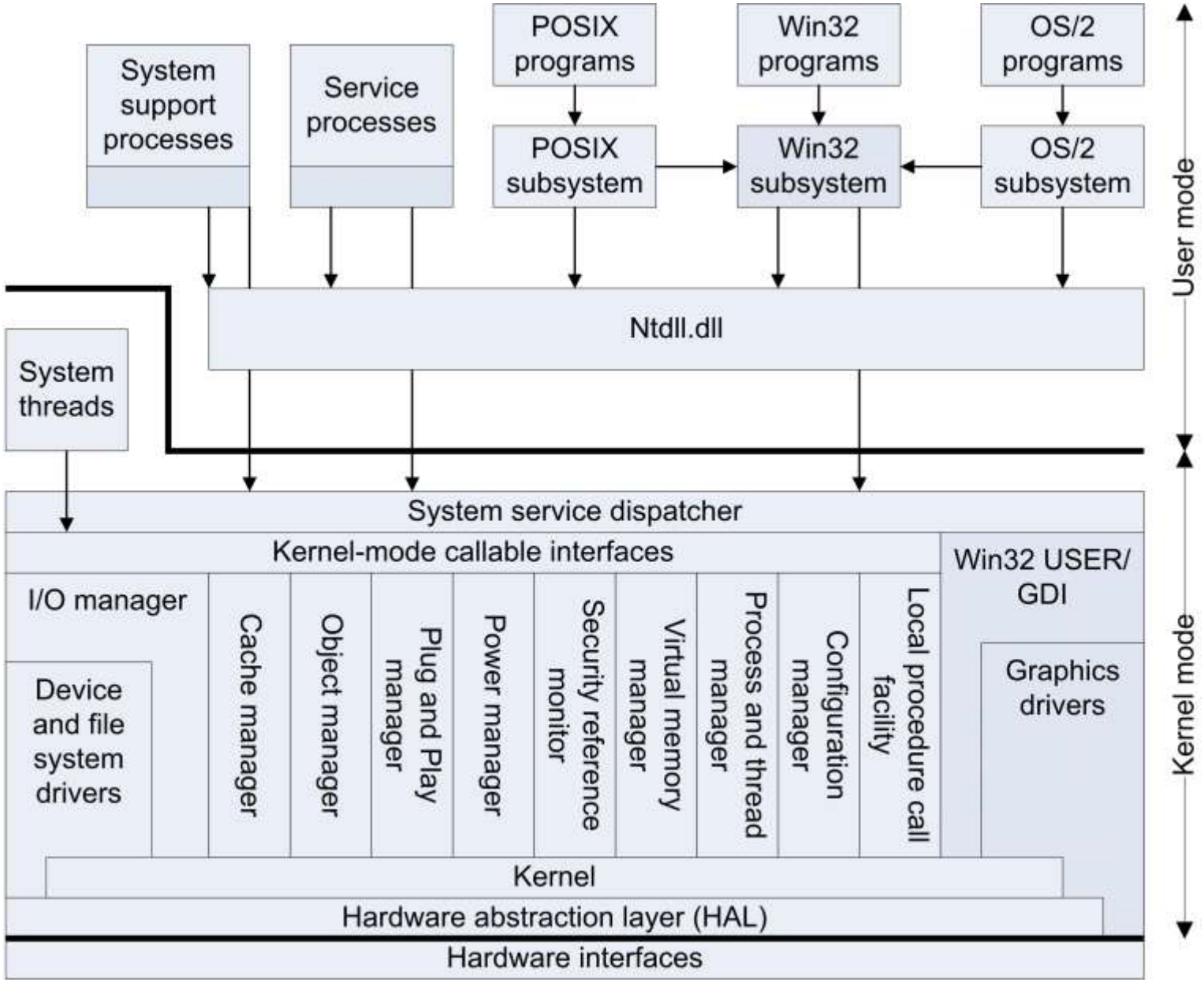
Windows for Reverse Engineers (+ Rootkit Basics)

T-110.6220 Reverse Engineering Malware

Course Spring 2015

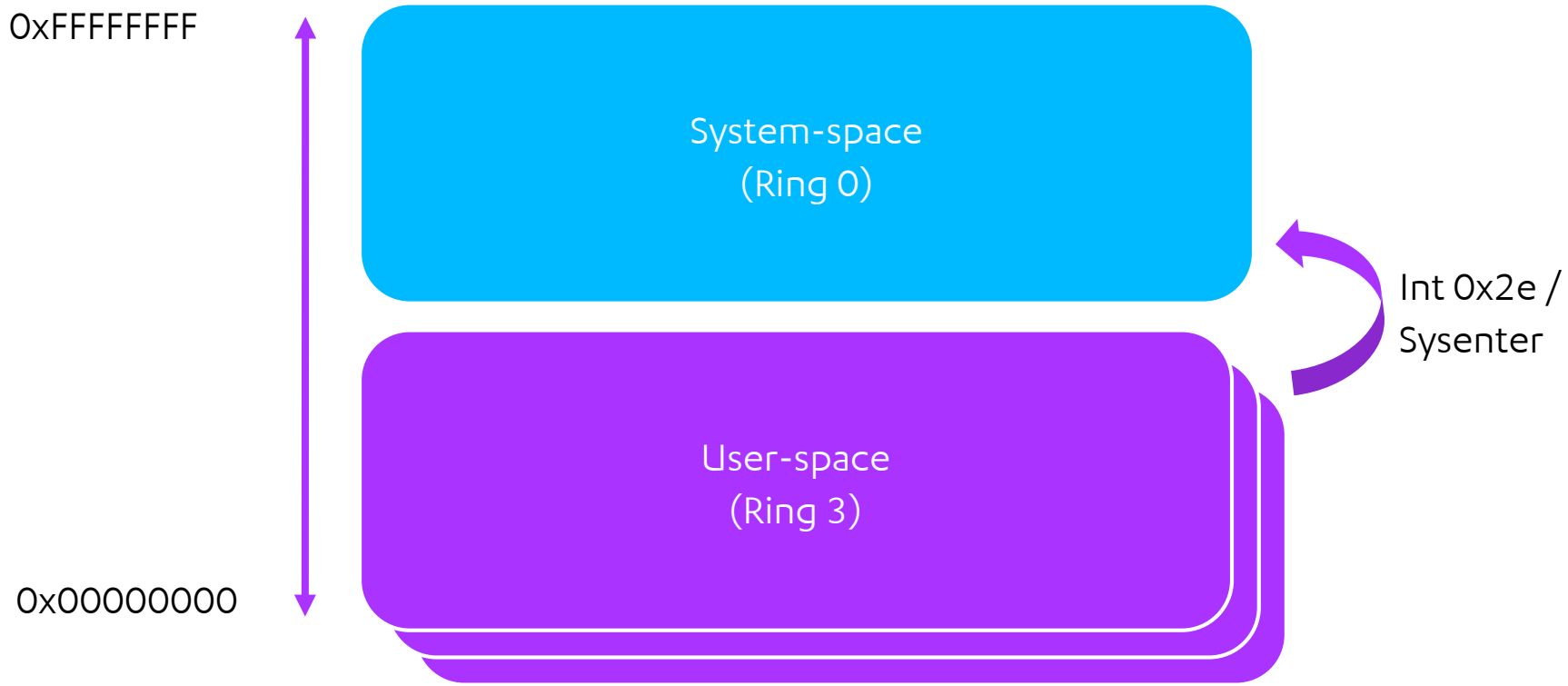
Windows Architecture


System support processes
Service processes
POSIX programs
Win32 programs
OS/2 programs
POSIX subsystem
Win32 subsystem
OS/2 subsystem
User mode
Ntdll.dll
System threads
System service dispatcher
Kernel-mode callable interfaces
Win32 USER/ GDI
I/O manager
Device and file system drivers
Cache manager
Object manager
Plug and Play manager
Power manager
Security reference monitor
Virtual memory manager
Process and thread manager
Configuration manager
Local procedure call facility
Graphics drivers
Kernel
Hardware abstraction layer (HAL)
Hardware interfaces
Kernel mode

F-Secure

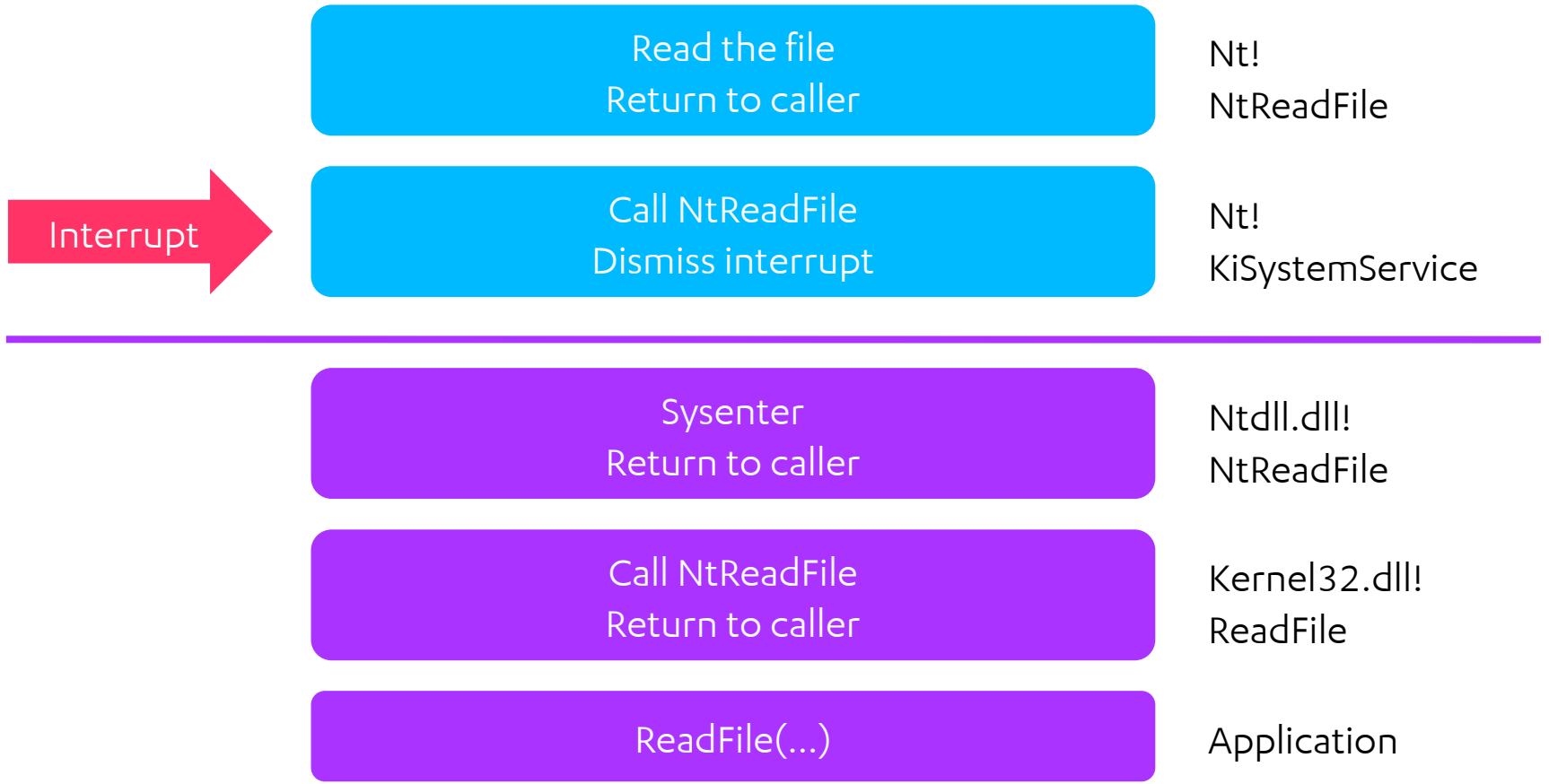
System Mechanisms

Kernelmode and Usermode



F-Secure.

Service Dispatching

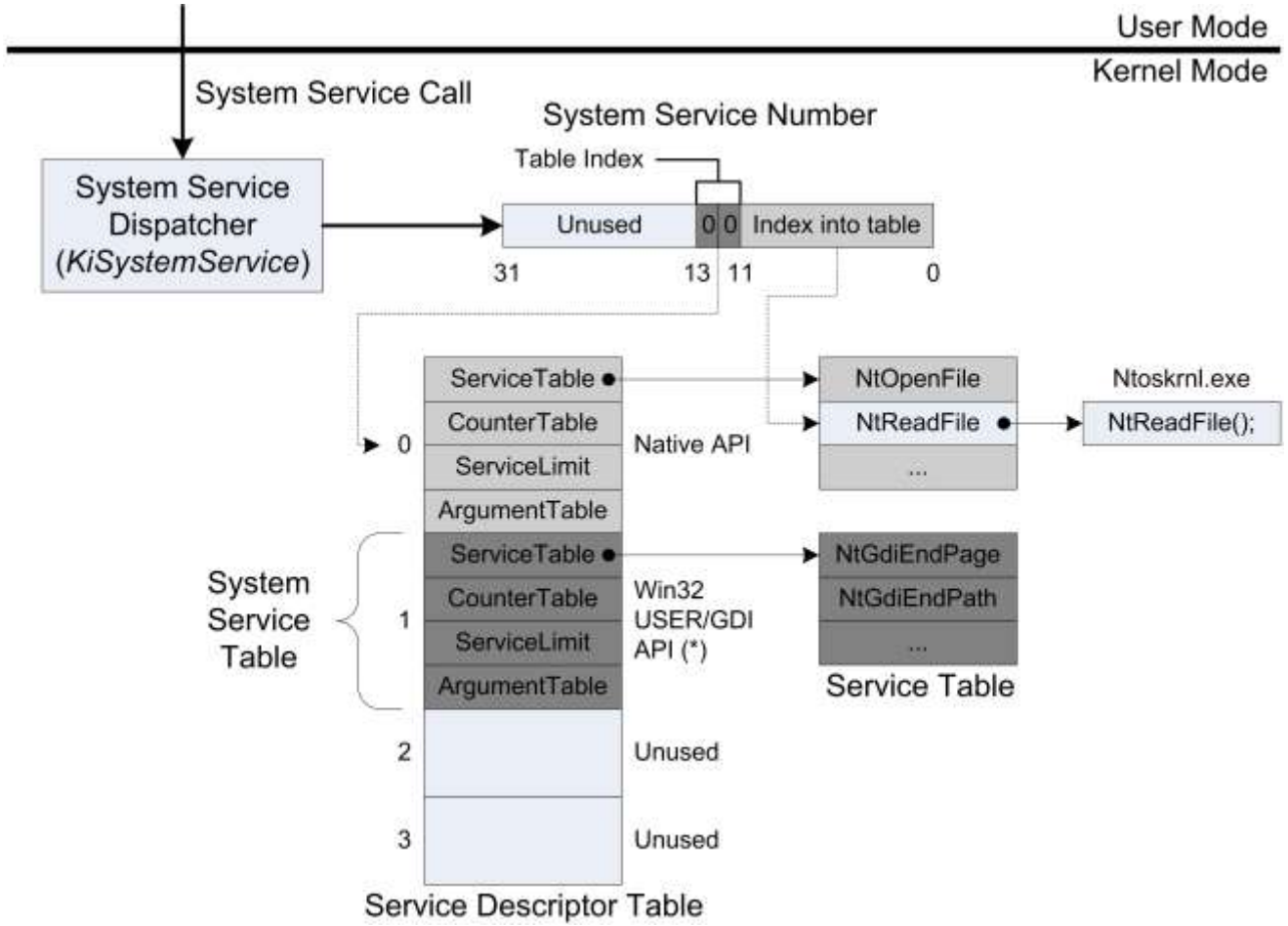


F-Secure.

Service Dispatching

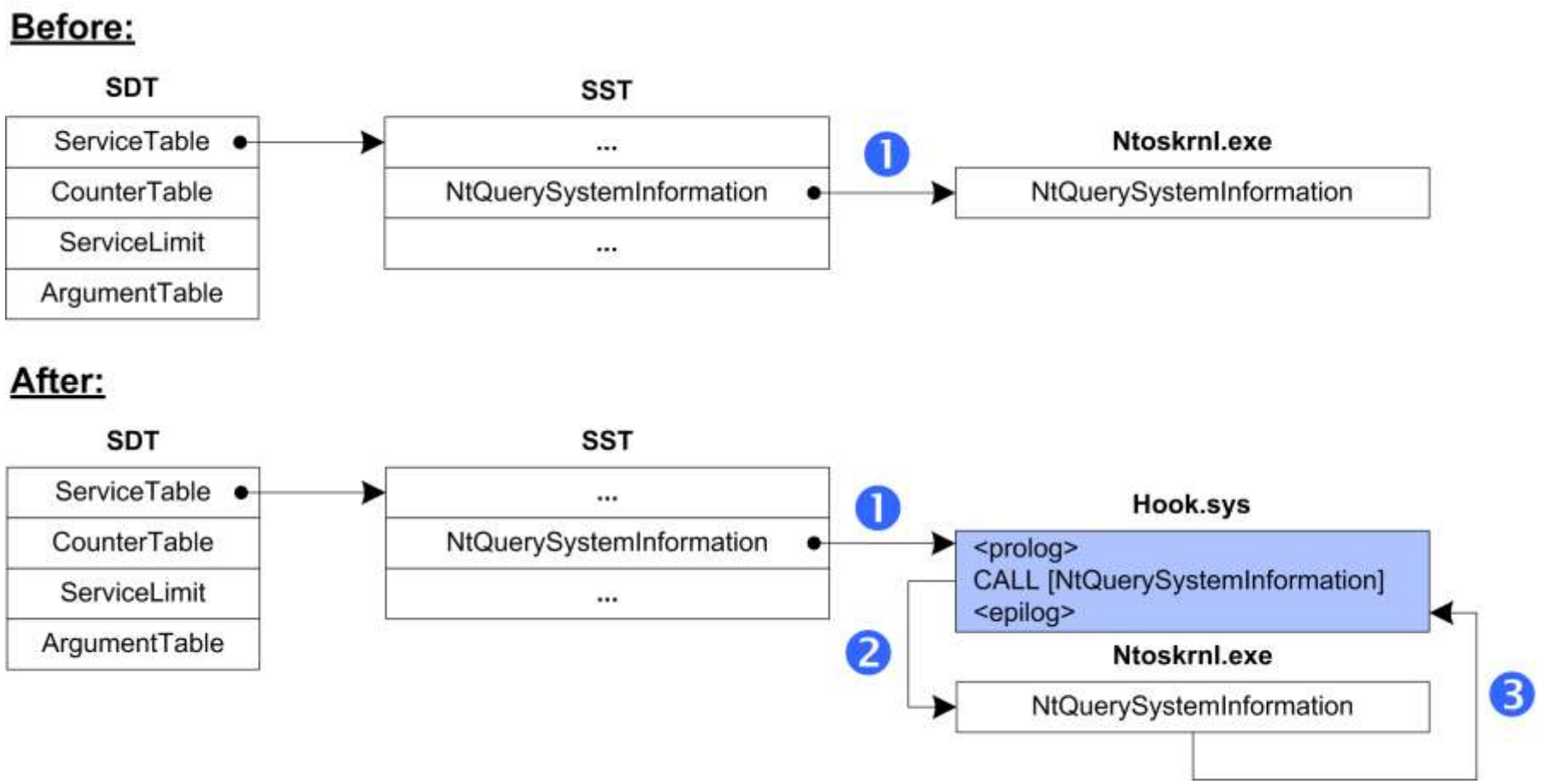
```
Windows 2000:

77F8C552:               mov     eax, 0A1h       ; NtReadFile
77F8C557:               lea     edx, [esp+4]
77F8C55B:               int     2Eh
77F8C55D:               retn    24h

Windows XP:

77F5BFA8:               mov     eax, 0B7h       ; NtReadFile
77F5BFAD:               mov     edx, 7FFE0300h
77F5BFB2:               call    edx
77F5BFB4:               retn    24h

7FFE0300:               mov     edx, esp
7FFE0302:               sysenter
7FFE0304:               ret
```

Service Dispatching



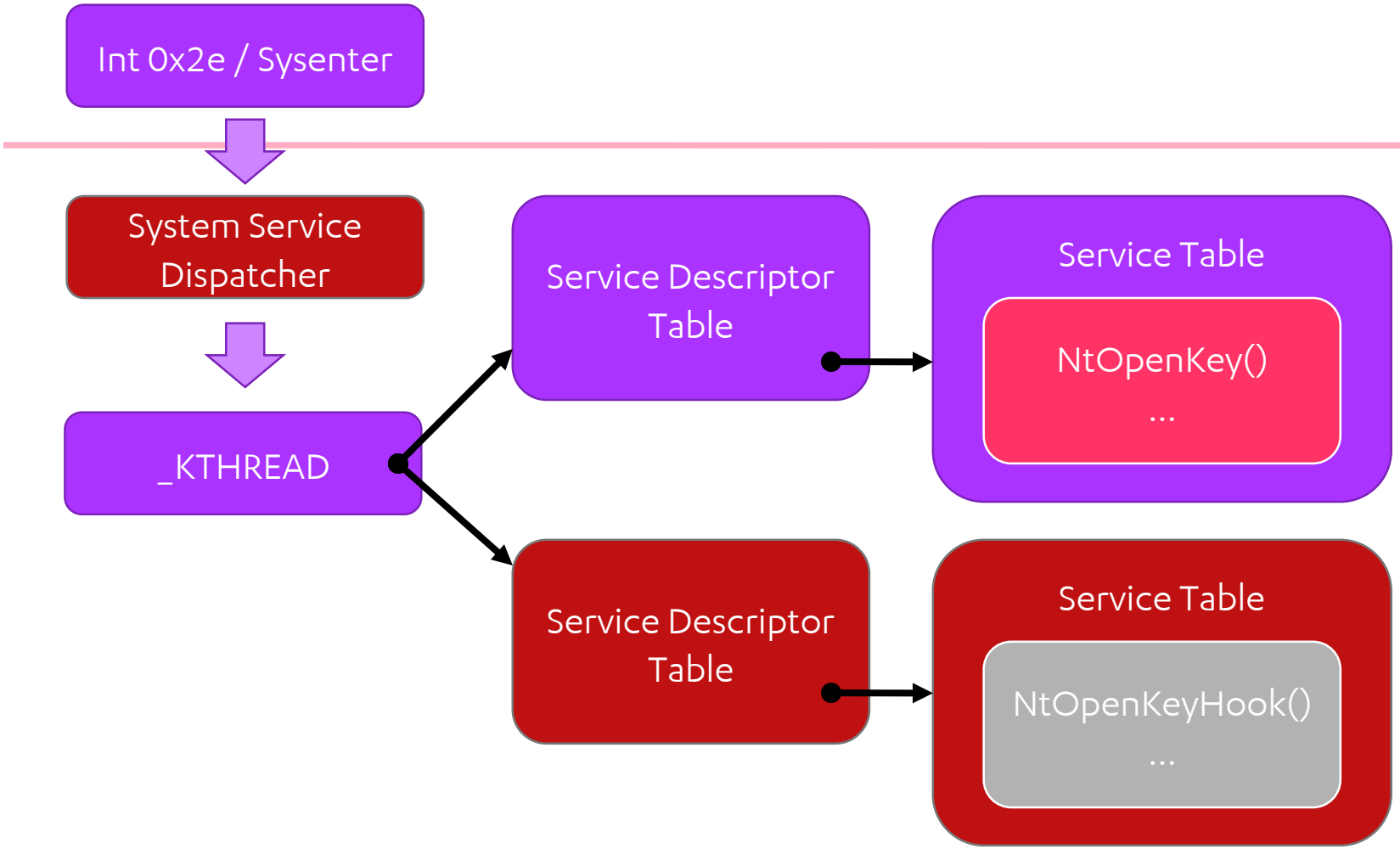
F-Secure.

(Old) Rootkit Techniques



F-Secure.

Variation of Syscall Hooking



F-Secure.

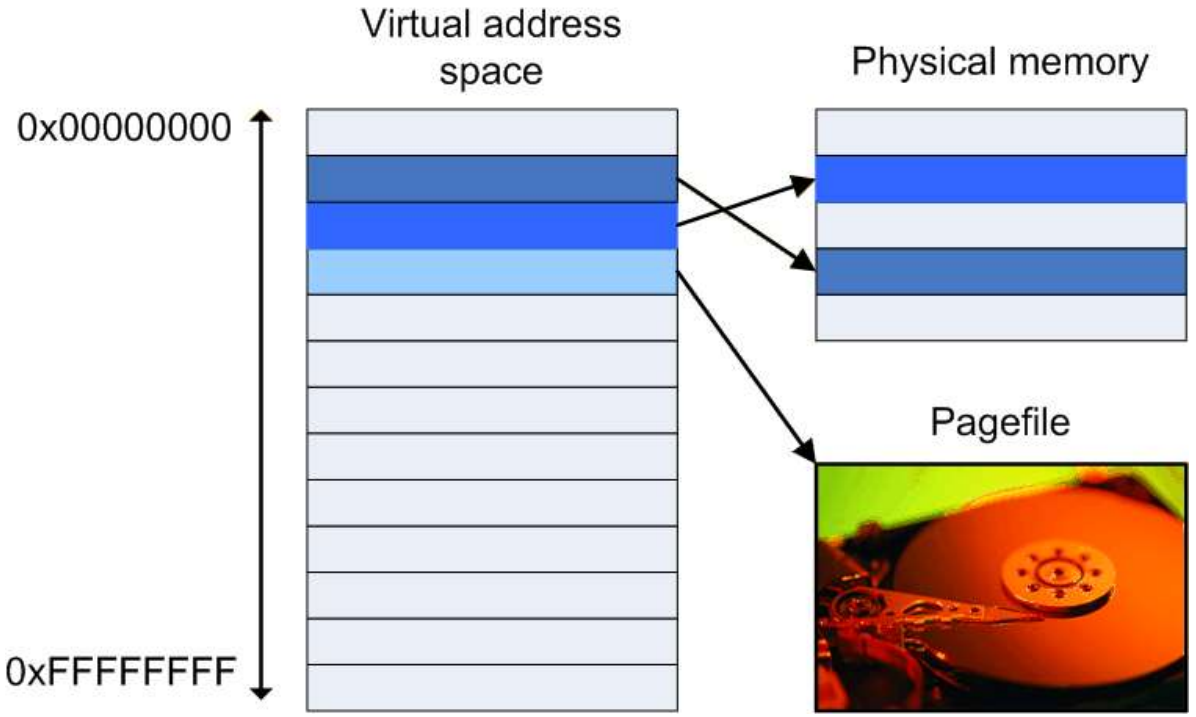
Memory Management

Memory Manager

- Each process sees a large and contiguous private address space
- The memory manager has two important tasks
  1. Mapping access to virtual memory into physical memory
  2. Paging contents of memory to disk as physical memory runs out; and paging the data back into memory when needed

Virtual Memory

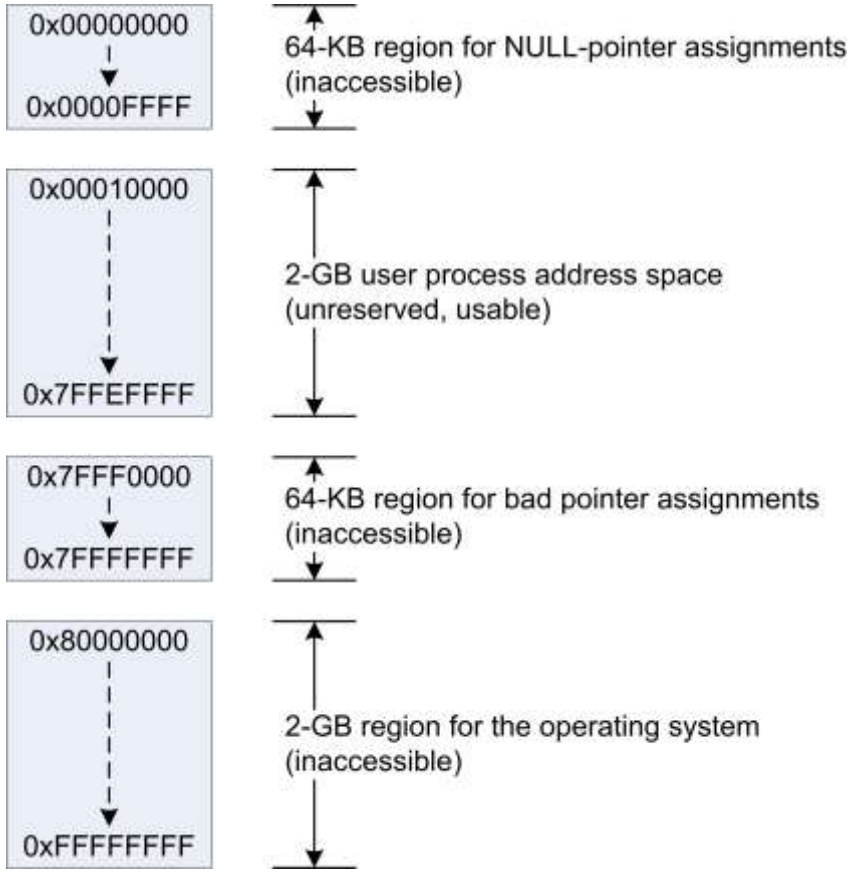
- Every process has its own virtual address space
- Virtual memory provides a logical view of the memory that might not correspond to its physical layout
- Paging is the process of transferring memory contents to and from the disk
  - Virtual memory can exceed available physical memory



F-Secure.

Virtual Memory (x86)

- Flat 32-bit address space, total of 4GB virtual memory
- By default, only the lower half can be used by a process for its private storage because the OS takes the upper half for its own protected OS memory utilization.
- The memory mappings of the lower half is changed to match the virtual address space of the currently running process



F-Secure.

Processes and Threads

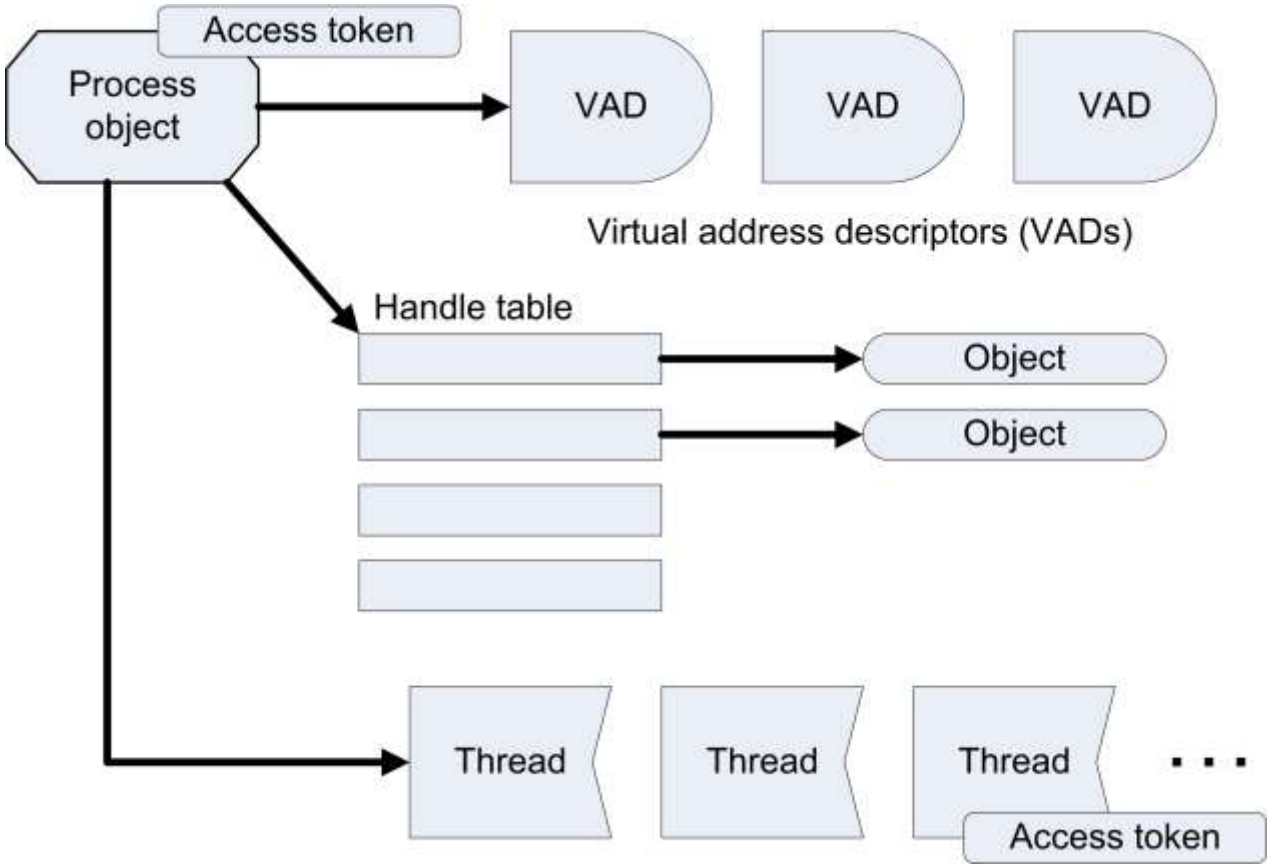
Process

- Process is an abstraction of a running program
- Process consists of following essential components:
  - A private virtual address space
  - An executable program (“the base image”)
  - A list of open handles to resources allocated by the operating system
  - An access token, which uniquely identifies the owner, security groups, and privileges associated with the process
  - A process ID
  - One or more threads
- Important structures: EPROCESS (KM) and PEB (UM)

Thread

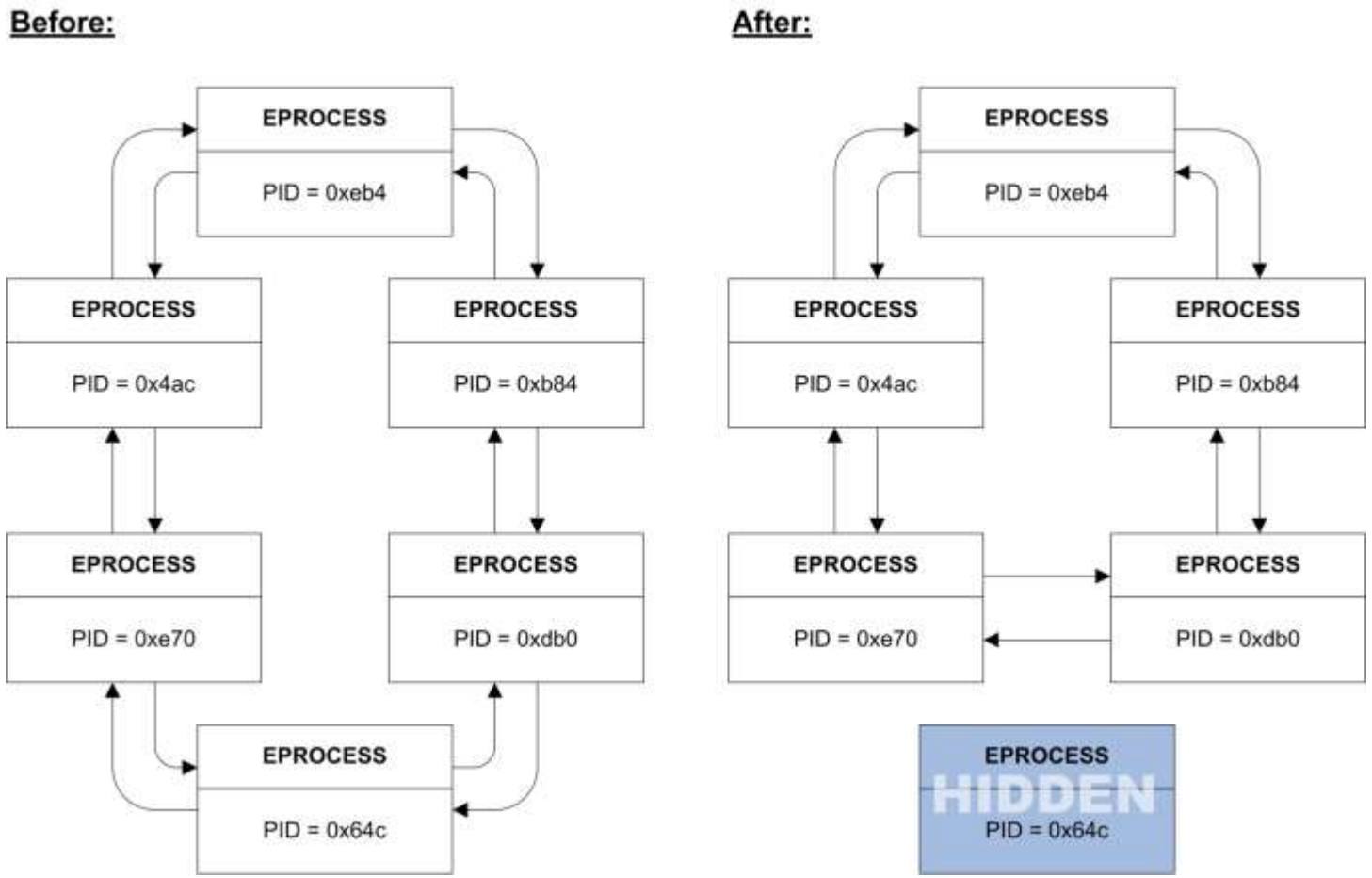
- Thread is an entity scheduled for execution on the CPU
- Thread consists of following essential components:
  - The CPU state
  - Two stacks, one for kernel-mode and one for user-mode
  - Thread-Local Storage (TLS), a private storage area that can be used by subsystems, run-time libraries, and DLLs
  - A thread ID
  - An access token, which uniquely identifies the owner, security groups, and privileges associated with the thread
- Important structures: ETHREAD (KM) and TEB (UM)

Processes and Threads



F-Secure.

Rootkit Techniques: DKOM

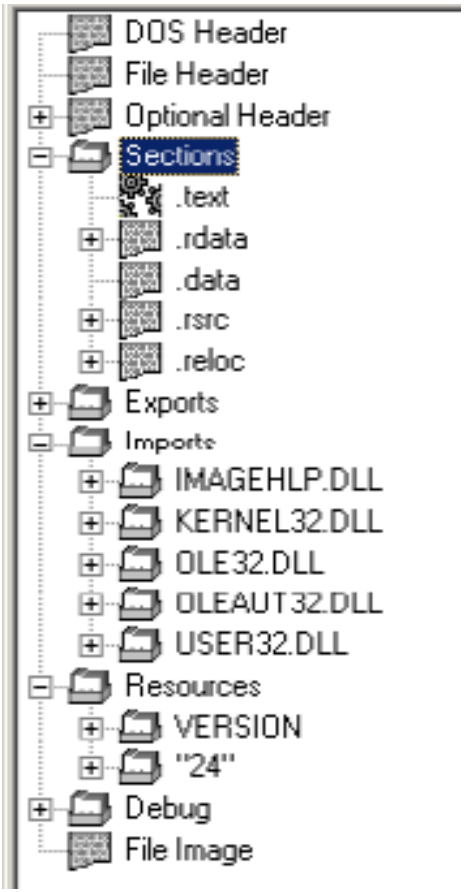


F-Secure.

Applications on Windows

Executable Format

- Object files and executables follow the PE (Portable Executable) file format
- Full specification available online
  - http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx
- Best viewed with your hex editor (HT) or specialized PE viewer (PEBrowsePro ->)
- File extensions commonly used by executables:
  - EXE, DLL, SYS and CPL



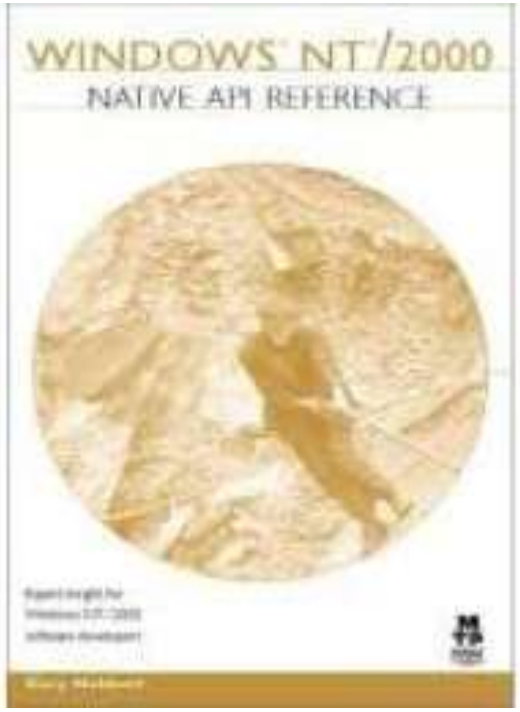
F-Secure.

Windows API

- Windows API is the interface to the operating system for applications
  - Exposed by a set of system libraries: kernel32.dll, user32.dll, …
  - Windows 7 refactored the system libraries so you will see e.g. kernelbase.dll
- Several subcategories
  - Administration and management (WMI, …)
  - Diagnostics (event logging, …)
  - Networking
  - Security
  - System services (processes, threads, registry…)
- MSDN is the reverse engineers best friend for Windows binaries
  - http://msdn2.microsoft.com/en-us/library/default.aspx

Native API

- Undocumented interface to OS functionality
  - One level below Windows API
  - Some low-level functionality **only** available through Native API
- Examples of interesting functions
  - NtSetSystemInformation
  - NtQuerySystemInformation
  - NtQueryDirectoryFile
- See “Windows NT/2000 Native API Reference” by Nebbett or
  - ReactOS project - http://www.reactos.org/

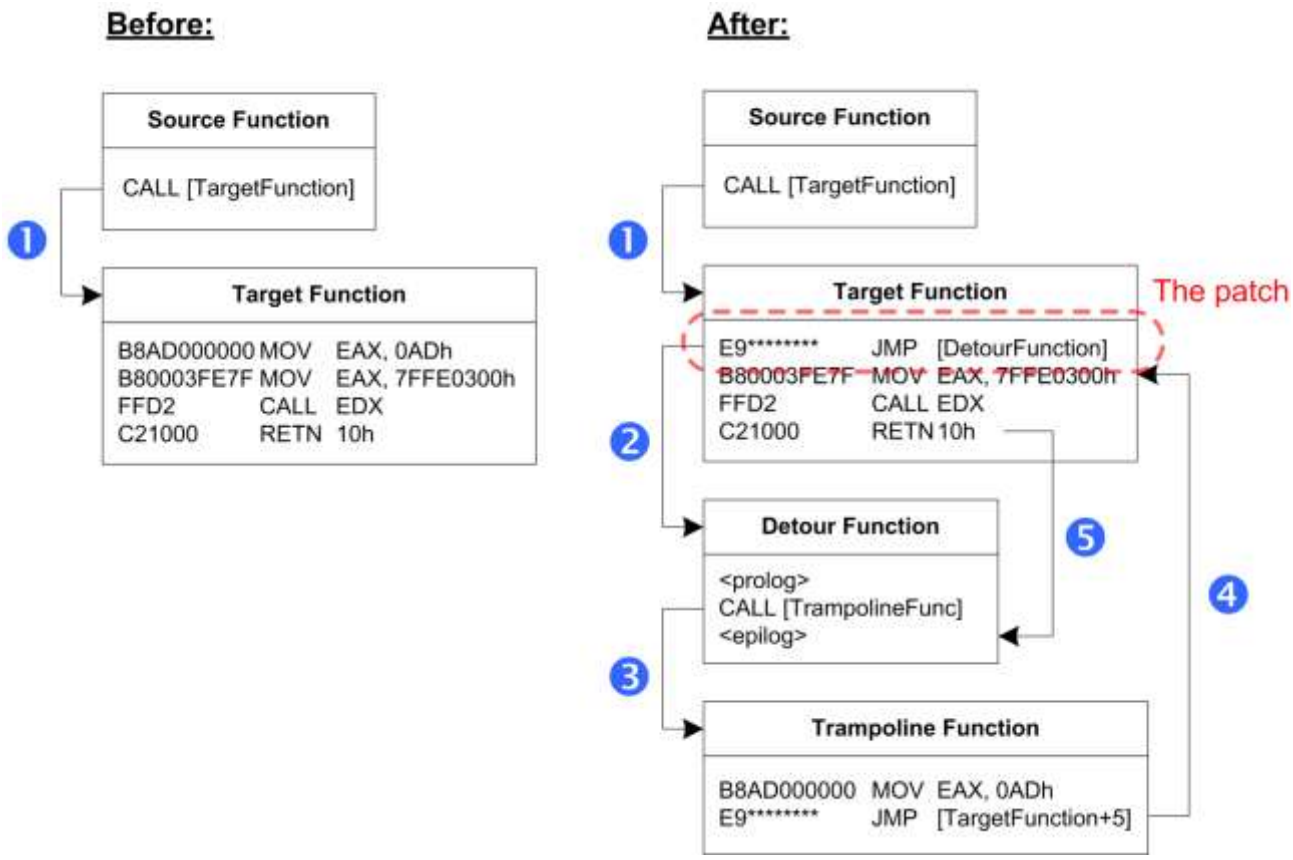


F-Secure.

API Hooking

- Hooking is a technique to instrument functions and extend or replace their functionality
  - For example, you want to know each time a program calls CreateFile() and strip write access from the caller
- Many implementations
  - Hooking a function table (IAT, SSDT, IDT, …)
  - Inline hooking (patching the first code bytes of a function)
- Hooking is used by rootkits to hide or protect objects

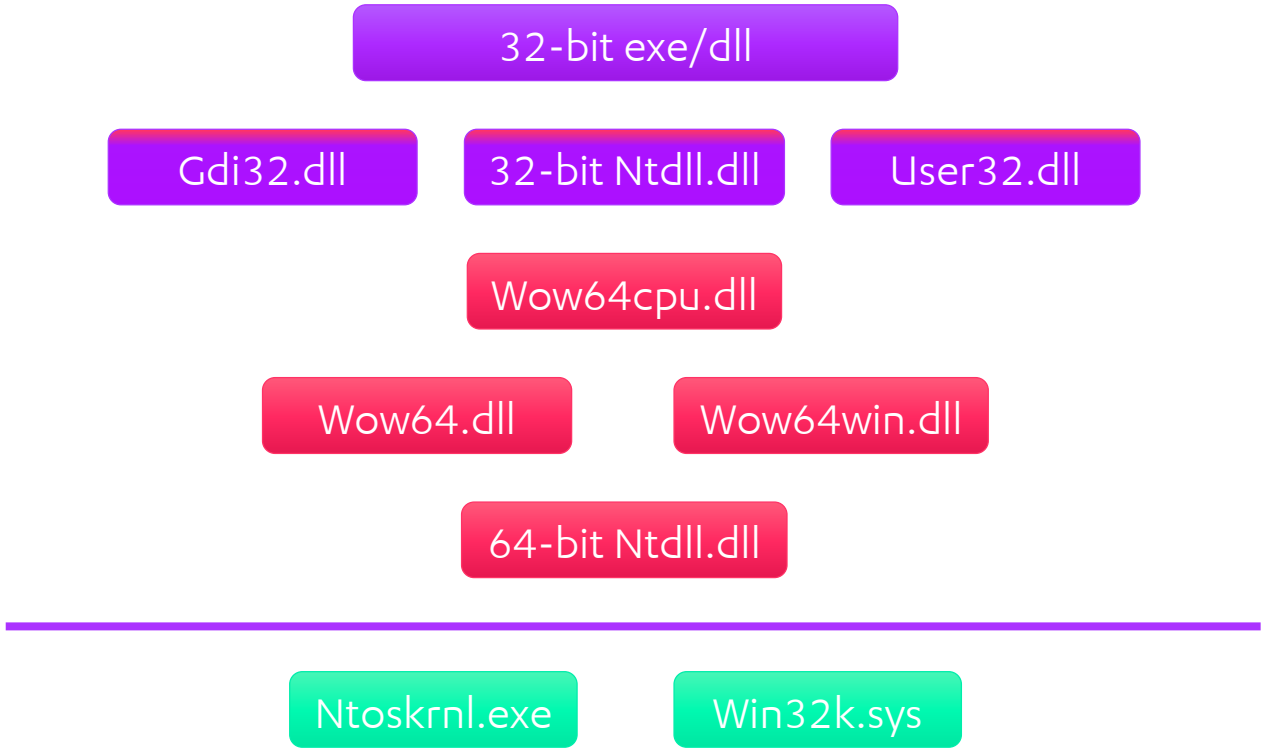
Inline Hooking



F-Secure

WOW64

- Win32 emulation on 64-bit Windows
- Implemented as a set of user-mode DLLs, with some support from kernel



F-Secure.

WOW64

32-bits ntdll from Win7 x86

```
mov     eax, X

mov     edx, 7FFE0300h
call    dword ptr [edx]
        ;ntdll.KiFastSystemCall

retn    Z
```

32-bits ntdll from Win7 x64

```
mov     eax, X
mov     ecx, Y
lea     edx, [esp+4]
call    dword ptr fs:[0C0h]
        ;wow64cpu!X86SwitchTo64BitMode
add     esp, 4
ret     Z
```

```
wow64cpu!X86SwitchTo64BitMode:
748c2320 jmp     0033:748C271E   ;wow64cpu!CpupReturnFromSimulatedCode
```

Source: http://blog.rewolf.pl/blog/?p=102

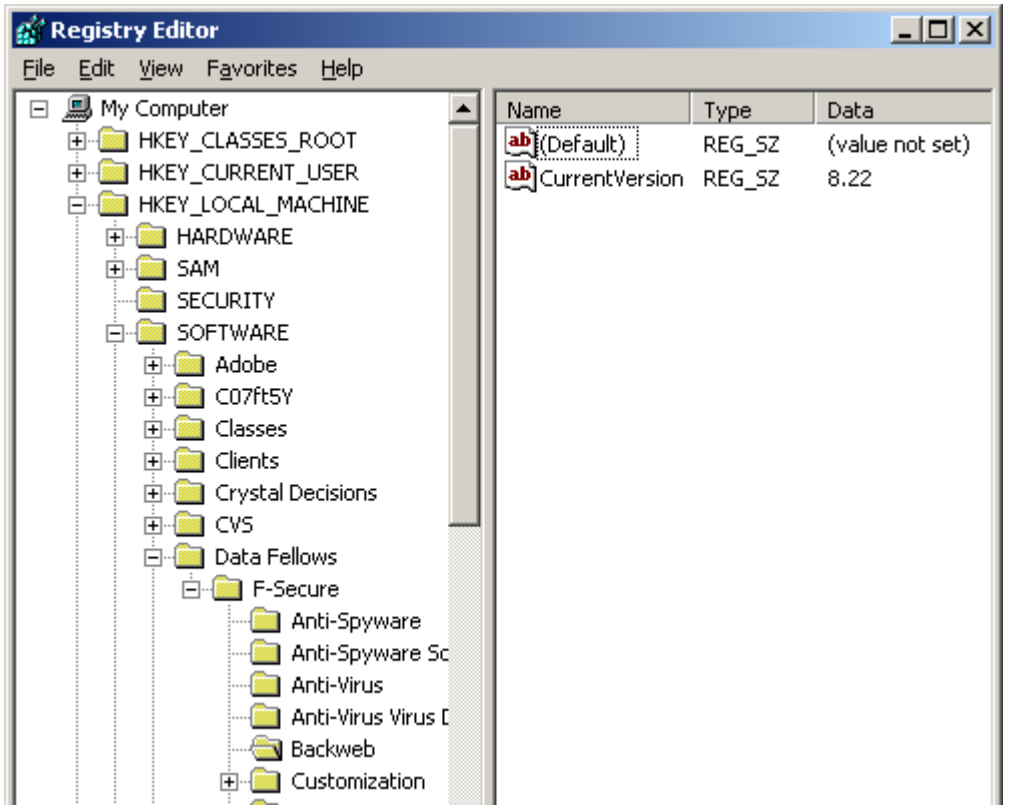
WOW64: Filesystem

- Folder \Windows\System32 stores native 64-bit images
  - Calls from 32-bit code redirected to \Windows\SysWOW64
- A few subdirectories are excluded from redirections for compatibility reasons
  - %windir%\system32\drivers\etc and %windir%\system32\spool
  - %windir%\system32\catroot and %windir%\system32\catroot2
  - %windir%\system32\logfiles and %windir%\system32\driverstore
- Other common folders are handled via system environment variables
  - 64-bit: %ProgramFiles% -> ”C:\Program Files”
  - 32-bit: %ProgramFiles% -> ”C:\Program Files (x86)”
- Automatic redirections can be enabled/disabled per thread with Wow64 APIs:
  - Wow64DisableWow64FsRedirection and Wow64RevertWow64FsRedirection

Management Mechanisms

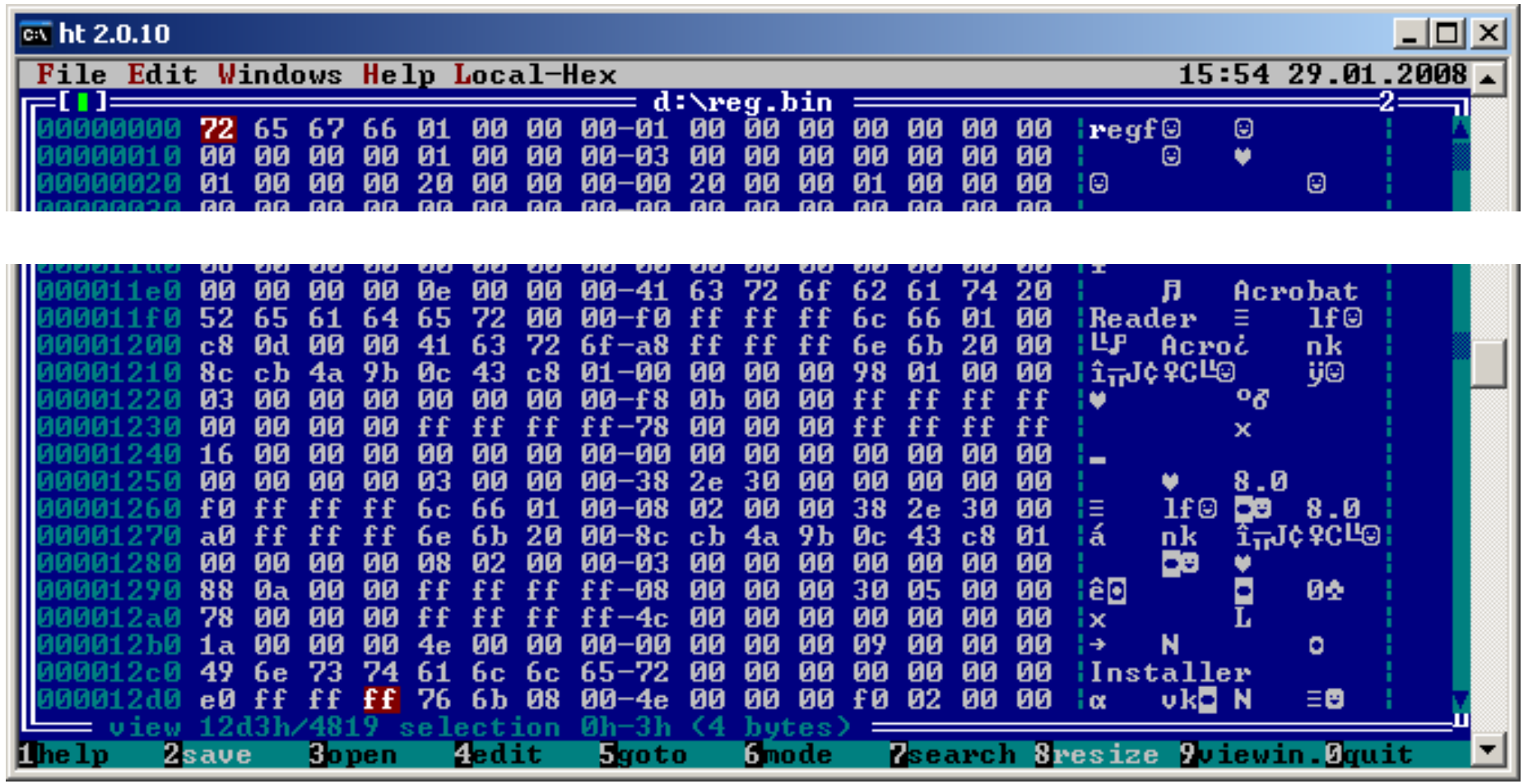
Registry

- A tree that contains all settings and configuration data for the OS and other software
- Basic concepts: hive, key, value
- Also contains in-memory volatile data
  - Current HW configuration, …
- Hives are just files, most under SystemRoot%\System32\Config\



F-Secure.

Registry Hive



F-Secure.

Registry Roots

- HKEY_LOCAL_MACHINE
  - System-related information
- HKEY_USERS
  - User-specific information for all accounts
- HKEY_CURRENT_USER
  - User-specific info for current user, links to HKEY_USERS
- HKEY_CLASSES_ROOT
  - File associations and COM registration, links to HKLM\Software\Classes
- HKEY_CURRENT_CONFIG
  - Current hardware profile, links to HKLM\System\CurrentControlSet\Hardware Profiles\Current

Registry and Malware

- Malware typically wants to survive a reboot
  - The registry is the most common place to do this
  - Hundreds of launchpoints
    - HKLM\Software\Microsoft\Windows\CurrentVersion\Run:MyApp
    - HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\explorer.exe:Debugger
- Malware also wants to change (security) settings for other components
  - Windows Firewall, IE extensions and settings, Windows File Protection, …
- The registry is also a great source for forensic data, for example:
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\ShellNoRoam\MUICache
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist

Services

- Services are background processes that usually perform a specific task and require no user-interaction
  - For example, Automatic Updates
- Controlled by the Service Control Manager (SCM), services.exe
  - Configuration data under HKLM\System\CurrentControlSet\Services
- Different types of services
  - Kernel drivers
  - Separate process
  - Shared process (hosted by svchost.exe)

Services and Malware

- You should be able to identify three kinds of components
  - Programs that control services (SCP's, service control programs)
  - Services
  - Drivers
- Imports are a giveaway:
  - SCP's: OpenSCManager, CreateService, StartService, …
  - Services: StartServiceCtrlDispatcher, RegisterServiceCtrlHandler
- Drivers:
  - Optional header subsystem: Native (1)
  - Imports

File Systems

Filesystems

- Windows supports the following file system formats
  - CDFS
    - Read-only filesystem for CD's
  - UDF
    - For DVD's, read+write support (since Vista)
  - FAT12, FAT16, FAT32
    - Older format
  - exFAT
    - Optimized for flash drives, supports large disk sizes (since XP SP2)
  - NTFS
    - Native file system format

NTFS

- Designed to improve perfomance and reliability over FAT
- Some interesting NTFS Features
  - Disk quotas
  - Encrypting File System (EFS)
  - **Multiple data streams**
  - Hard links and junction points
  - Unicode-based naming

I/O Subsystem

I/O Subsystem

- A set of components in the kernel that manage and provide access to hardware devices
  - I/O Manager
  - Plug and Play Manager
  - Power Manager
- Key concepts
  - Driver
  - Device
  - I/O requests

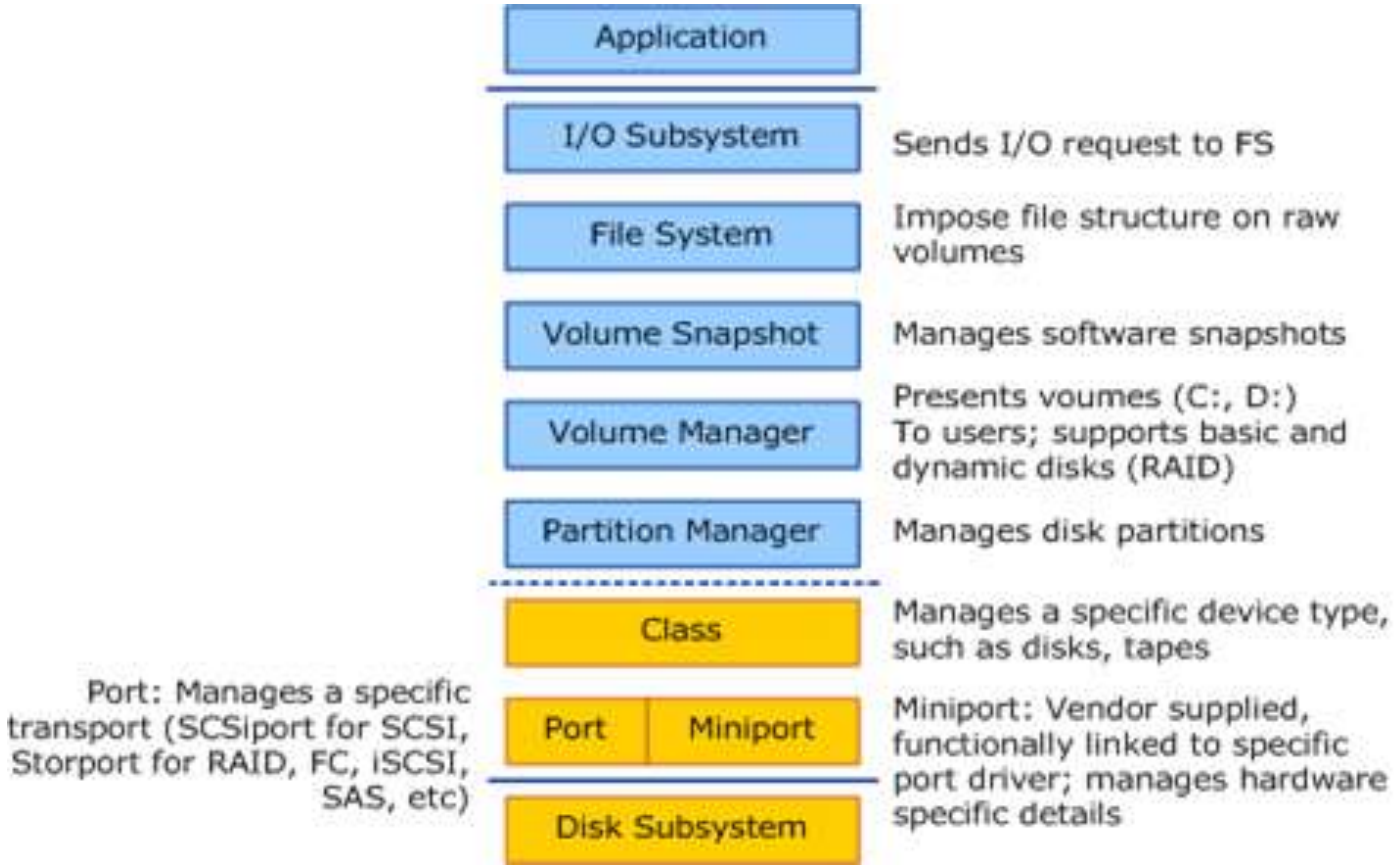
I/O Manager

- The core of the I/O system
  - Provides a framework for other components to have device independent I/O services
  - Responsible for dispatching the service requests to the appropriate device drivers for further processing
- Packet-driven (IRP's, I/O request packets)
- Handles creation and destruction of IRP's
- Offers uniform interface for drivers that handle IRP's

Device Drivers

- Drivers are loadable kernel-mode components
- Code in drivers gets executed in different contexts:
  1. In the user thread that initiated I/O
  2. A system thread
  3. As a result of an interrupt (any thread)
- Different types: file system drivers, protocol drivers, hardware drivers
- **Layered driver model**

Layered Driver Model



F-Secure.

Reversing Drivers: Starting Points

1. The initialization routine (DriverEntry)
   - The entry point of the driver
2. Add-device routine
   - For PnP drivers, called by the PnP manager when a new device for the driver appears
3. IRP dispatch routines
   - Main functionality (”read”, ”write”, ”close”)
   - In many cases the most interesting part

Windbg Demo

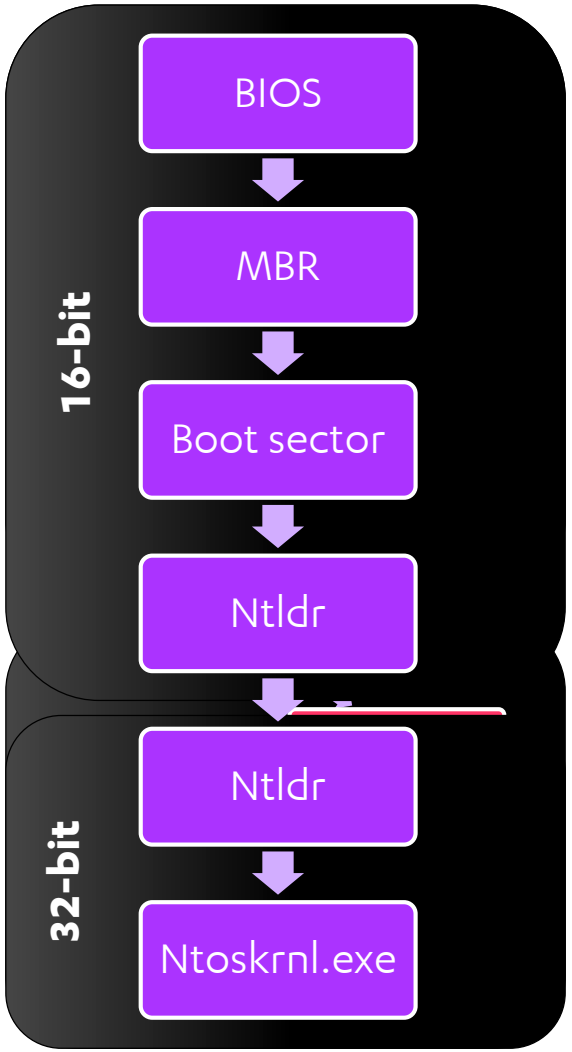
Securing Windows: Driver Signing

- Introduced with 64-bit versions of Windows Vista
- Enforces that following types of drivers are digitally signed:
  - All kernel-mode software
  - User-mode drivers, such as printer drivers
  - Drivers that stream protected content (DRM) are signed with ”special” keys
- Windows 8 UEFI Secure Boot-enabled platforms have additional signing requirements
- Main motivation was to increase the safety and stability of Windows platform
  - Kernel-mode rootkits were becoming too powerful
  - 3rd-party kernel hooks were causing instability and disoptimal performance
- http://msdn.microsoft.com/en-us/library/windows/hardware/ff548231(v=vs.85).aspx

Securing Windows: Kernel Patch Protection

- Introduced in Win2003 SP1 x64 and Windows XP x64 edition
- Prohibits kernel-mode drivers that extend or replace kernel services through undocumented means
- Monitors for any modifications to following critical places:
  - System Service Tables
  - Interrupt Descriptor Table (IDT)
  - Global Descriptor Table (GDT)
  - Model Specific Registers (MSRs)
  - Kernel functions and debug routines
- Triggers Bug Check 0x109: CRITICAL_STRUCTURE_CORRUPTION
  - http://msdn.microsoft.com/en-us/library/windows/hardware/ff557228(v=vs.85).aspx

Rootkit Examples: Mebroot



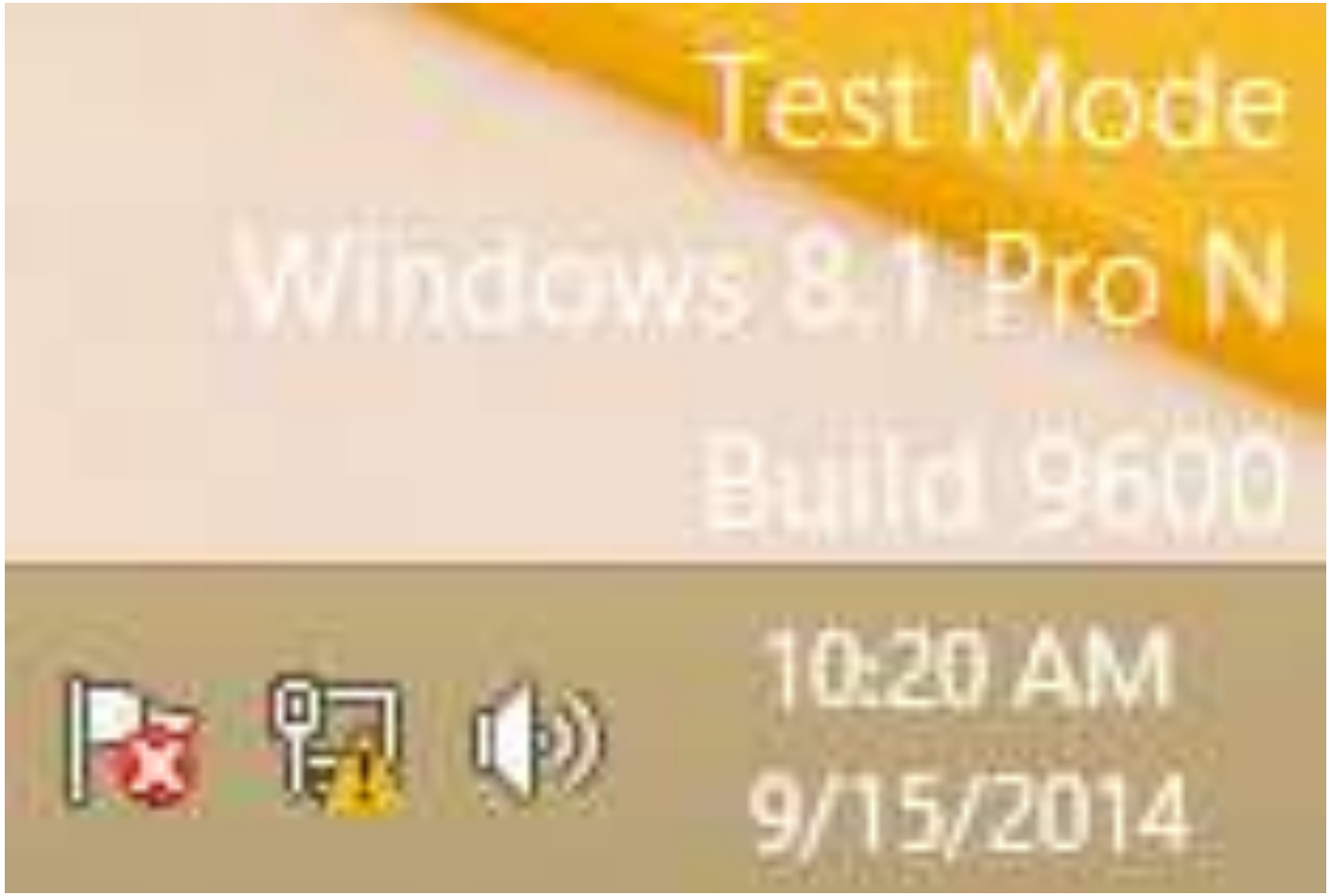
Infected MBR loads and runs “ldr16” which hooks INT13. Original MBR is then called.

INT13 hook patches the real mode Ntldr to disable its code integrity checks and to hook its protected mode part.

”ldr32” patches nt!Phase1Initialization function from ntoskrnl.exe to hook nt!IoInitSystem call.

”ldrdrv” loads Mebroot driver from raw sectors and executes it.

BlackEnergy



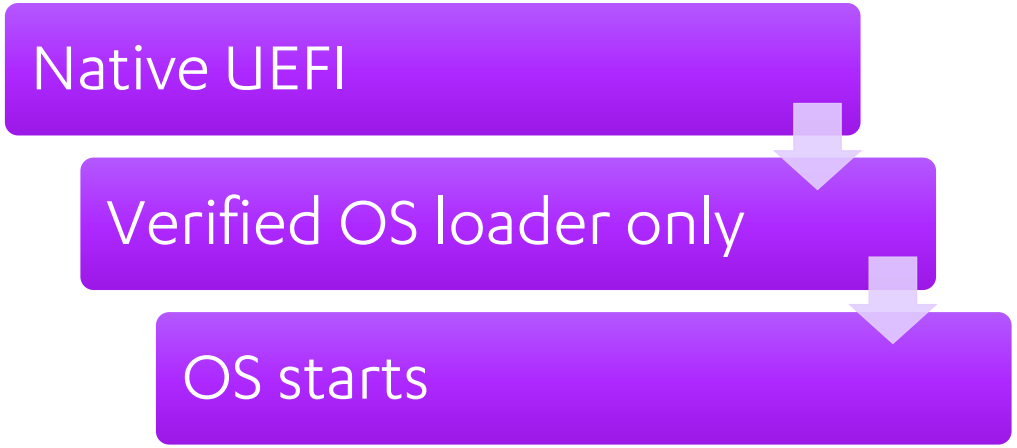
F-Secure.

Turla

```
1.  // Get file handle of VBoxDrv device driver
2.  hVBoxDrvObj = CreateFile("\\\\.\\VBoxDrv", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_
    READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, NULL);
3.
4.  // Step 1 Initialize VBoxDrv's cookie
5.  DeviceIoControl(hVBoxDrvObj, SUP_IOCTL_COOKIE, &Cookie, SUP_IOCTL_COOKIE_SIZE_IN, &C
    ookie, SUP_IOCTL_COOKIE_SIZE_OUT, &lpBytesReturned, NULL);
6.
7.  // Step 2 Creates a fake image
8.  DeviceIoControl(hVBoxDrvObj, SUP_IOCTL_LDR_OPEN, &OpenLdrReq, 0x40, &OpenLdrReq, 0x2
    8, &lpBytesReturned, NULL);
9.
10. // Step 3 Register the fake image and copy shellcode buffer to fake image buffer
11. DeviceIoControl(hVBoxDrvObj, SUP_IOCTL_LDR_LOAD, &LdrLoadReq, 0x88, &LdrLoadReq, 0x1
    8, &lpBytesReturned, NULL);
12.
13. // Step 4 Turn on and initialize the fast VMenter entry point (VMMR0)
14. DeviceIoControl(hVBoxDrvObj, SUP_IOCTL_SET_VM_FOR_FAST, &pVmFastRequest, 0x20, &pVmF
    astRequest, 0x18, &lpBytesReturned, NULL);
15.
16. // Step 5 Call VMMR0 entry point which in turn execute the shellcode that disables g
    _ciEnabled
17. DeviceIoControl(hVBoxDrvObj, SUP_IOCTL_FAST_DO_NOP, g_ciEnabled, 0, g_ciEnabled, 0,
    &lpBytesReturned, NULL);
```

Securing Windows: Secure Boot

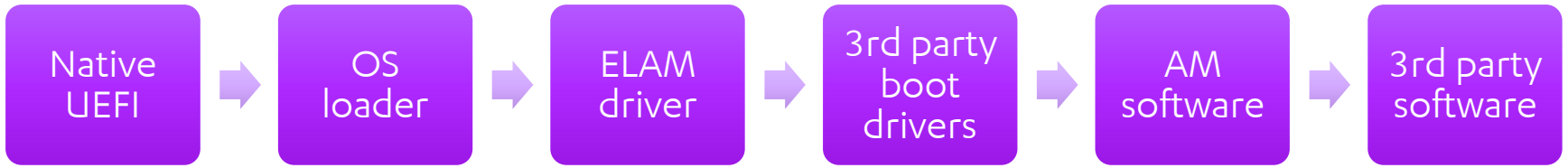
- The firmware enforces policy, only executes signed OS loaders
- OS loader enforces signature verification of Windows components
- Secure Boot is required for Windows 8 certification
- This effectively prevents bootkits from changing the boot or kernel components



F-Secure.

Securing Windows: ELAM

- A Microsoft supported mechanism for AM software to start before all other 3rd party components
- Can prevent malicious boot drivers from executing depending on policy settings
- ELAM drivers must be signed by a special Microsoft certificate



F-Secure.

Securing Windows: Protected Processes

- Introduced in Windows 8.1
- Generalization of protected process technology and applied to critical system processes
  - csrss.exe, services.exe, smss.exe, …
- Protected Service is a service running as a system protected process
  - Only for code signed by a special certificate provided at runtime to Windows