Tracking Windows Rootkit Design

October 19, 2011

Outline

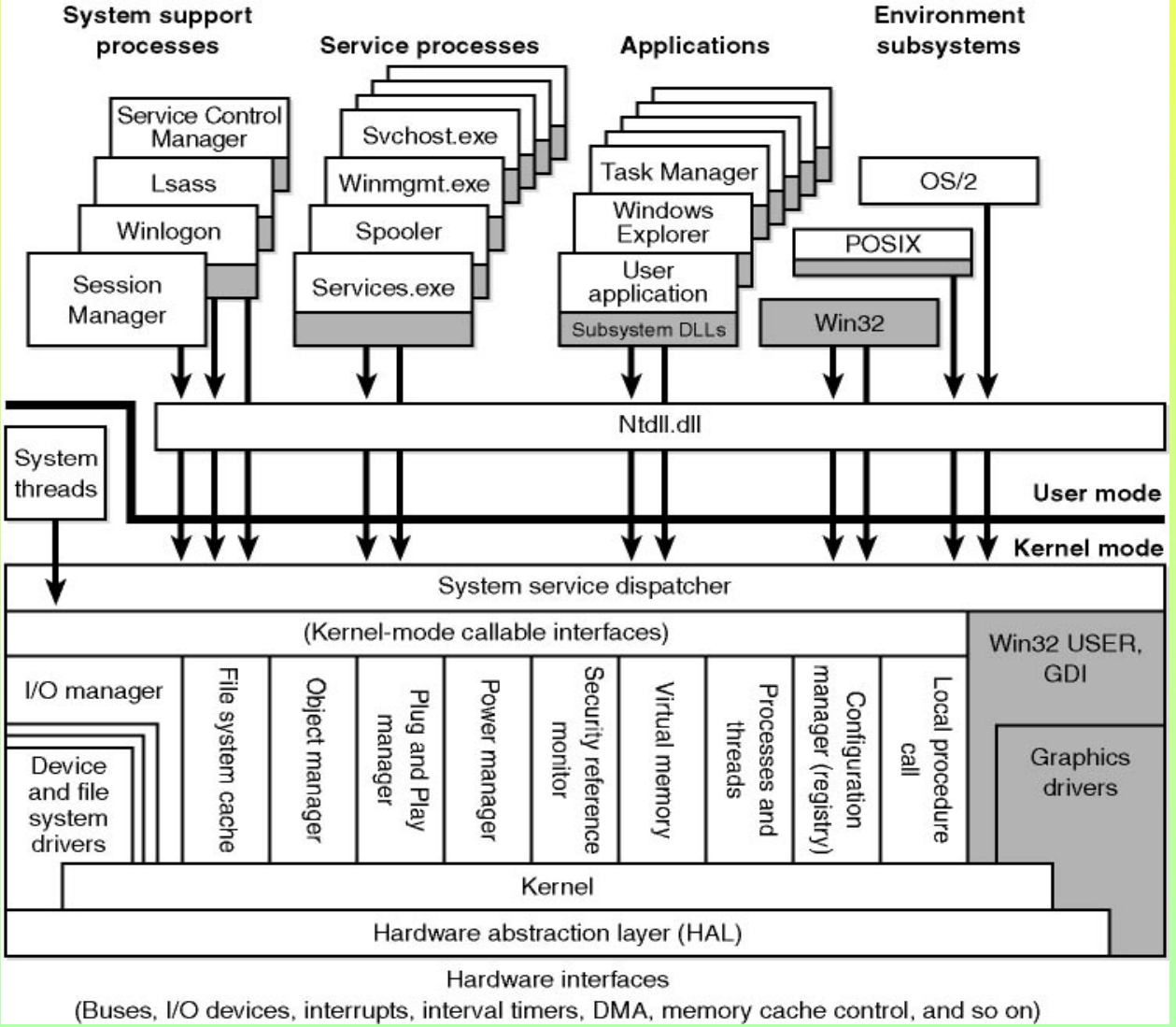
What is ROOTKIT?

Different Definitions

- Root $\Rightarrow$ Super User
  Kit $\Rightarrow$ Super User Abilities
- a set of tools used by an attacker to be hidden in the victim system after getting access to it.
- a type of malware that attempts to hide its presence on a victim system by compromising the Operating System.

Q. What are the main features of rootkits?

Windows OS Architecture



Let's simplify the architecture...

- **Q1. How an applications runs in Windows?**
- **Q2. What is the difference between system calls and library calls in Windows?**

Motivating Example

- An intruder gains access to your system.
- She puts a malicious executable file,"worm.exe", in your system and makes it run in every system startup.
- You can easily find "worm.exe" is running in your system by tracking the list of running processes in your task manager.
- She wants to hide "worm.exe" process in your task manager. **HOW?** $\rightarrow$ Rootkit development begins...

First Generation of Rootkits

- The intruder wonders about the way task manager shows the list of running processes:
  - Task manager calls a function,NtQuerySystemInformation, in NTDLL.DLL to get the list of running processes...
  - Found it!...I'll replace the real NTDLL.DLL file with the fake one in which I changed the function to not list any process with the name "worm.exe"!
- First Generation of rootkit changes the important files in the **disk**.
  Q.Is it really a good technique? How can we detect it?

First Generation of Rootkits (Simple Diagram)

Rootkit Generation Track Table

Generation	Modification Type	Location	Detectibility
1	direct(code)	Disk	Easy

Second Generation of Rootkits

- The intruder wonders about the way task manager shows the list of running processes:
  - Is it possible to manipulate NtQuerySystemInformation function without making any changes on NTDLL.DLL in the disk?...
  - Found it!...I will change NtQuerySystemInformation function in the loaded image of NTDLL.DLL in the **memory** $\rightarrow$ **Hooking Techniques!**
- Second Generation of rootkit changes the important functions in the **memory**.
  Q.Is it really a good technique? How can we detect it?

Second Generation of Rootkits(Simple Diagram)

Second Generation of Rootkits(User mode vs Kernel mode Hooking)

- The intruder wonders about the way task manager shows the list of running processes:
  - I only hooked NtQuerySystemInformation function in NTDLL.DLL loaded in the address space of the task manager. What if other applications(ie. anti-virus) invoke this function to check the list of running processes?
  - Found it! ... I put **rootkit.sys** driver in the victim computer. By hooking **ZWQuerySystemInformation** I'll change **all** user-mode applications which call NtQuerySystemInformation and **all** kernel-mode applications(drivers) that invoke ZWQuerySystemInformation.

Second Generation of Rootkits(User mode and Kernel mode Hooking Diagram)

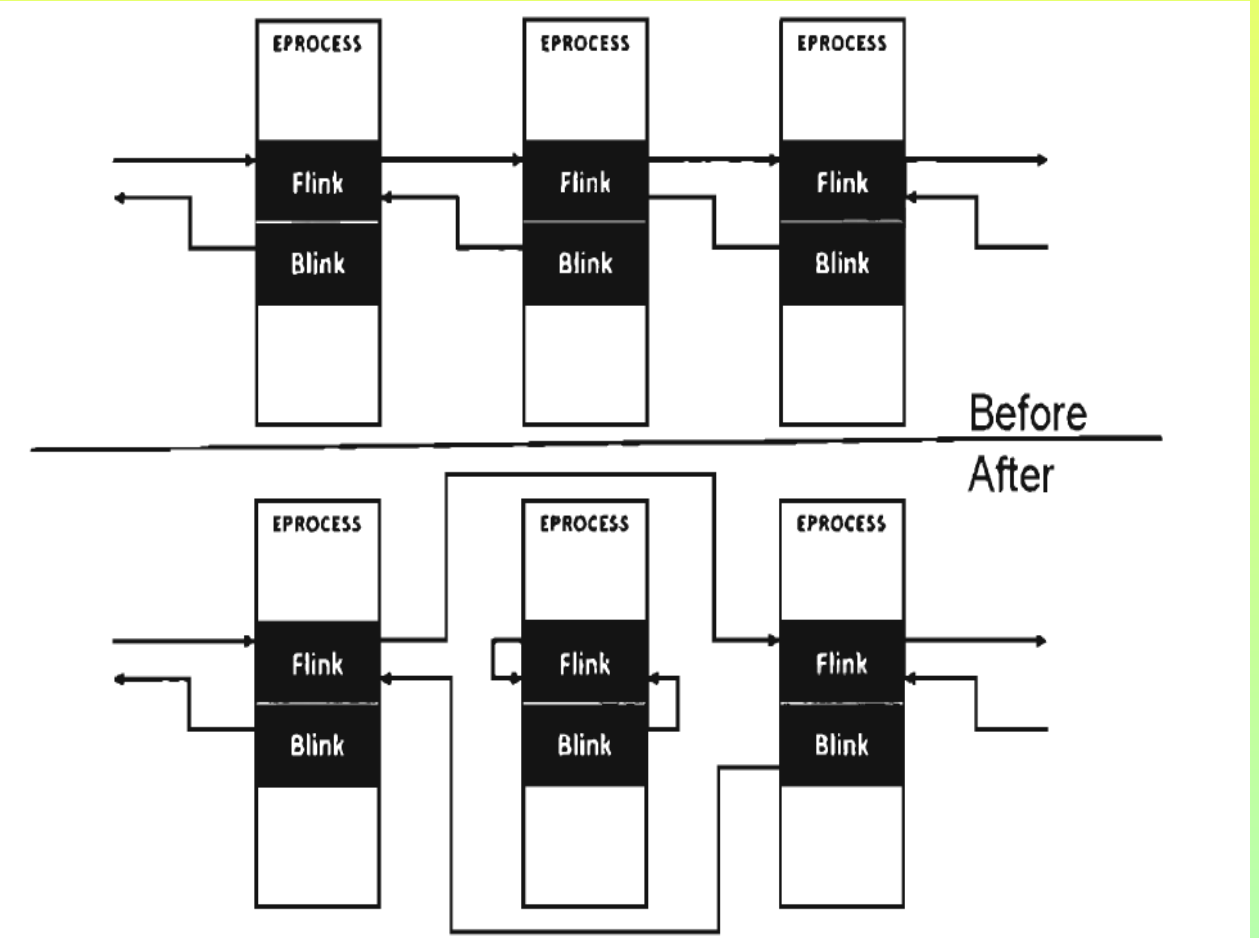
Rootkit Generation Track Table

Generation	Modification Type	Location	Detectibility
1	direct(code)	Disk	Easy
2	direct(code)	Memory	Moderate

Third Generation of Rootkits

- The intruder wonders about the way to bypass anti-hooker applications?
  - Is there any way to change ZWQuerySystemInformation operation without any hooks?
  - Found it!...Change the object not the subject!!! I'll change the data structures/objects that this function uses without making any modification in the function code.
- Third Generation of rootkit changes the important **objects** in the operating system instead of changing functions codes.
  Q.Is it really a good technique? How can we detect it?

Third Generation of Rootkits- Example: DKOM (Direct Kernel Object Manipulation)

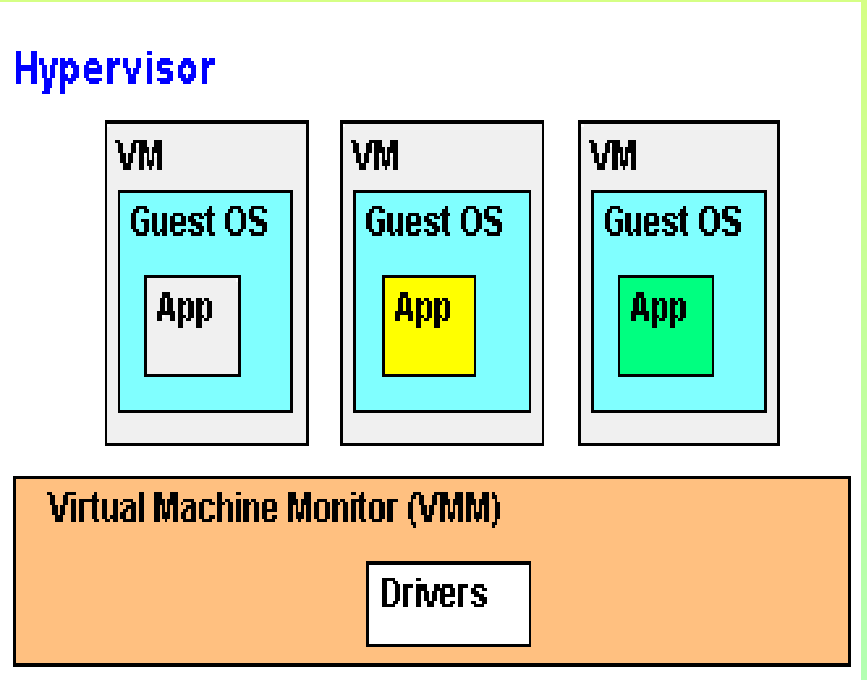


Rootkit Generation Track Table

Generation	Modification Type	Location	Detectibility
1	direct(code)	Disk	Easy
2	direct(code)	Memory	Moderate
3	indirect(data)	Memory	Moderate/Hard

Advanced Generation of Rootkits

- The intruder wonders about designing a rootkit independent from Operating system → Advanced rootkits development begins...
- Example: Hypervisor Rootkit
  Q. How can we exploit this structure to implement an OS-independent rootkit?



Rootkit Generation Track Table

Generation	Modification Type	Location	Detectability
1	direct(code)	Disk	Easy
2	direct(code)	Memory	Moderate
3	indirect(data)	Memory	Moderate/Hard
3+	No Modification	Memory	So Difficult