Windows for Reverse Engineers

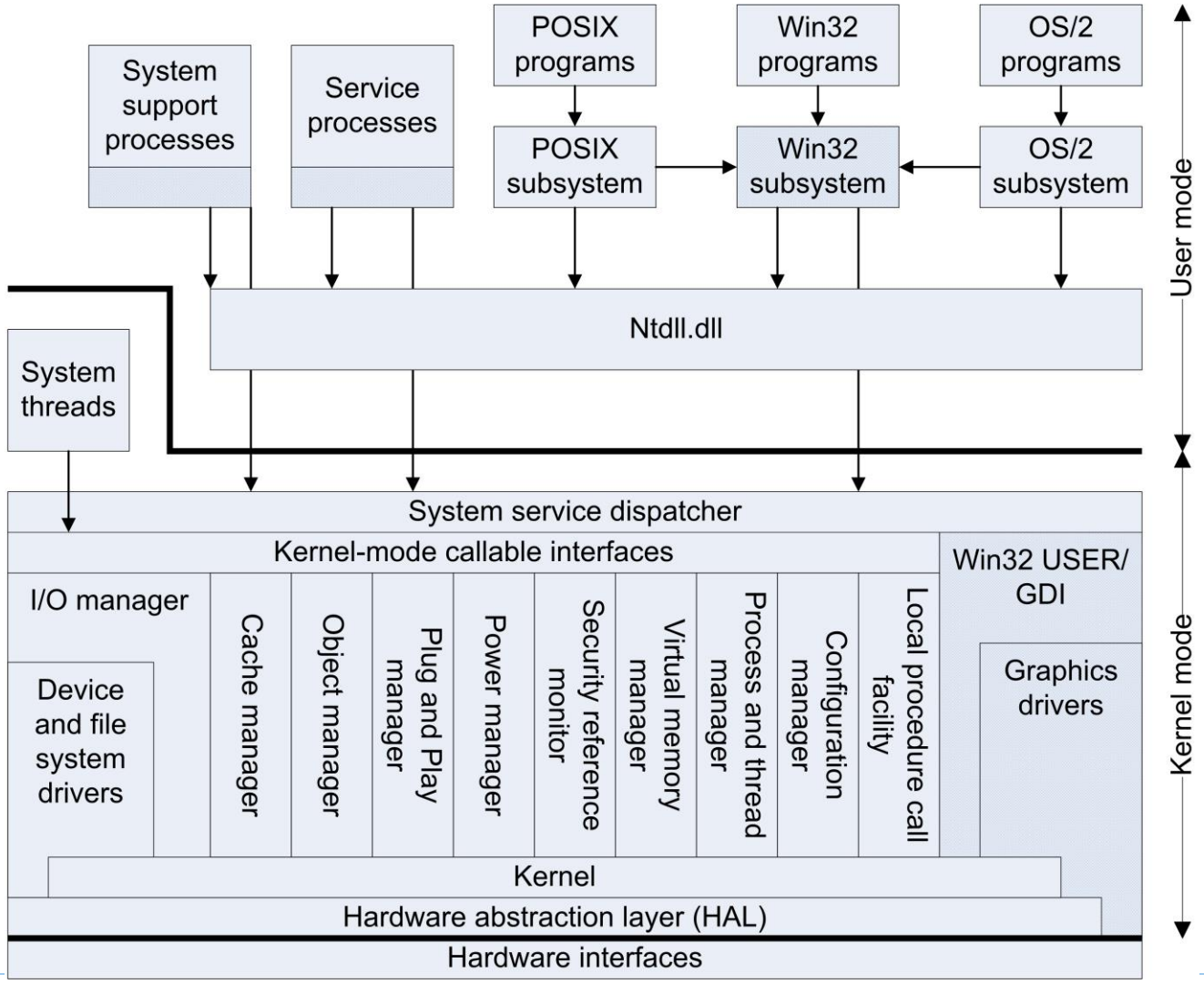
- **T-110.6220 Special Course in Information Security**

Kimmo Kasslin, 26th Feb 2014

F-Secure

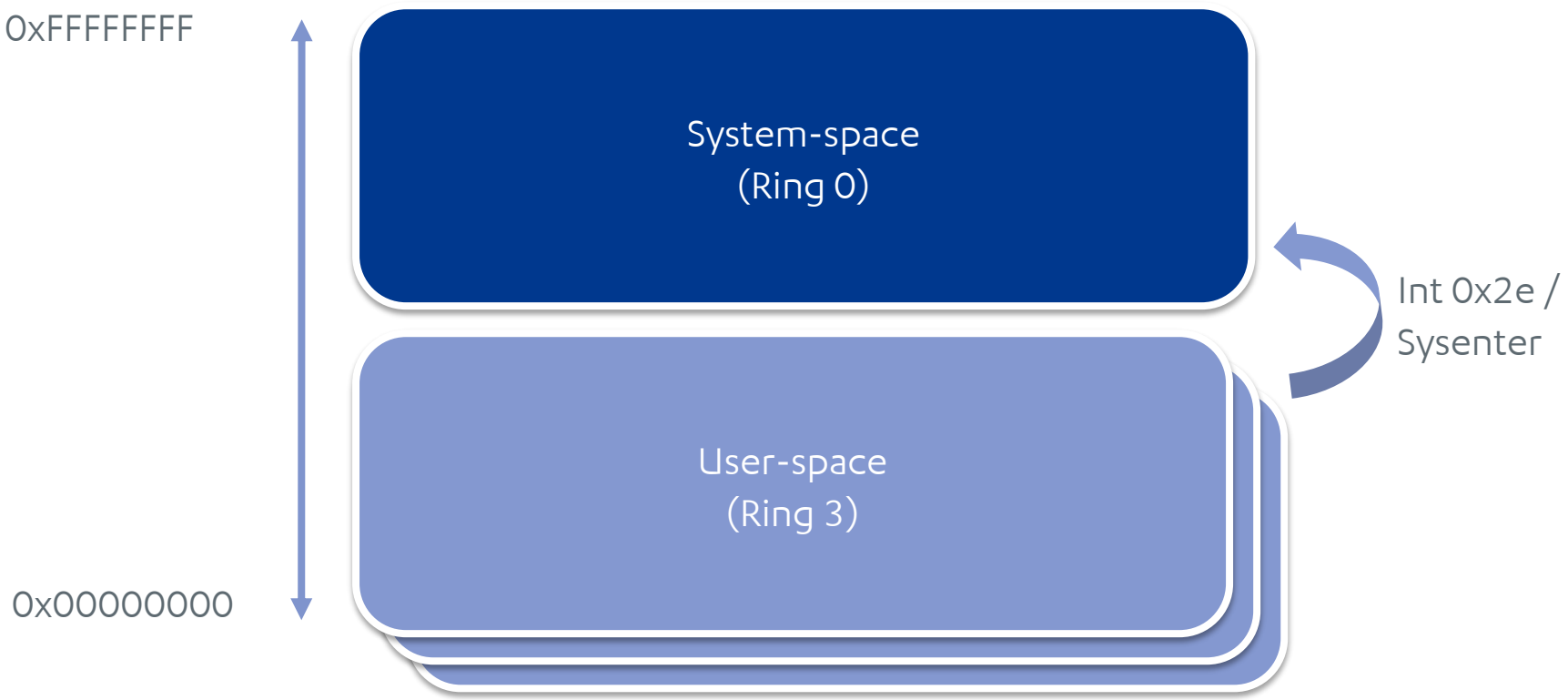
Architecture

Windows architecture

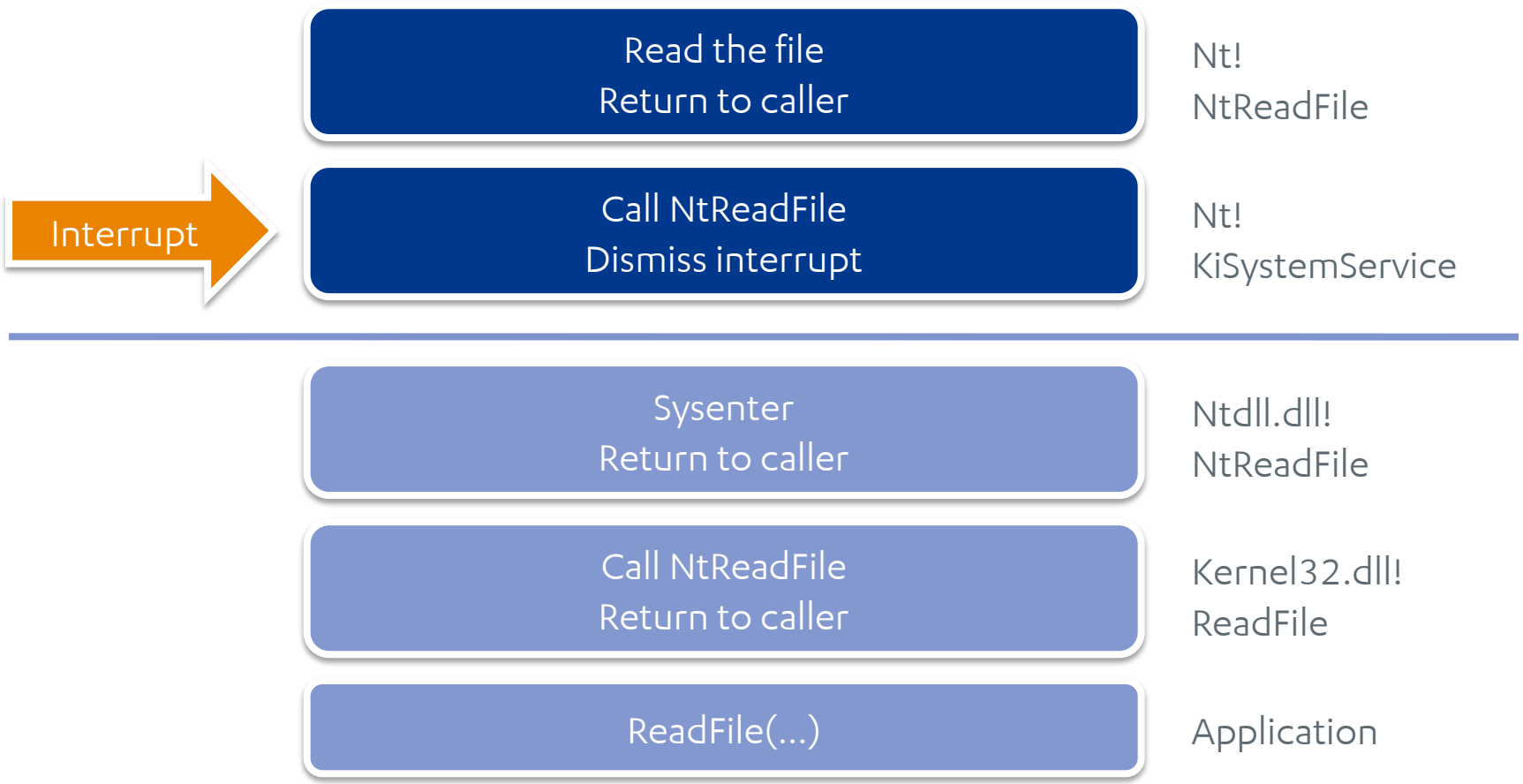


System Mechanisms

Kernel-mode & user-mode



System Service Dispatching

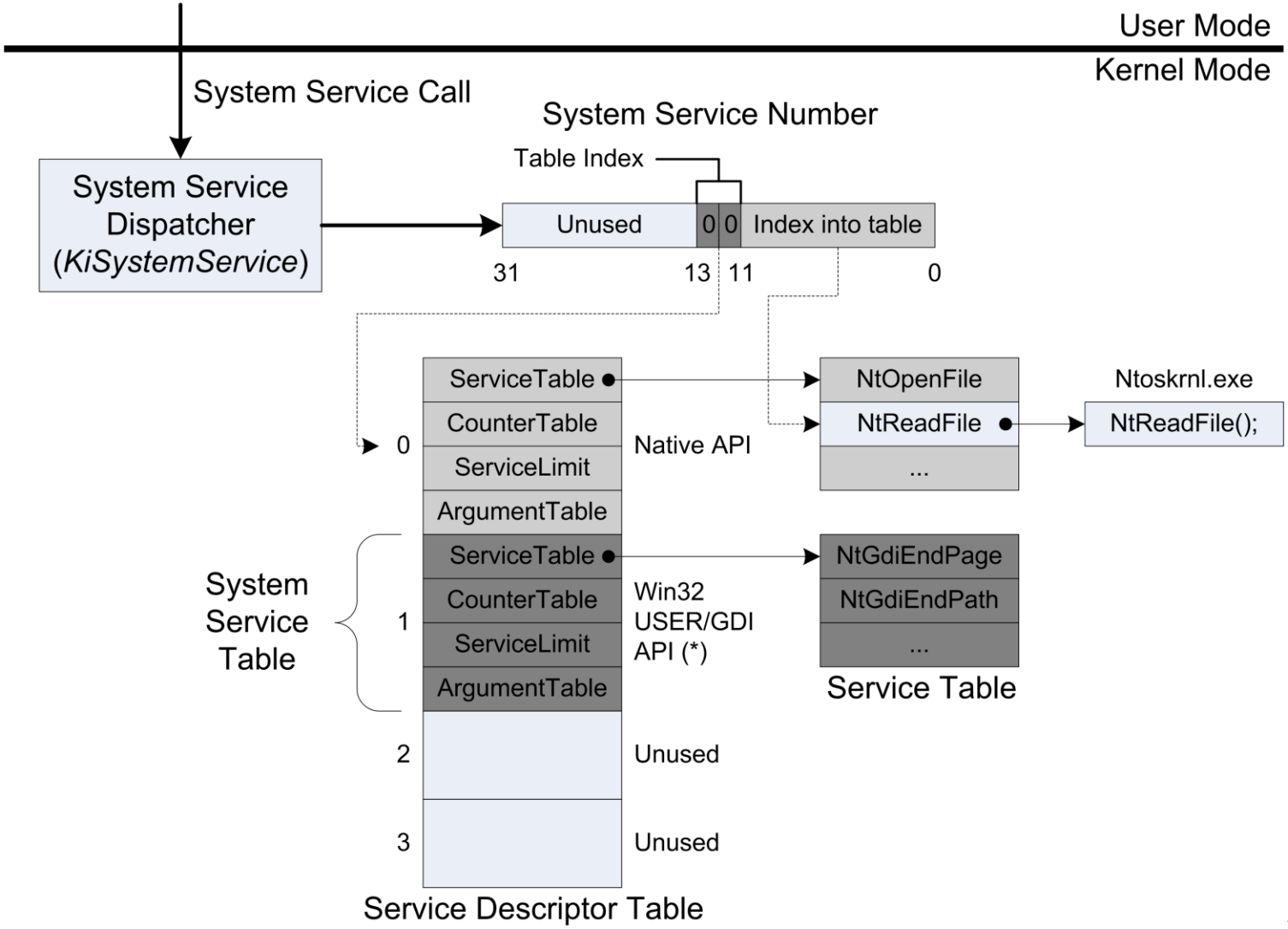


F-Secure.

System Service Dispatching

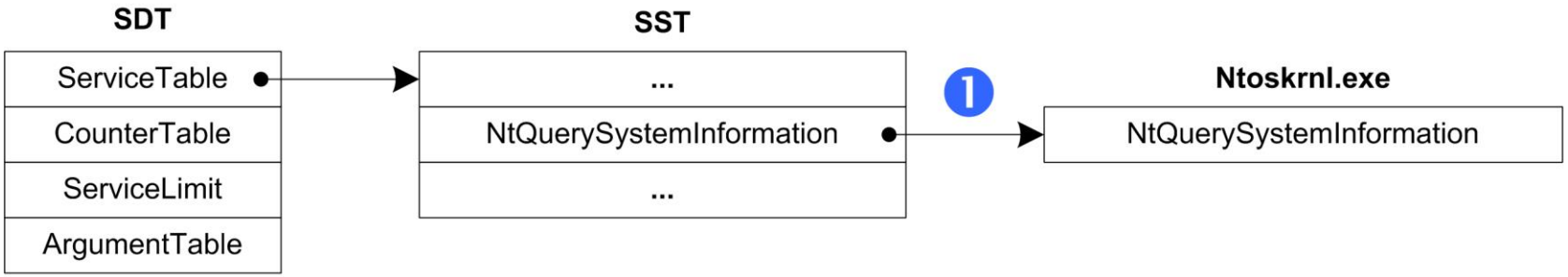
```
Windows 2000:

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
```

System Service Dispatching

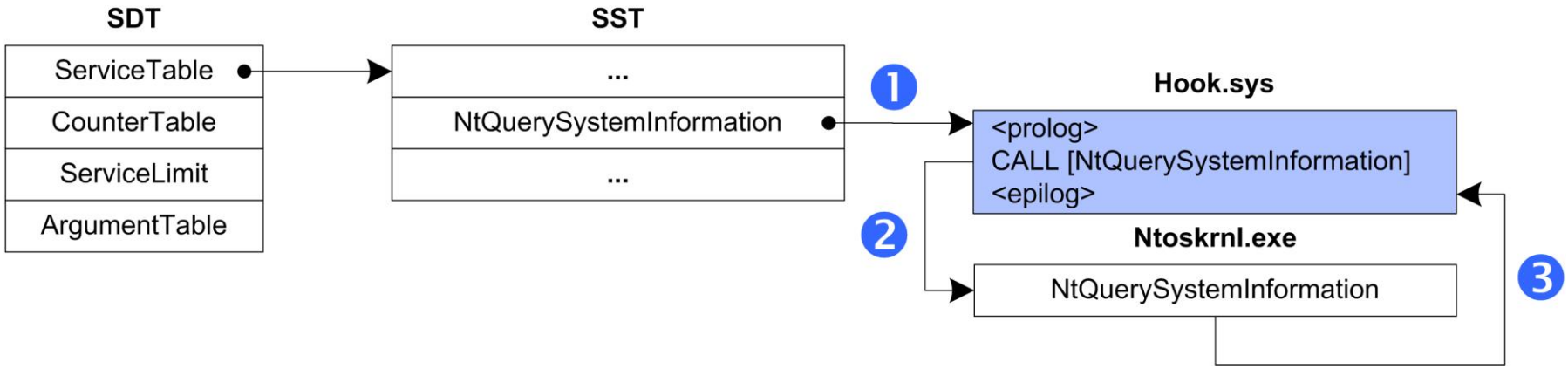


Rootkit Techniques:
Hooking the Handler Table

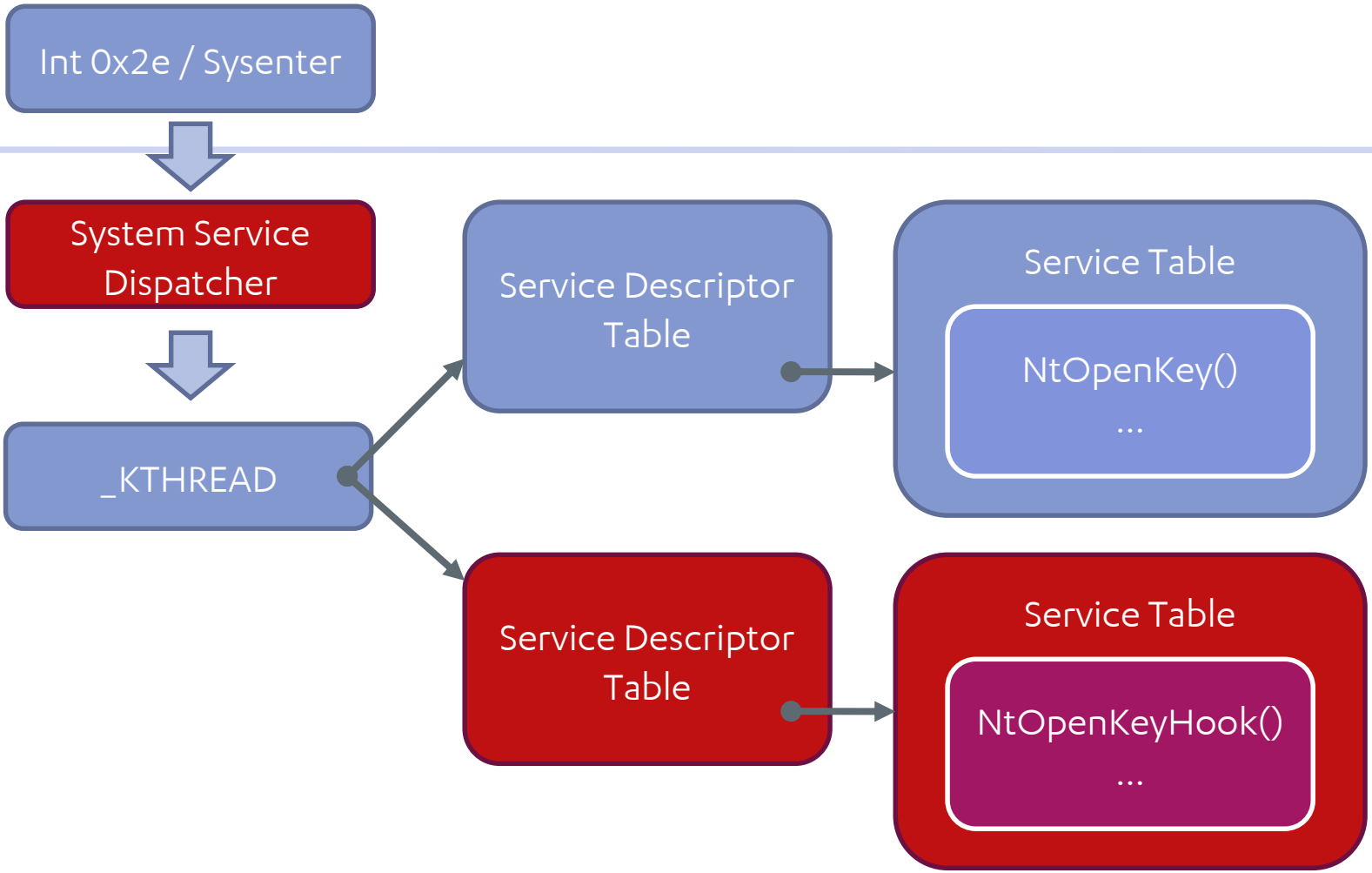
Before:



After:



Rustock – System Call Hooking



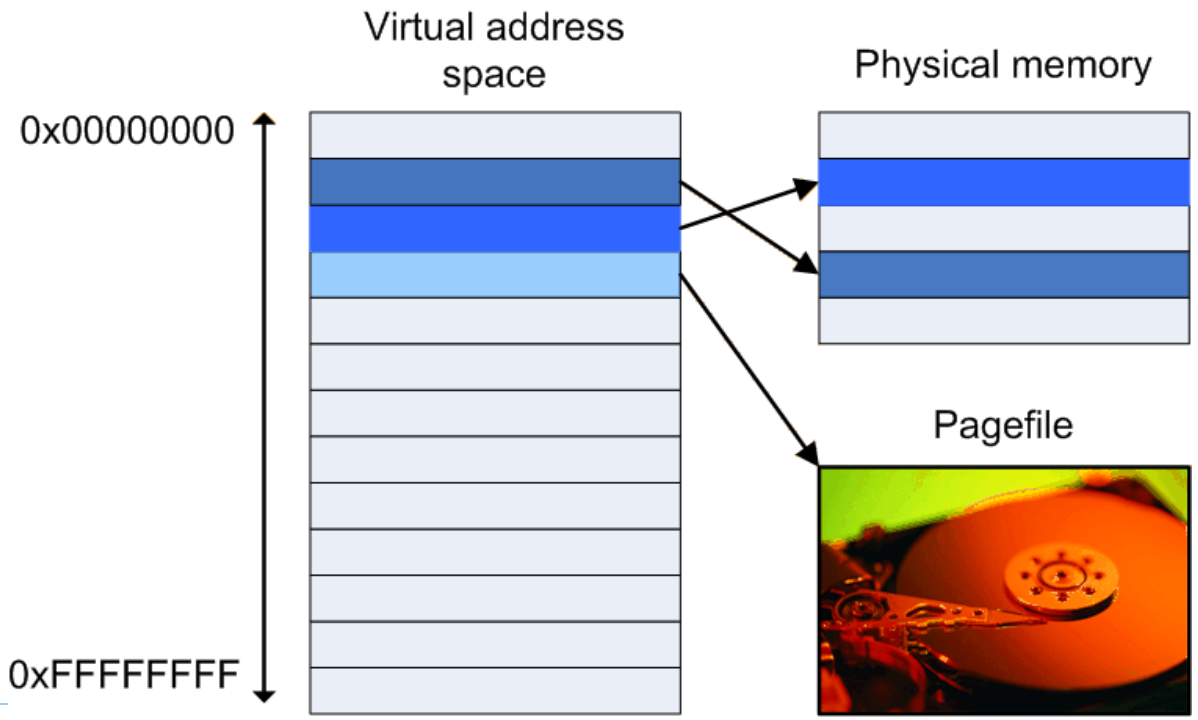
Memory Management

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

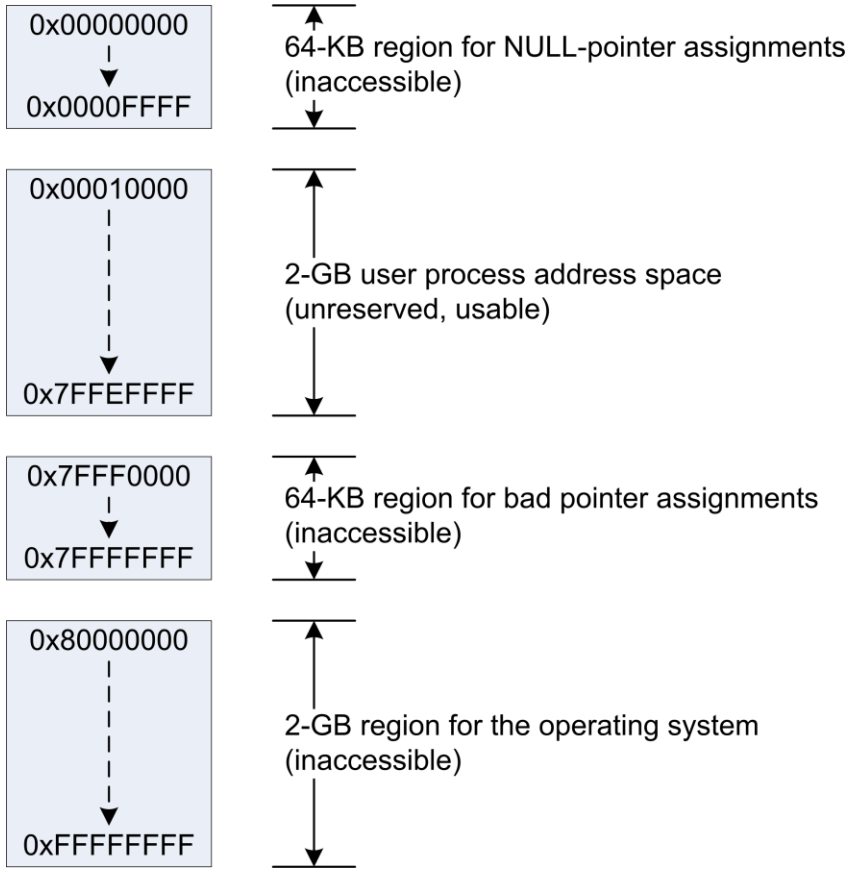
Virtual memory

- 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



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



Processes and Threads

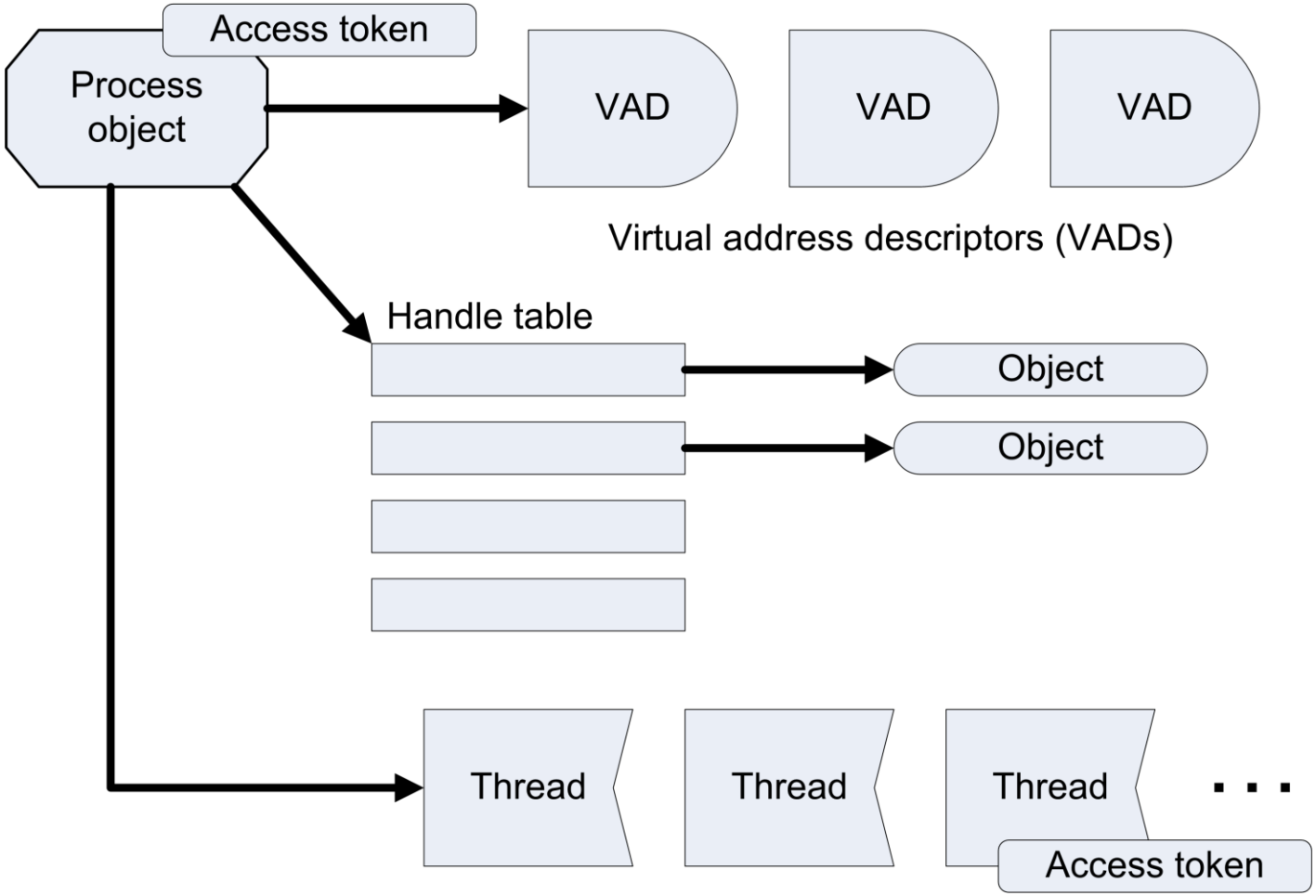
Processes

- Process is an abstraction of a running program
- Process consists of following essential components:
  - A private virtual address space
  - An executable program
  - 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)

Threads

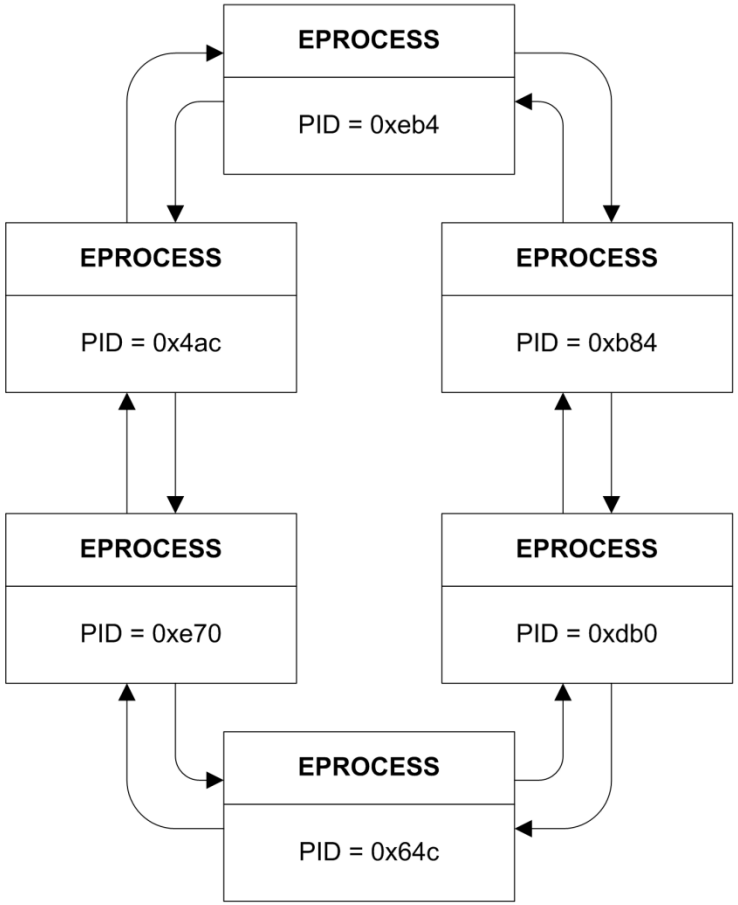
- Thread is an entity scheduled for execution on the CPU
- Thread consists of following essential components:
  - The CPU state
  - Two stacks, one for kernel-mode and one for user-mode
  - Thread-Local Storage (TLS), a private storage area that can be used by subsystems, run-time libraries, and DLLs
  - 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

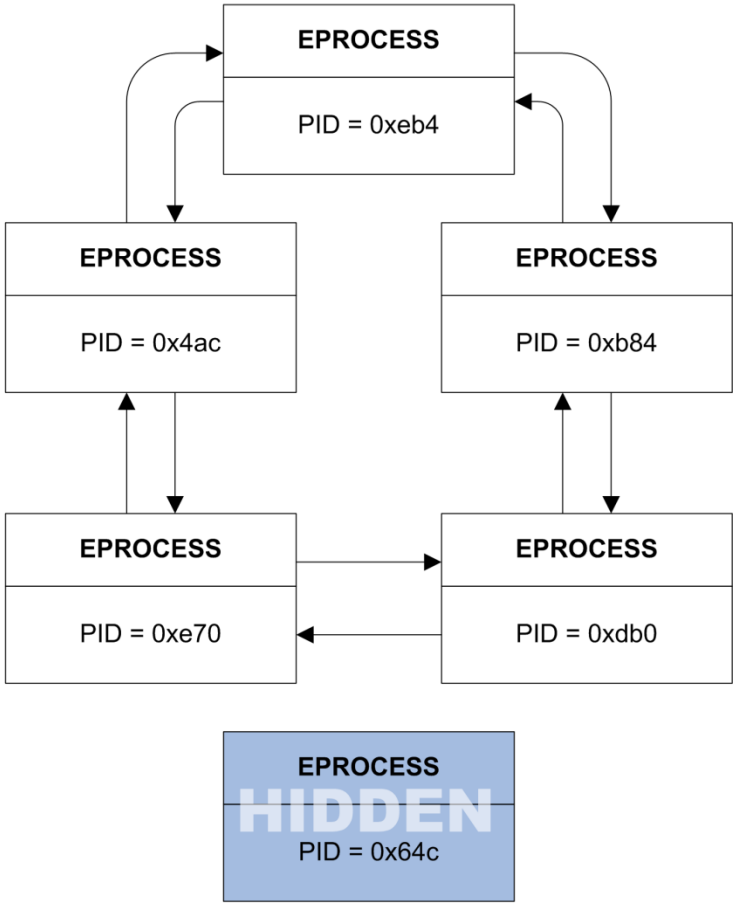


Rootkit techniques:
in-memory data structure manipulation

Before:



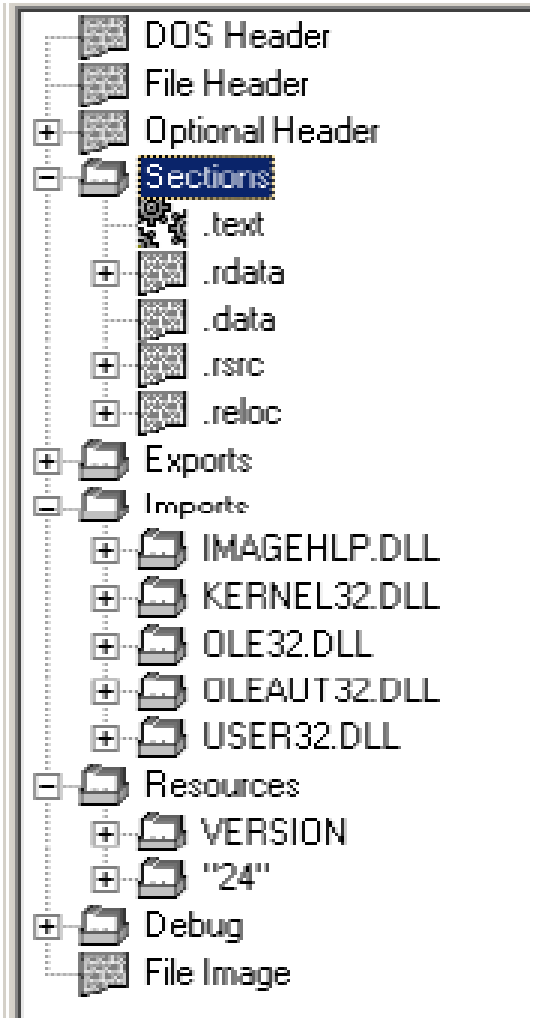
After:



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

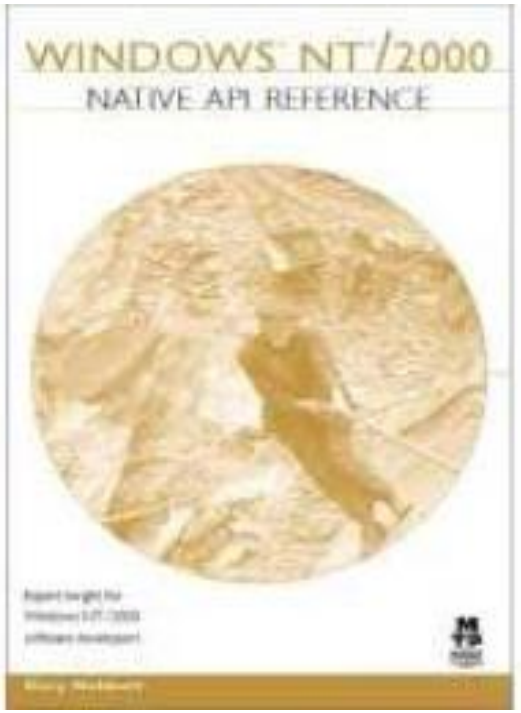


Windows API

- 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/

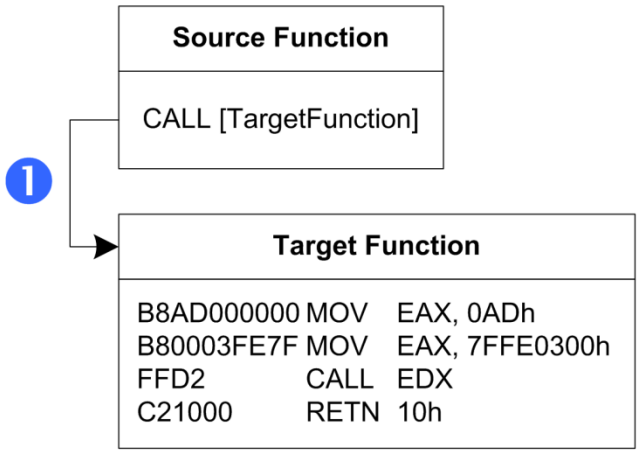


API Hooking

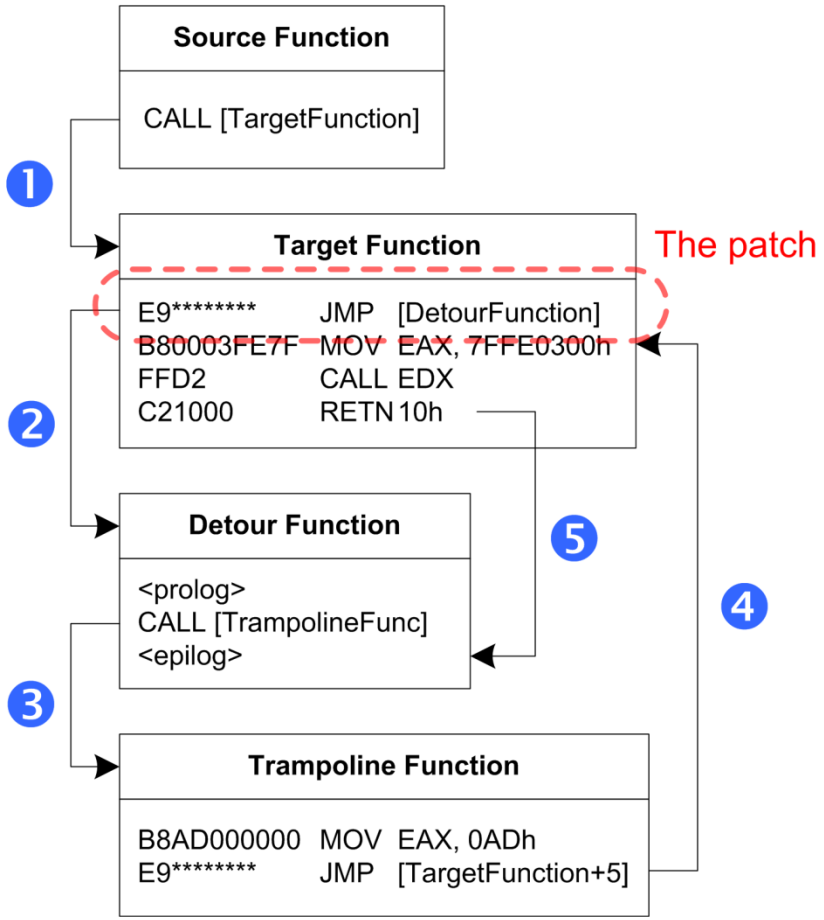
- Hooking is a technique to instrument functions and extend or replace their functionality
  - For example, you want to know each time a program calls CreateFile() and strip write access from the caller
- Many implementations
  - Hooking a function table (IAT, SSDT, IDT, ...)
  - Inline hooking (patching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

Rootkit techniques: Inline Hooking

Before:

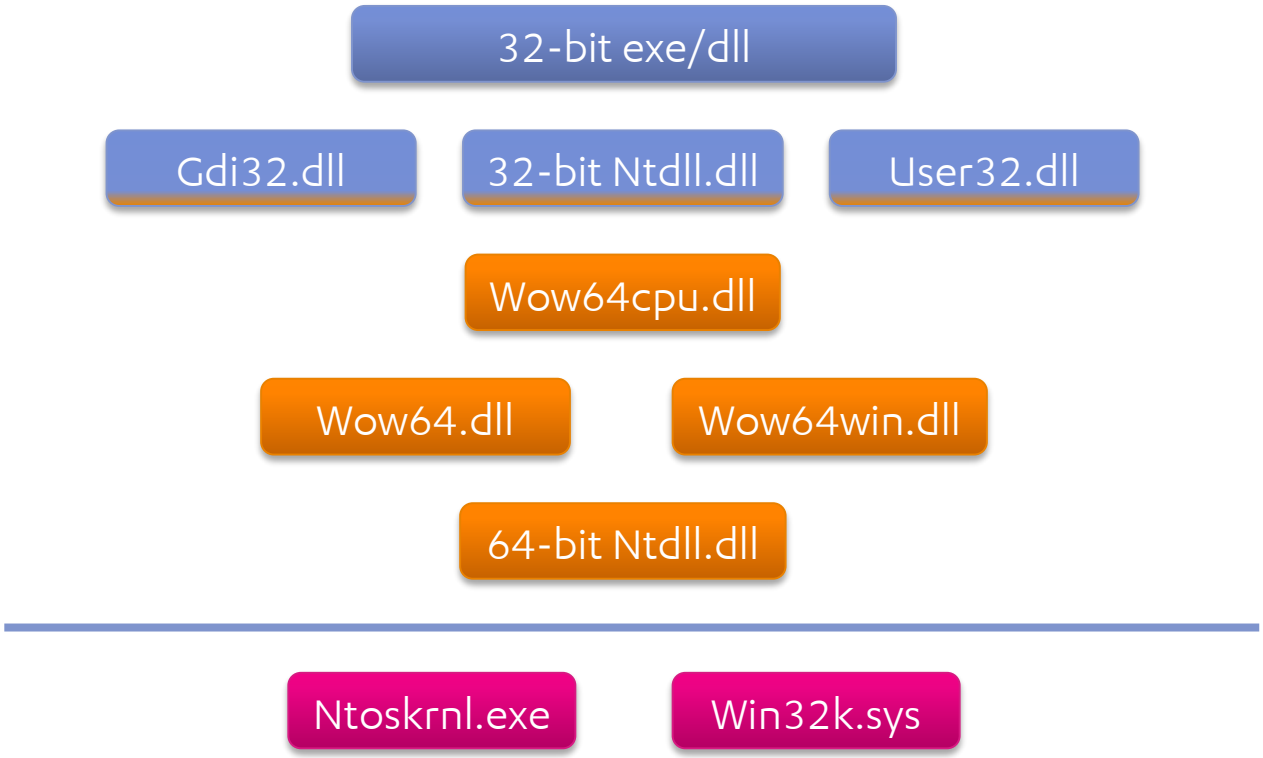


After:



Wow64

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



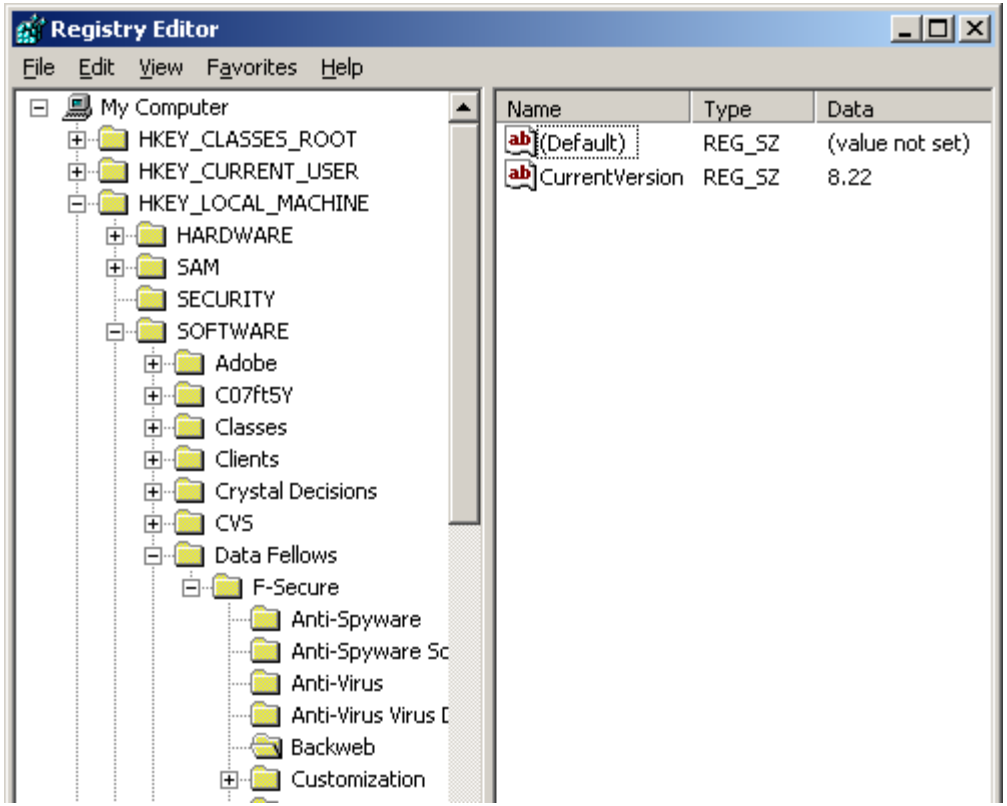
Wow64 – File System Redirection

- 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

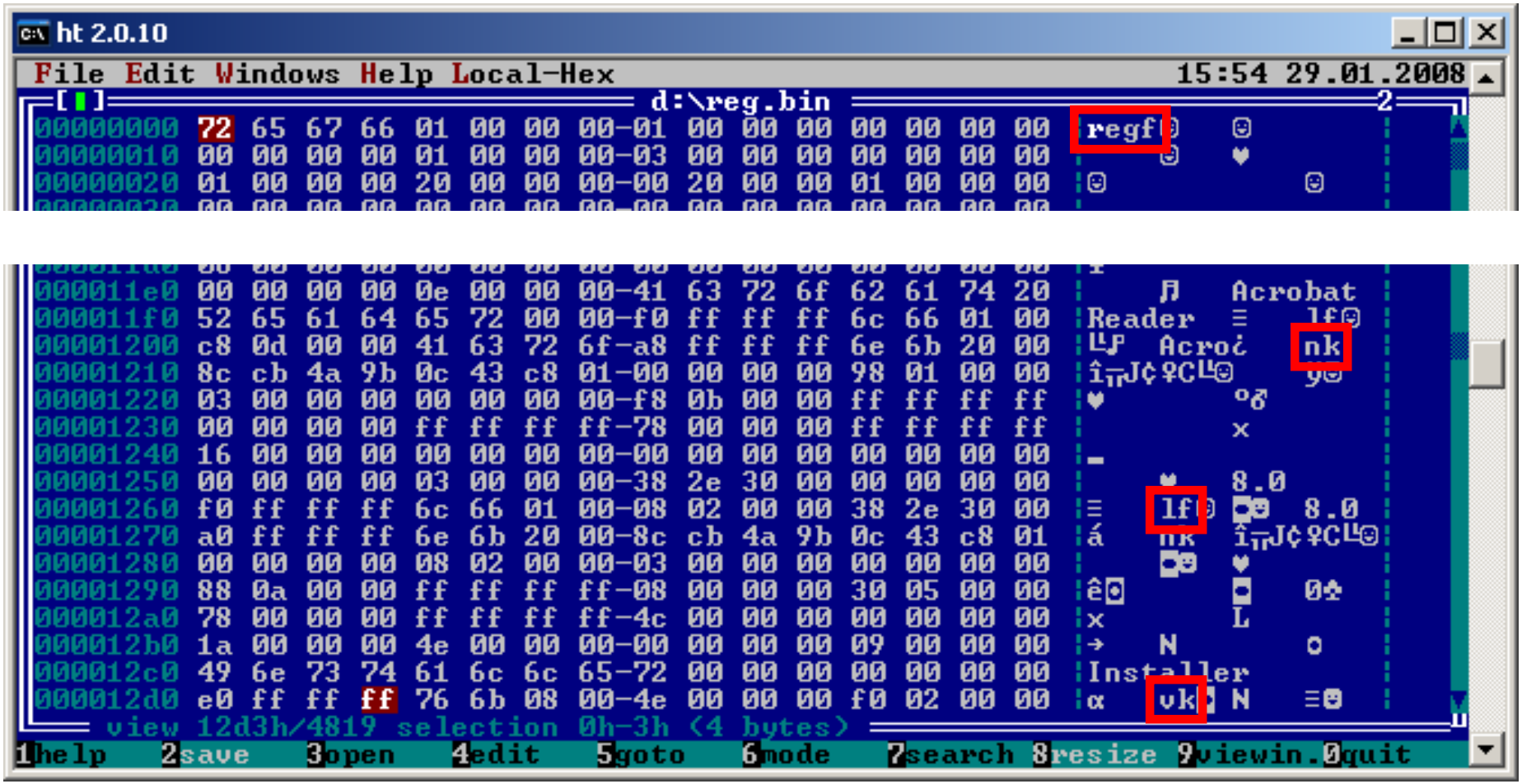
Management Mechanisms

Registry

- A directory 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\



Registry hive format



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\User Assist

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

Demo: services and drivers

- Let's look at c:\windows\system32\smss.exe.
  - Is it a service?
  - An application that controls a service?
  - A driver?

File Systems

Windows File System Formats

- 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

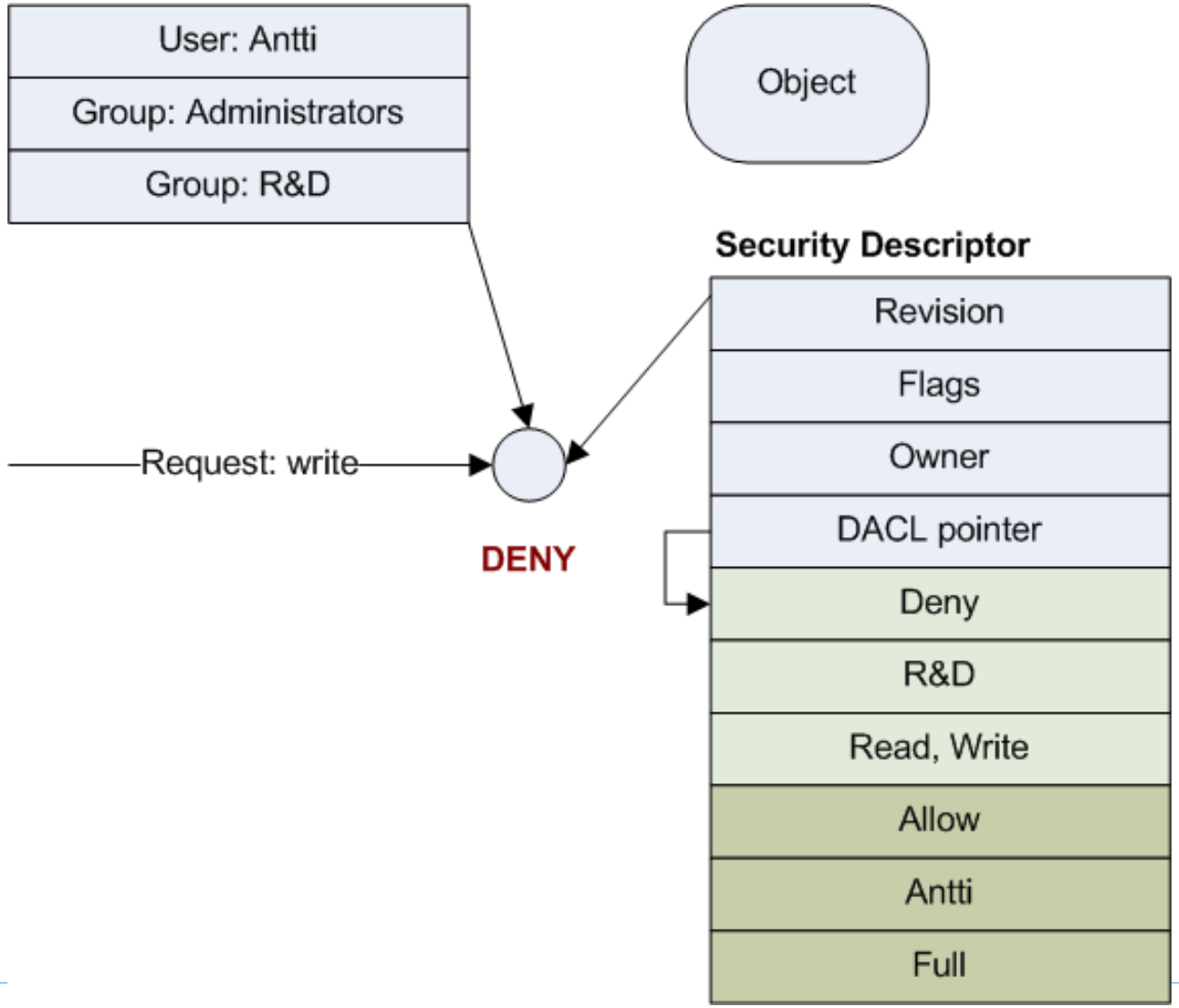
- Designed to improve perfomance and reliability over FAT
- Some interesting NTFS Features
  - Disk quotas
  - Encrypting File System (EFS)
  - Multiple data streams
  - Hard links and junction points
  - Unicode-based naming

Security Mechanisms

Objects and how to protect them

- Almost everything is an object (file, process, thread, desktop, ...)
- Basic concepts
  - Security Identifier (SID) is a unique ID for any actor
    - "S-1-5-21-525843606-2469437151-111719316-1006"
  - A token identifies the security context of a process
    - "Member of Administrators group, can shut down OS"
  - Security Descriptor specifies who can do what to an object
    - Owner
    - Discretionary Access Control List (DACL)
  - Privileges
    - Most dangerous: Debug programs, Take ownership, load and unload device drivers

Access check



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

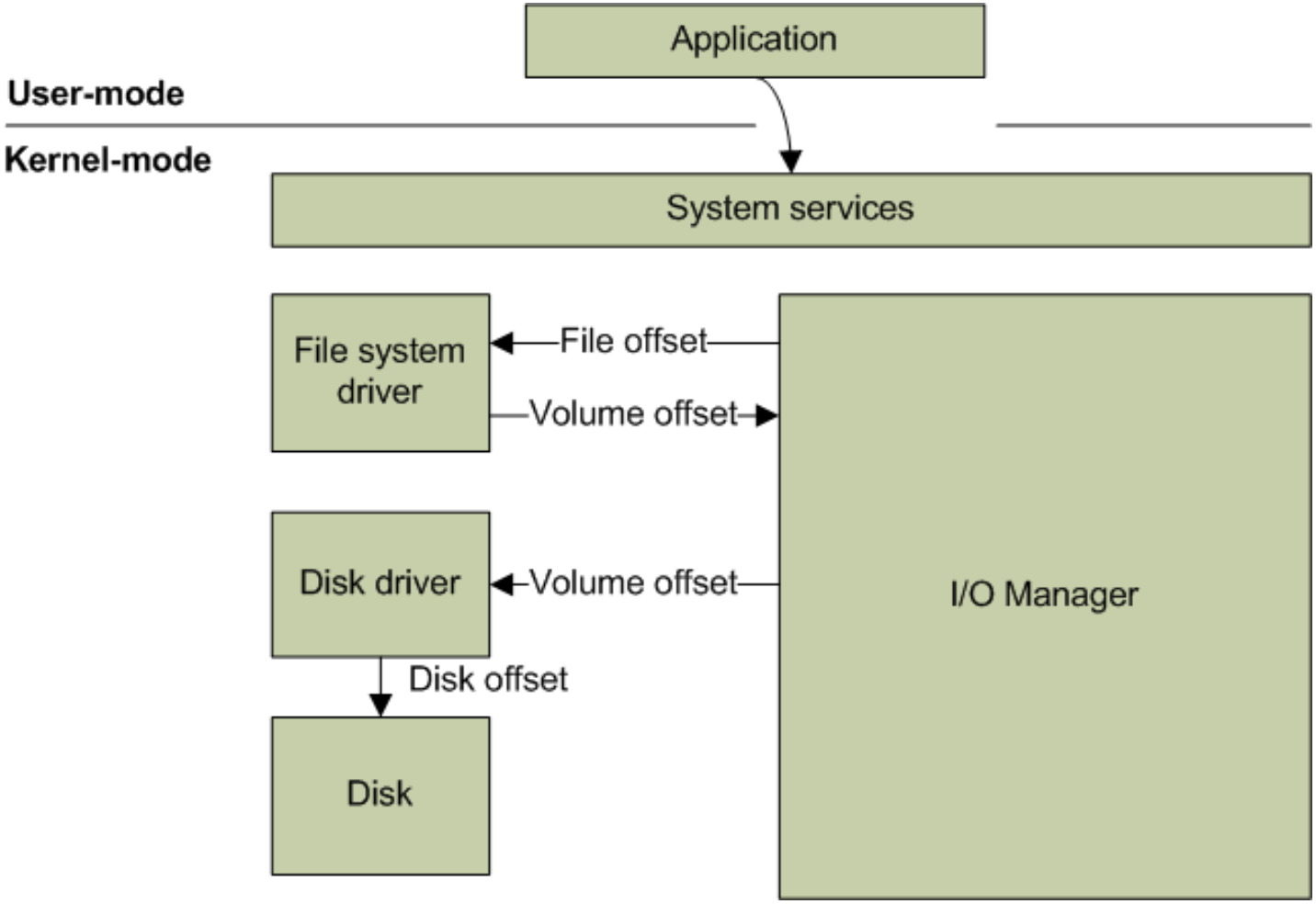
I/O Manager

- 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

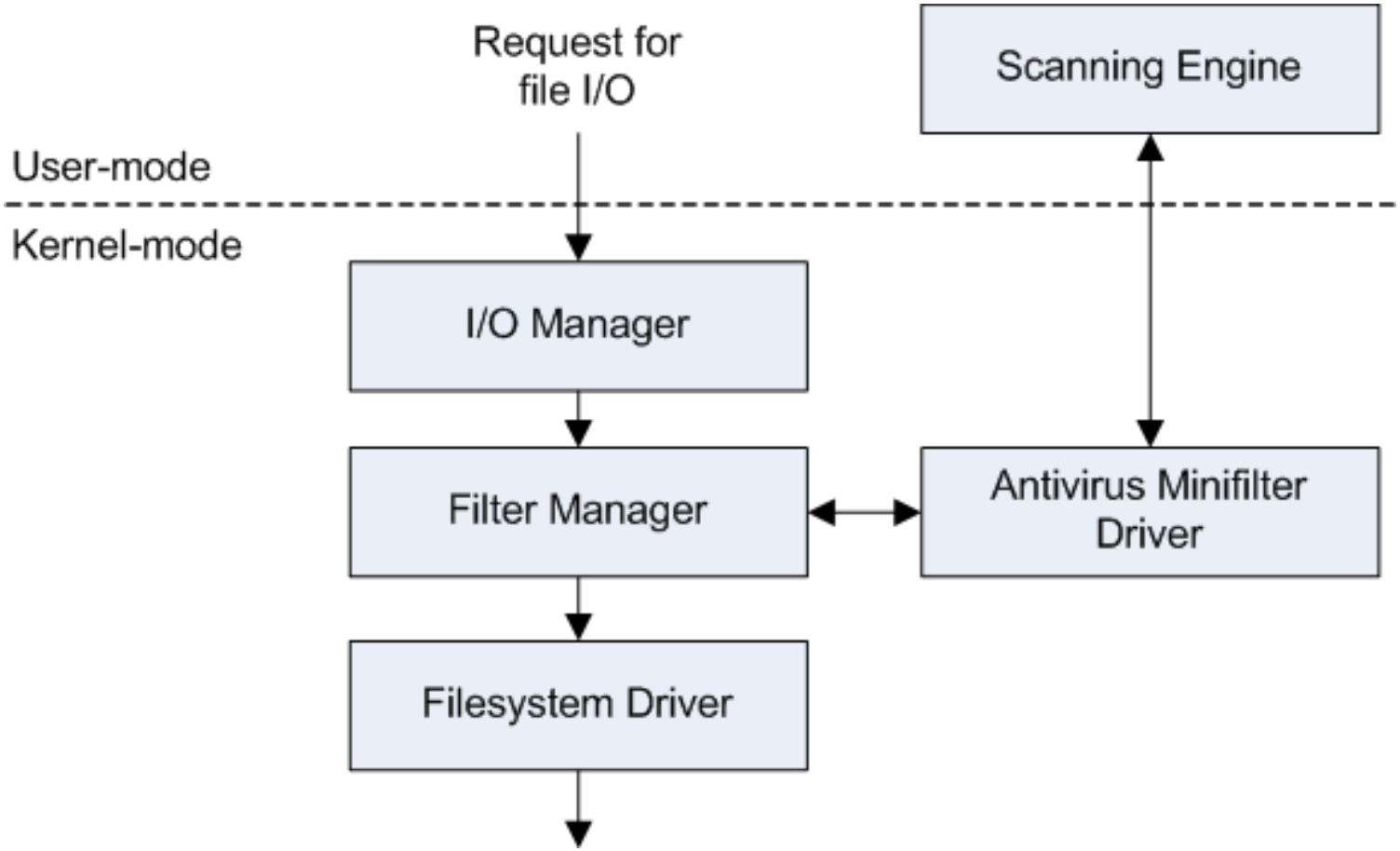
Device drivers

- 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



Driver example:
How on-access scanning might work



Interesting elements of a driver

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

Windows API for Malware Analysis

Processes and threads

- CreateProcess, TerminateProcess
- CreateThread, _beginthread
- CreateRemoteThread
- GetThreadContext, SetThreadContext
- CreateToolhelp32Snapshot
- Process32First, Process32Next
- NtQueryInformationProcess
- NtQueryInformationThread

Memory

- ReadProcessMemory
- WriteProcessMemory
- VirtualAlloc
- VirtualProtect

Files and registry

- CreateFile
- FindFirstFile, FindNextFile
- RegOpenKey
- RegCreateKey
- RegEnumKey
- RegEnumValue
- ... and lots more

Services

- OpenSCManager
- CreateService
- StartService
- StartServiceCtrlDispatcher
- RegisterServiceCtrlHandler

Miscellaneous

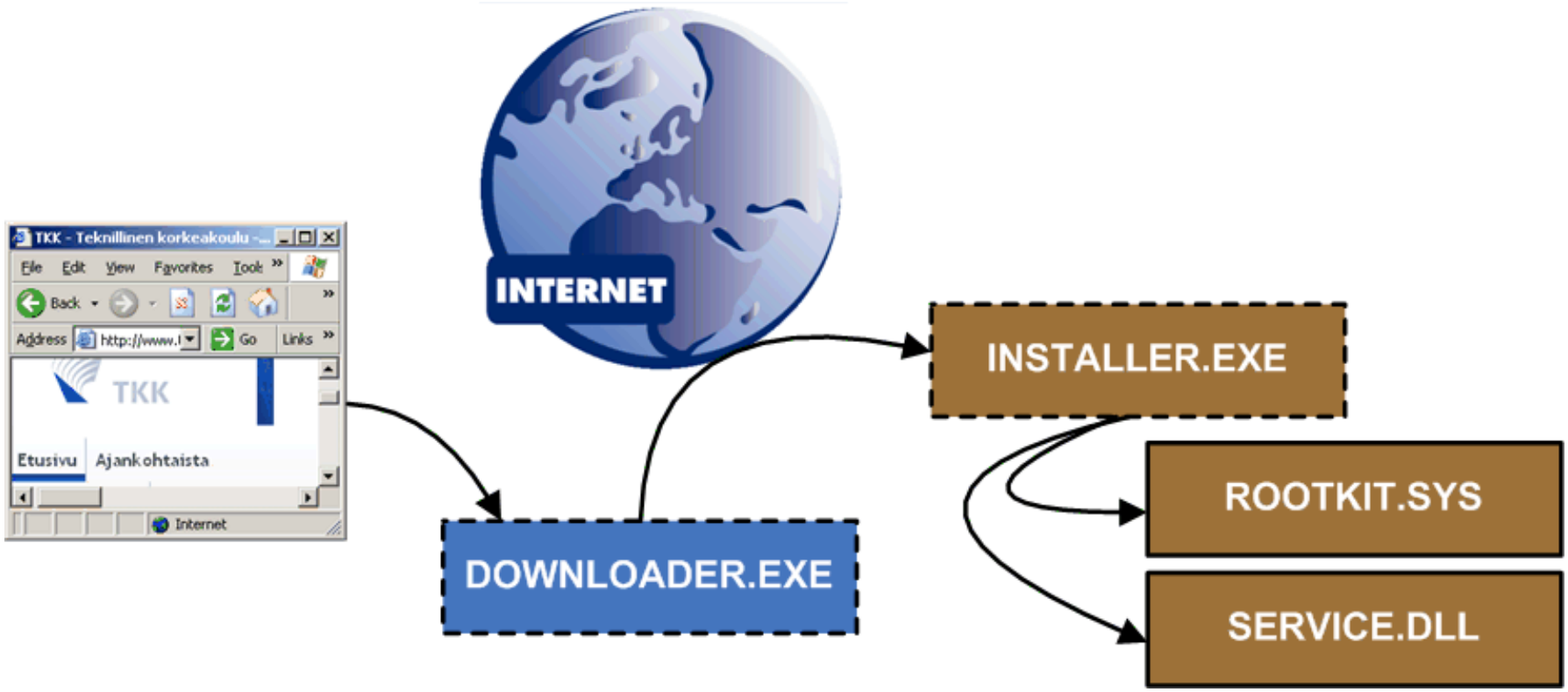
- LoadLibrary
- GetProcAddress
- IsDebuggerPresent
- DeviceIoControl
- FindResource, LoadResource, LockResource
- SetWindowsHook

Arms Race Between Malware Authors and Microsoft

What Is a Rootkit?

- In the early 1990s rootkits used to be a set of tools that allowed root-level access to the system, hence the name
  - Back then, hiding malware was called "stealth"
- Currently the word "rootkit" is used to describe an application that uses some kind of filtering for hiding things
  - This "rootkit" is actually feature - not a class of programs
  - Rootkits usually hide files, processes, network connections, and registry keys
  - So, the term "rootkit" has replaced "stealth"

Example: Anatomy of a Rootkit Infection

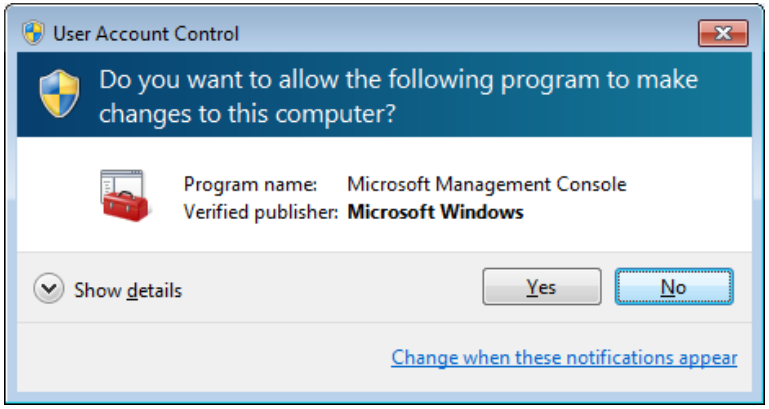


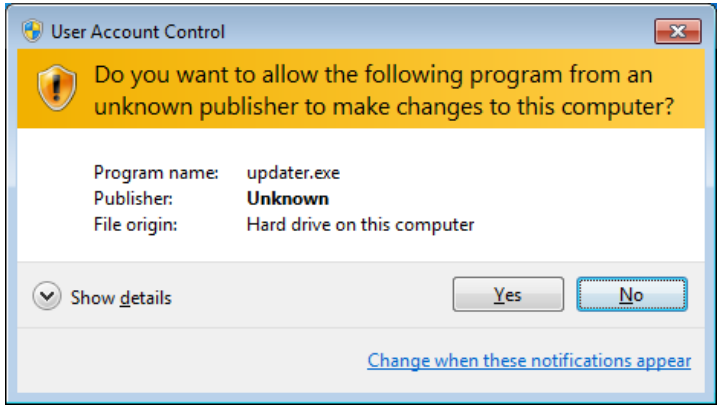
Vista Introduced New Security Features

- Following features were introduced to mitigate initial infections
  - User Account Control
  - Application Sandboxing
  - Protected Processes (mostly for DRM)
  - Address Space Layout Randomization and other compiler improvements
- Following improvements were done to mitigate the rootkit problem
  - Driver Signing Policy restrictions for 64-bit Windows
  - Kernel Patch Protection

User Account Control (UAC)

- UAC is meant to **enable** users to run with standard user rights, as opposed to administrative rights
- Achieved by creating two tokens when user logs in to an administrative account
  - Filtered admin token and normal admin token
- Elevation is required to switch to normal admin token
- Confirmation (or admin redentials) asked in Secure Desktop mode





F-Secure.

Sandboxing Applications

- Especially in later Windows versions (Vista, Windows 7), extensions to the security model can be used to isolate less trustworthy applications
  - Prevent exploited applications from changing the system
  - In practice, requires support from the application itself
  - Protected Mode Internet Explorer, Adobe Reader X, Google Chrome
- A process is isolated from the rest of the system with additional mechanisms, e.g.
  - Restricted tokens
    - See CreateRestrictedToken() on MSDN
  - Job objects
    - CreateJobObject(), AssignToJobObject(), SetInformationJobObject()
  - Integrity Levels
    - Security descriptors and tokens are assigned an Integrity Level
    - A process with a lower IL cannot modify (sometimes not read) higher IL objects

Protected Process

- Process running with the debug privilege can do anything on other processes
  - Read and write arbitrary process memory
  - Inject code
  - Suspend and resume threads, and change their context
- Protected processes disallow these actions at kernel level
  - Bulk of process creation occurs in kernel mode
  - Process manager denies all dangerous access rights from non-protected processes
- Implemented to support reliable and protected playback of DRM content

Driver Signing Policy

- 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

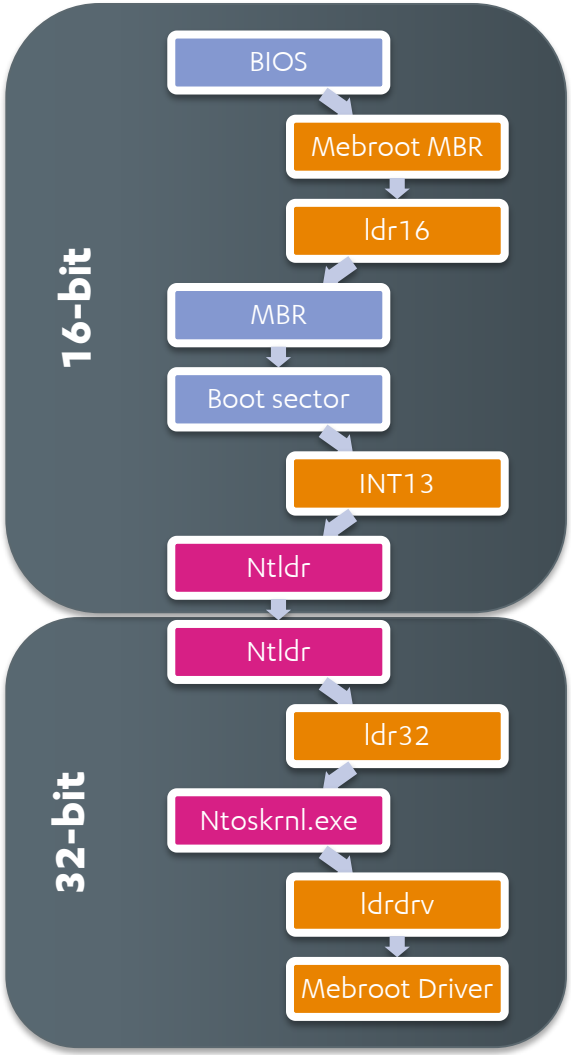
Kernel Patch Protection

- 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

Malware Authors Respond

- To some extend Microsoft was successful in their efforts
  - Kernel-mode rootkits practically disappeared
  - Only a few KPP PoC attacks demonstrated but no evidence of real attacks so far
- However, malware adapted and continued to cause harm to the ecosystem
  - Bootkits appeared and were able to bypass both KPP and signing enforcement
  - Rootkits moved to user mode where they can still be effective
  - Malware targeting modern web services just need to compromise user's browser which does not require admin privileges

Mebroot: Boot Sequence



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

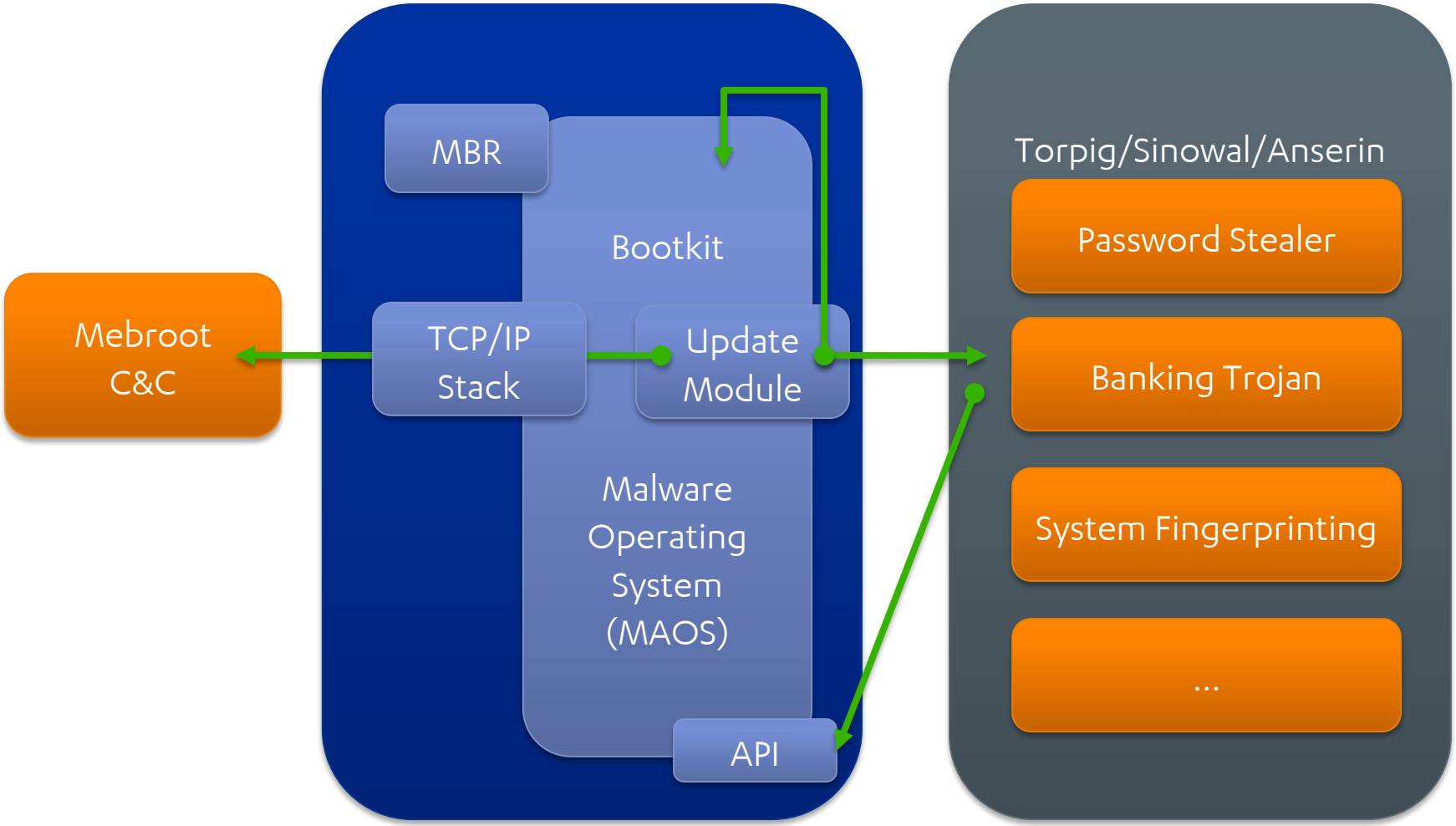
INT13 hook patches the real mode Ntldr to disable its code integrity checks and to hook its protected mode part.

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

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

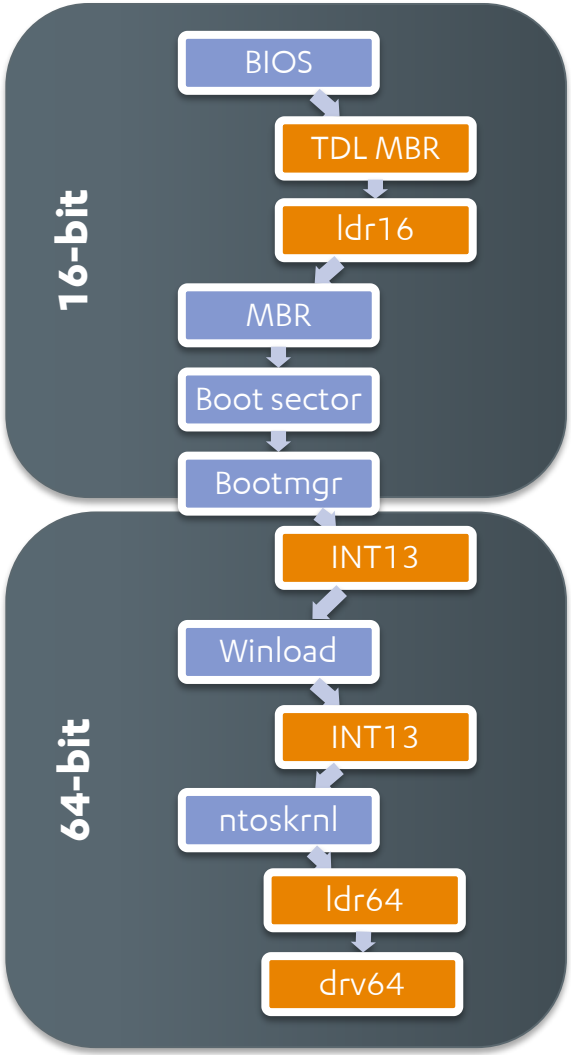
F-Secure.

Mebroot – Architecture



Adapted from Andreas Greulich, MELANI/GovCERT.ch

TDL: Boot Sequence



Infected MBR decrypts and runs "ldr16" which hooks INT13. Original MBR is then called.

INT13 hook tricks boot into Windows PE mode by modifying BCD (*BcdLibraryBoolean_EmsEnabled* → *BcdOSLoaderBoolean_WinPEMode*) and patching winload.exe (*"/MININT"* → *"IN/MINT"*)

INT13 hook replaces kdcom.dll with "ldr64"

"ldr64" decrypts and runs "drv64"

F-Secure.

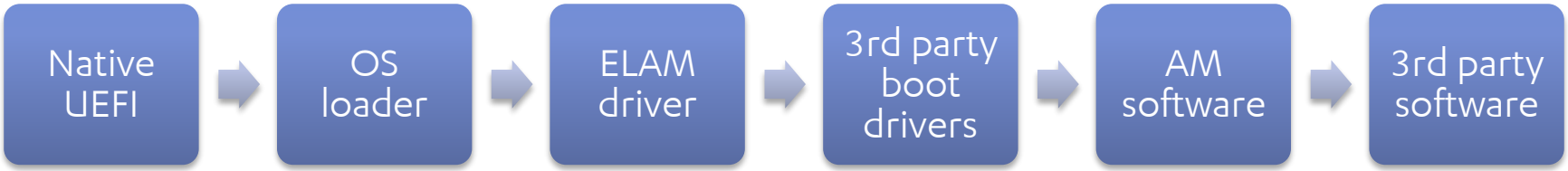
Windows 8 comes with new tricks

- Bootkits and signed malicious kernel drivers were countered by these features:
  - Secure Boot
  - Early Launch Antimalware (ELAM) driver
- Remote code injection to trusted and high-privilege processes in user mode brought us System Protected Processes and Protected Services

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

Early Launch Antimalware (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

System Protected Process

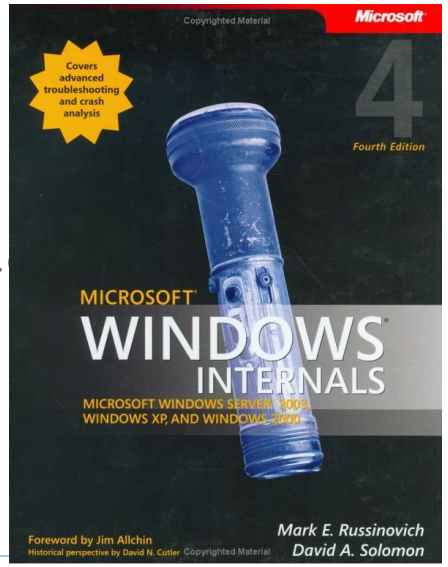
- Introduced in Windows 8.1
- Generalization of protected process technology and applied to critical system processes
  - csrss.exe, services.exe, smss.exe, ...
- Protected Service is a service running as a system protected process
  - Only for code signed by a special certificate provided at runtime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

Additional Information about R(B)ootkits

- 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
- Kasslin, K.; Tikkanen, A. (2010). Rootkits in the real world today. #days Security & Risk Conference
  - http://www.youtube.com/watch?v=mmjMqu2w4p0
- http://go.eset.com/us/resources/white-papers/Rodionov-Matrosov.pdf
- http://www.nvlabs.in/uploads/projects/vbootkit2/vbootkit2.0-AttackingWindows7viaBootSectors.odp

Suggested tools & reading

- Hex editors
  - HT (http://hte.sourceforge.net/)
- Sysinternals tools (www.sysinternals.com)
  - Process Explorer
  - Autoruns
  - Process Monitor
- The Art of Computer Virus Research and Defense
  - Chapter 3: Malicious Code Environments, from 3.1 through 3.
  - Chapter 12: Memory Scanning and Disinfection
- Microsoft Windows Internals (M. Russinovich & D. Solomon)
  - New Vista edition out soon



F-Secure

Protecting the irreplaceable

F-Secure