

ROOTKITS THEN AND NOW

Presenter – txs

www.donkeyonawaffle.org (Donkey on a Waffle)

Who The HELL Is This GUY?!



- txs (aka Tyler)
 - Security consultant for a security company
 - I consult to fortune 500 companies, government, educational facilities
 - Member of 0x90.org, Ghetto Hackers, Atarininja
 - Lead researcher at Donkey On A Waffle (www.donkeyonawaffle.org)
 - Spoke at H2K2, CarolinaCon [1,2,4], and many other major conferences
 - Lazy bastard who makes his brother do all the grunt work and system administration for his sites

Overview



- Definition
- History And Evolution
- Classes of Rootkits
 - ▣ Application Level
 - ▣ Library Level
 - ▣ Kernel Level
 - ▣ Firmware Level
- Detection Techniques
- Forensic Implications
 - ▣ Live State Evidence Acquisition
 - ▣ Rootkit Data as Source of Evidence
- Current and Future Research
- Questions and Discussion

Definition

What is a Rootkit?



Definition



Wikipedia says:

*“A **rootkit** is a program (or combination of several programs) designed to take fundamental control (in Unix terms “root” access, in Windows terms “Administrator” access) of a computer system, without authorization by the system's owners and legitimate managers. Access to the hardware (ie, the reset switch) is rarely required as a rootkit is intended to seize control of the operating system running on the hardware. Typically, rootkits act to obscure their presence on the system through subversion or evasion of standard operating system security mechanisms. Often, they are also Trojans as well, thus fooling users into believing they are safe to run on their systems. Techniques used to accomplish this can include concealing running processes from monitoring programs, or hiding files or system data from the operating system.”*

Definition



h-spot.net says:

“a set of software utilities designed to run on specifically Unix/Linux type computers and assume control of them as the root user without the knowledge or permission of the owner. More recently, the term has been broadened to encompass kits of software utilities that are able to hide files, folders, programs or processes on any type of computer and allow them to evade detection by the computer operator.”

5starsupport.com says:

“A set of programs used by hackers to gain access to information contained in your operating system and can even mask its presence. The program can also be used to access computers within a network. Usually, the rootkit is written for malicious purposes.”

Definition



CompuKiss Techtionary says:

“A rootkit is software that runs at the lowest level of the computer. It infiltrates the kernel of the computer. A rootkit is a technique that is often used by hackers and virus creators to hide the files they create. It has also been used by manufacturers to hide digital right management software, much to the chagrin of the average computer user.”

Texas State Library and Archives says:

“A hacker security tool that captures passwords and message traffic to and from a computer. A collection of tools that allows a hacker to provide a backdoor into a system, collect information on other systems on the network, mask the fact that the system is compromised, and much more. Rootkit is a classic example of Trojan Horse software. Rootkit is available for a wide range of operating systems.

Definition



TXS SAYS (don't believe everything I say!):

“Rootkit technology, as defined in this presentation, includes any code that is implemented in an effort to hide the existence of the code itself, and to allow surreptitious execution and control of a target system.”

This presentation will focus on the hiding of code used to maintain long term compromise of a target system.

History and Evolution



- Stealth technologies have been in existence since at least the mid 1980s
 - ▣ First notable piece of “stealth” code was the Brain Virus
 - Boot Sector Virus for FAT
 - First virus in existence to include code created to hide the virus from detection
 - Hooks INT13 and intercepts attempts to read the infected boot sector presenting a modified boot sector instead.

History and Evolution



- Stealth moves into the UNIX world
 - ▣ Modified system binaries and “log cleaning” software began to surface
 - ▣ SUN Microsystems based machines are compromised and “kits” are detected that attackers have utilized to cover their tracks
 - ▣ Kit Goals – continuing root level remote access, privilege escalation, hiding potential evidence, going undetected by administrators

History and Evolution



- Packet capture and “sniffing” code added
 - ▣ Captured data as it traversed the network
 - ▣ Extended compromise to other targets
- Extended to libraries for quicker easier installation
- Mid 1990
 - ▣ Userland Rootkits are getting detected
 - Too easy to detect (hashing)
- 1997-1999
 - ▣ Relocation to kernel level occurs
 - ▣ Similar results, far more difficult to detect

History and Evolution



- Kernel level rootkits in the unix world based on LKM (Loadable Kernel Modules)
 - ▣ No longer do we have to rewrite individual binaries
 - ▣ Rewrite the kernel code affecting all binaries
- In the windows world this is achieved via hooking, device drivers and direct kernel object modification (DKOM)
- The very core of the OS can not be trusted!

History and Evolution



- Rootkit detection tools move to the kernel level to match the offensive
 - ▣ Cat and mouse game begins
- Early 2000s
 - ▣ Virtualized computing systems are created
 - ▣ A layer is created that is lower than the kernel (Virtual Machine Monitors)
 - ▣ If Rootkit loads under the kernel it may be possible to intercept calls from the kernel to the hardware itself.

History and Evolution



- Get lower
 - ▣ Firmware rootkits
- Today
 - ▣ Continued research into virtualized rootkits
 - ▣ Continued research into detection of virtualized rootkits

Classes of Rootkits



- Application Rootkits (Userland rootkits)
- Library Rootkits (Userland as well)
- Kernel Rootkits (Kernel space)
- Firmware Rootkits (Hardware?!)

Operating System Layers And Rootkit Models



Executable Program (Application Level)

System Libraries (Library Level)

Operating System Kernel (Kernel Level)

Optional Virtual Machine Monitor (Virtualized Rootkits)

Hardware (Firmware Level)

Application Layer Rootkits



- Also referred to as userland rootkits
- Recompiled binaries to replace system binaries and operate in a malicious way
- Modification of dll or exe files in windows world
- Typically Trojan binaries
- Binaries grouped together into “kits”, thus the term “Root Kit”

Library Layer Rootkits



- ❑ Patch, hook, or replace calls to system libraries
- ❑ Technically reside in userspace
- ❑ Affects a large number of system binaries while only modifying a small number of libraries
- ❑ Rootkit's library intercepts the call to the regular library, calls regular library, returns modified results.

Kernel Layer Rootkits



- Implemented by replacing or creating new code in the kernel
- Windows
 - ▣ Device driver code
 - ▣ Direct Kernel Modification (DKOM)
- UNIX
 - ▣ Loadable Kernel Module (LKM)

Virtualized Rootkits



- Lowest level rootkit in existence
- Inserted below the general operating system, one layer abstracted from the hardware
- Can trap, drop, created, modify, all requests to the hardware
- Two types
 - ▣ Software based
 - ▣ Hardware assisted
- Not limited by size (bigger is easier to see)

Software Based



- ❑ Uses a virtual machine monitor (VMM) to manage resources
- ❑ Intermediary between the hardware and the operating system
- ❑ Requires reboot to load under the OS
- ❑ Traps all requests to the hardware filtering responses
- ❑ Presents falsified and filtered results
- ❑ Nearly undetectable to tools installed even in the kernel of the OS

Hardware Assisted



- Also run at one abstraction layer below the operating system
- Loads itself under an already running operating system
- Forks or migrates the OS to a guest state
- Hardware itself must support virtual hosts
 - ▣ AMD-V
 - ▣ Intel VT-x

Firmware Layer Rootkits



- ❑ Implemented at the hardware level
- ❑ Bottom of the stack
- ❑ Peripheral devices, disk controllers, USB keys, processors, firmware memory, etc
- ❑ Firmware can be modified by administratively run code (Think flashing the BIOS!)
- ❑ Many pieces of firmware are run before the OS is loaded
- ❑ Removal is difficult

Detection Techniques



- General Detection
 - ▣ System monitoring tools
 - ▣ Diff based detection
 - Review multiple witnesses and different angles
 - Differences equals liars
 - ▣ Directly examine the underlying hardware
 - Can be done live (does not always require a reboot)
 - Use statically compiled tools
 - Use tools that bypass the kernel and directly access the hardware (sleuth kit)
 - ▣ Directly review kernel memory bypassing APIs

Application Layer Detection



- Use trusted binaries
 - ▣ Compiled offline
 - ▣ Statically linked
 - ▣ Use a CDROM and mount save tools
 - What if the mount system was compromised?!
- Live state analysis can never be fool-proof
- Gather cryptographic hashes
 - ▣ Compare against known good hashes
- Check configuration files for modifications
- Use a host based intrusion detection system (HIDS)
- Get to the kernel layer and check (get lower)

Library Layer Detection



- Very similar to those used in Application Layer
- Trusted binaries
 - ▣ This time targets libraries and not binaries
- Static compilation is mandatory
- Check configuration files
- Implement a strong HIDS
- Get to the kernel layer and check (get lower)

Kernel Layer Detection



- ❑ Significantly more difficult
- ❑ Safe binaries wont work, still relies on kernel
- ❑ Detect modification of the kernel tables
 - ▣ SSDT has an expected range of addresses
 - ▣ Addresses outside this range are suspect
 - ▣ Same method used for kernel hooking in Interrupt Descriptor Table (IDT), Import Address Table (IAT), and Drivers' I/O Request Packet (IRP) handler

Kernel Layer Detection



- Detecting DKOM
 - ▣ First to arrive has distinct advantage
 - ▣ Hope to use a section of kernel that is not subverted
 - ▣ Thread analysis
 - ▣ Heuristic memory searching
- Use a virtual machine monitor
 - ▣ Get to the virtual machine layer and check (get lower)
- Binary runtime analysis
 - ▣ Check that the binaries that are run do not directly modify the kernel

Firmware Layer Detection



- Theoretical and cutting edge
- Disassemble the ROMs
 - ▣ ACK! No thanks
- Signed firmware updates
- Physical jumper on writeable ROMs

Virtualized Layer Detection

- Install the detection engine below the Rootkit
 - ▣ Utilize secure hardware
 - ▣ Boot from a CDROM or other secure boot system
 - ▣ Create the VM FIRST below the Rootkit
- From above the Rootkit
 - ▣ Monitor and compare timing based functions
 - ▣ Detect the additional overhead
 - Can be hindered by temperature or power fluctuations
 - ▣ Execute “buggy” behavior directly in the processor

Virtualized Layer Detection

- Dual Core Techniques
 - ▣ Run a counter thread on one core
 - ▣ Execute a synchronized thread in the main core that will be intercepted by the Rootkit
 - ▣ Compare the results, skew indicates overhead
- Prevention
 - ▣ Load a preemptive hypervisor that prevents the loading of a malicious hypervisor
 - ▣ Intercept and deny/alert on any attempt at installation of a new hypervisor
 - ▣ Minimalist design as stealth is not a requirement, limited overhead

Forensic Implications



- Turn off the machine?
 - ▣ Results will be steadfast
 - ▣ Loses access to data stored in memory

- Live state analysis?
 - ▣ Results are less than certain
 - ▣ Rootkits will screw with live state analysis

Live State Analysis



- Determine the validity of the incident
- First responder activities
- When you don't want the subject to know
- Memory acquisition
 - ▣ Software based memory acquisition
 - Can be subverted
 - ▣ Hardware based memory acquisition
 - Physical hardware required
 - May also be subverted (in theory)

Live Disc Data Acquisition



- Can be subverted by rootkits
- Why bother?! Dead state analysis gives same results

Rookit As Evidence



- What is the goal of evidence gathering
 - ▣ Prove or disprove that an event has occurred on the system
 - ▣ The existence of a Rootkit can directly be used as evidence when attempting to prove events
 - ▣ Take care not to disturb or otherwise taint the evidence

Current and Future Research



- Creation and detection of virtualized Rootkits
- Creation and detection of hardware assisted virtual Rootkits
- Moving firmware Rootkits from the realm of theoretical to practical
- IMPLEMENT FIXES IN THE HARDWARE!
 - ▣ Create a secure computing platform for high assurance use

Questions and Discussion



Check out www.donkeyonawaffle.org for interesting posts, comments, and research

Leave comments – I promise I read them!