

# Windows NT Rootkits

Cosmin Stejerean

cstejerean@gmail.com



# What is a rootkit?

- ◆ A rootkit is a set of programs which \*PATCH\* and \*TROJAN\* existing execution paths within the system.

–

Greg Hoglund

- ◆ Involves itself in preexisting architecture so that it goes unnoticed



# What is a rootkit... (cont)

- ◆ Netbus, SubSeven and Back Orifice are not rootkits.
- ◆ A rootkit could however be used however to hide their presence.



# What can a rootkit do?

- ◆ Hide processes
- ◆ Hide files or file contents
- ◆ Hide registry keys and values
- ◆ Hide open ports
- ◆ Create and/or hide a backdoor
- ◆ Sniff network traffic or key presses
- ◆ Etc... there are many possibilities

# How does a rootkit work?

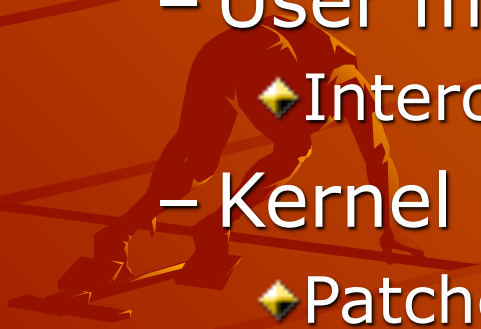
◆ Before we can answer that we need to identify the two types of rootkits

– User mode rootkits

◆ Intercept system calls of other processes

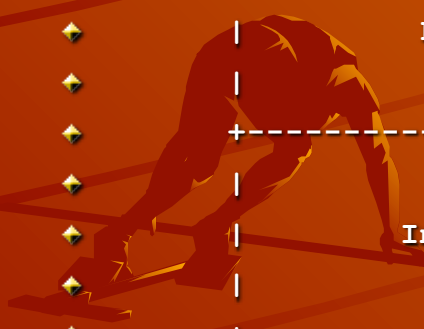
– Kernel mode rootkits

◆ Patches the kernel





# User mode Rootkits



◆	+-----+	- offset 0
◆	MS DOS Header ("MZ") and stub	
◆	+-----+	
◆	PE signature ("PE")	
◆	+-----+	
◆	.text	- module code
◆	Program Code	
◆		
◆	+-----+	
◆	.data	- initialized (global static) data
◆	Initialized Data	
◆		
◆	+-----+	
◆	.idata	- information for imported functions
◆	Import Table	and data
◆		
◆	+-----+	
◆	.edata	- information for exported functions
◆	Export Table	and data

# User mode Rootkits

- ◆ Must 'patch' all running processes as well as processes that will be created in the future
- ◆ Lists all running processes and intercepts the API calls that we want to modify
- ◆ To hook future processes we modify the NtResumeThread and LdrInitializeThunk

# User mode Rootkits ... (cont)

- ◆ These functions are called whenever a new processes is created
- ◆ We can then modify the memory space of the new processes before it begins to execute
- ◆ This way we can patch all processes on a system



# User mode Rootkits ... (cont)

- ◆ To hide files we patch NtQueryDirectoryFile and NtVdmControl to exclude any files we want to hide from being returned
- ◆ To hide processes we modify the return value of NtQuerySystemInformation

# User mode Rootkits ... (cont)

- ◆ We can use similar techniques to hide keys in the registry, hide open ports, etc
- ◆ How can a process hook an API Call?



# Hook API Calls

- ◆ Every running process has a copy of the kernel as well as any DLLs that were loaded when the program started
- ◆ This contains pointers to the actual DLL functions
- ◆ We can change these pointers to point to our own modified function

# Hook API Calls... (cont)

- ◆ We can either implement a new function or run the original function and modify the value it returns
- ◆ The key is to locate where in the process's memory the function pointer resides
- ◆ We can also modify the actual function by patching it's first 5 bytes with a jmp instruction

# Hook API Calls... (cont)

- ◆ We can also load arbitrary code or DLLs into running processes
- ◆ If the code runs with Administrator privileges we can access the memory of any process
- ◆ We can then write our code to it and call the `CreateRemoteThread` function to run our code



# Kernel Mode Rootkits

- ◆ Instead of modifying each new process we can modify the kernel
- ◆ We can modify the code of kernel functions to provide desired functionality
- ◆ We can modify kernel data structures to hide running processes

# Interesting things to modify in the kernel

- ◆ GINA.dll – the login screen (capture passwords)
- ◆ LSA (Local System Authority) – backdoors
- ◆ SST, IDT – add new system services to modify system functionality
- ◆ SRM – Change the way access control works

# Wow, that's pretty bad...

- ◆ So how can I detect these 'rootkits'?
- ◆ This is very hard to do and it depends on which rootkit you are trying to detect.
- ◆ Can use data structures and functions that you 'hope' were not altered

# Detecting rootkits... (cont)

- ◆ Detect hidden files by listing all files from the OS and then booting from a CD or another trusted OS and listing the files that way then compare the differences
- ◆ Access the system hive and compare the entries with entries in the registry

# Detecting rootkits... (cont)

- ◆ List running processes using low level system calls instead of API
- ◆ Use some commercial software that might do some of the above...





# Rootkit detectors

- ◆ F-Secure BlackLight
- ◆ Sysinternals RootkitRevealer
- ◆ UnHackMe
- ◆ RootKit Shark
- ◆ RegdatXP
- ◆ Malicious Software Removal Tool
- ◆ Flister
- ◆ Find Hidden Service
- ◆ Kernel SC
- ◆ Kernel PS
- ◆ Klister
- ◆ Process Magic
- ◆ KProcCheck
- ◆ TaskInfo

# Bad news...

- ◆ Hacker Defender Gold (paid version) cannot be detected by any of them
- ◆ Best solution is to analyze the system from a CD (Knoppix or Windows PE)



# For more info

- ◆ <http://www.rootkit.com>
- ◆ “A real NT rootkit” – Greg Hoglund
- ◆ “Hooking Windows API” – Holy Father
- ◆ “Invisibility on NT boxes” – Holy Father
- ◆ Rootkits: Subverting the Windows Kernel – Greg Hoglund and Jamie Butler