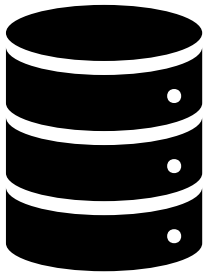
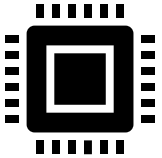
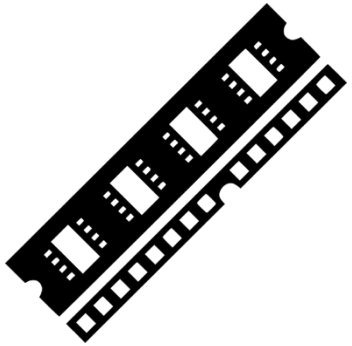
COMPX508 – Malware Analysis

Week 8

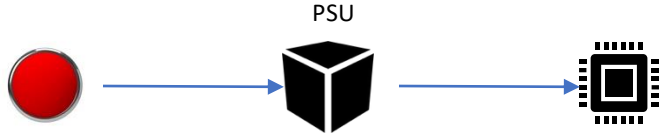
Lecture 1: Windows bootstrapping and processes

Vimal Kumar

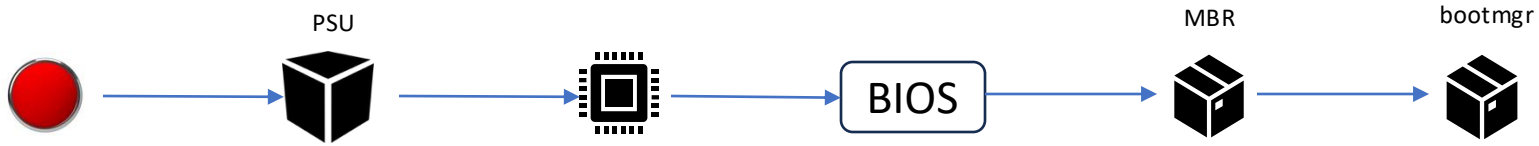
Windows bootstrapping

- Power on
- PSU checks electrical circuitry and send a signal to the CPU to wake up
- Once the CPU receives the wake-up signal it starts executing a pre-defined set of instructions
  - These are hard-wired on the CPU
- One of these instructions is an un-conditional jump into the BIOS/UEFI code



Windows bootstrapping

- CPU runs BIOS/UEFI code
  - Usually stored on a separate chip on the motherboard
- BIOS performs some checks
  - Are all the critical devices (CPU, RAM, HDD/SSD, peripherals, etc) available and functioning? This is called POST
  - BIOS sets the CPU and RAM up for basic execution and then loads the code from the MBR
    - The MBR (Master Boot Record) is the first sector on the disk. The very first 512 bytes on the disk
    - The MBR code loads `bootmanager` from the hidden system partition



Windows bootstrapping

- Bootmanager stores information about where physically on the actual disk the `winload.exe` for the operating system is.
  - This information along with other user preferences data is called the Boot Configuration Data
  - The bootmanager loads `winload.exe` from the disk into the RAM.
- Winload.exe prepares the system for loading the kernel (`ntoskernel.exe`)
  - Partially sets up the virtual memory and page tables
  - Loads `hal.dll` (Hardware Abstraction Layer) so that the kernel can work with the CPU on the system
  - Loads the registry in the memory
  - Loads drivers for accessing storage, file system etc.
  - And finally loads the kernel

Windows bootstrapping

- The execution of `ntoskernel.exe` (kernel) is where the Windows OS actually starts
- The kernel sets up the full virtual memory manager
- Loads device drivers to interact with devices
- Sets up the operating system environment

Processes on Windows

- **Process** – a program in execution

- A program is ***passive*** entity stored on disk (**executable PE file**)
- A process is ***active***
  - A process is created when an executable file is loaded into memory
  - The operating system then tracks its state, memory, resources, etc.
  - This could be via GUI mouse clicks, command line, etc.
- One program can result in the creation of several processes
  - E.g. Google Chrome
  - E.g. executing a program several times

Processes on Windows

- A process can be thought of as a container holding an execution context
- It consists of
  - A process ID
  - Its own virtual address space
  - Executable code (from the .text section of the PE file)
  - Handles (references) to resources such as files, registry keys, sockets, etc.
  - Process Control/Environment Block, which is the data structure where all the above are stored.
  - Threads (At least one)
- Process can either be in user-mode or kernel-mode

Processes on Windows

- In Windows all new process are created using the `CreateProcess` API call*
- Processes have a hierarchical relationship.
  - A process that creates another process is called the parent and the new process is called the child process.

Processes on Windows

- Registry and System Idle Process are not really processes
- They are shown as processes in monitoring tools for accounting and visibility
- System Idle Process
  - A (non) process with one or more kernel threads (one for each processor) that are always in “ready” state.
  - The threads don’t do anything but are scheduled when no threads are available to run
  - PID = 0
- Registry
  - Is not a process
  - Useful to see the amount of memory being taken by the registry hive
  - Is given a pseudo PID

Processes on Windows

- System
  - Special, kernel process
    - Handles kernel level tasks and loads drivers
  - On Windows 10, also handles memory compression
  - PID = 4
  - Created by `ntoskrnl.exe` when it is loaded by winload
  - Creates `smss` process
  - The system process has no parent process

Processes on Windows

- Smss
  - Once all system drivers are loaded the system process starts smss.exe
  - Windows session manager subsystem
    - Manages desktops and all processes belonging to users
  - Sets up session 0
    - Windows uses sessions to separate user environments
    - Session 0 is non-interactive and for system services
  - First user level process created by system
    - Creates two session
      - Session 0 for OS services
        - Non-interactive
      - Session 1 for User services
        - User logs in to session 1
      - More sessions?
  - Spawns winlogon, wininit and csrss
  - After spawning these processes it goes into a suspend state

Processes on Windows

- winlogon
  - Runs for each session and manages access to the desktop
  - Once a user is authenticated, checks in the registry to find initialization process to run next
    - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon
    - At least one executable at that location
- csrss
  - Client server runtime subsystem
  - Before Windows NT it was responsible for the entire UI
  - Now responsible for the console window and the shutdown process
    - At least two versions one for session 0 and one for session 1

Processes on Windows

- wininit.exe
  - Launches system applications
  - Starts services.exe
  - Starts lsass.exe
  - Runs in session 0
- Services
  - Services or Service Control Manager (SCM) is responsible for running, ending, and interacting with system services
  - Launched by Wininit
  - Starts all services configured for automatic startup
  - Looks at the HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Svchost key in the registry
- svchost
  - Host process for windows and third party services
- lsass
  - Local Security Authority Subsystem
  - Enforces security policy
  - Handles user access control

Processes on Windows

- explorer.exe
  - Handles the Graphical UI
  - Every program that is launched will appear as a child process