

CS 798: Digital Forensics and Incident Response

Lecture 14 - Digital Stratigraphy & Memory Forensics

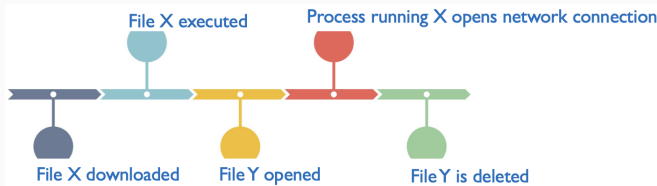
Diogo Barradas

Winter 2025

University of Waterloo

Temporal Analysis

- Now that we have all those OS-, network- and FS-related artifacts, we want to establish a **timeline of events**:
 - Help identify patterns, gaps, sequencing past actions, and lead to other sources of evidence



- Investigators resort to **timestamps**:
 - From all sorts of sources: logs, file metadata, Registry, etc.
- Can we **rely on timestamps**? What if they have been **tampered with**? Do we even **have timestamps** to start with?

1. Temporal Analysis

- Time tracking

- Timestomping

- Digital Stratigraphy

2. Memory analysis

- Memory analysis methodology

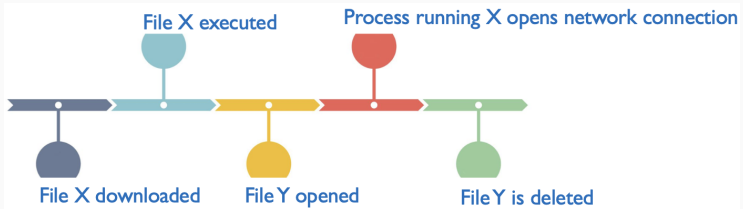
- Recovery of memory artifacts

- Memory acquisition techniques

Temporal Analysis

Temporal Analysis

- Forensic investigations usually require to know the **time and sequence** of events
- Fortunately, computers keep ample account of time
- Temporal analysis consists in analyzing timing information from digital artifacts
- File systems, in particular, are rich in maintaining **timestamps**



Computer timekeeping

- **Real Time Clock (RTC)**
 - Battery powered; keeps time while computer is shut down
 - Used as basis for determining time when computer boots
- **System clock**
 - SW clock set from the RTC at boot plus HW timer
- **Network Time Protocol (NTP)**
 - Protocol for reliable synchronization of computer clocks
- **Network Identity and Time Zone (NITZ)**
 - Method to obtain time info from GSM network
- **Global Positioning System (GPS)**
 - Device sets its clock based on signals received from GPS

Time tracking in file systems

- **MACtimes:** three time attributes attached to any file or directory in UNIX, Windows, and other systems
 - **atime:** Last time the file or directory was accessed
 - **mtime:** changes when a file's contents are modified
 - **ctime:** keeps track of when the contents or meta-data about the file has changed: owner, group, file permissions, etc.
- Sometimes this information is enriched with **creation** time

TSK tools for timestamp analysis

- TSK has several tools: mactime, fls, icat
- Example of mactime output:

```
Columns:
  Date/Time      Size  Activity  Unix   User   Group   inode   File Name
              (Bytes)  Type      Permissions  Id      Id
Example:
[...]
Thu Aug 21 2003 01:20:38    512    m.c.    -/-rwxrwxrwx    0      0      4      /file1.dat
                        900    m.c.    -/-rwxrwxrwx    0      0      8      /file3.dat
Thu Aug 21 2003 01:21:36    512    m.c.    -/-rwxrwxrwx    0      0     12      /_ILES.DAT (deleted)
Thu Aug 21 2003 01:22:56    512    .a..    -/-rwxrwxrwx    0      0      4      /file1.dat
[...]
```

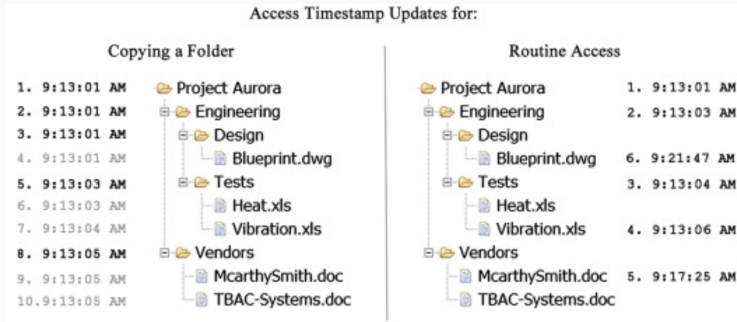
MAC Meaning by File System

File System	m	a	c	b
Ext4	Modified	Accessed	Changed	Created
Ext2/3	Modified	Accessed	Changed	N/A
FAT	Written	Accessed	N/A	Created
NTFS	File Modified	Accessed	MFT Modified	Created

M = Modified
A = Accessed
C = Changed
B = Created (Birthed)

Access time patterns may emerge

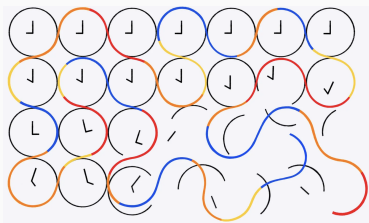
- Check the two access time patterns below in two cases



- Access patterns may disclose past activities
 - Useful in intellectual property theft – date-time stamps may show what files were copied and when!

How reliable are timestamps?

- Several factors influence the accuracy of timekeeping on computers and the interpretation of timestamps:
 - System clock implementation
 - Clock configuration
 - Tampering
 - Synchronization protocol
 - Misinterpretation of timestamps
 - Bugs in software



Time and date issues in forensic investigation

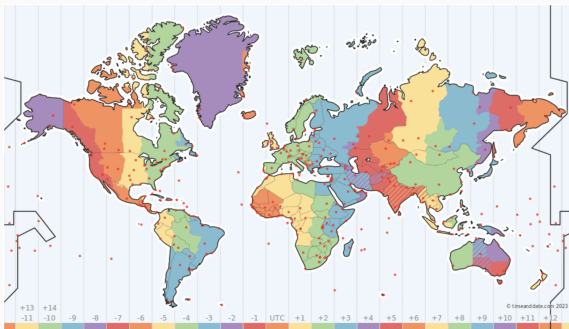
- **Clock skew**
 - The amount of time units (often seconds or milliseconds) by which a clock deviates from the “real” time
- **Unnormalized timestamps**
 - Represent timestamps taken in different time zones

Detection of clock skew

- In any investigation, it is important to assert if the device's system clock was **registering the correct time**
- Compare with the time of an NTP-synced computer
 - When the device is still operational
- Compare with external timestamps
 - Generated outside the investigated device
 - e.g., call logs, network logs, etc.

Normalization of timestamps

- Requires proper understanding of the various **formats and time zones** in which timestamps are stored
 - Ext2 timestamps in Unix time = total number of seconds since 01-01-1970 00:00:00 UTC
- If timestamps of different time zones, translate them into a **common time zone**, e.g., UTC

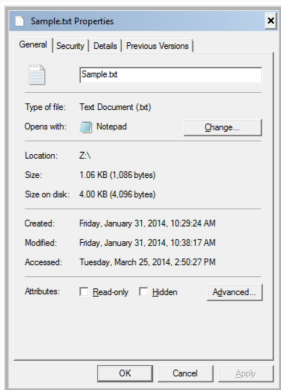


Time forgery analysis and timestamp resolution

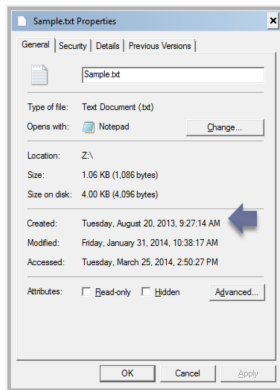
- Timestamps are **not meant to be manipulated** by the end user
 - Mostly generated by OS (e.g., FS updates) and applications (e.g., email)
- However, resourceful users can modify these timestamps using various methods: this is called **timestomping**
- Criminals leverage timestomping so that file accesses / modifications do not appear conspicuous to investigators
- Can we determine that timestomp programs have been used for anti-forensic purposes?

Example date forgery analysis scenario

- Consider the following file, with local (PT) timestamps
 - On the left, File Properties shows timestamps with second-precision
 - On the right, after using a time modification tool to change timestamps



timestomper.exe



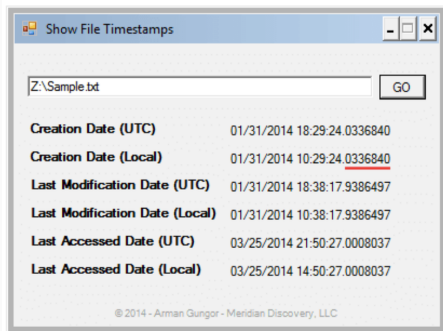
NTFS timestamp resolution

- In NTFS, timestamps are stored as 8-byte file time values
 - Represent the number of 100-nanosecond intervals that have elapsed since 12:00 A.M. January 1, 1601
 - NTFS timestamps have 100 nanosecond precision
- Below, the creation time before timestamp modification

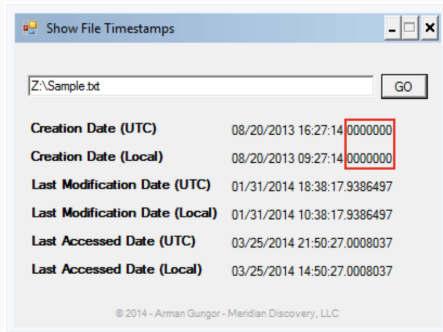
[illegible]

Gather information about the root partition

- After the timestamp manipulation, the updated creation timestamp has **lost its resolution** beyond seconds



Original



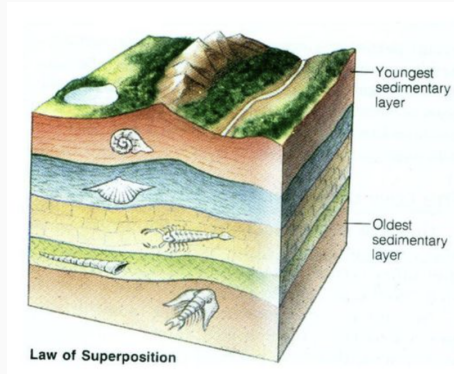
After modification

Some strategies for timestomping detection

- **Anomalies in timestamp format (ending in zeros)**
 - Relies on the limitation of tools that are used to modify the timestamp; the timestamp resolution stops at second level and everything else (all the way to the last 100ns) is set to zero
- **Inconsistencies with other timing sources**
 - e.g., compared with the Windows Event log

Digital stratigraphy

- Useful when time markers are obliterated
- **Stratigraphy:** The study of rock layers
- Predicts the age of natural artifacts based on the principle that upper layers are younger than the layers underneath
- The idea is to employ a similar approach in the digital realm



Establishing a time order

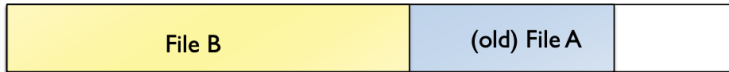
- **Digital stratigraphy** studies file system traces and writing patterns to infer time-related facts
- Let's showcase two representative techniques with a challenge:
 - Consider files A and B – was A written earlier or later than B?
- In both techniques, the idea is to **establish a time order** between both these files, but using different “digital layers”

1. Based on how data was overwritten

- Suppose file A was first created and fits into one block

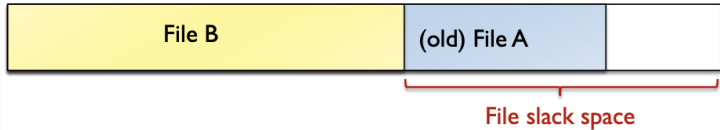


- Then, A was deleted and that block was unallocated
- Later file B was created, the OS reallocated the same block, but B is a smaller file than A

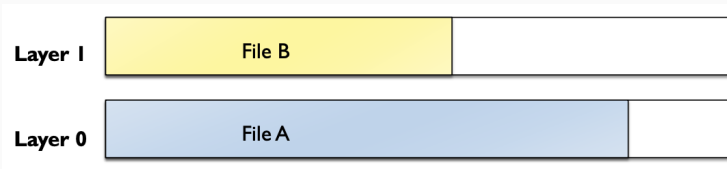


Breaking it down

- This is what the forensic analyst will be able to retrieve



- We can conceptualize this into two different layers:



- So, even if we do not have timestamp info about B, but we have about A, then we can still say that B is likely more recent than A

Additional considerations

- May need to look for **other time reference** points
 - Date-time info exists in Word files, directory entries, cookie files, Internet- related files, event logs, etc.
- The insight was based on **knowledge on how the OS reallocates blocks** of formerly deleted files and effect of slack space
 - Once deleted, these files form an underlying layer of time-related data upon which newer files are saved
- But this principle can be more broadly applied, e.g., **layers preserved after disk formatting**
 - e.g., computer running Linux was found with numerous Windows files in unallocated space that contain hardware specific info (e.g., NIC address)
 - It is likely that the computer was running Windows before!

2. Based on data positioning

- Suppose now that A uses 3 blocks when created

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- Now, file B is created in block #1, and it keeps increasing until it requires 5 blocks:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- The OS will typically try to find contiguous blocks but, in this case, fragmentation will occur

Breaking it down

- This is what the forensic analyst will be able to retrieve

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

- We can conceptualize this into two different layers:

Layer 1	1	2	3	4	5	6	7	8	9	10
Layer 0	1	2	3	4	5	6	7	8	9	10

- So, even if we do not have timestamp info about neither of files, we may likely infer that **B is likely more recent than A**

Additional considerations

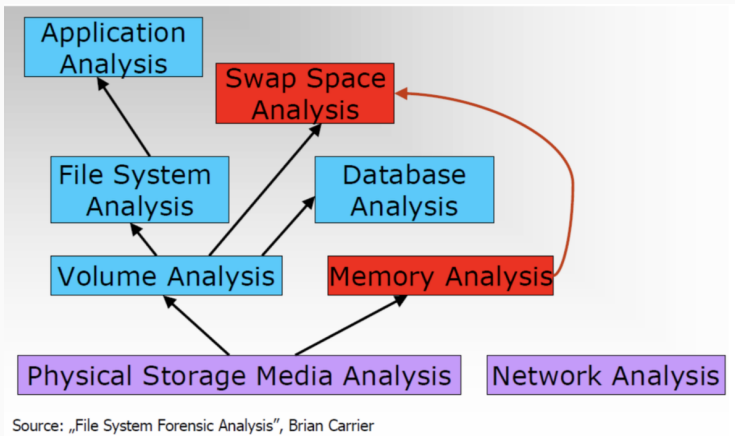
- The insight was based on knowledge on how the OS allocates blocks to files and effect of fragmentation
 - Two pieces of a file located in blocks on either side of a large, contiguous file
 - It is likely that the contiguous file is older
- However, **these techniques are not infalible** because there might be other effects taking place
- Thus, it is always important to look for other potential sources for supporting or dispelling these hypotheses

Takeaways (I)

- Temporal analysis is fundamental in digital forensics for establishing a timeline of events
- In performing temporal analysis, we may have to deal with several challenges, including timestomping, and may leverage emerging stratigraphy techniques

- **Textbook:**
 - Casey – Chapters 16.6, 17.1.2–4, Carrier – Chapters 11–12, Luttgens – Chapter 12.1

We talked a lot about file systems...



- Is it possible to recover formerly deleted files?
- How to recover deleted files when no metadata is available?

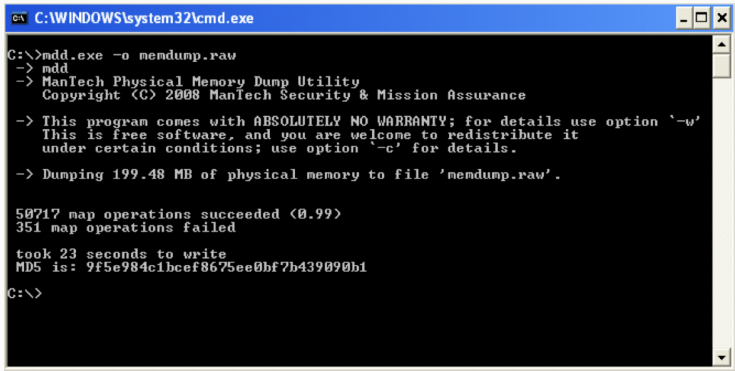
Memory analysis

Memory forensics: Dump & analyze

- Dump physical memory (RAM)
- Lots of potentially relevant data
 - Current running processes and terminated processes
 - Open TCP/UDP ports/raw sockets/active connections
 - Memory mapped files: executable image, shared libs, modules/drivers, text files
 - Caches: Web addresses, typed commands, passwords, clipboards, SAM database, edited files
 - Hidden data, encryption keys and many more
- Analyze the RAM
 - Enumerate different program structures, signature based carving, find text strings, virus scans, network connections, etc.

Example: Dump and analyze RAM memory

- We are analyzing a Windows workstation, suspected to be infected by malware
 - Look for evidence of that program residing in memory
- First, make a dump of physical memory inside the workstation
 - mdd.exe copies the contents of a computer's physical memory to a file



```
C:\WINDOWS\system32\cmd.exe

C:\>mdd.exe -o memdump.raw
-> mdd
-> ManTech Physical Memory Dump Utility
   Copyright (C) 2008 ManTech Security & Mission Assurance

-> This program comes with ABSOLUTELY NO WARRANTY; for details use option '-w'
   This is free software, and you are welcome to redistribute it
   under certain conditions; use option '-c' for details.

-> Dumping 199.48 MB of physical memory to file 'memdump.raw'.

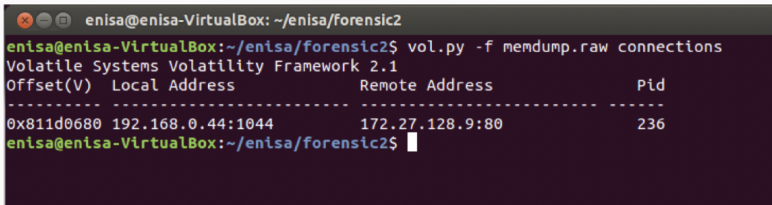
50717 map operations succeeded (0.99)
351 map operations failed

took 23 seconds to write
MD5 is: 9f5e984c1bcef8675ee0bf7b439090b1

C:\>
```

Evaluate gathered evidence

- The volatility framework helps interpret kernel data structures from the memory dump
- First, check if the malware was communicating with external server by displaying active connections



```
enisa@enisa-VirtualBox: ~/enisa/forensic2
enisa@enisa-VirtualBox:~/enisa/forensic2$ vol.py -f memdump.raw connections
Volatile Systems Volatility Framework 2.1
Offset(V)  Local Address          Remote Address          Pid
-----
0x811d0680 192.168.0.44:1044        172.27.128.9:80        236
enisa@enisa-VirtualBox:~/enisa/forensic2$
```

The screenshot shows a terminal window with the command `vol.py -f memdump.raw connections` executed. The output displays a table of active network connections. The first entry shows a connection from local address 192.168.0.44:1044 to remote address 172.27.128.9:80, associated with PID 236.

- The computer had an active connection with IP 172.27.128.9:80 (the suspect malicious server)
 - Next, follow the lead of the associated PID

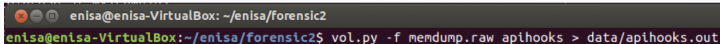
Evaluate gathered evidence

- Use volatility to list which windows process ID (PID) has opened the connections: 236

```
enisa@enisa-VirtualBox: ~/enisa/forensic2
enisa@enisa-VirtualBox:~/enisa/forensic2$ vol.py -f memdump.raw pslist
Volatile Systems Volatility Framework 2.1
Offset(V) Name PID PPID Thds Hnds Sess Wow64 Start Exit
-----
0x8132b020 System 4 0 54 523 ----- 0
0x81231c60 smss.exe 408 4 3 21 ----- 0 2013-08-26 15:32:08
0x8119b698 csrss.exe 564 408 10 341 0 0 2013-08-26 15:32:09
0x811f4b00 winlogon.exe 588 408 18 503 0 0 2013-08-26 15:32:09
0x81232020 services.exe 792 588 15 253 0 0 2013-08-26 15:32:09
0x81235020 lsass.exe 804 588 20 331 0 0 2013-08-26 15:32:09
0xffbd5530 VBoxService.exe 988 792 8 107 0 0 2013-08-26 15:32:09
0x81230020 svchost.exe 1060 792 22 215 0 0 2013-08-26 15:32:09
0x81195790 svchost.exe 1172 792 9 239 0 0 2013-08-26 15:32:10
0xffac11d8 svchost.exe 1360 792 59 1139 0 0 2013-08-26 15:32:10
0xffab8688 svchost.exe 1424 792 6 76 0 0 2013-08-26 15:32:10
0x811d7760 svchost.exe 1456 792 14 206 0 0 2013-08-26 15:32:10
0xffa92c08 spoolsv.exe 2008 792 10 106 0 0 2013-08-26 15:32:11
0x811c62f0 explorer.exe 236 216 18 365 0 0 2013-08-26 15:32:12
0xffa7b8c0 VBoxTray.exe 288 236 7 71 0 0 2013-08-26 15:32:12
0xffa7b280 msmsgs.exe 296 236 3 176 0 0 2013-08-26 15:32:12
0xffa7a660 enneo.exe 304 236 0 ----- 0 2013-08-26 15:32:12 2013-08-26 15:32:13
0xff9d6d08 alg.exe 868 792 6 104 0 0 2013-08-26 15:32:29
0xffbb7228 wscntfy.exe 200 1368 3 48 0 0 2013-08-26 15:32:30
0x811d5c08 taskmgr.exe 340 588 3 76 0 0 2013-08-26 15:32:35
0x8119a130 cmd.exe 1432 236 1 53 0 0 2013-08-26 15:32:56
0x811e3a58 wpabaln.exe 1356 588 1 77 0 0 2013-08-26 15:34:08
0xffa9f800 mdd.exe 744 1432 1 41 0 0 2013-08-26 15:42:21
enisa@enisa-VirtualBox:~/enisa/forensic2$
```

Evaluate gathered evidence

- Under normal circumstances, `explorer.exe` should not make any external connections
 - Hypothesis: check whether the binary has been modified or if other process injected malicious code into one of explorer's threads
- Test by checking if API hooking was used for modifying a program at runtime
 - Run volatility to extract the API hooks used by the application for making system calls



```
enisa@enisa-VirtualBox: ~/enisa/forensic2
enisa@enisa-VirtualBox:~/enisa/forensic2$ vol.py -f mendum.raw apihooks > data/apihooks.out
```

Evaluate gathered evidence

- Let's find what processes memory space contained API hooks with something registered:

```
enisa@enisa-VirtualBox: ~/enisa/forensic2/data
enisa@enisa-VirtualBox:~/enisa/forensic2/data$ cat apihooks.out | grep Process | sort -u
Process: 1356 (wpabaln.exe)
Process: 1432 (cmd.exe)
Process: 200 (wscntfy.exe)
Process: 236 (explorer.exe)
Process: 288 (VBoxTray.exe)
Process: 340 (taskmgr.exe)
Process: 744 (mdd.exe)
enisa@enisa-VirtualBox:~/enisa/forensic2/data$
```

- Let's check the explorer.exe hook reported inside apihooks.out:

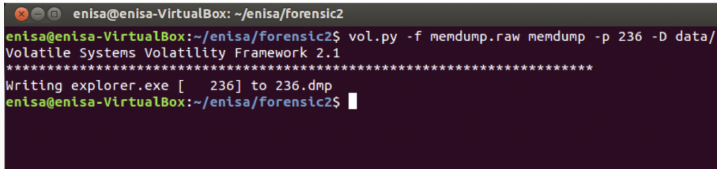
```
*****
Hook mode: Usermode
Hook type: Inline/Trampoline
Process: 236 (explorer.exe)
Victim module: ntdll.dll (0x7c900000 - 0x7c9b0000)
Function: ntdll.dll!NtCreateThread at 0x7c90d7d2
Hook address: 0x1246989
Hooking module: <unknown>

```

Plain Text ▾	Tab Width: 8 ▾	Ln 1, Col 1	INS
--------------	----------------	-------------	-----

Evaluate gathered evidence

- Extract the memory space of explorer.exe from the computer memory image:

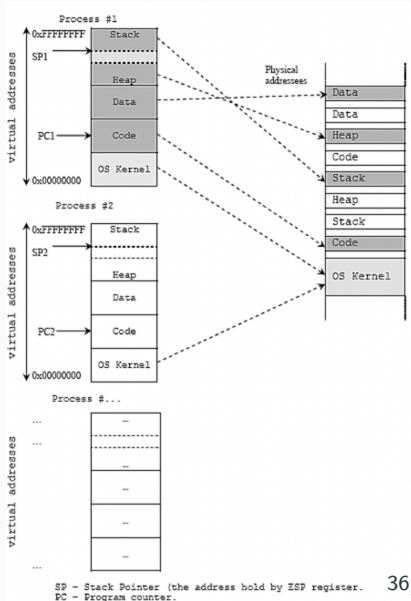


```
enisa@enisa-VirtualBox: ~/enisa/forensic2
enisa@enisa-VirtualBox:~/enisa/forensic2$ vol.py -f memdump.raw memdump -p 236 -D data/
Volatile Systems Volatility Framework 2.1
*****
Writing explorer.exe [ 236] to 236.dmp
enisa@enisa-VirtualBox:~/enisa/forensic2$
```

- Continue with the investigation looking for evidence of malicious code injected into explorer.exe process ...

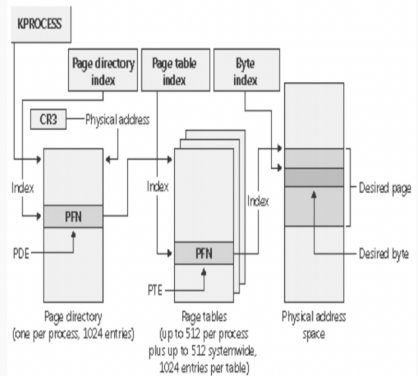
Primer on memory organization

- Processors with MMU (Memory Management Unit) support the concept of virtual memory
- However, it has limitations in general case:
 - Page tables are set up by the kernel to map virtual addresses to physical addresses



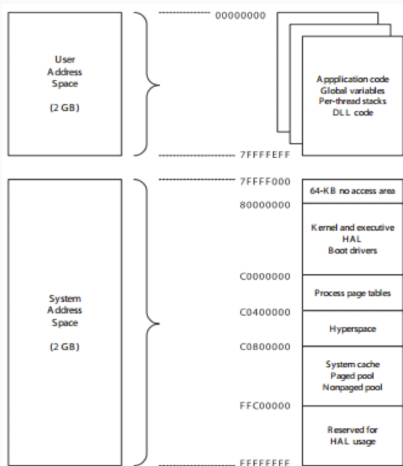
Virtual address translation

- Programs operate on virtual memory regions
- Volatile storage is organized into units called pages
- Size of pages is 4 KB on x86 platforms
- Two level approach to reference a page



Memory address space layout

- In Microsoft Windows, each process has its own private virtual address space
- 32 bit x86 user is equipped with 2 GB of virtual memory
 - In 64-bit Windows, the theoretical amount of virtual address space is 264 bytes (16 exabytes), but only a small portion of this range is used.
- Size of pages is 4 KB on x86
- Kernel space is shared among system components

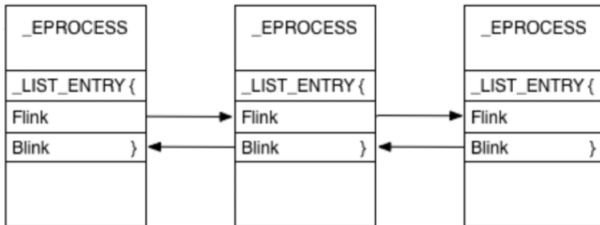


Interpretation of physical memory dumps

- Once a memory dump has been performed, it is necessary to interpret the raw memory contents
- Main approaches to memory reconstruction:
 - Tree and list traversal (data structure traversal)
 - Memparser, KnTTools and KnTList, WMFT
- Object fingerprint / pattern searches
 - PTFinder / PoolFinder
- Both methods (modern tools)
 - Volatility, Mandiant Memoryze

Tree / list traversal basics

- Find index into lists and tables of interesting structure, and follow them through to reconstruct the data



- EPROCESS linked list is a common example, with pointers to:
 - `_ETHREAD` structures
 - Security identifier (SID) of starting user
 - Start time, PID and other metadata in PEB (Process Environment Block)
 - Process virtual memory pages

Fingerprint / pattern searching basics

- Search for relevant patterns in memory
- Scan for sufficiently unique structure signatures
 - e.g., PoolFinder parses kernel pool memory (pre-allocated 4k memory pool pages)
 - e.g., PTFinder works with EPROCESS and ETHREAD structs (`_DISPATCHER_HEADER`)
- Perform basic sanity checks on data to weed out corrupt records, duplicates etc.

Pros and cons

Pros

- List traversal
 - Can stitch together more related records from kernel perspective
- Pattern search
 - Find unlinked, dead structures (warm reboot)
 - Can work with imperfect dumps

Cons

- List traversal
 - Can miss unlinked, dead structures
- Pattern search
 - Less context without following related structures/objects
 - Susceptible to rubbish

Volatility: a memory analysis framework

- Volatility modules (Python)
 - Functionality can be extended w/ plugins
- Implements the methods outlined before
- Framework
 - Suitable for high degree forensic tasks
 - Requires expertise
 - Aimed at academic researchers and professionals

Table A.1 – Integral modules of the volatility memory forensics framework

Module name	Description
connections	Locates the <code>_TCBTable</code> hash table in the <code>tcpip.sys</code> driver file and traverses the singly linked list of <code>_TCPT_OBJECT</code> entries to enumerate open network connections on the target system (see Section 4.4).
connscan2	Scans the non-paged pool of the operating system for allocations that contain information about open network connections (see Section 4.4).
dlllist	Retrieves the base address, size, and path of all dynamically loaded libraries (DLLs) that are referenced by a running application. For this operation, the <code>Ldr</code> structure of the <i>Process Environment Block</i> (PEB) is parsed. It stores three doubly linked lists of <code>_LDR_DATA_TABLE_ENTRY</code> types that hold the respective information (see Section 4.5).
files	Retrieves the list of open file handles that is maintained by a process (see Section 4.5).
getsids	Reconstructs the security context of a process to retrieve the list of user and group SIDs (Security Identifiers) the application is associated with (see Section 4.6).
hivelist	Prints the virtual address and name of hive structures that internally represent parts of the system registry (see Section

Acquisition of volatile memory

- Several tradeoffs when choosing the acquisition technique:
 - Time of installation: prior to incident or post incident
 - Access to system: local or remote
 - Access to main memory: pure hardware vs. software
 - Required privileges: user vs. administrator
 - Impact on system: live vs. post mortem
- Two main factors can be used to help make a decision:
 - **Atomicity**: how close to the present memory state can the forensic memory snapshot be retrieved
 - **Availability**: whether the tools necessary to perform memory acquisition are available or not

Software acquisition methods: User level applications

- Based on user-level applications for memory dumping
 - Acquire copy of physical memory (e.g., avml)
 - Attach to target process or read from OS-provided interface (e.g., /dev/mem, /dev/crash, /proc/kcore)
- **Strengths** of this technique:
 - Good for incident scenarios
 - Capturing forensic image even in situations with little time
- **Weaknesses** of this technique:
 - May only work on specific operating systems
 - Applications must be loaded into memory before execution
 - Depends on functions of the operating system (rootkit may deny access to physical memory or modify the RAM)

Software acquisition levels: Kernel level applications

- Leverage a kernel driver / module to access physical memory without restrictions (e.g., LiME)
- For incidents involving Linux, `fmem` kernel module, e.g., allows for dumping the target's memory
 - To use `fmem`, compile the module and then load it into the running kernel
 - This operation creates a new `/dev/fmem` pseudodevice
 - After loading `fmem`, use `dd` to dump the memory to a file

```
dd if=/dev/fmem of=memory.dd bs=1MB count=512
```

- Downsides: this process causes changes in the system state

Software acquisition levels: Software crash dumps

- Microsoft Windows dumps files to hard disk in case of failure
- Preserves the contents of processor registers
- Dump files can be opened
 - Debugging Tools
 - Manually
- System services may be interrupted
 - Third party application
 - Built-in CrashOnCtrlScroll
- If explicitly triggered, this acquisition is more invasive

Software acquisition methods: Warm and cold boots

- RAM retains memory as long as power is provided
- Warm boots refer to reboot methods in which power is never removed from the memory module (e.g., press reset button)
 - Tools like `msrampdump` or `afterlife` act like minimal OS-es that can save memory to disk
 - When the RAM is cleared by the BIOS, replacing the BIOS can be an option
- Cold boot refers to reboot methods in which power is removed from the memory module (e.g., pull the plug and reboot)
- RAM retains contents for 2-3 seconds
- Retention time can be extended for up to an hour by cooling the memory chip



Software acquisition methods: More methods

- Operating system injection
 - Injects OS into the subverted kernel of target machine
- Hibernation file
 - Windows Hibernation file: `hiberfil.sys`
- Virtual machine imaging
 - If the target is a virtual machine running on a VM monitor (e.g, Xen, QEMU, VirtualBox, etc.), collect memory image by:
 - Pause / stop the system and collect the image into a file (downside the VM is offline), or
 - Live dump a memory image (e.g., QEMU's `pmemsave`)

Hardware acquisition methods: Dedicated hardware card

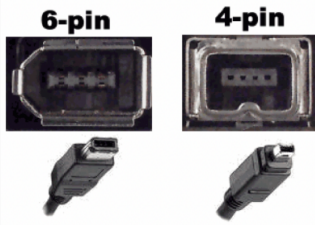
- Use of special hardware card to obtain the forensic image of a computer's RAM
- Uses Direct Memory Access (DMA)
 - Hardware Card is installed as a dedicated PCI device and is capable of saving volatile information
- Limitation: prior installation of PCI card before its use
- Beneficial when installed on critical servers



Hardware acquisition methods: Special hardware bus

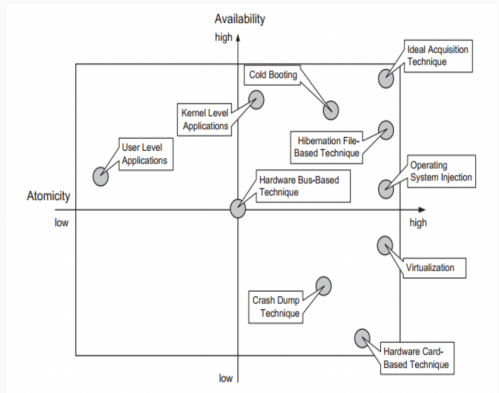
- Alternative to PCI cards, several authors suggest reading volatile memory via IEEE 1394 bus - Firewire
 - Firewire devices have direct access to the computer's memory
 - Rarely used: random system crashes and reliability problems

IEEE 1394(AKA Firewire, i-Link)



Decision Matrix

- Decision matrix helps investigators in choosing a specific memory acquisition technique
- An ideal acquisition method is characterized by both a high atomicity and availability



Takeaways

- Volatile memory contains a wealth of valuable information
- Memory forensics deals with obtaining a memory dump and performing analysis of relevant artifacts therein contained
- There are several memory acquisition techniques that can be adopted offering different tradeoffs between availability and atomicity

- **Textbook:**
 - Casey – Chapter 13.3
 - Luttgens – Chapters 7.5, 7.6, 12.7
- **Acknowledgements:**
 - Slides adapted from Nuno Santos's Forensics Cyber-Security course at Técnico Lisbon