Kernel Malware: The Attack from Within

Kimmo Kasslin / 5th December 2006

F-SECURE®

BE SURE.

Contents

Kernel Malware

“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 to subcategories

- Full-Kernel Malware
- Semi-Kernel Malware

Kernel Malware – Past

Kernel malware is not new – it has just been rare

WinNT/Infis

- Discovered in November 1999
- Full-Kernel malware

Virus.Win32.Chatter

- Discovered in January 2003
- Semi-Kernel malware

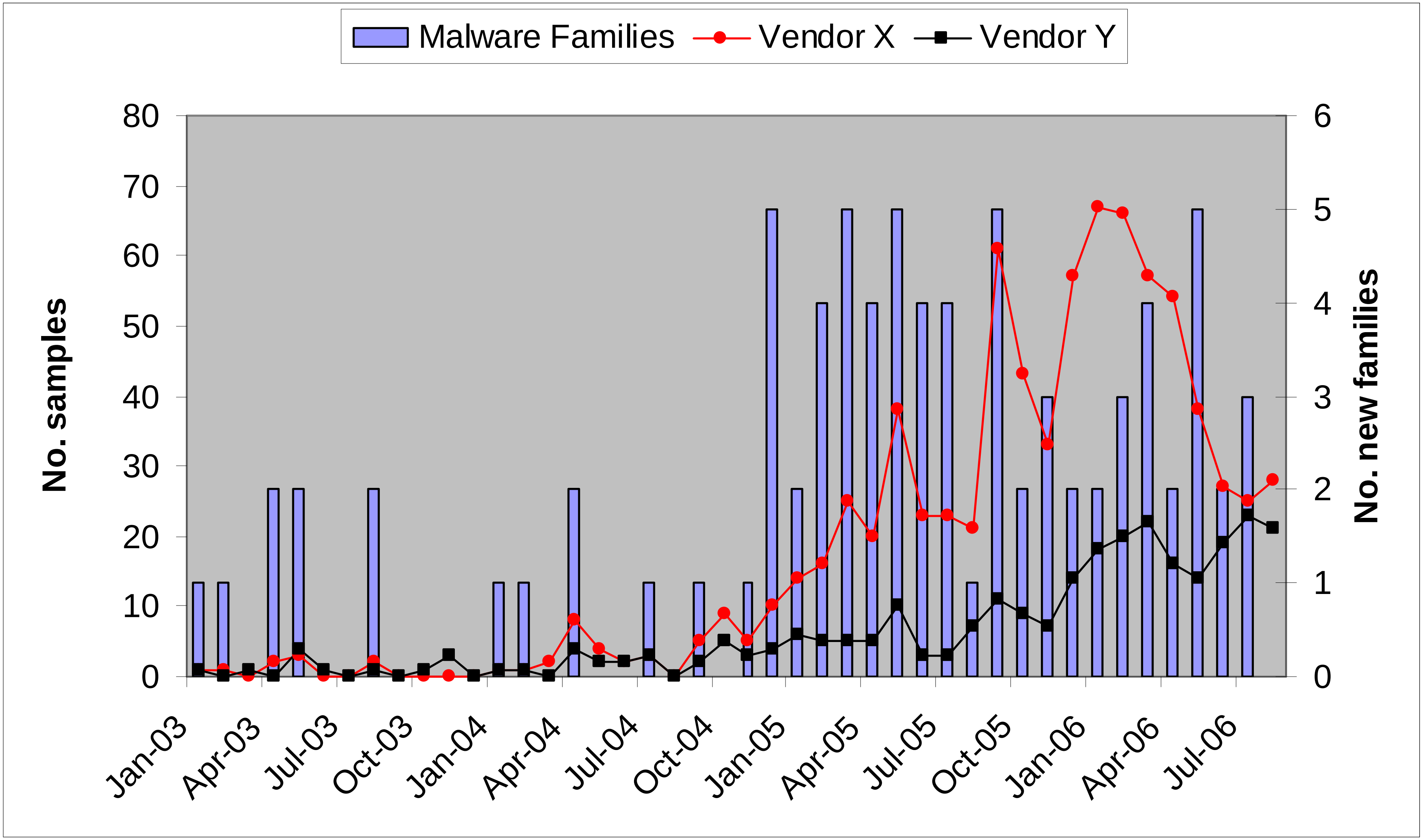
Kernel Malware – Present

Today, the number of kernel malware compared to all malware is still very small

To see how the trend has changed a statistical analysis of malware samples was conducted

- Two separate collections from Jan 2003 to Aug 2006
- On average 100000 samples per collection
- Kernel-Mode drivers were selected based on their PE header information
- Their detection names were used to identify families using kernel-mode components

Kernel Malware – Present



Kernel Malware – Present

F-Secure	Symantec	McAfee
Backdoor.Win32.Haxdoor	Backdoor.Haxdoor	Backdoor-BAC
Backdoor.Win32.HacDef	Backdoor.HackDefender	HackerDefender trojan
Trojan-Spy.Win32.Banker	Infostealer.Bancos	PWS-Banker trojan
Backdoor.Win32.PcClient	Backdoor.Formador	BackDoor-CKB trojan
Trojan-Spy.Win32.Goldun	Trojan.Goldun	PWS-Goldun trojan
Trojan.Win32.Crypt.t	Spyware.Apropos.C	Adware-Apropos
SpamTool.Win32.Mailbot	Backdoor.Rustock.A	Spam-Mailbot trojan
Trojan-Clicker.Win32.Costrat	Backdoor.Rustock.B	Spam-Mailbot.c trojan

Key Techniques

F-SECURE® BE SURE.

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 = 0x1 or 0x2
- Start = 0x0 or 0x1 or 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

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

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

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

In some cases 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)
```

Case Study 1: Haxdoor

Haxdoor is a powerful backdoor with rootkit and spying capabilities

Consists of a PE executable, a DLL and a kernel-mode driver

Uses the driver to make its detection and removal more difficult and to bypass personal firewalls

- Hides its process and files
- Protects its own threads and processes against termination
- Protects its own files against any access
- Injects payload into created processes

Case Study 2: Mailbot aka Costrat

Mailbot is the most powerful and stealthiest rootkit seen so far

Consists of a single kernel-mode driver

Carries an encrypted DLL as a payload that is a sophisticated spambot with backdoor capabilities

Its detection and removal is still a challenge to most rootkit detectors and antivirus solutions

Case Study 2: Mailbot aka Costrat

Mailbot uses lots of unique techniques

- Driver is stored in “hidden” and protected ADS
- Hooking of nt!KiServiceTable on a thread-level basis
- INT 0x2E and SYSENTER hooks reside inside ntoskrnl module
- Has an advanced rootkit anti-detection engine
- Bypasses filter drivers by finding the lowest device object and sending IRPs directly to it
- Bypasses NDIS hooks by getting original pointers from ndis.sys image file

Demo

F-SECURE®

BE SURE.

Conclusions

Kernel Malware is becoming more popular

- Mostly driven by high interest in rootkits

One reason is that more documentation and examples is available to the public

Current security solutions, including antivirus scanners and firewalls, have not been designed to protect against kernel malware

Prevention might be the only solution

THANK YOU – QUESTIONS?

kimmo.kasslin@f-secure.com

F-SECURE®

BE SURE.