Kernel Modules and Rootkit Analysis

What is a Rootkit?

- A program that performs system hooking or modifies the functionality of OS
- Hide files, processes, other objects to conceal its presence
- Intercepts and alters the normal execution flow
- Can contain both user mode and kernel mode components
- Some rootkits can install as device drivers
- Types: User Mode and Kernel Mode Rootkits

User Mode Rootkits

- Hooking in user space or application space
- Some common user mode Rootkit techniques:
- IAT (Import Address Table) hooking
- Inline API hooking

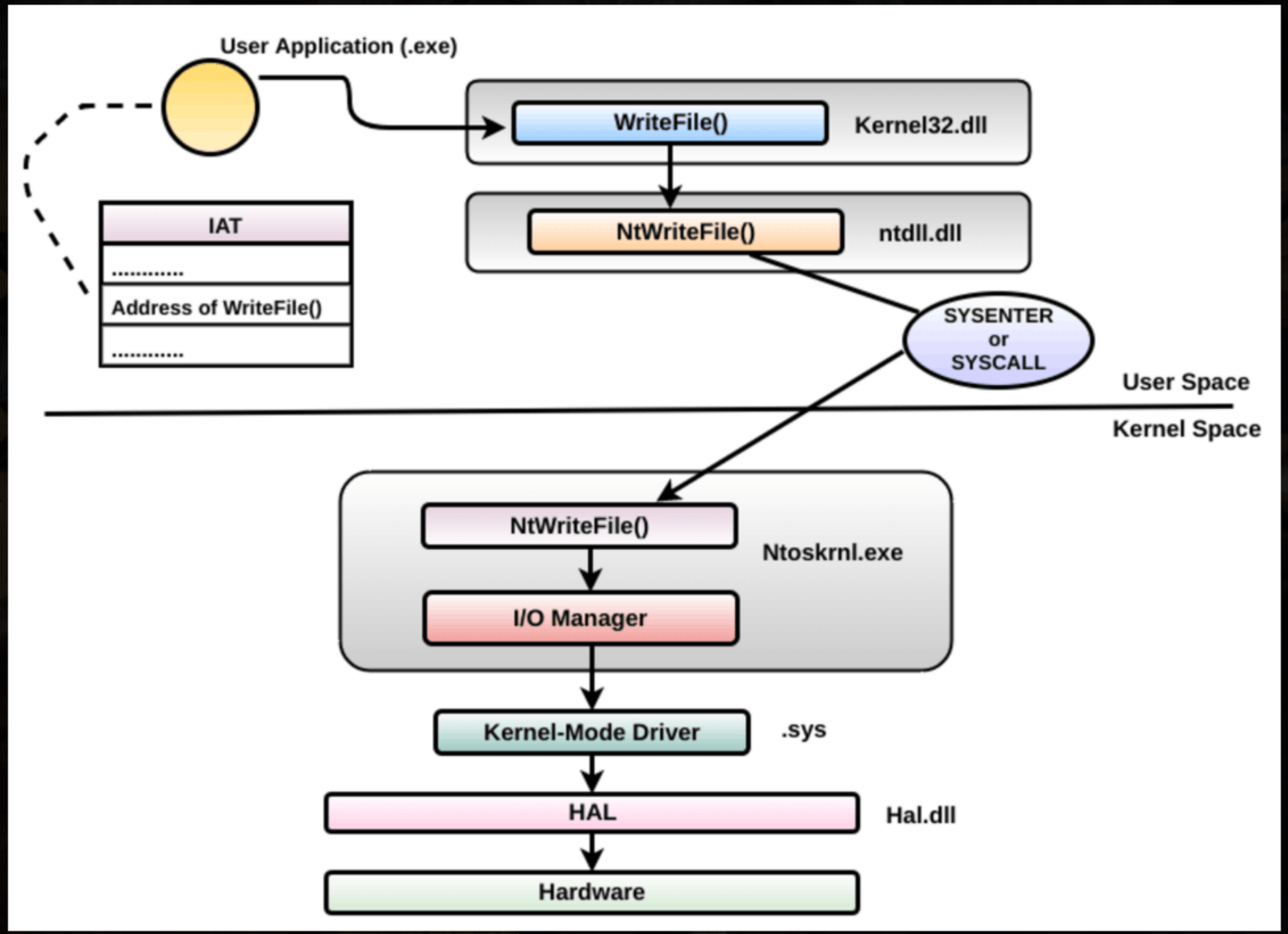
- Rootkit need to perform patching in the memory space of every running application

Kernel Mode Rootkits

- Runs in Ring 0
- System hooking or modification in kernel space

- Some Kernel mode Rootkit techniques:
  - SSDT (System Service Descriptor Table) hooking
  - DKOM (Direct Kernel Object Manipulation)
  - IDT (Interrupt Descriptor Table) hooking
  - Installing as Device Drivers
  - Driver IRP hooking

API call flow



Kernel Volatility Plugins

- ***modules*** - Prints list of loaded modules
- ***modscan*** - scanner for kernel modules
- ***driverscan*** - scanner for driver objects
- ***moddump*** - Dump a kernel driver to an executable file sample
- ***ssdt*** - Display SSDT entries
- ***callbacks*** -Print system-wide notification routines

Lab 17: The Case of Mader Rootkit

From the memory image (***mader.vmem***)

- Identify the SSDT hooks?
- Which kernel functions are hooked?
- Which malicious driver is implementing the hooks?
- Based on the hooks, what type of activity the Rootkit is monitoring?
- Is the driver using any other functionality to monitor the system activity?
- Can you dump the malicious driver and check if the driver is malicious?

Bonus Question:

- Can you disassemble the hooking function and identify the code where it is calling the hooked function?

Demo 15 - Investigating TDSS Rootkit