
# Windows Rootkits

**Chris Ries**
**Security Research Engineer**
**VigilantMinds Inc.**

# Introduction

- What I do
  - Research new vulnerabilities, malware, and other security threats
  - Create defensive measures
  - Evaluate security software
- What this talk is about
  - Windows rootkits
  - How they are used
  - How they work
  - Defensive measures

# What is a Rootkit?

- **“A rootkit is a set of programs and code that allows a permanent or consistent, undetectable presence on a computer”**
- **Goals:**
  - **Hide malicious resources**
    - **Processes, files, registry keys, open ports, etc.**
  - **Provide hidden backdoor access**

# Brief History

- Early rootkits targeted UNIX OSes
  - "Kits" to attain and maintain "root" access to machines
  - Replaced login, ls, ps, netstat, etc. to give an attacker hidden access
  - Eventually moved towards kernel
- Windows popularity brought Windows rootkits

# Why So Popular?

- Worms, trojans, malware are utilizing rootkits
  - Presence becomes hidden
  - Machines stay infected longer -> can send spam and steal info longer -> more money for attacker
- Some commercial software adopts rootkit technology
  - Sony DRM software

# How Rootkits Are Used

# Stages of An Attack

1. Vulnerability in a system is discovered
2. Vulnerability is exploited to gain access to the system
3. Attacker gains a foothold on the system by escalating privileges, installing backdoor, etc.
4. Attacker utilizes system access to steal information, launch other attacks, etc.
5. Compromise is discovered, and incident response is executed

# Where Rootkits Fit In

- Attacker uses a rootkit to gain a stronger foothold on the system
- Rootkits aim to prevent or delay discovery by hiding an attacker's resources on a compromised system
- Rootkit can also re-enforce an attacker's system access by providing a stealth backdoor

# Attack Scenario - Haxdoor

- Employee visits a malicious website that exploits an IE 0day
- Site installs malware that includes a rootkit
- While on the system, the malware steals usernames and passwords, periodically emailing them to an attacker
- Malware also installs a backdoor, which the attacker uses to steal confidential documents
- Malware goes undiscovered for a long period of time, allowing the attacker to steal large amounts of information

# Attack Scenario - Insider

- IT worker discovers that he will be fired
- He installs a kernel-level rootkit on the web server
- After he is fired, the system is audited for backdoors or security holes, but none are found (hidden by rootkit)
- Attacker uses access to the web server to steal information, take down site, etc., causing financial loss for his former employer

# How Rootkits Work

# How They Work

- To access files, registry, etc. on system…
  - User interacts with GUI or CLI
  - Application developer interacts with Win32 API
  - Most rootkits are implanted at a much lower level, deep within the operating system

# User-mode vs. Kernel-mode

- **Applications run with *user-mode* privileges**
  - **Cannot access operating system's memory**
  - **Limited access to other process's memory**
  - **Limited access to instruction set**
- **This provides**
  - **Stability**
  - **Security**

# User-mode vs. Kernel-mode

- **Most operating system code and drivers run with *kernel-mode* privileges**
  - **Access to all memory**
  - **Access to all instructions**
  - **Can directly access system's resources**
- **User-mode code usually accesses resources with the Win32 API**
- **Win32 API uses the Native API, which uses kernel-mode system services (system calls) to access resources**

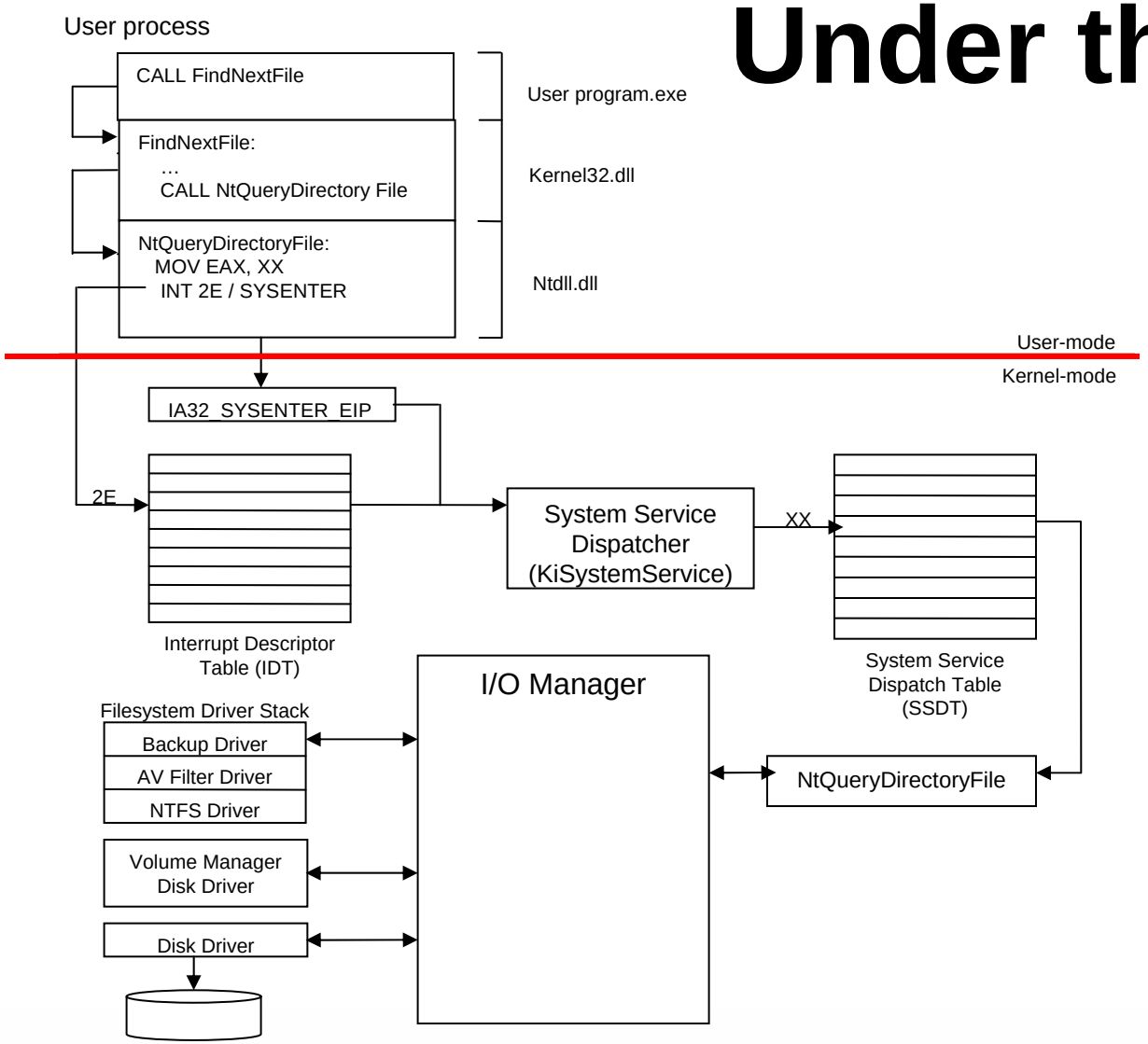
# User-mode vs. Kernel-mode

| User-mode | Kernel-mode |
|---|---|
| Limited memory access | Unlimited memory access |
| Limited instruction set | Full instruction set |
| Needs to access system resources through kernel | Can directly access system resources |

# How Rootkits Work

- **Example: Listing files in a directory**
  - **User ‘dir’, Explorer, etc.**
  - **Win32 Programmer: FindFirstFile() and FindNextFile()**
  - **Under the hood…**

# Under the Hood



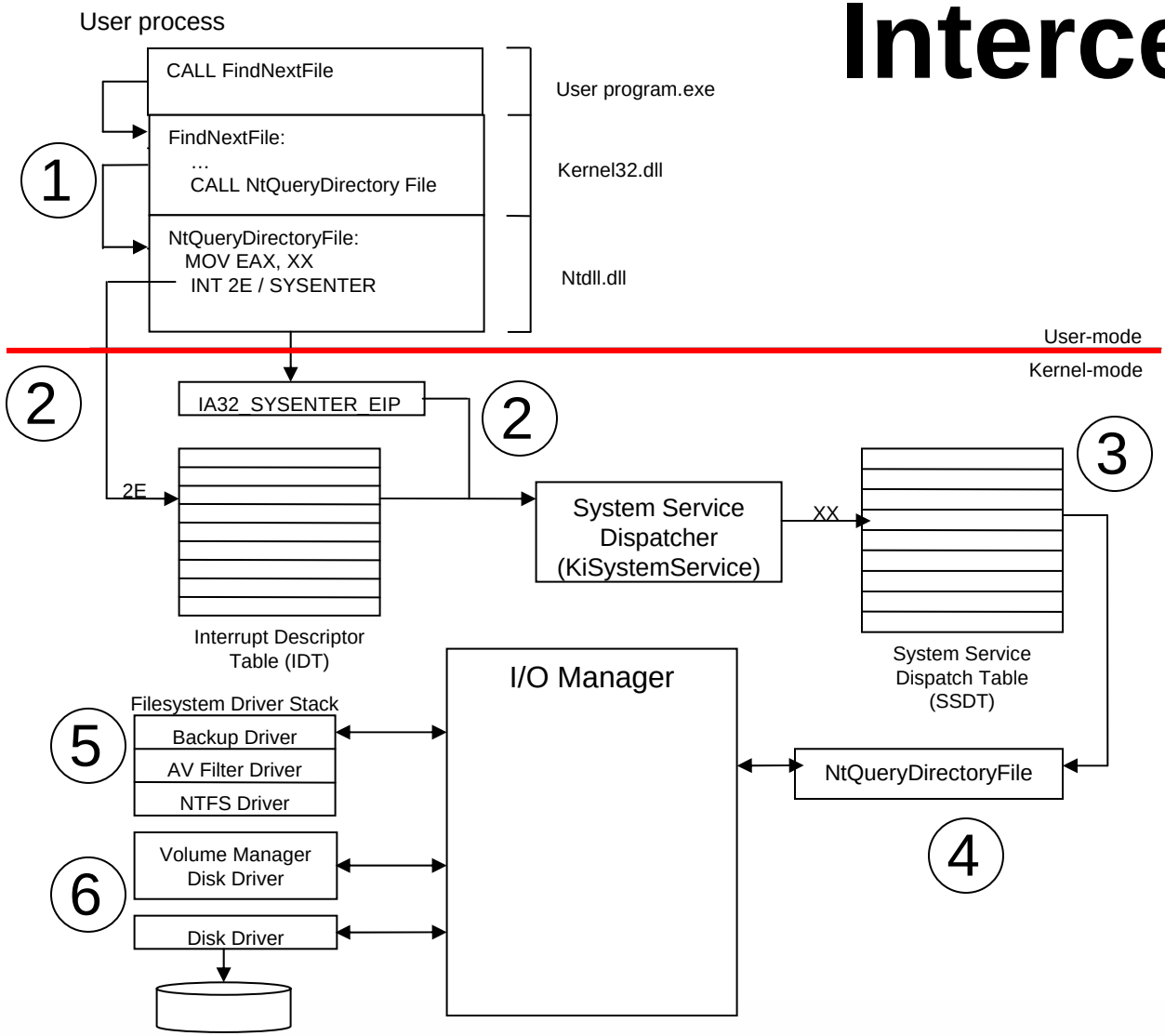

## Why So Complex?

- Convenience
- Flexibility
- Portability

# Interception

- Rootkits can intercept requests to:
  - Block request
  - Alter request
  - Fabricate results
  - Alter results
- Interception is also useful for stealing information

# Interception




1. User-mode hooks
2. IDT / SYSENTER hooks
3. SSDT hooks
4. Kernel code patching
5. Layered driver
6. Driver hooks

# User-Mode Interception

- Pro: Easier to develop code
- Con: Easier to detect

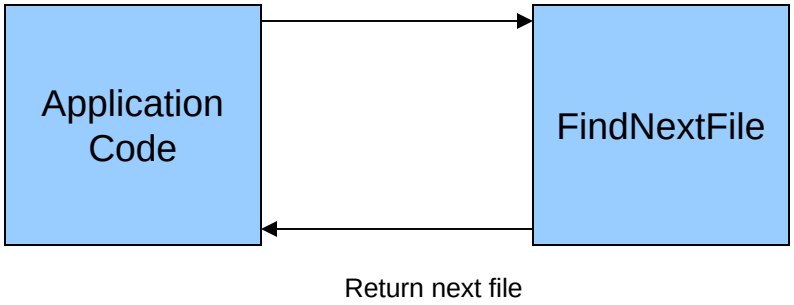
- Methods:
  - Import Address Table (IAT) Hooks
  - Export Address Table (EAT) Hooks
  - Inline Hooks
- Examples: Vanquish, Haxdoor, Hacker Defender (some are hybrids)

# Inline hooking

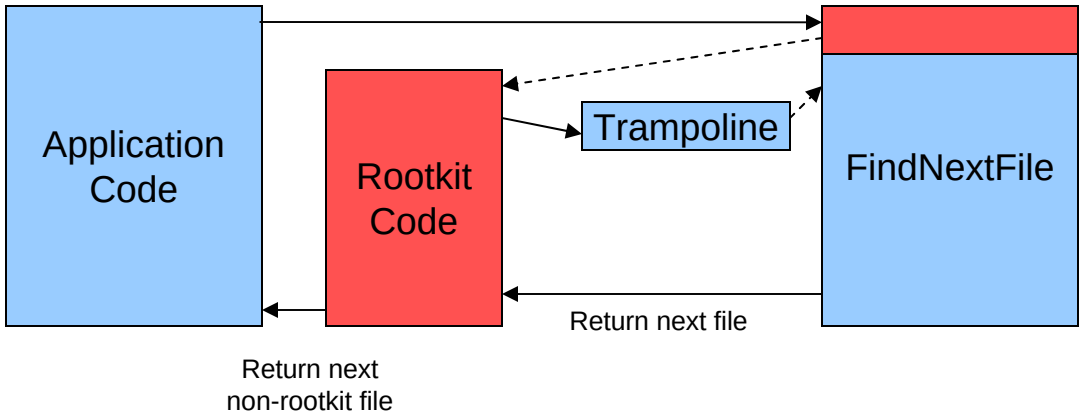
- Overwrite first few bytes of target function with a jump to rootkit code
- Create “trampoline” function that first executes overwritten bytes from original function, then jumps back to original function
- When function is called, rootkit code executes
- Rootkit code calls trampoline, which executes original function

# Inline hooking

**Before:**




**After:**



# Installation – User-mode

- In order to hook functions in a given process, rootkit can inject code into process
- Win32 API provides functions for this
  - WriteProcessMemory()
  - CreateRemoteThread() or SetThreadContext()
- Injected code can insert jumps and create trampoline functions

# Kernel-mode Interception

- **Pros: Can be difficult to detect, many places to intercept**
- **Cons: Complex to implement, can make system unstable**

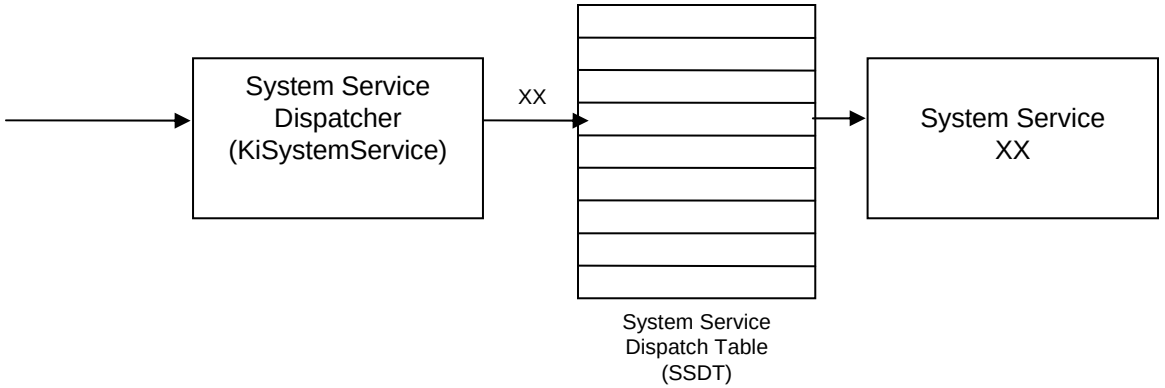
- **Methods:**
  - **IDT, SYSENTER, SSDT, driver hooks**
  - **Layered drivers**
  - **Code patching**

# SSDT Hooking

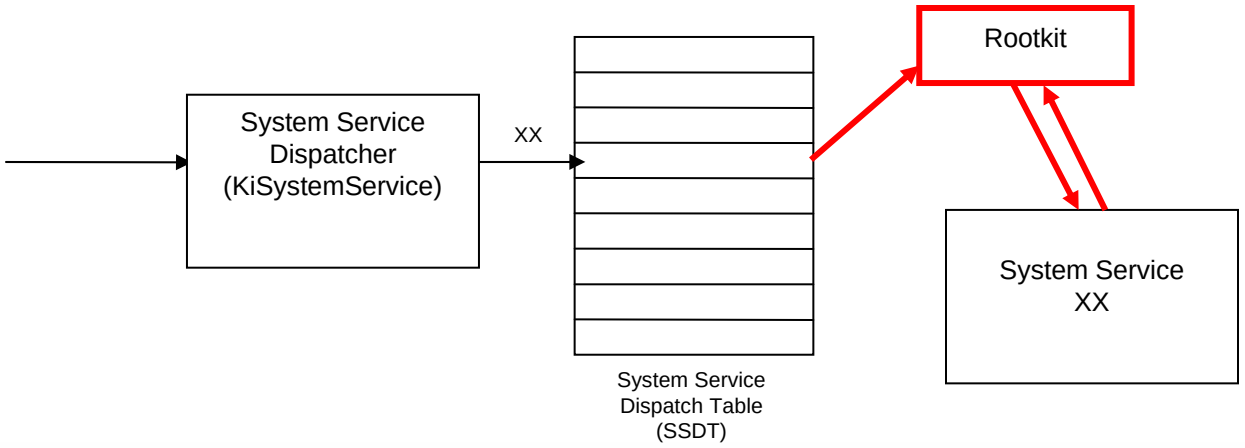
- System services (system calls) used to access/manipulate:
  - Filesystem
  - Registry
  - Processes and Threads
  - Memory
- System Service Dispatch Table (SSDT) has an entry for each system service that contains the service's address

# Example: SSDT Hooking

**Before:**




**After:**

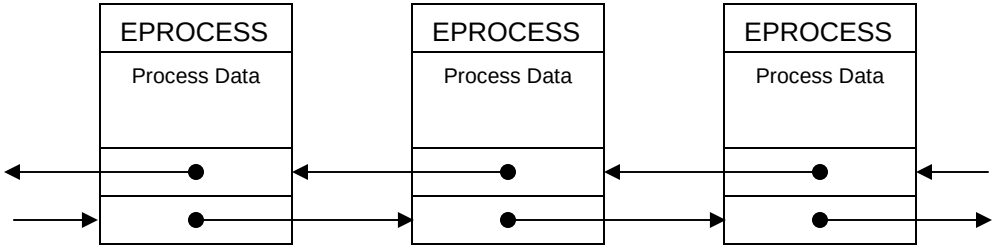


# Direct Kernel Object Manipulation

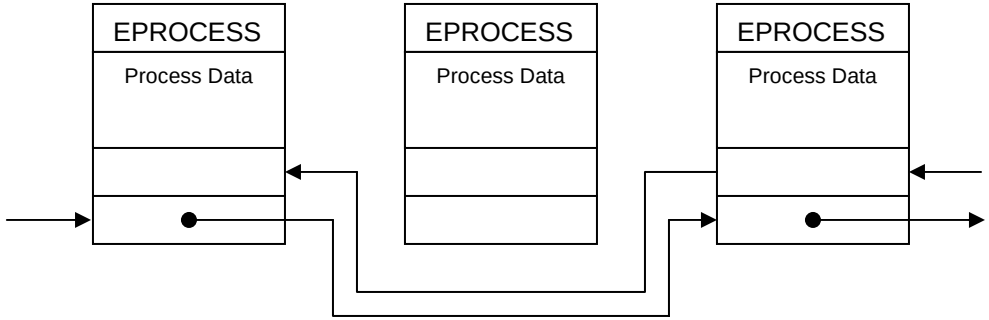
- Kernel uses data objects to keep track of almost everything
  - Processes, loaded drivers, etc.
- Instead of using code to hide resources, manipulate objects
- Take advantage of redundancy
- Examples: FU, FUTO

# DKOM with Processes

**Before:**




**After:**



# Installation – Kernel-Mode

- How an Attacker can inject code into the kernel from an Admin account
  - Load a driver
  - Manipulate \Device\PhysicalMemory
  - Exploit kernel vulnerability

# Stealth Backdoors

- Not usually focused on by rootkit authors
- Possibilities
  - Steganography – Hide data in TCP packet Fields
  - Hide in “normal” traffic
    - HTTP, DNS
    - Covert
    - Can bypass network filtering


# Defending Against Rootkits

# Defensive Measures

- **Reactive**
  - **Detect Rootkit AFTER it has been installed**
- **Proactive**
  - **Prevent rootkit from being installed**
  - **Prevent compromise in the first place**

# Detection

- **Difficult, because…**
  - **Rootkit's goal is to hide**
  - **Usually cannot trust operating system**

# Integrity-based Detection

- Use checksums to monitor system files for changes
- Ex.  Tripwire
- Successful against early rootkits that modified system utilities
- Most modern rootkits target memory, so not as successful today

# Signature-based Detection

- Develop “signatures” for known rootkits
  - Sequence of bytes
- Scan files / memory for signatures
- Cannot detect unknown rootkits

# Hook Detection

- Most hooks can be detected using heuristics
  - Jumps at the start of a function
  - Table entries in memory vs. in binary file do not match
- Examples: VICE, SDTRestore

# Hook Detection

- False positives
  - Some functions appear to be hooked
  - Some legitimate software uses hooks
    - Personal Firewalls
    - Host Intrusion Prevention Systems

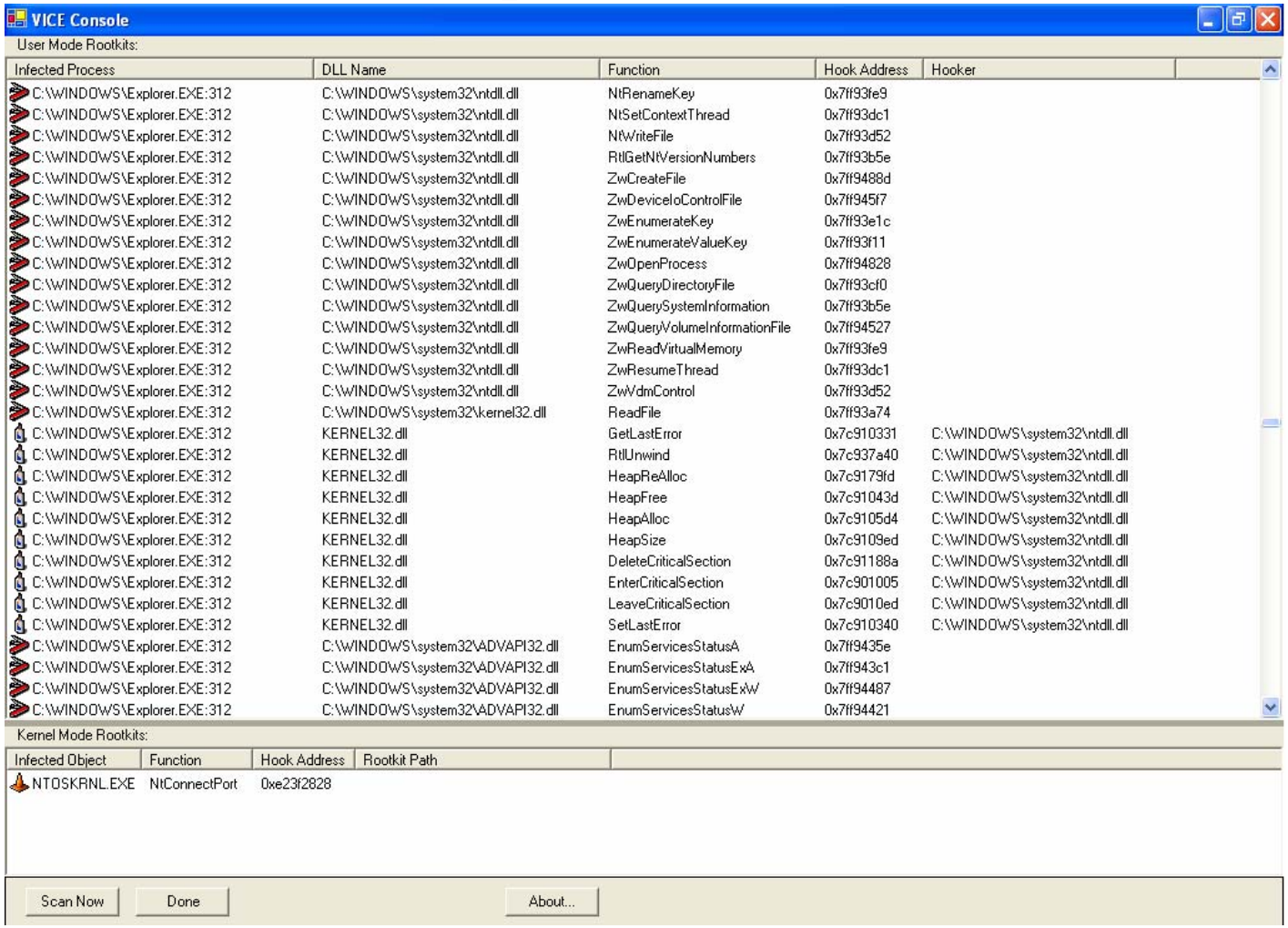
VIGILANTMINDS

# Example: VICE

Hacker Defender

False Positives

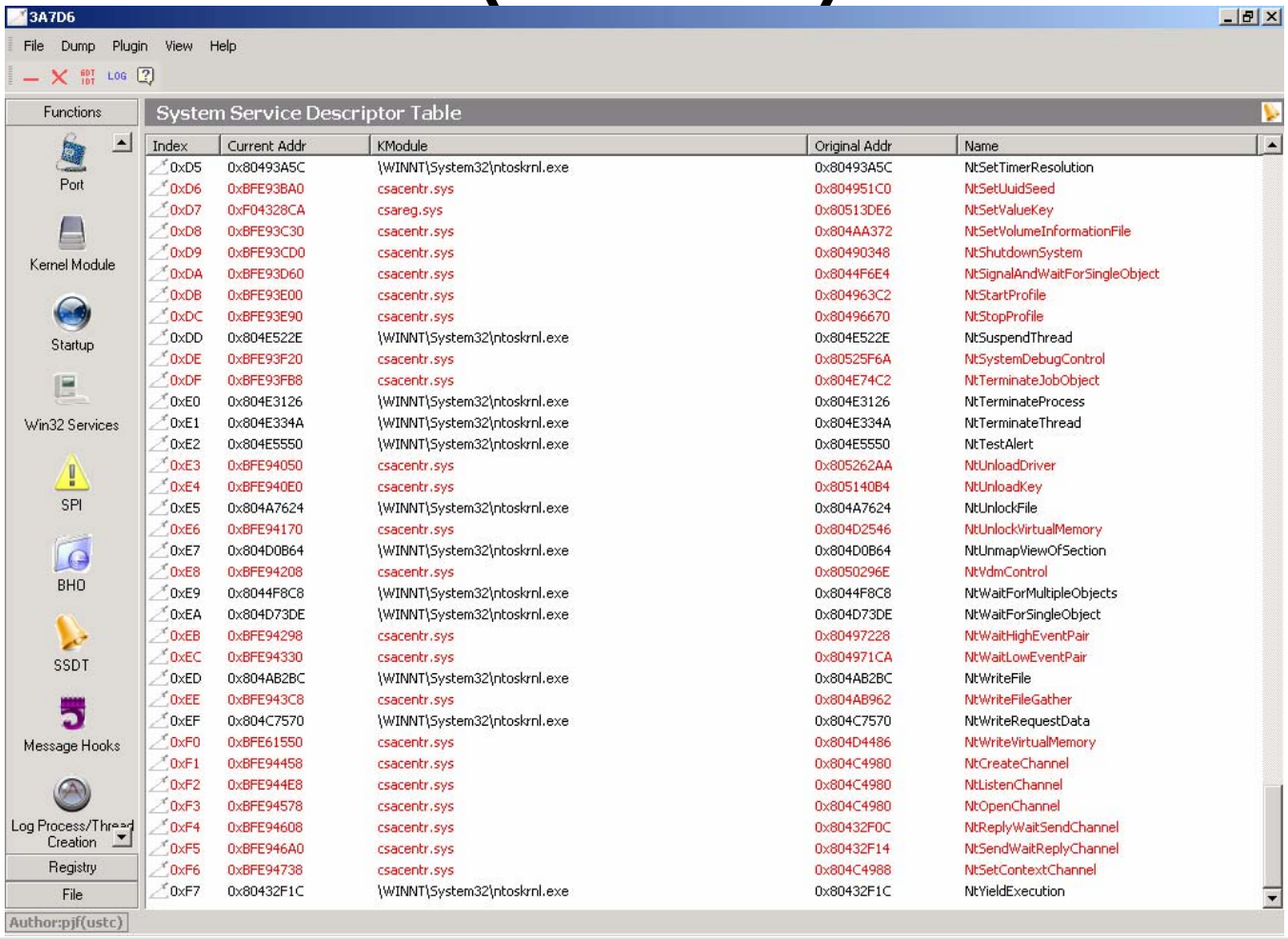
Hacker Defender



VIGILANTMINDS

# Example: False Positives

## (IceSword)

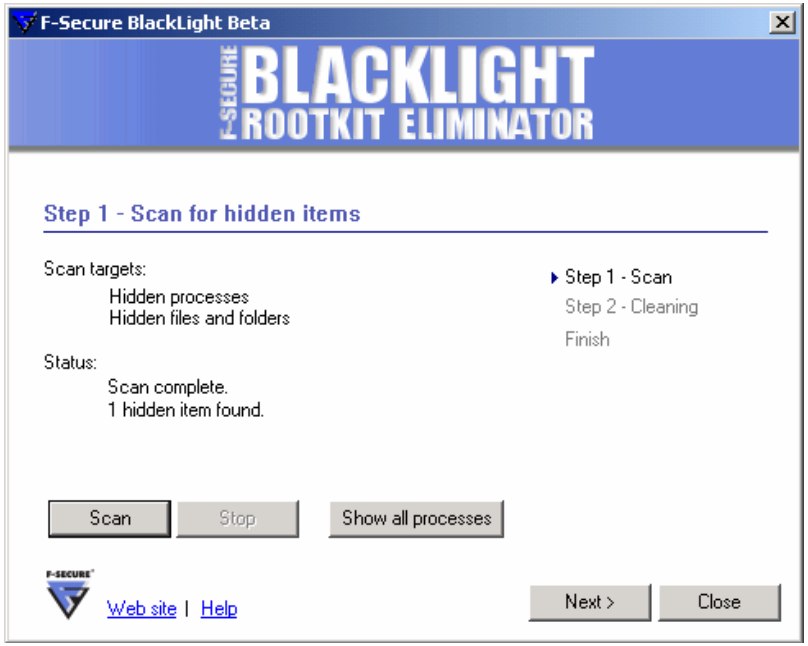


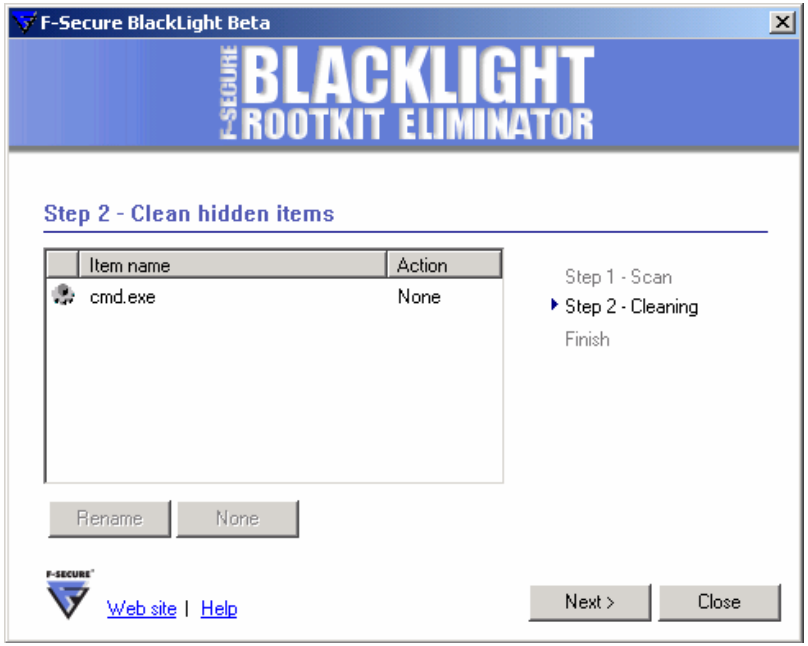

| Index | Current Addr | KModule | Original Addr | Name |
|---|---|---|---|---|
| 0xD5 | 0x80493A5C | \WINNT\System32\ntoskrnl.exe | 0x80493A5C | NtSetTimerResolution |
| 0xD6 | 0xBFE93BA0 | csacentr.sys | 0x804951C0 | NtSetUuidSeed |
| 0xD7 | 0xF04328CA | csareg.sys | 0x80513DE6 | NtSetValueKey |
| 0xD8 | 0xBFE93C30 | csacentr.sys | 0x804AA372 | NtSetVolumeInformationFile |
| 0xD9 | 0xBFE93CD0 | csacentr.sys | 0x80490348 | NtShutdownSystem |
| 0xDA | 0xBFE93D60 | csacentr.sys | 0x8044F6E4 | NtSignalAndWaitForSingleObject |
| 0xDB | 0xBFE93E00 | csacentr.sys | 0x804963C2 | NtStartProfile |
| 0xDC | 0xBFE93E90 | csacentr.sys | 0x80496670 | NtStopProfile |
| 0xDD | 0x804E522E | \WINNT\System32\ntoskrnl.exe | 0x804E522E | NtSuspendThread |
| 0xDE | 0xBFE93F20 | csacentr.sys | 0x80525F6A | NtSystemDebugControl |
| 0xDF | 0xBFE93FB8 | csacentr.sys | 0x804E74C2 | NtTerminateJobObject |
| 0xE0 | 0x804E3126 | \WINNT\System32\ntoskrnl.exe | 0x804E3126 | NtTerminateProcess |
| 0xE1 | 0x804E334A | \WINNT\System32\ntoskrnl.exe | 0x804E334A | NtTerminateThread |
| 0xE2 | 0x804E5550 | \WINNT\System32\ntoskrnl.exe | 0x804E5550 | NtTestAlert |
| 0xE3 | 0xBFE94050 | csacentr.sys | 0x805262AA | NtUnloadDriver |
| 0xE4 | 0xBFE940E0 | csacentr.sys | 0x805140B4 | NtUnloadKey |
| 0xE5 | 0x804A7624 | \WINNT\System32\ntoskrnl.exe | 0x804A7624 | NtUnlockFile |
| 0xE6 | 0xBFE94170 | csacentr.sys | 0x804D2546 | NtUnlockVirtualMemory |
| 0xE7 | 0x804D0B64 | \WINNT\System32\ntoskrnl.exe | 0x804D0B64 | NtUnmapViewOfSection |
| 0xE8 | 0xBFE94208 | csacentr.sys | 0x8050296E | NtVdmControl |
| 0xE9 | 0x8044F8C8 | \WINNT\System32\ntoskrnl.exe | 0x8044F8C8 | NtWaitForMultipleObjects |
| 0xEA | 0x804D73DE | \WINNT\System32\ntoskrnl.exe | 0x804D73DE | NtWaitForSingleObject |
| 0xEB | 0xBFE94298 | csacentr.sys | 0x80497228 | NtWaitHighEventPair |
| 0xEC | 0xBFE94330 | csacentr.sys | 0x804971CA | NtWaitLowEventPair |
| 0xED | 0x804AB2BC | \WINNT\System32\ntoskrnl.exe | 0x804AB2BC | NtWriteFile |
| 0xEE | 0xBFE943C8 | csacentr.sys | 0x804AB962 | NtWriteFileGather |
| 0xEF | 0x804C7570 | \WINNT\System32\ntoskrnl.exe | 0x804C7570 | NtWriteRequestData |
| 0xF0 | 0xBFE61550 | csacentr.sys | 0x804D4486 | NtWriteVirtualMemory |
| 0xF1 | 0xBFE94458 | csacentr.sys | 0x804C4980 | NtCreateChannel |
| 0xF2 | 0xBFE944E8 | csacentr.sys | 0x804C4980 | NtListenChannel |
| 0xF3 | 0xBFE94578 | csacentr.sys | 0x804C4980 | NtOpenChannel |
| 0xF4 | 0xBFE94608 | csacentr.sys | 0x80432F0C | NtReplyWaitSendChannel |
| 0xF5 | 0xBFE946A0 | csacentr.sys | 0x80432F14 | NtSendWaitReplyChannel |
| 0xF6 | 0xBFE94738 | csacentr.sys | 0x804C4988 | NtSetContextChannel |
| 0xF7 | 0x80432F1C | \WINNT\System32\ntoskrnl.exe | 0x80432F1C | NtYieldExecution |

# Cross-view Detection

- Try to look at same data using multiple methods, look for discrepancies in different views
- Two Approaches
  - Look at multiple places when data is redundantly stored
  - Look at same place from high level and low level
- Examples:  Rootkit Revealer, F-Secure BlackLight, MS Strider GhostBuster

# Example: BlackLight



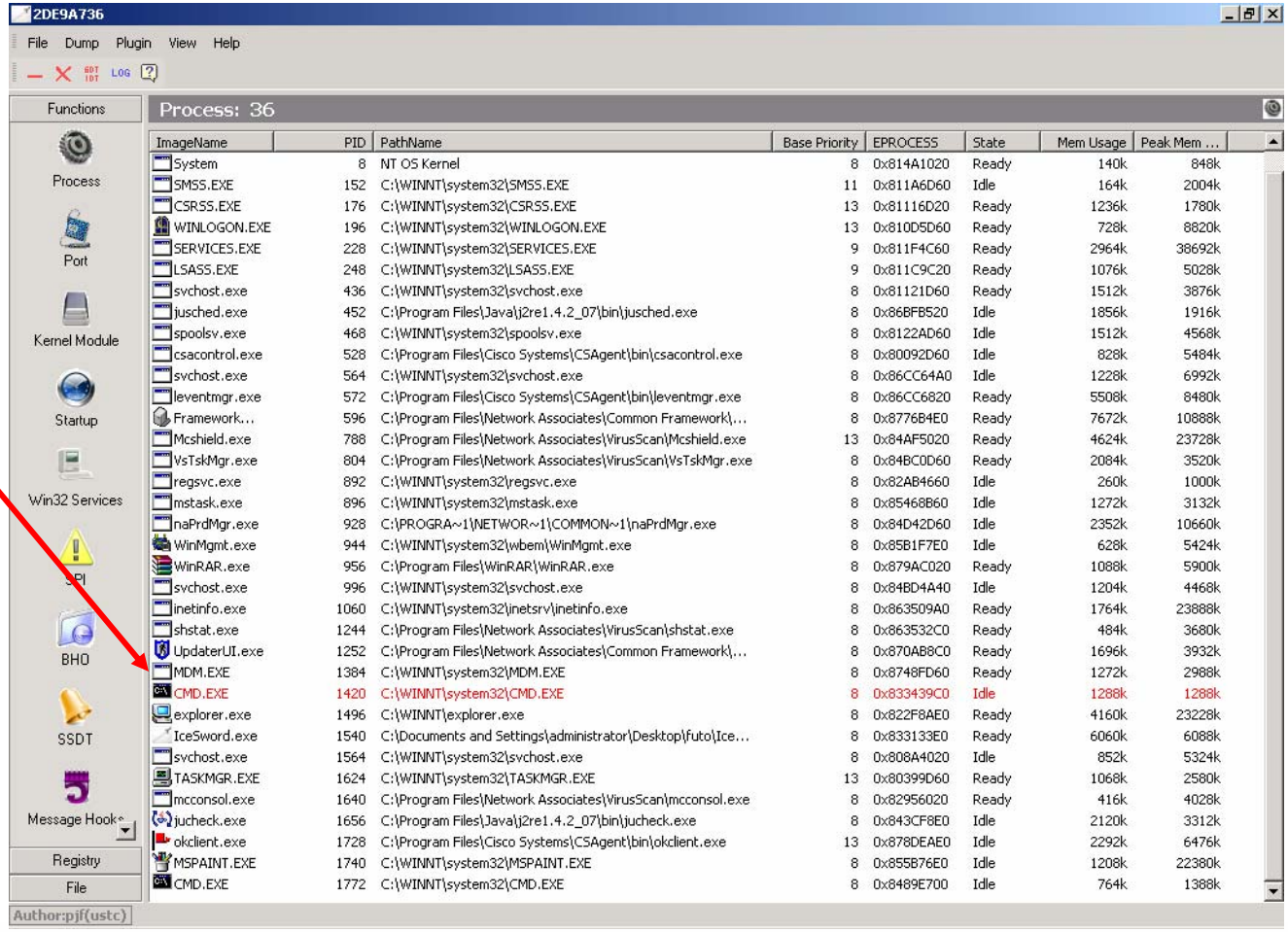

CMD.exe process hidden by FU rootkit

# Combination Tools

- Each method has strengths and weaknesses, so combine methods
- Examples:
  - IceSword
  - RAIDE
  - KProcCheck

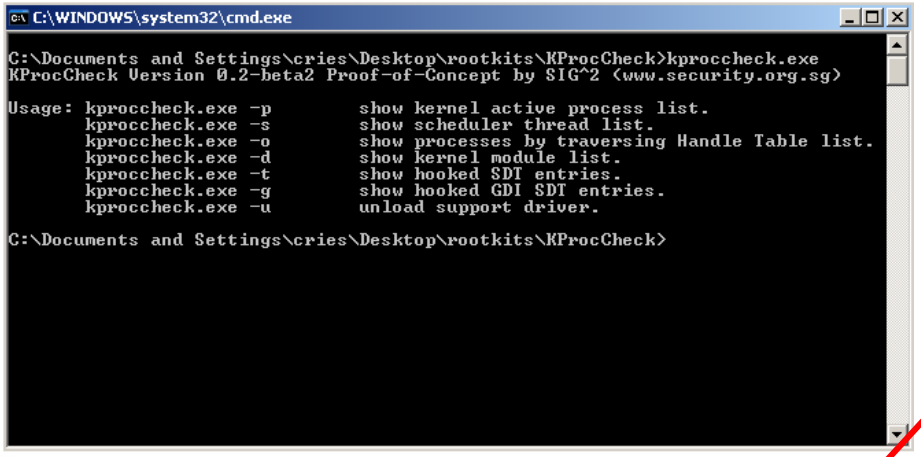
# Example: IceSword

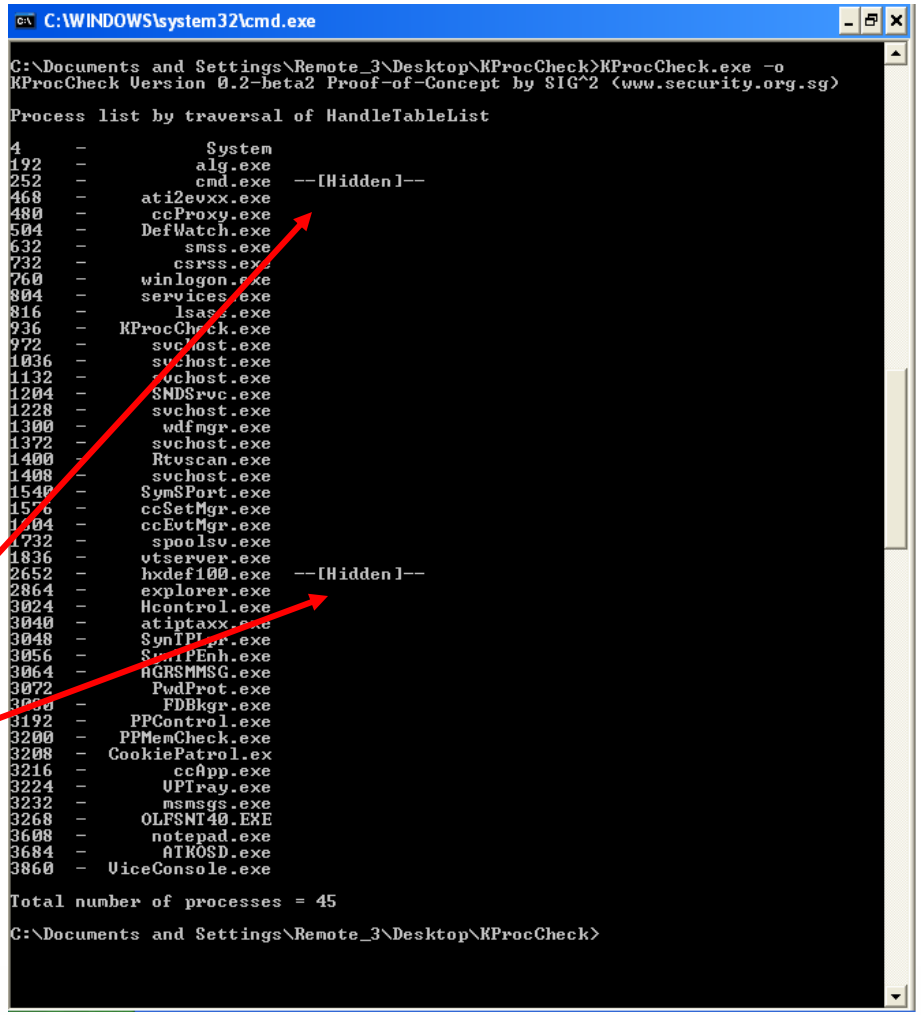
**CMD.exe process hidden by FU rootkit**



VIGILANTMINDS

# Example: KProcCheck




CMD.exe and hxdef100.exe processes hidden by Hacker Defender rootkit

# Removal

- Best solution is to rebuild system
- Clean the infection
  - Remember, OS cannot be trusted!
  - Either disable rootkit or boot with clean CD, and remove rootkit's resources
  - Need backups or baselines to verify integrity of system files and data

# Anti-detection

- Rootkit authors respond to detection tools:
  - Exploit weaknesses in tool to evade detection – FUTO
  - Use signatures to detect detectors – Hacker Defender Gold commercial rootkit
    - When detector is detected, either disable rootkit or patch detector

# DEMO (After talk)

- FUTO vs. BlackLight
- FUTO vs. KProcCheck

# Prevention

- **Prevent compromise in the first place!**
- **Good security practices**
  - **System hardening**
  - **Patch Management**
  - **Up to date Anti-virus**
  - **Least Privileges**
  - **Periodic auditing of critical systems**

# Stages of An Attack

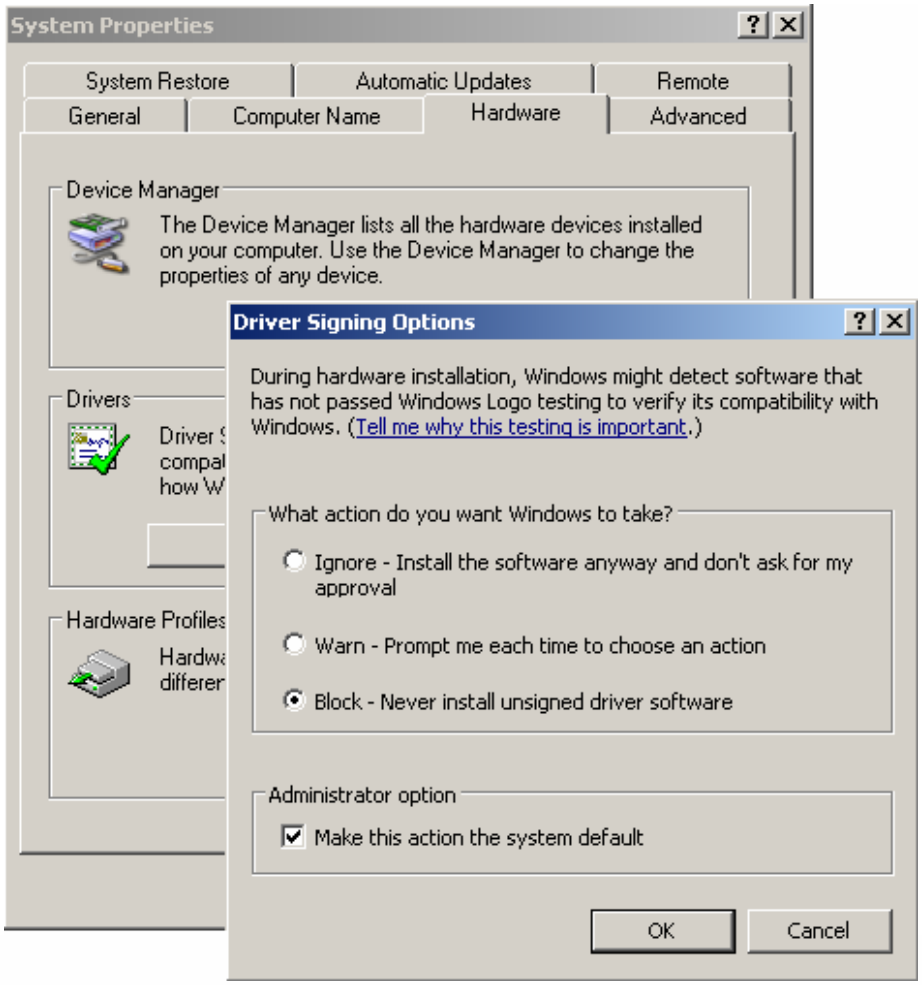
1. Vulnerability in a system is discovered <- IDS
2. Vulnerability is exploited to gain access to the system <- Patch Management
3. Attacker gains a foothold on the system by escalating privileges, installing backdoor, etc. <- Host-based security software, least privilege
4. Attacker utilizes system access to steal information, launch other attacks, etc. <- filtering, segmentation
5. Compromise is discovered, and incident response is executed <- Periodic auditing

# Prevention

- **Rootkit-specific**
  - **Disable user-mode access to physical memory**
  - **Do not allow new drivers to be loaded, or only allow digitally signed drivers**
  - **Monitor interactions between processes**

VIGILANTMINDS

# Digitally Signed Drivers

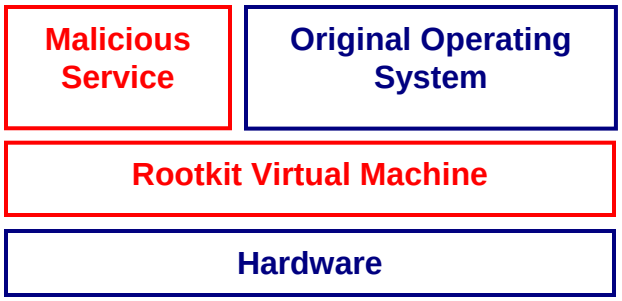


# Host Intrusion Prevention Systems

- Prevent new drivers from being loaded
- Protect Physical Memory
- Detect and prevent buffer overflow exploits
- Restrict interactions between processes such as process injection

# Future of Rootkits

- **Subvert OS at boot**
  - **Like boot sector virus**
  - **eEye BootRoot**
  - **SubVirt**
- **Hardware / Firmware**
  - **NGS ACPI BIOS Rootkit**
  - **NIC Card firmware**
- **Hardware-based Detection**
  - **Copilot**




SubVirt

VIGILANTMINDS

# Thank You!

## Questions? Comments?

Contact Info:
412-661-5700 x258
Chris.Ries@vigilantminds.com