Texas Tech University

A Course on Digital forensics

Memory Forensics

Module 1 – System Overview

Akbar S. Namin, Spring 2018

CHAPTER 1 – SYSTEMS OVERVIEW

OUTLINE

1. Digital Environment
2. PC Architecture
    a. Physical Organization
    - CPU and MMU
    - North and Southbridge
    - Direct Memory Access
    - Volatile Memory (RAM)
    b. CPU Architectures
    - Address Spaces
    - Intel IA-32 Architecture
    - Registers
    - Segmentation
    - Paging
    - Address Translation
    - Physical Address Extension
    - Intel 64 Architecture
    - Interrupt Descriptor Table
3. Operating Systems
    a. Privilege Separation
    b. System Calls
4. Process Management
    a. Threads
    b. CPU Scheduling
    c. System Resources
5. Memory Management
    a. Virtual Memory
    b. Demand Paging
    c. Shared Memory
    d. Stacks and Heaps
6. File System
7. I/O Subsystem
    a. Device Drivers
    b. I/O Controls (IOCTLs)

CONTENT

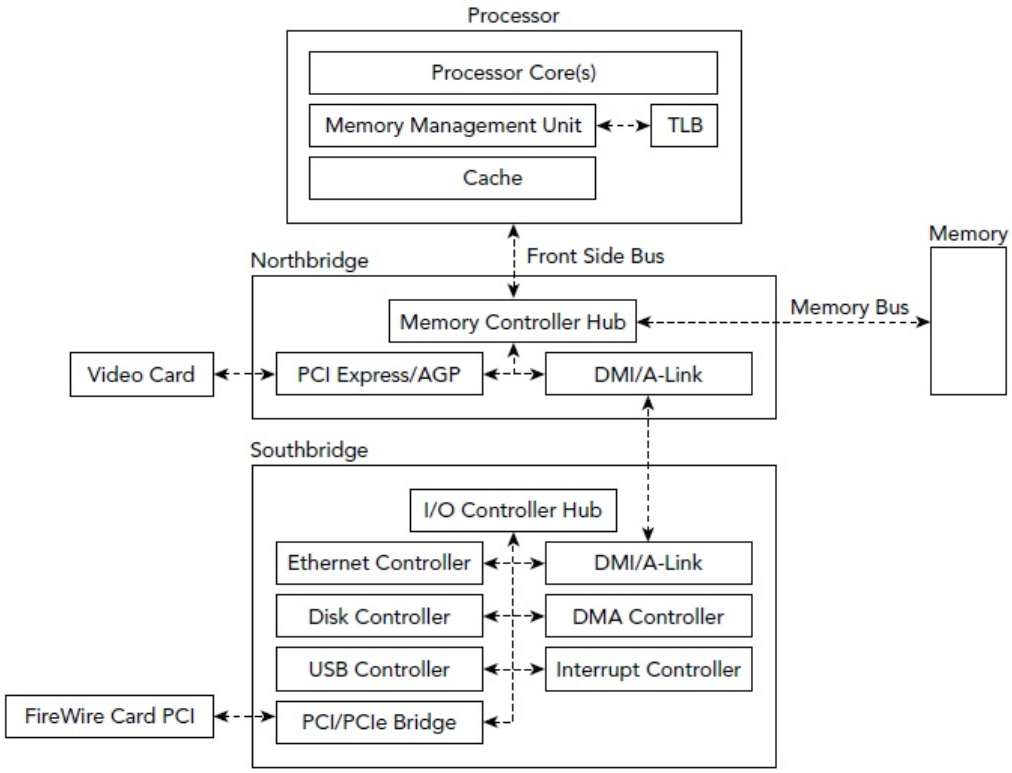
1. Digital Environment

- On most platforms, the hardware is accessed through a layer of software called an *operating system*, which controls processing, manages resources, and facilitates communication with external devices.
- Operating systems must deal with the low-level details of the particular processor, devices, and memory hardware installed in a given system.
- During an investigation, you look for artifacts that suspected software or users might have introduced into the digital environment and try to determine how the digital environment changed in response to those artifacts.

2. PC Architecture

a. Physical Organization: A PC is composed of printed circuit boards that interconnect various components and provide connectors for peripheral devices.

• CPU and MMU: CPU executes programs, and the main memory which temporarily stores the executed programs and their associated data.



Physical organization of a modern system

- Modern systems leverage multiple layers of fast memory, called *caches*, to help offset this disparity.
- In most systems, these caches are built into the processor and each of its cores. If data is not found within a given cache, the data must be fetched from the next level cache or main memory.
- The CPU relies on its *memory management unit* (*MMU*) to help find where the data is stored. The MMU is the hardware unit that translates the address that the processor requests to its corresponding address in main memory.

• **North and Southbridge**

- The CPU relies on the *memory controller* to manage communication with main memory.
- The memory controller is responsible for mediating potentially concurrent requests for system memory from the processor(s) and devices.

- On older PCs, CPU connected to the northbridge. And devices were connected via another chip called southbridge. But to improve performance and reduce the costs of newer systems, most capabilities associated with the memory controller hub are now integrated into the processor.

- **Direct Memory Access**

- Most modern systems provide I/O devices the capability to directly transfer data stored in system memory without processor intervention.
- This capability is called *direct memory access* (*DMA*).
- Before DMA was introduced, the CPU would be fully consumed during I/O transfers and often acted as an intermediaryVolatile Memory (RAM).
- Besides its obvious impact on system performance, DMA also has important ramifications for memory forensics. It provides a mechanism to directly access the contents of physical memory from a peripheral device without involving the untrusted software running on the machine.
- For example, the PCI bus supports devices that act as *bus masters*, which means they can request control of the bus to initiate transactions. As a result, a PCI device with bus master functionality and DMA support can access the system's memory without involving the CPU.

- **Volatile Memory**

- The main memory of a PC is implemented with *random access memory* (*RAM*), which stores the code and data that the processor actively accesses and stores.
- RAM is considered *volatile memory* because it requires power for the data to remain accessible. Thus, except in the case of cold boot attacks (https://citp.princeton.edu/research/memory), after a PC is powered down, the volatile memory is lost.
  - ✓ Cold boot attack: https://www.youtube.com/watch?v=Ej-Nr79bVjg

b. CPU Architectures: To effectively extract structure from physical memory and understand how malicious code can compromise system security, you should have a firm understanding of the programming model that the CPU provides for accessing memory.

- Address Spaces

- For the CPU to execute instructions and access data stored in main memory, it must specify a unique address for that data. The processors discussed in this book leverage byte.

- Intel IA-32 Architecture

- The IA-32 architecture commonly refers to the family of x86 architectures that support 32-bit computation.

- Registers

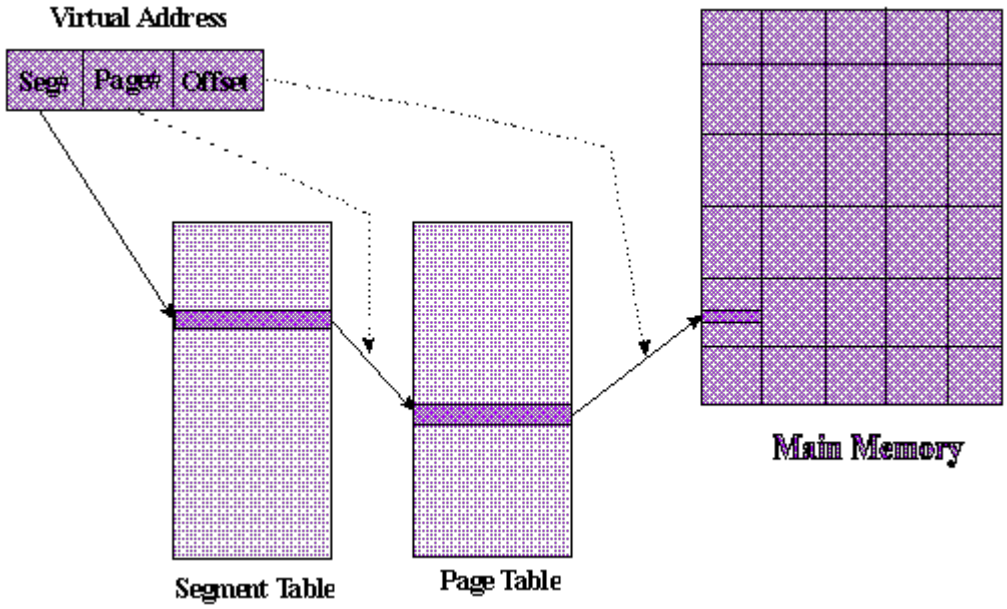
- The IA-32 architecture defines a small amount of extremely fast memory, called *registers*, which the CPU uses for temporary storage during processing. Each processor core contains

- Segmentation

- IA-32 processors implement two memory management mechanisms: *segmentation* and *paging*.
- Segmentation divides the 32-bit linear address space into multiple variable-lengthsegments.
- All IA-32 memory references are addressed using a 16-bit segment selector, which identifies a particular segment descriptor, and a 32-bit offset into the specified segment.

- Paging

- Paging provides the ability to virtualize the linear address space. It creates an executionenvironment in which a large linear address space is simulated with a modest amountof physical memory and disk storage.



Retrieved from: http://denninginstitute.com/modules/vm/purple/segment.html

- **Address Translation**
- **Physical Address Extension**

- The IA-32 architecture's paging mechanism also supports PAE.
- This extension allows the processor to support physical address spaces greater than 4GB. Although programs still possess linear address spaces of up to 4GB, the memory management unit maps those addresses into the expanded 64GB physical address space.

- **Intel 64 Architecture**

- The execution environment for the Intel 64 architecture is similar to IA-32, but there are a few differences.
- The Intel 64 architecture supports a linear address space up to 2^{64} bytes.

- **Interrupt Descriptor Table**

- PC architectures often provide a mechanism for interrupting process execution and passing control to a privileged mode software routine.
- After most interrupts, the operating system will resume execution where it was originally interrupted.
  - ✓ WARNING: Given the critical role that the IDT performs for operating systems, it has been a frequent target of malicious software. Malicious software might try to redirect entries, modify handler code, add new entries, or even create entirely new interrupt tables. (For example: https://www.blackhat.com/presentations/bh-jp-05/bh-jp-05-sparks-butler.pdf)

3. Operating Systems

a. Privilege Separation

- To prevent potentially malfunctioning or malicious user applications from accessing or manipulating the critical components of the operating system, most modern operating systems implement some form of user and kernel mode privilege isolation.

b. System Calls

- A user application requests a service from the operating system's kernel using a system call.
- System calls define the low-level API between user applications and the operating system kernel.
- Because it is such a critical bridge between user applications and the operating system, the code used to service system call interrupts is commonly intercepted by security products and targeted by malicious software.

4. Process Management:

- *Process* is an instance of a program executing in memory. The operating system is responsible for managing process creation, suspension, and termination.
- The *process address space* becomes a container for the application's code, shared libraries, dynamic data, and runtime stack.
- A process provides the execution environment, resources, and context for threads to run.
- An important aspect of memory analysis involves enumerating the processes that were executing on a system and analyzing the data stored within their address spaces, including passwords, URLs, encryption keys, e‑mail, and chat logs.

a. Threads

- A *thread* is the basic unit of CPU utilization and execution.
- In terms of memory forensics, thread data structures are useful because they often contain timestamps and starting addresses.
- This information can help you determine what code in a process has executed and when it began.

b. CPU Scheduling

- The operating system's capability to distribute CPU execution time among multiple threads is referred to as *CPU scheduling*.
- One goal of scheduling is to optimize CPU utilization.

c. System Resources

- Another important service that an operating system provides is helping to manage a process' resources. As previously mentioned, a process acts as a container for system resources that are accessible to its threads.

2. Memory Management: Memory management refers to the operating system's algorithms for managing the allocation, deallocation, and organization of physical memory.

a. Virtual Memory

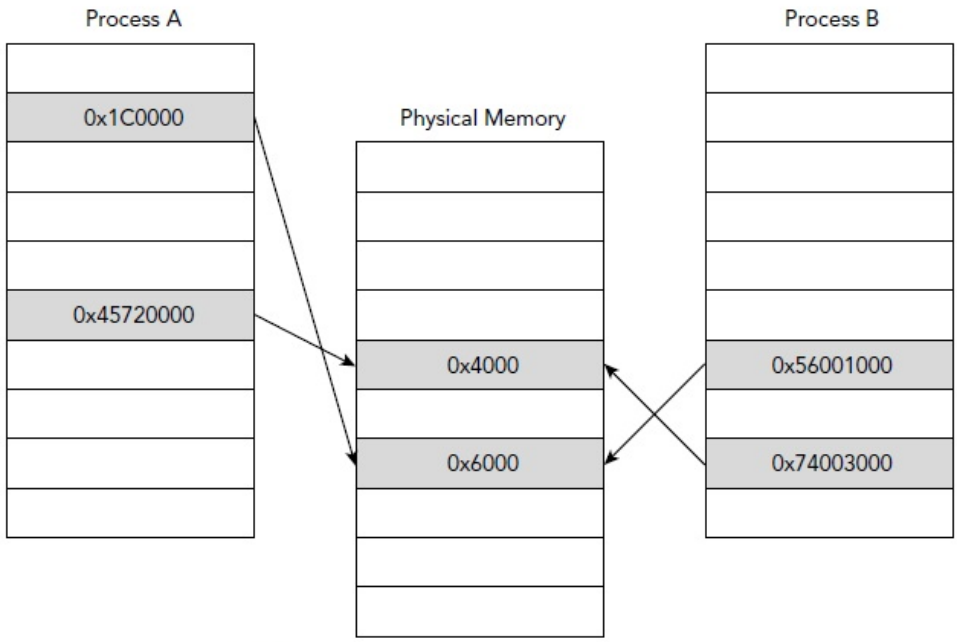
- Operating systems provide each process with its own private virtual address space. This abstraction creates a separation between the logical memory that a process sees and the actual physical memory installed on the machine.
- Behind the scenes, the memory manager is responsible for transferring regions of memory to secondary storage to free up space in physical memory.
- With the support of the hardware, the memory manager can partition the data to prevent a malicious or misbehaving process from reading or writing memory that belongs to kernel memory or other processes.

b. Demand Paging

- The mechanism that is commonly used to implement virtual memory is *demand paging*, which is a memory management policy for determining which regions are resident in main memory and which are moved to a slower secondary storage when the need arises.
- It does add some complexity to memory forensics because some pages might not be memory resident at the time the memory sample is collected.

c. Shared Memory

- You can view shared memory as memory that is accessible from more than one virtual address space.
- For example, figure shows that Process A and Process B have regions of their private virtual address space that map to common pages in physical memory.
- One common use for shared memory is to provide an efficient means of communication between processes.



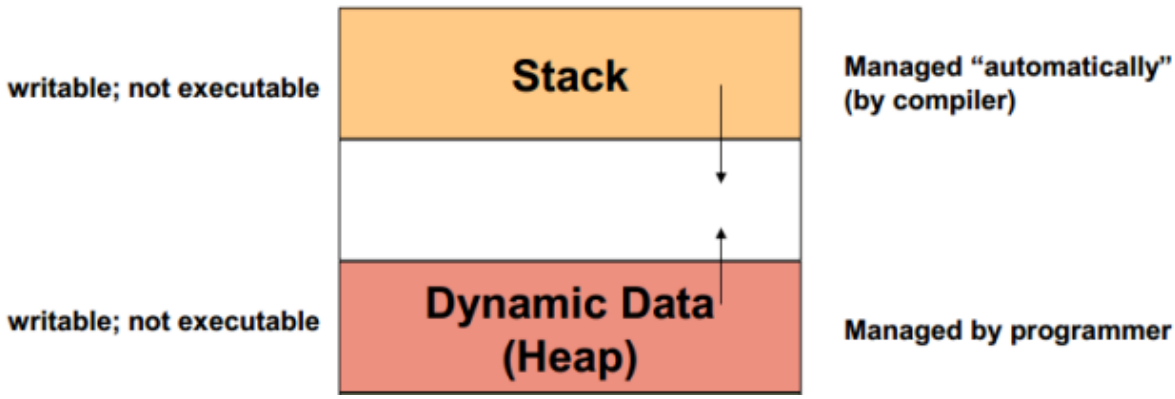
Example of shared memory mappings between two processes

- Shared memory is also commonly used to conserve physical memory. Instead of allocating multiple physical pages that contain the same data, you can create a single instance of the data in physical memory and map various regions of virtual memory to it. Examples include shared or dynamic libraries that contain common code and data.
- Both shared memory and copy-on-write mappings are frequently encountered during memory forensics because malicious software often attempts to modify the code of shared libraries to hijack the flow of execution.

d. Stacks and Heaps

- The *stack* region holds the temporary data associated with executing functions.
- The data in this region is stored in a data structure called a *stack frame*.
- During malware analysis, stack frames can be used to infer what part of the malware was active and what parts of the system the malware was interacting with.
- The application's data that needs to be dynamically allocated is stored within the region called the *heap*.

- The data allocated within the heap can persist for the lifetime of the process.



Retrieved from: http://stackoverflow.com/questions/32418750/stack-and-heap-locations-in-ram

3. File System

- Unlike volatile main memory, secondary storage is typically composed of nonvolatile block devices such as hard disks.
- The collection of data structures that allow an application to perform primitive operations on the stored data is called a *file system*.
- Data stored in files and the directory structures must be loaded into memory when they are needed.
- The operating system also caches frequently accessed data in main memory to reduce the overhead associated with repetitively querying slower secondary storage.

4. I/O Subsystem

a. Device Drivers

- Device drivers abstract away the details of how a device controls and transfers data.
- Software commonly abuses device drivers to modify the state of the system.
- This device interface has been commonly used to collect forensic samples of physical memory. Note that device memory and registers might also be mapped into memory, which can have interesting consequences for memory acquisition (**see Chapter 4**).

b. I/O Controls (IOCTLs)

- *I/O Control* (*IOCTL*) commands are another common mechanism for communicatingbetween user mode and kernel mode.
- IOCTLs allow a user application to communicate with a kernel mode device driver.
- As with system calls, kernel-level malware might hook IOCTL functions in order to filter results or modify control flow. Malware has also used IOCTL handlers to communicate between user mode and kernel mode components (for example, to request that a kernel component elevate privileges, disable a service, or modify firewall settings).
- Memory forensics can detect modified or unknown IOCTLs and provide valuable insight into how attackers leverage them.