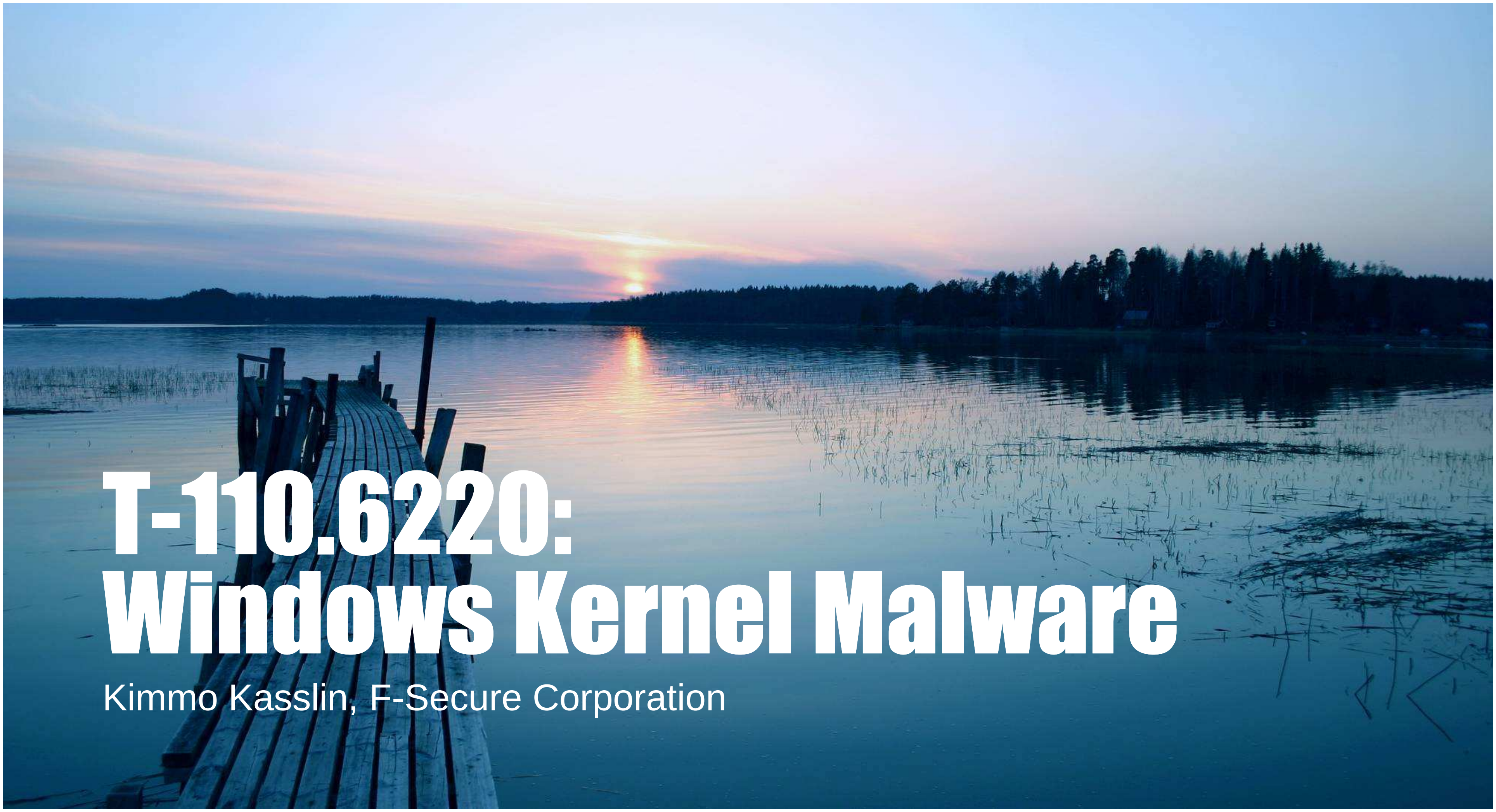


T-110.6220: Windows Kernel Malware

Kimmo Kasslin, F-Secure Corporation

F-SECURE®

BE SURE.

Agenda

- Definition of kernel-mode malware
- History
- Trend and present situation
- Techniques
- Evolution
  - The average Joe
  - Haxdoor, Apropos, Rustock, Srizbi, Mebroot
- Conclusions

Definition

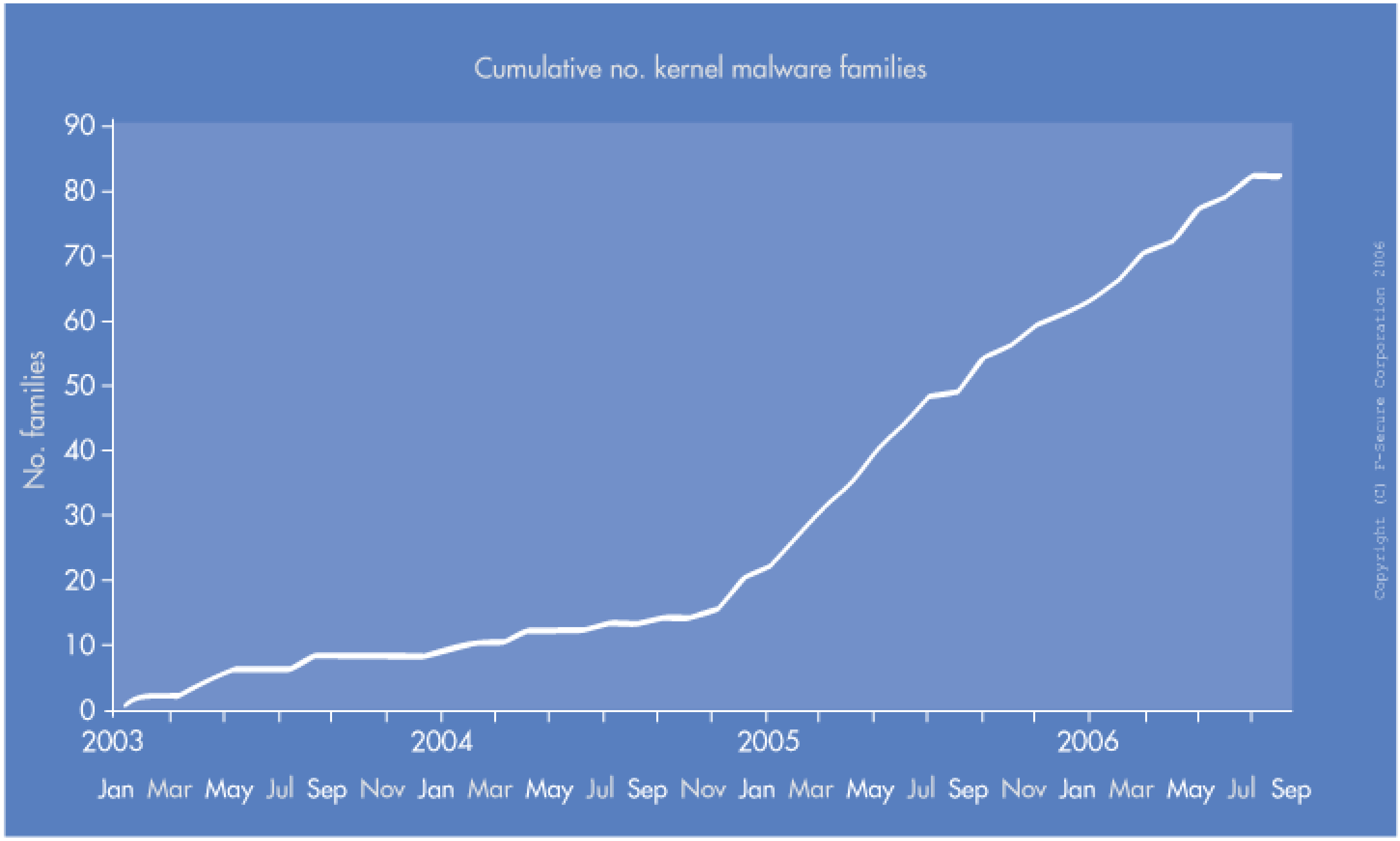
"Kernel malware is malicious software that runs fully or partially at the most privileged execution level, ring 0, having full access to memory, all CPU instructions, and all hardware."

- Can be divided into two subcategories
  - Full-Kernel malware
  - Semi-Kernel malware

History

- Kernel malware is not new – it has just been rare
- WinNT/Infis
  - Discovered in November 1999
  - Full-Kernel malware
  - Payload – PE EXE file infector
- Virus.Win32.Chatter
  - Discovered in January 2003
  - Semi-Kernel malware
  - Payload – PE SYS file infector
- Mostly proof of concepts

Increase of Kernel-Mode Malware



Situation Today

- Growth of kernel malware has been steady
- More main stream malware is utilizing kernel-mode techniques
  - Storm, Srizbi, Pandex, various banking trojans and password stealers
- Over half of the biggest spam botnets are kernel malware! [1]
  - Number 1 – Srizbi, 315.000 bots
  - Number 3 – Rustock, 150.000 bots
  - Number 4 – Pandex, 125.000 bots
  - Number 5 – Storm/Peacomm, 85.000 bots
- Malware is moving to kernel to protect themselves against security products and against other malware

1. Steward, Joe. (2008). Top Spam Botnets Exposed. http://www.secureworks.com/research/threats/topbotnets/

Key Techniques

- Majority of existing kernel malware is semi-kernel malware where their function is to hide and protect the main payload that executes in user mode
- Implementing a full-kernel malware can vary from hard to impossible depending on its features
- Basic downloader does following tasks when it executes:
  - Allocates memory for storing temporary data
  - Accesses internet to download the new payload
  - Stores the file on the file system
  - Modifies the registry to add a launch point
  - Executes the new payload

Executing Code in Ring 0

- The only documented way to execute third party KM code is to load a kernel-mode driver
- They are loaded at boot time if they have an entry in HKLM\System\CurrentControlSet\Services
  - Type = SERVICE_KERNEL_DRIVER (0x1) or SERVICE_FILE_SYSTEM_DRIVER (0x2)
  - Start = SERVICE_BOOT_START (0x0) or SERVICE_SYSTEM_START (0x1) or SERVICE_AUTO_START (0x2)
- They can also be installed and loaded at run time
  - CreateService + StartService Windows APIs
- There is also an undocumented way to do this
  - ntdll!ZwSetSystemInformation

Demo – Executing a Driver

Welcome to Ring 0!

Executing Code in Ring 0

- There are other undocumented ways of executing third party code in Ring 0
  - Code injection into system address space
  - Exploits
  - Call gates
- Both ways require write access to system address space from Ring 3
  - \Device\PhysicalMemory
  - ntdll!ZwSystemDebugControl
- Microsoft fixed this problem in Windows Server 2003 SP1 and later operating systems versions [2]

2. Ionescu, Alex. (2006). Subverting Windows 2003 SP1 Kernel Integrity Protection

Kernel-Mode Support Routines

- Windows kernel provides an API for kernel-mode drivers to do basic tasks
  - ExAllocatePoolWithTag / ExFreePoolWithTag
  - ZwCreateFile / ZwWriteFile / ZwClose
  - ZwCreateKey / ZwSetValueKey / ZwClose
- Only a subset of Native API functions exported by ntdll.dll are available for drivers
- The solution - use ntdll.dll to get correct index to nt!KiServiceTable and fetch the pointer
  - Read index from ntdll.dll in user mode and pass it to the driver
  - Driver loads the ntdll.dll file into kernel memory and reads index from it

Demo – Finding Unexported Functions

Some Ring 0 tricks…

Kernel-Mode Support Routines

```
; Exported entry 365. NtWriteVirtualMemory
; Exported entry 1162. ZwWriteVirtualMemory



; __stdcall NtWriteVirtualMemory(x, x, x, x, x)
public _NtWriteVirtualMemory@20
_NtWriteVirtualMemory@20 proc near
mov     eax, 115h       ; NtWriteVirtualMemory
mov     edx, 7FFE0300h
call    edx
retn    14h
_NtWriteVirtualMemory@20 endp
```

Executing Code in Ring 3

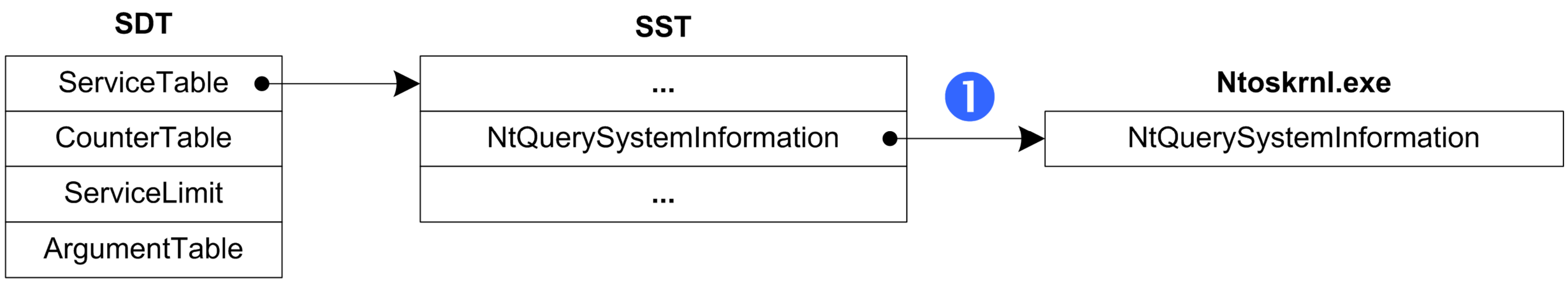
- Sometimes it is not feasible for kernel malware to execute all code in Ring 0
  - Launching of new processes
  - Complex libraries
  - Information stealing and encryption
- Two different approaches
  - Injecting payload into target process context
  - Queuing an user-mode Asynchronous Procedure Call

Executing Code in Ring 3

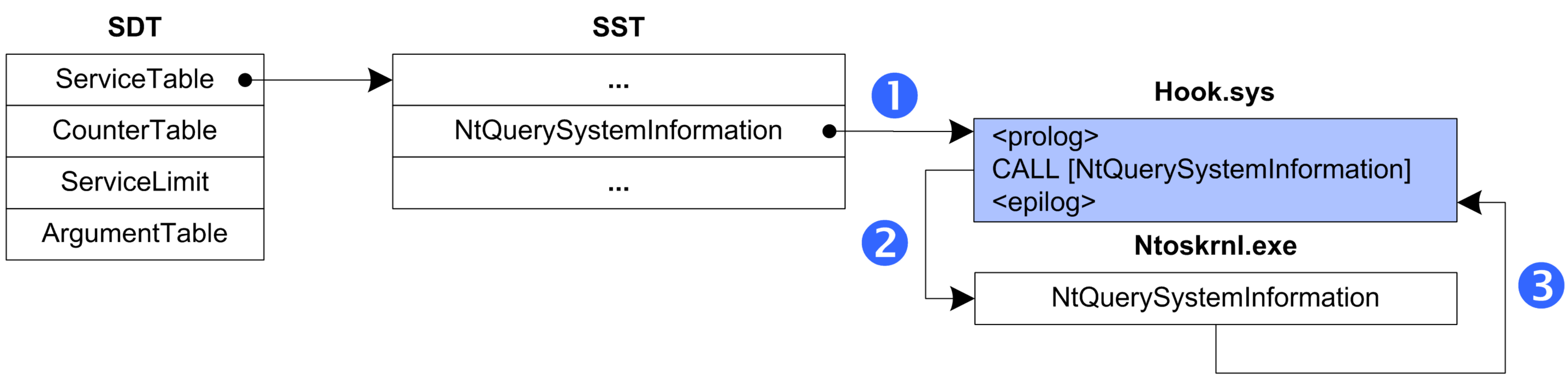
```
• pMdl = IoAllocateMdl(pPayloadBuf, dwBufSize, FALSE, FALSE, NULL);
• // Lock the pages in memory
• __try {
•     MmProbeAndLockPages(pMdl, KernelMode, IoWriteAccess);
• }
• __except (EXCEPTION_EXECUTE_HANDLER){}
• // Map the pages into the specified process
• KeStackAttachProcess(pTargetProcess, &ApcState);
• MappedAddress = MmMapLockedPagesSpecifyCache(pMdl,
•             UserMode, MmCached, NULL, FALSE, NormalPagePriority);
• KeUnstackDetachProcess(&ApcState);
• // Initialize APC
• KeInitializeEvent(pEvent, NotificationEvent, FALSE);
• KeInitializeApc(pApc, pTargetThread, OriginalApcEnvironment,
•      &MyKernelRoutine, NULL, MappedAddress, UserMode, (PVOID)NULL);
• // Schedule APC
• KeInsertQueueApc(pApc, pEvent, NULL, 0)
```

Rootkit techniques: hooking the handler table

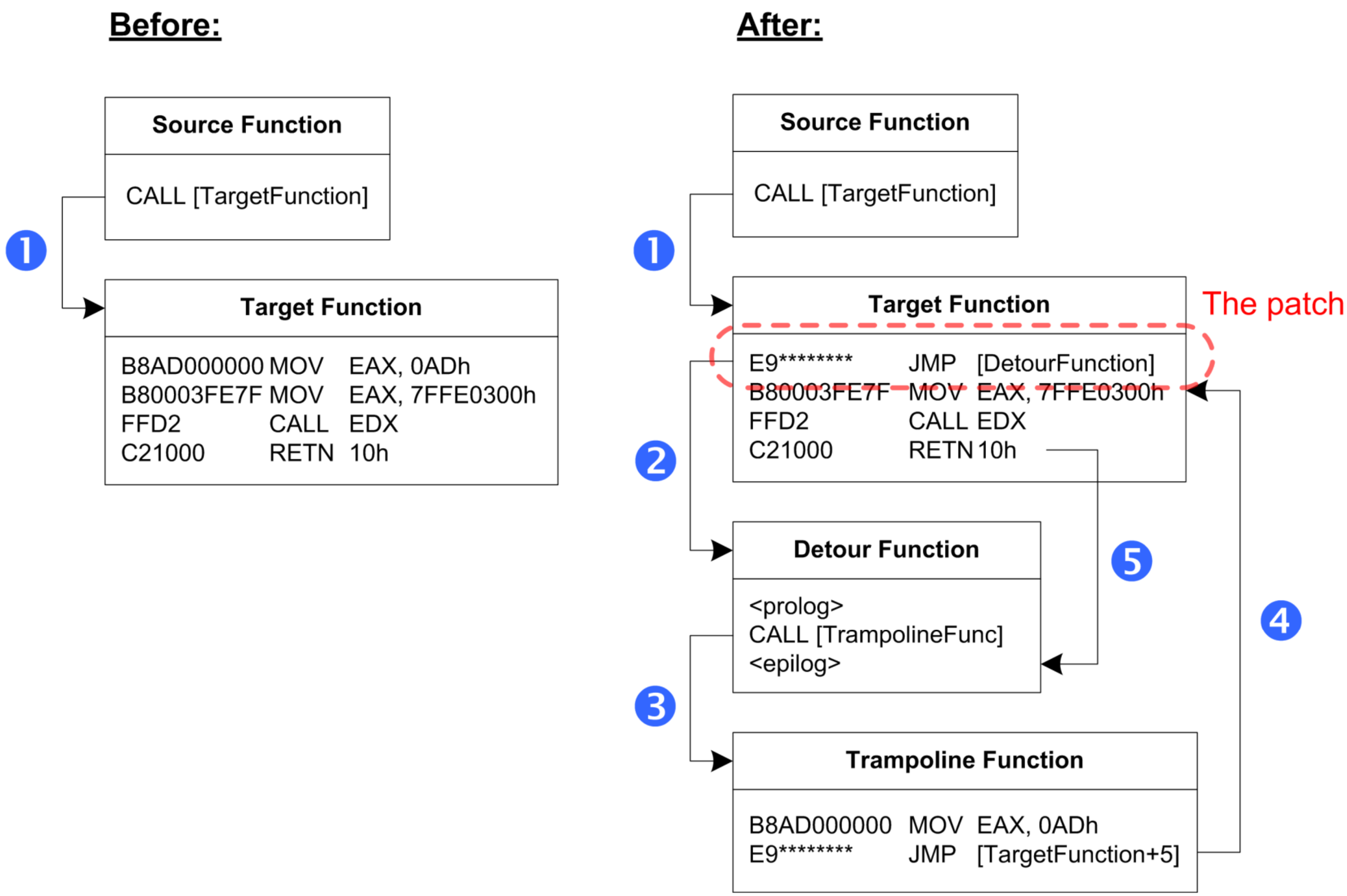
Before:



After:

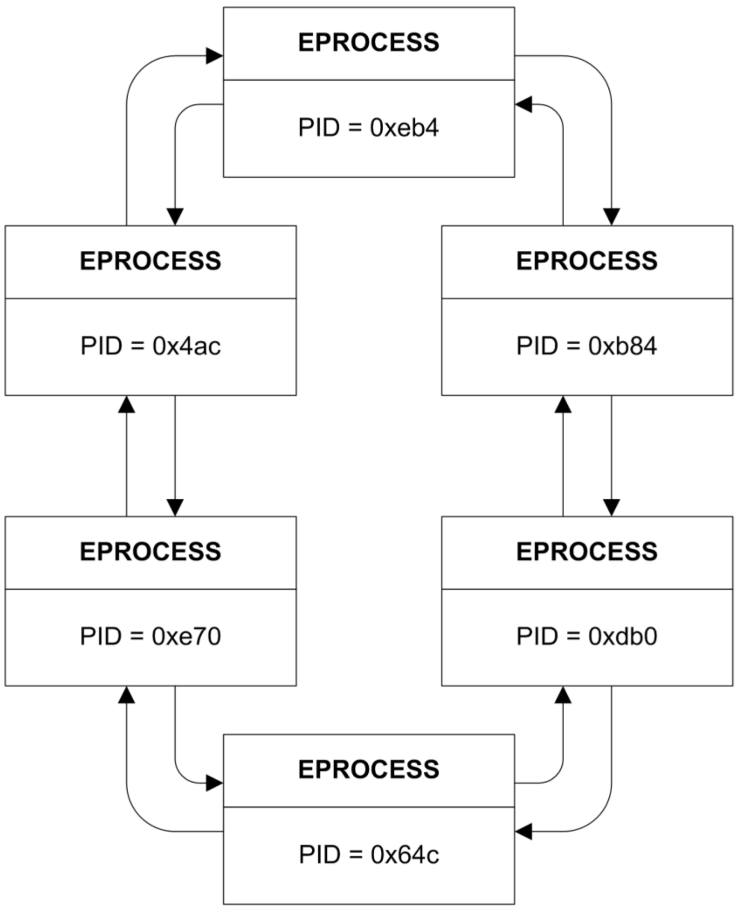


Rootkit techniques: inline hooking

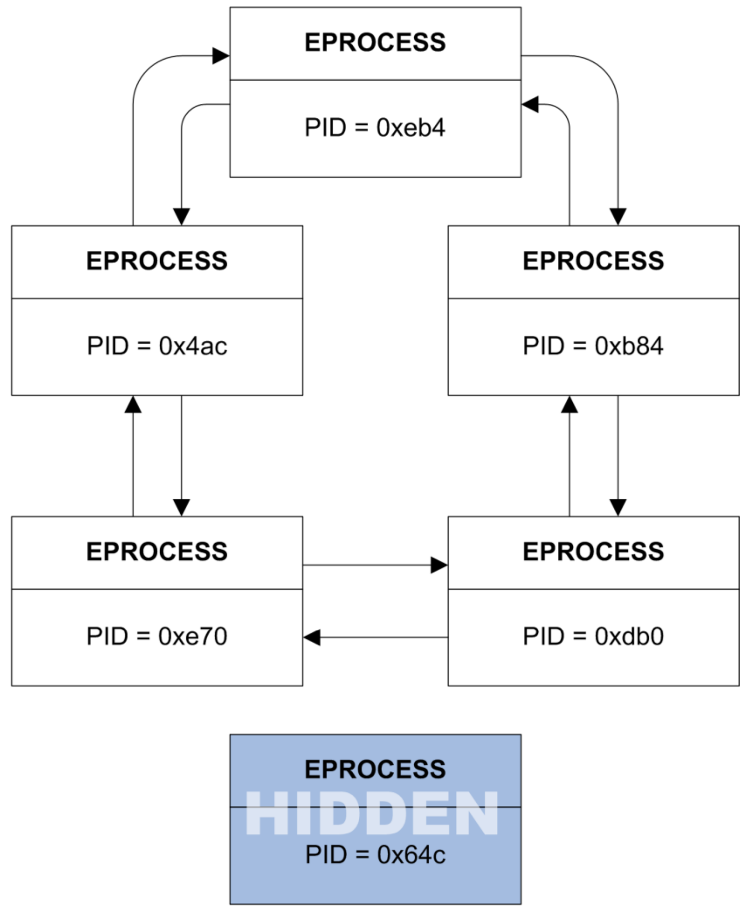


Rootkit techniques: in-memory data structure manipulation

Before:



After:



Demo – Hiding Processes

I am invisible!

Evolution – The Average Joe

- A simple piece of code whose purpose is to perform a specific task on behalf of the main malware component
- No code obfuscation or packing
- Usually a rootkit that hides
  - Files/Directories
  - Registry keys/values
  - Network connections
- Uses System Service Table and IRP handler hooks
- Easy to find and remove by modern AV solutions

Evolution – Haxdoor

- Backdoor with rootkit and spying capabilities
  - First variant found in August 2003
- Has three components – EXE (installer), DLL (payload), SYS (rootkit)
- Uses the driver to make its detection and removal more difficult
  - Hides its process and files
  - Protects its own threads and processes against termination
  - Protects its own files against any access
  - Injects the main payload into newly created processes
- First widely utilized kernel-mode malware

Demo – Haxdoor

Don’t mess with me!

Evolution – Apropos

- Adware/Spyware with rootkit capabilities
  - Emerged in October 2005
- Has multiple components – EXEs (installer), DLLs (payload), SYS (rootkit)
- Uses the driver to make its removal more difficult and to bypass personal firewalls
  - Hides its directory, files, registry entries and processes
  - Driver is obfuscated
  - Uses inline patching with Interrupt handler hooking to hook kernel functions
  - Hooks ndis.sys module to bypass personal firewalls
- First kernel-mode malware to utilize code obfuscation and NDIS hooking

Demo – Apropos

Blue screen of death?

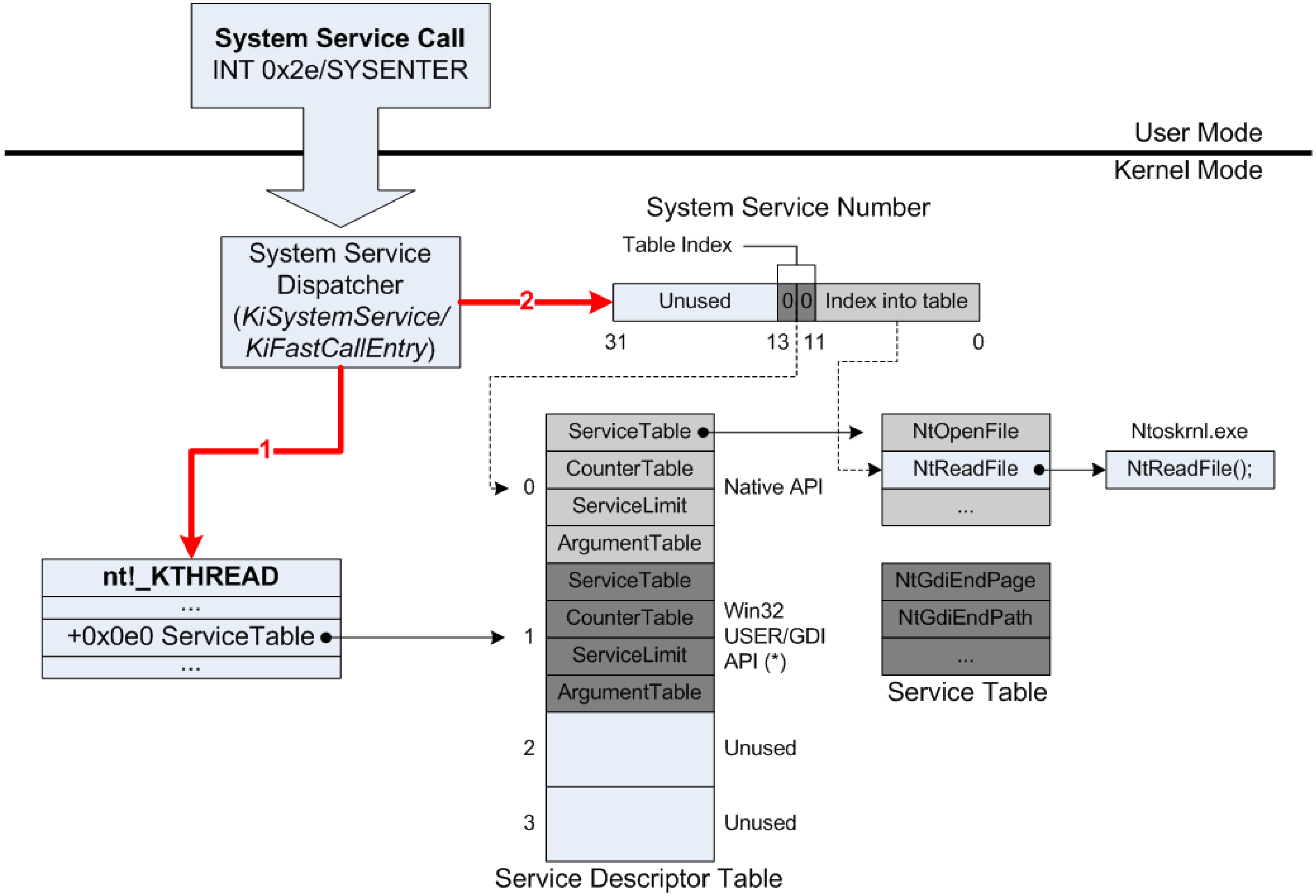
Evolution – Rustock

- Spambot and backdoor with rootkit capabilities
  - First variant found in December 2005
  - Rustock.A was found in 27th May 2006
  - Rustock.B was found in 3rd July 2006
- Consists of a single kernel-mode driver
  - EXE file loads the driver and deletes itself
  - SYS file carries the main payload inside an encrypted user-mode DLL
- The driver loads the main payload and acts as a rootkit to complicate its detection/removal and to bypass personal firewalls
- The most powerful and stealthiest rootkit seen by that time

Rustock – Details

- Rustock introduced new techniques to the stealth malware scene
  - Consists of a single driver which starts early during the boot process
  - Obvious traces of the loaded driver are removed from the memory
  - Driver is stored in a “hidden” and protected NTFS Alternate Data Stream
  - Driver uses obfuscation and a polymorphic packer
  - Hooks INT 0x2E and SYSENTER handler functions to control system calls
  - System Service Table hooks are present only when needed
  - Has an advanced rootkit anti-detection engine
  - Bypasses filter drivers by communicating directly to the lowest level device
  - Bypasses NDIS hooks by getting original pointers from ndis.sys file
  - Uses Asynchronous Procedure Call mechanism to execute the DLL in user mode
  - Tunnels network traffic from the DLL directly to the NDIS layer

Rustock – System Call Hooking



Demo – Rustock

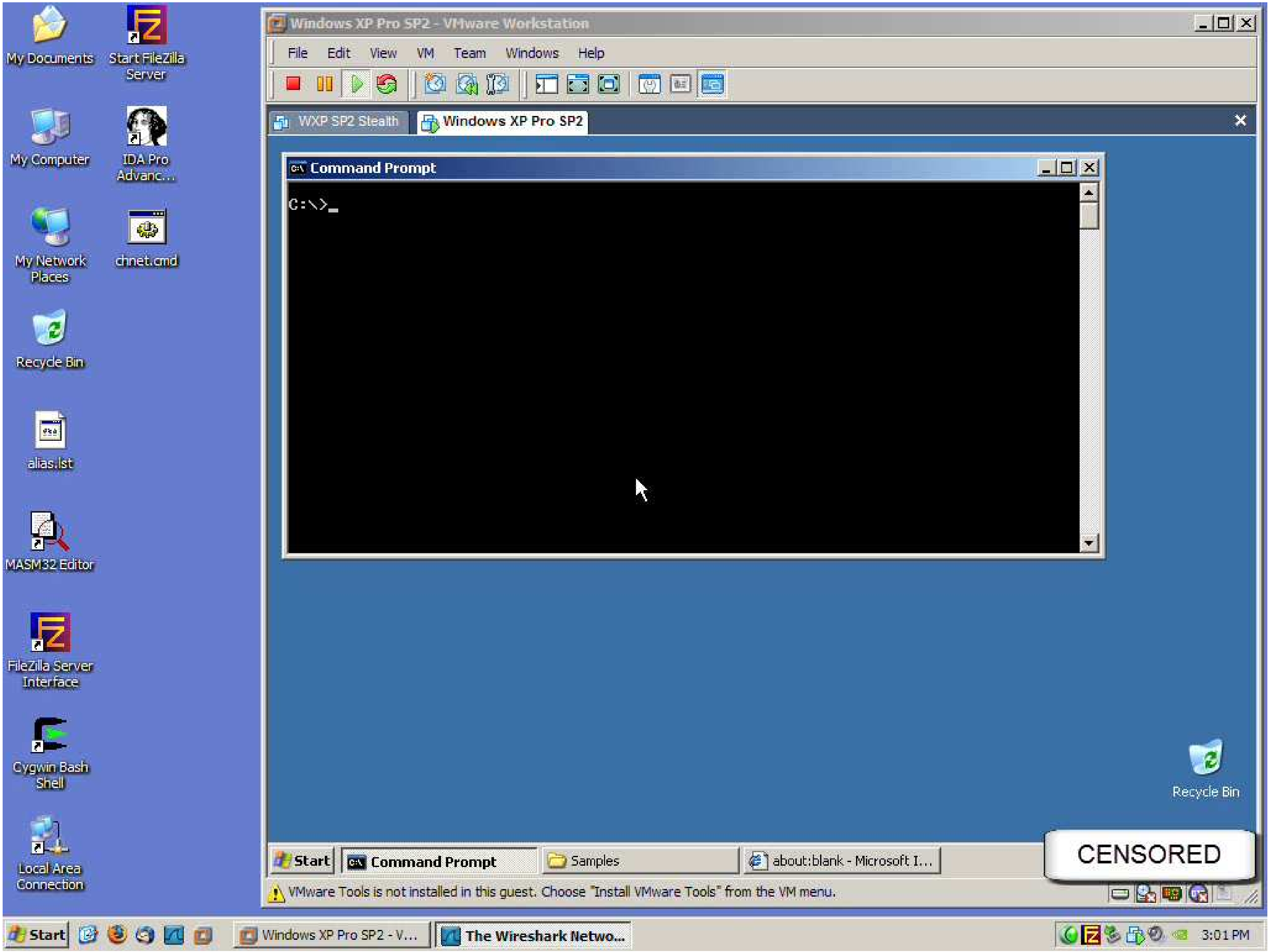
Hide'n Seek

Evolution – Srizbi

- Spambot and backdoor with rootkit capabilities
  - Emerged in April 2007
- Consists of a single kernel-mode driver
  - EXE file loads the driver and deletes itself
- First complex full-kernel malware!
  - Implements a fully blown spam client with a HTTP based C&C infrastructure
  - Uses low-level NDIS hooks and private TCP/IP stack to send/receive packets
  - Has complex code to bypass memory hooks
- The first malware to bypass virtually every personal firewall!
- Basic rootkit – easy to detect and remove by modern AV software

Demo – Srizbi

Spam from the kernel!


My Documents
Start FileZilla Server
My Computer
IDA Pro Advanc...
My Network Places
chnet.cmd
Recycle Bin
alias.lst
MASM32 Editor
FileZilla Server Interface
Cygwin Bash Shell
Local Area Connection
Windows XP Pro SP2 - VMware Workstation
File Edit View VM Team Windows Help
WXP SP2 Stealth
Windows XP Pro SP2
Command Prompt
C:\>_
Recycle Bin
Start
Command Prompt
Samples
about:blank - Microsoft I...
CENSORED
VMware Tools is not installed in this guest. Choose "Install VMware Tools" from the VM menu.
Start
Windows XP Pro SP2 - V...
The Wireshark Netwo...
3:01 PM

Evolution – Mebroot

- Downloader and backdoor with rootkit capabilities
  - First variant found in November 2007
- Consists of a custom MBR (loader) and a custom kernel-mode driver
  - EXE file replaces the MBR and writes the driver to raw disk sectors located in unpartitioned slack space at the end of the disk
- The most advanced and stealthiest malware seen so far!
  - Uses MBR as its launch point
  - All non-volatile data is stored in physical sectors outside of the file system
  - Driver uses polymorphic packer and advanced code obfuscation
  - Uses advanced NDIS hooks and private TCP/IP stack to send/receive packets
  - Utilizes “code pullout” technique to bypass memory hooks
  - Active Anti-Removal protection
  - Totally generic, open malware platform (MAOS)

Demo – Mebroot

Infecting the MBR!

Conclusions

- Kernel malware is a threat that has to be taken seriously
  - Wide distribution – Srizbi and Pandex spam runs, Mebroot drive-by-downloads from high volume web sites in Italy and other parts of Europe
- Today's kernel-mode malware is robust and effective
  - Biggest spam botnets are kernel-mode malware
  - Rustock, Srizbi and Mebroot are written by professional developers
- Detection and removal is becoming very challenging
  - How do you fight against someone who cheats?
- Prevention is a solution but how about false positives?
  - Please digitally sign your drivers

Additional Information

- Kasslin, K. (2006). Kernel malware: The attack from within.
  - http://www.f-secure.com/weblog/archives/kasslin_AVAR2006_KernelMalware_paper.pdf
- Florio, E.; Pathak P. (2006). Raising the bar: Rustock and advances in rootkits
  - http://www.virusbtn.com/virusbulletin/archive/2006/09/vb200609-rustock
- Kasslin, K.; Florio E. (2007). Spam from the kernel.
  - http://www.virusbtn.com/virusbulletin/archive/2007/11/vb200711-srizbi
- Kasslin, K.; Florio E. (2008). Your computer is now stoned (…again!).
  - http://www.virusbtn.com/virusbulletin/archive/2008/04/vb200804-MBR-rootkit
- Kasslin, K.; Florio E. (2008). Your computer is now stoned (…again!). The rise of MBR rootkits.
  - http://www.f-secure.com/weblog/archives/Kasslin-Florio-VB2008.pdf

QUESTIONS?

F-SECURE®

BE SURE.

BE SURE.

F-SECURE®