edi,    0008:F6A5F51A 817DFC03010000 JNZ   +F6A5F529
not      ecx,    0008:F6A5F521 7506 PUSH   EDI, [F6A6?C07]
sub      edi,    0008:F6A5F523 57 CALL   AND,  DWORD PTR [ESI+101,00]
mov      esi,    0008:F6A5F524 E8DE860000 JMP   EB06, [F6A5F535]
mov      ebx,    0008:F6A5F529 83661000 CALL   [HAL!KiReleaseSpinLock]
mov      ebx,    0008:F6A5F52D EB06 POP    EDI, EBX
mov      ebx,    0008:F6A5F52F FF15607FA6F6 CALL   EDI, EBX
mov      ebx,    0008:F6A5F535 5F POP    ESI, EBX
mov      ebx,    0008:F6A5F536 5E POP    EBX
mov      ebx,    0008:F6A5F537 5B LEAVE  \ 4 bytes
cmp      eax,    0008:F6A5F538 C9 RET    0004 indirect call hook
jnz      loc_67, 0008:F6A5F539 C20400
jnz      loc_67, 0008:F6A5F53C 90 NOP
jnz      loc_67, 0008:F6A5F53D 90 NOP
jnz      loc_67, 0008:F6A5F53E 90 NOP

lea      ecx,    (DISPATCH)-KTEB(80551920)-TID(0000)-tcpip!.text+00036188
push    104
push    ecx
push    2
call    sub
add    esp
test   eax
jnz    shor
lea     edx
push   edx
call    sub
mov    edi
or     ecx
xor    eax
lea     edx
repne scasb
not    ecx
sub    edi
mov    esi
mov    ebx

104
0008: 8053C7B7 0F8345FDFFFF JAE   +8053C502
0008: 8053C7BD 83F910 CMP   ECX, 10
0008: 8053C7C0 751A JNZ   +8053C7DC
0008: 8053C7C2 8B0D18F0DFFF MOU   ECX, [FFDFF018]
0008: 8053C7C8 33DB XOR   EBX, EBX
0008: 8053C7C9 0B997000F0000 OR    EBX, [ECX+00000F70]
0008: 8053C7D0 740A JZ    +8053C7DC
0008: 8053C7D2 52 PUSH  EDX
0008: 8053C7D3 50 PUSH  EAX
0008: 8053C7D4 F15C4215580 CALL  [805521C4]
0008: 8053C7DA 58 POP   EAX
0008: 8053C7DB 5A POP   EDX
0008: 8053C7DC FF0538F6DFFF INC   DWORD PTR [FFDFF638]
0008: 8053C7E2 8BF2 MOU   ESI, EDX
0008: 8053C7E4 8B5F0C MOU   EBX, LEDI+0C1
0008: 8053C7E7 33C9 XOR   ECX, ECX
0008: 8053C7E9 8ABC18 MOU   CL, [EBX+EAX]
0008: 8053C7EC 0F PUSH  CS
0008: 8053C7ED 90 NOP
0008: 8053C7EE 83EC04 SUB   ESP, 04
0008: 8053C7F1 C70424FEA7D585 MOU   DWORD PTR [ESP], 85D5A7FE
0008: 8053C7F8 CB RETF
0008: 8053C802 8D457000000000 XRA   EBX, HAL!KfReleaseSpinLock
0008: 8053C804 0F83A8010000 JAE   +8053C9AC
0008: 8053C805 F3A5 REPZ MOUSD
0008: 8053C806 FFD3 CALL  EBX
0008: 8053C808 8BE5 MOU   ESP, EBP
0008: 8053C80A 8B0D24F1DFFF MOU   ECX, [FFDFF124]
0008: 8053C810 8B553C MOU   EDX, [EBP+3C]
0008: 8053C813 899134010000 MOU   [ECX+00000134], EDX
0008: 8053C819 FA CLI
0008: 8053C81A F14570000000200 TEST  DWORD PTR [EBP+70], 00020000 ; "?"
0008: 8053C821 7506 JNZ   +8053C829
0008: 8053C823 F6456C01 TEST  BYTE PTR [EBP+6C], 01
0008: 8053C827 7457 JZ    +8053C880
0008: 8053C829 8B1D24F1DFFF MOU   EBX, [FFDFF124]

(DISPATCH)-KTEB(80551920)-TID(0000)-toskron!.text+00065189

```



```
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, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, 104h
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+LibFileName]
push   edx
call   sub_672B35F0
mov    edi, off_672CA058
or     ecx, 0xFFFFFFFFh
xor    eax, eax
lea    edx, [esp+114h+LibFileName]
repne scasb    |
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

The botnet user mode code



```
push    Z
call    sub_672B3730
add    eax, eax
test   eax, eax
jnz    short loc_672B5428
lea    edx, [esp+110h+LibFileName]
push   edx
call   sub_672B2552
mov    edi, offset FFFFFFFE
or     eax, eax
xor    edx, [esp+114h+LibFileName]
lea    edx, [esp+110h+LibFileName]
repne scasb
not    ecx
sub    edi, ecx
mov    esi, edi
mov    ebx, ecx
```

The botnet user mode code

- The first variants had the name botdll.dll and send spam the classic way using port 25 (SMTP)
- But as more and more SMTP gateways successfully detect such spam bots, a new user mode payload was distributed in march 2008 and changed to HTTP-mode spamming over hotmail with stolen accounts (hotsend.dll)
- Spam templates are downloaded from the C&C server, which are temporarily stored as tmpcode.bin
- Currently it is unknown what malware steals the hotmail accounts involved in spamming
- To communicate with the kernel INT 2Eh is used, to inform about new tasks, e.g. self-disinfection or a new C&C



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

Lessons learned

- Kernel mode driver could easily host other user mode payload, e.g. banking trojans, DDoS client ...
- Without automated deobfuscation scripts, it would be nearly impossible to analyze the code
- Brute forcing would have been impossible, if a stronger encryption had been applied
- Disinfection wouldn't be that easy, if the original driver in the last PE-section would have been better encrypted



```
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, 0xFFFFFFFFh
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    edx, [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, 0xFFFFFFFFh
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 review fly to:

UG North

Elia Florio

Sergei Shevchenko

Lukasz Kwiatak