Practical Malware Analysis and Memory Forensics for Incident Response

Ricardo J. Rodríguez

rjrodriguez@unizar.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

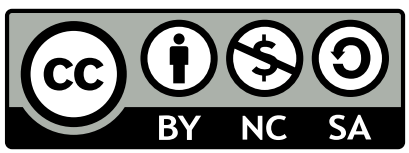
Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

April 1, 2025

DFRWS EU 2025

Brno, Czech Republic



$whoami

- **Associate Professor at the University of Zaragoza**
- **Research lines**:
  - Program binary analysis
  - Digital forensics
  - System security
  - Formal methods applied to cybersecurity
- Speaker and trainer at different infosec conferences (NcN, HackLU, RootedCON, STIC CCN-CERT, HIP, MalCON, HITB...)

Universidad Zaragoza

$whoami

- **Associate Professor at the University of Zaragoza**
- **Research lines**:
  - Program binary analysis
  - Digital forensics
  - System security
  - Formal methods applied to cybersecurity
- Speaker and trainer at different infosec conferences (NcN, HackLU, RootedCON, STIC CCN-CERT, HIP, MalCON, HITB...)
- **Research team** – *we make really good stuff!* 🙂
  - `https://reversea.me` / `https://t.me/reverseame`

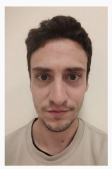
Universidad Zaragoza

~~$whoami~~ $whoarewe

https://reversea.me/index.php/people/

Faculty





Dr. Ricardo J. Rodríguez | Dr. Javier Carrillo-Mondéjar | Dr. José Roldán-Gómez

PostDoc Staff

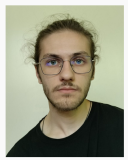
Dr. Daniel Uroz

PhD Students



Razvan Raducu | Tomás Pelayo | Daniel Huici (former MSc. student)

Technical Staff


Daniel Lastanao | León Abascal | Pablo Ruiz


Héctor Toral | Luis Palazón

Administrative Staff

Virginia Giménez

Master & Bachelor Students



Miguel Moniente (MSc student) | Christian Lin Jiang (MSc student) | Alain Villagrasa (BSc student)

Internships


Martina Gracia [July to August 2023] | Zineb Helali [July to August 2024]

Visitors

Ailton S. da Silva (UFAM, Brazil)
[August to November 2018]
[September 2021 to March 2022]

Universidad Zaragoza 1474

Agenda

Universidad Zaragoza

Agenda

Universidad Zaragoza

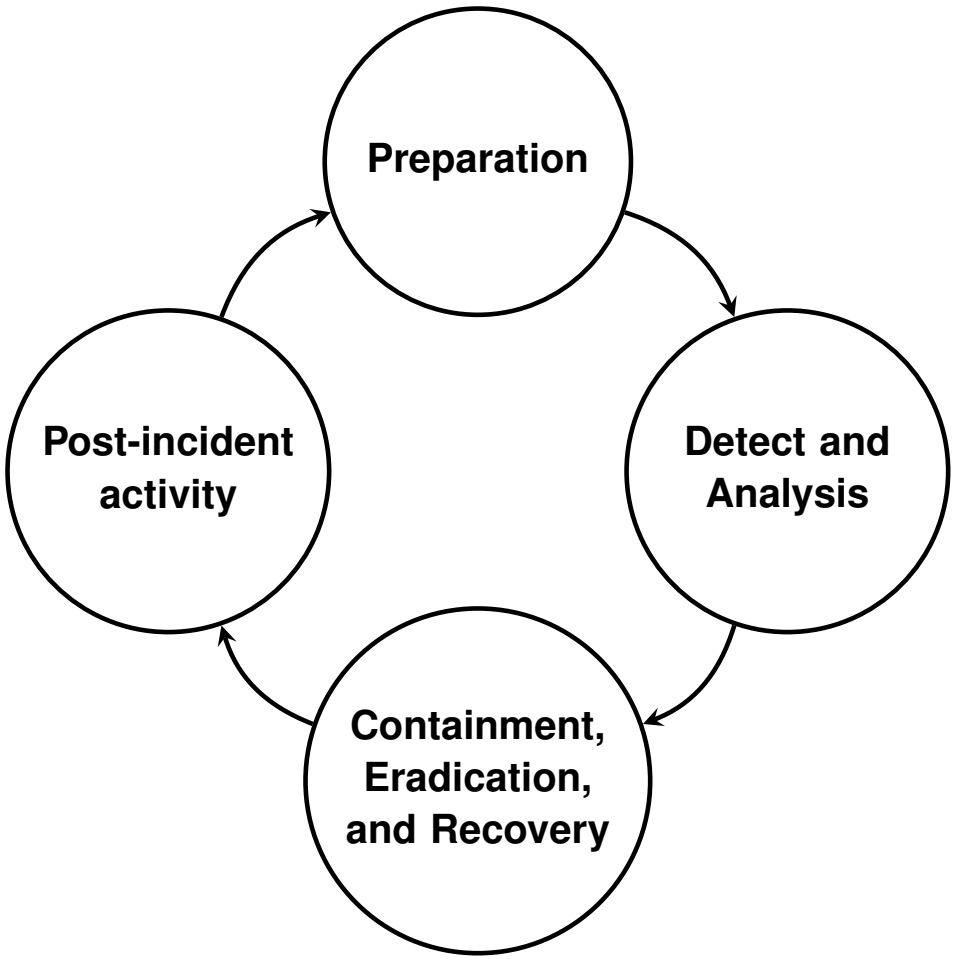
Introduction
Workshop Goals

- Practical intro to the **analysis of memory dumps and malware artifacts**
- Explore both **static *and dynamic* malware analysis techniques**
- Gain an **understanding of memory acquisition best practices**
- Apply **forensics skills in real-world scenarios**
- Understand **malicious behavior**

From Alert → Memory Dump → Analysis → Report

Universidad Zaragoza

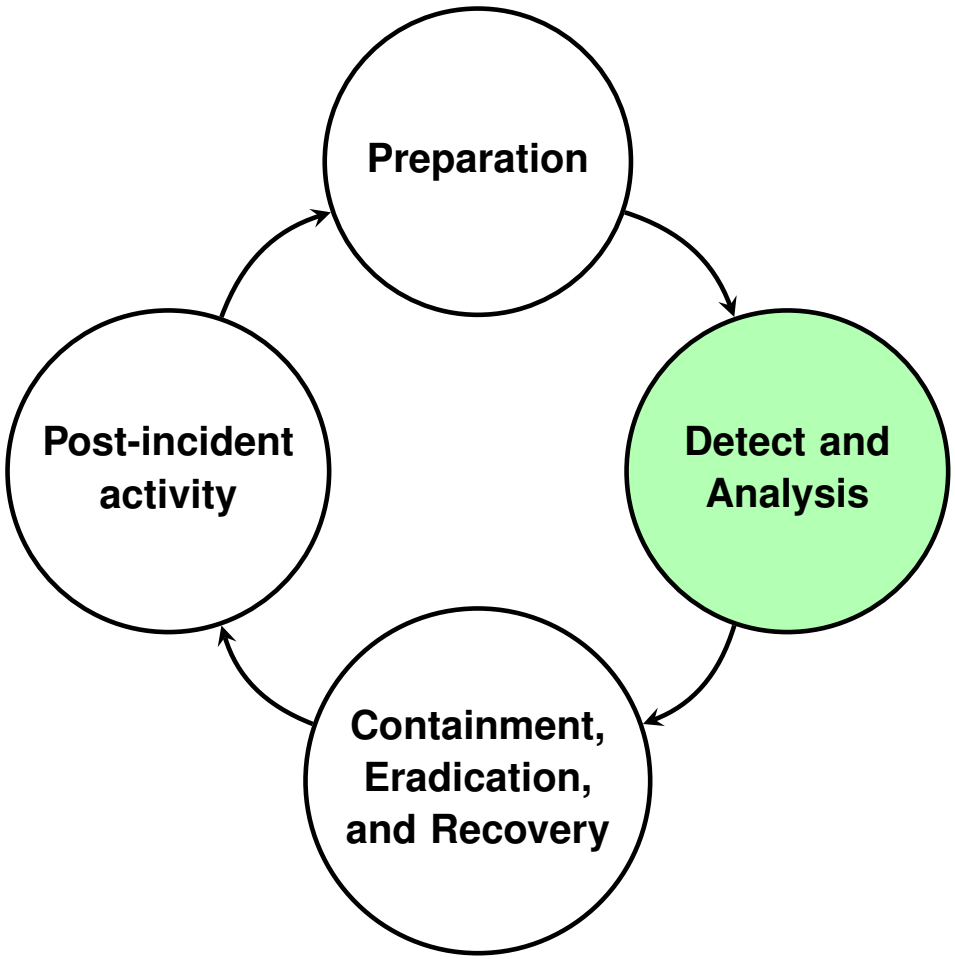
Introduction
Incident Response



Incident response

- **Figure out what happened**, while preserving incident-related data
- **Ask the well-known 6 W's** (what, who, why, how, when, and where)
- Common incident: **presence of malicious software** (malware)

Introduction
Incident Response



(as defined by NIST)

Incident response

- **Figure out what happened**, while preserving incident-related data
- **Ask the well-known 6 W's** (what, who, why, how, when, and where)
- Common incident: **presence of malicious software** (malware)
- **Detect and Analysis phase**:
  - Identify, verify, and analyze potential security incidents
  - Determine their scope, impact, and appropriate response actions
  - *Forensics help here*

Universidad Zaragoza 1474

Introduction

Meet Peter Griffin...

Credits: https://en.wikipedia.org/wiki/File:Peter_Griffin.png

Introduction

Meet Peter Griffin... and alert him!

Credits: https://en.wikipedia.org/wiki/File:Peter_Griffin.png

Introduction

Meet Peter Griffin... and alert him!

Credits: https://en.wikipedia.org/wiki/File:Peter_Griffin.png

Introduction

Meet Peter Griffin... and alert him!

Credits: https://en.wikipedia.org/wiki/File:Peter_Griffin.png

Introduction

Too late. Bye Peter Griffin!

Credits: https://knowyourmeme.com/memes/family-guy-death-pose-peter-falls-down-the-stairs

Introduction

Can we discover what happened to him? → **forensics analysis**

Universidad Zaragoza

Introduction

Can we discover what happened to him? → **forensics analysis**

- **Systems use (physical) memory to run and do their work**
  - *Let Peter Griffin be a system. Then, his brain is* the memory *of the system*
- The memory is always a ***snapshot* of the current system state**
  - Memory can contain indicators of what happened (i.e., *indicators of compromise related to the security incidents*)

Universidad Zaragoza

Introduction

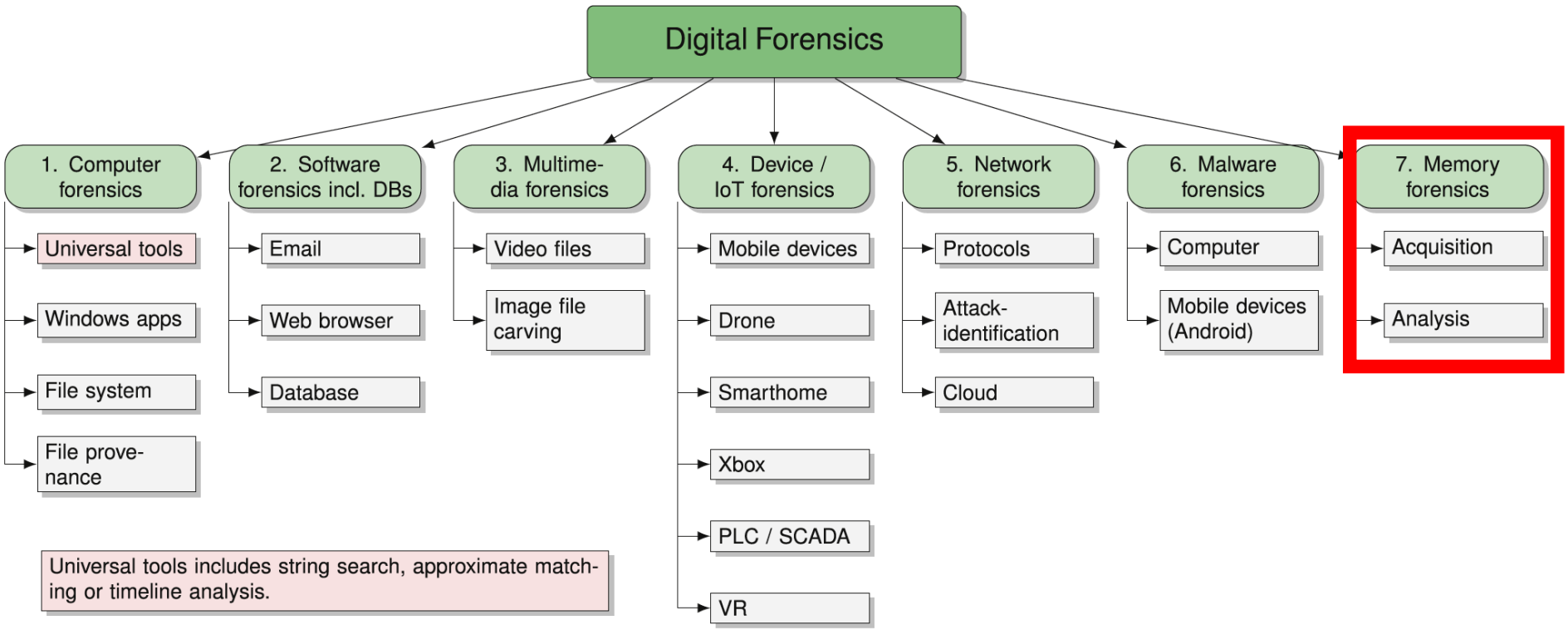
Can we discover what happened to him? → **forensics analysis**

- **Systems use (physical) memory to run and do their work**
  - *Let Peter Griffin be a system. Then, his brain is* the memory *of the system*
- The memory is always a ***snapshot* of the current system state**
  - Memory can contain indicators of what happened (i.e., *indicators of compromise related to the security incidents*)

In memory forensics,
memory becomes the *victim* to analyze

Universidad Zaragoza

Introduction



Credits: *Digital forensic tools: Recent advances and enhancing the status quo*. Wu et al., doi: 10.1016/j.fsidi.2020.300999

Universidad Zaragoza

Introduction

Memory forensics – some terminology

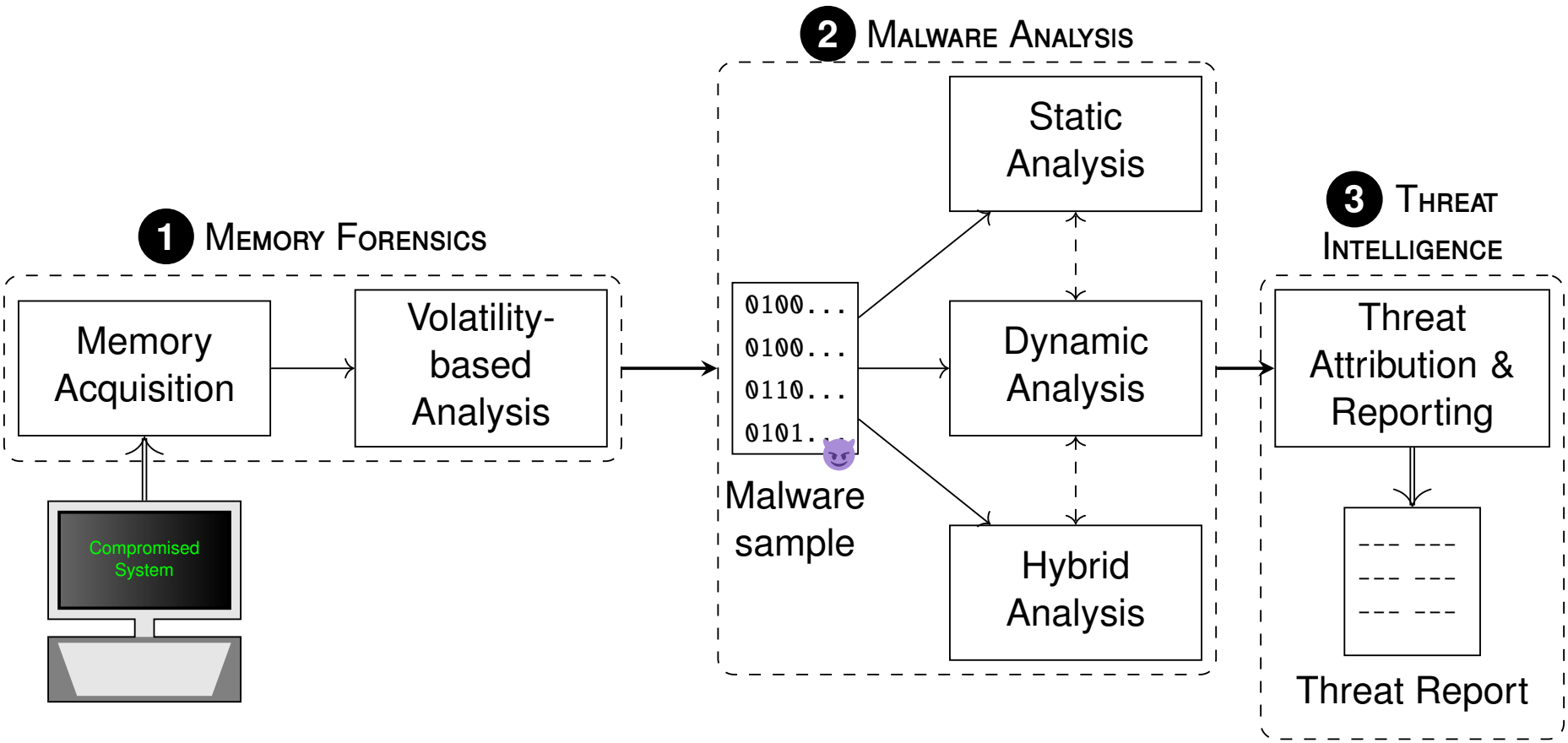
Memory dump

- **Full of data** to analyze
- **Each item to analyze is called *memory* artifact (or simply *artifact*)**
  - Retrieved via appropriate internal structures of the OS or using a pattern-like search
- Snapshot of running processes, logged in users, open files, or open network connections – **everything that was running at the time of acquisition**
  - Potential malware code (and in different forms: injected, unpacked, fileless...)
- May also contain **recently freed system resources**
  - Normally, memory is not zeroed when freed
- `Volatility`: **de facto standard** tool for analyzing memory dumps
  - Version 2 vs. version 3 ⇒ Python2[1] vs. Python3
  - *We will talk about Volatility 2 and Volatility 3 in a few slides...*

[1] I know, it is deprecated but Vol2 is certainly necessary on some analysis scenarios...

Introduction

End-to-End Analysis Workflow



Introduction

Lab Roadmap

Lab 1.- ***Getting Started with Volatility***

- Introduce memory forensics concepts and Volatility
- *Learning goals*:
  - List running processes and understand process hierarchies
  - Identify useful plugins and interpret their output

Lab 2.- ***Memory Dump Analysis with Volatility 3***

- Perform deeper analysis on a memory dump from a WannaCry-infected system
- *Learning goals*:
  - Locate malware-related processes and extract binaries
  - Identify registry keys, DLLs, and persistence mechanisms
  - Analyze malware execution patterns in memory

Lab 3.- ***Practical Malware Analysis: From Memory Forensics to Threat Attribution***

- Apply static analysis and triage techniques to extracted binaries, and connect findings to threat intelligence
- *Learning goals*:
  - Perform basic static triage: hash, strings, import analysis
  - Identify behavioral indicators and map to MITRE ATT&CK
  - Create a basic YARA rule and outline a threat report

Universidad Zaragoza 1474

What do we need for workshop labs?

1. Install Docker (and, optionally, Docker Desktop) on your host
2. Get the Dockerfile from the workshop website
3. Deploy it and test the SSH connection: `ssh forensic@localhost -p 2222`
   - The password is the same as the username: `forensic`

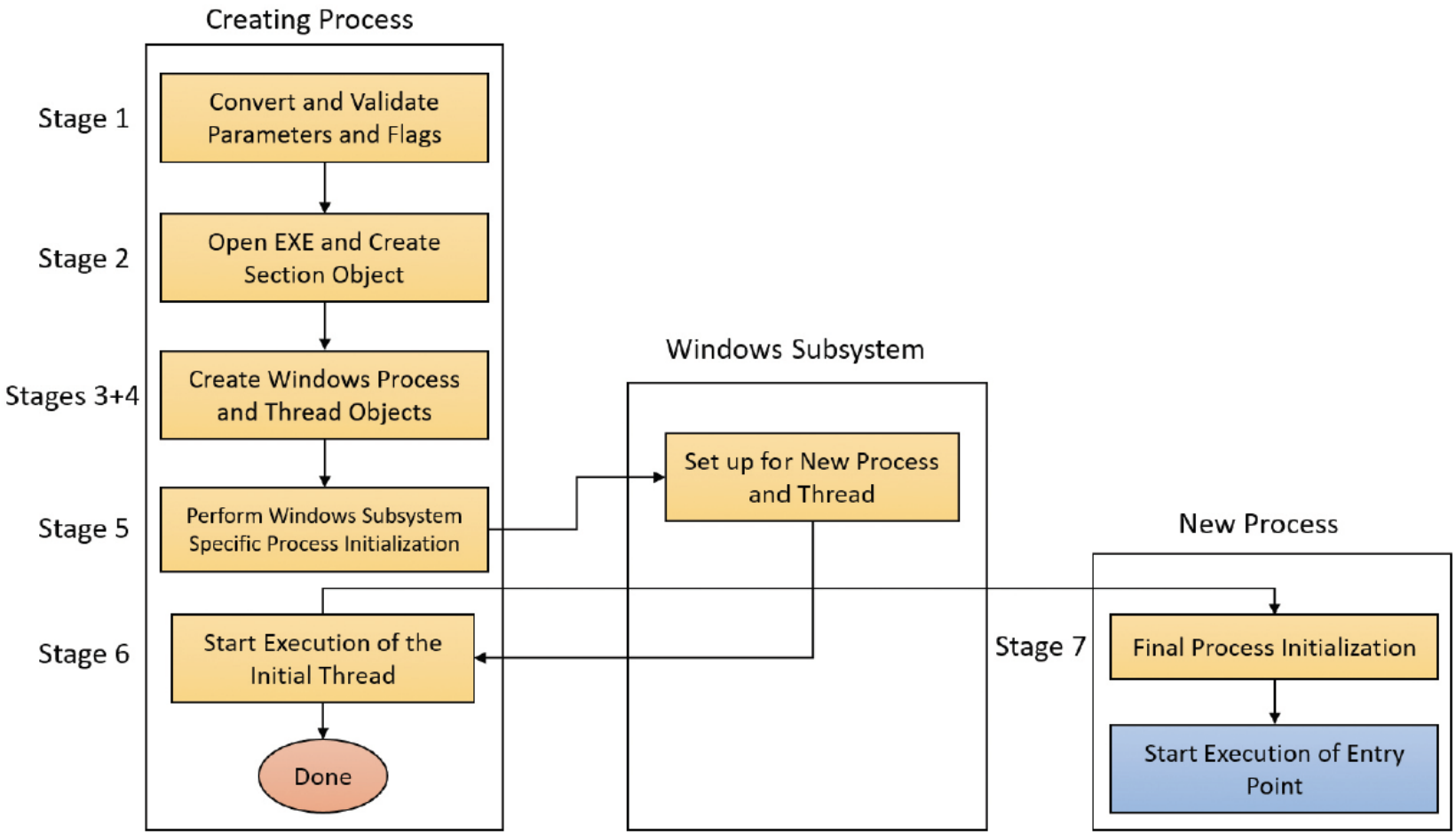
Universidad Zaragoza

Agenda

Universidad Zaragoza

Background

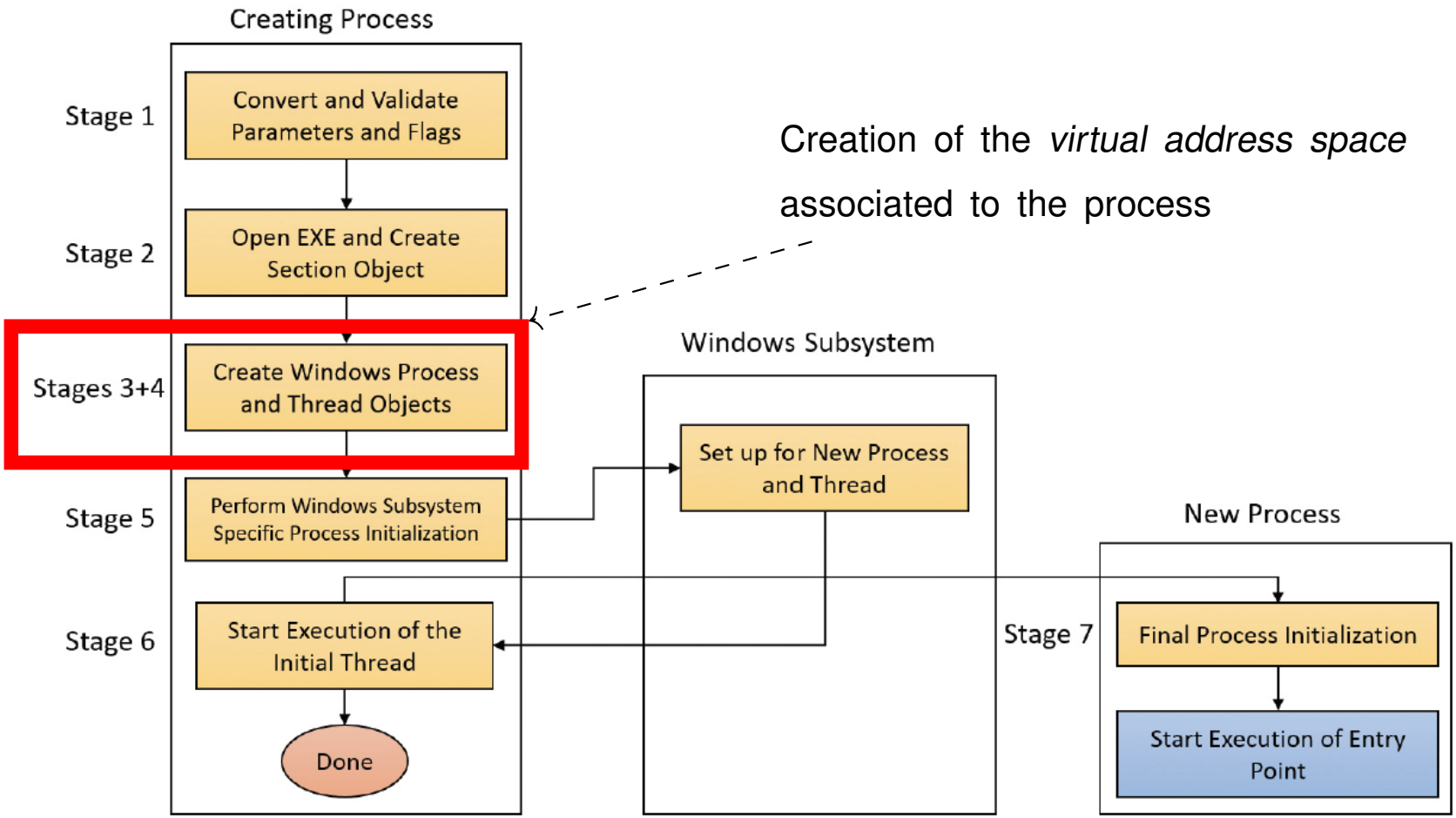
Creation of Windows Processes



Credits: *Windows Internals, 7th Ed., Part 1*. P. Yosifovich et al. Microsoft Press, ISBN 978-0735684188

Background

Creation of Windows Processes



Credits: *Windows Internals, 7th Ed., Part 1*. P. Yosifovich et al. Microsoft Press, ISBN 978-0735684188

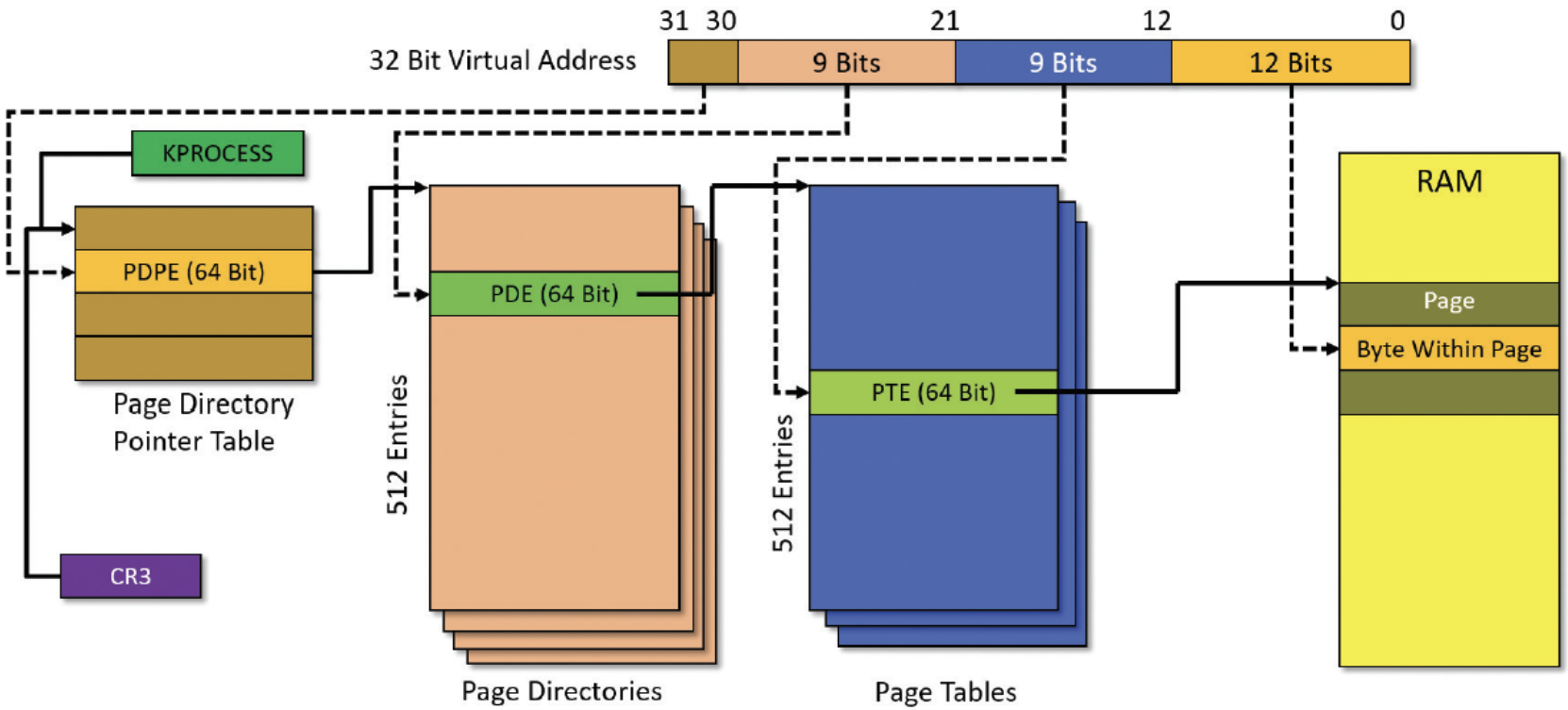
Background
Virtual Address Space

- **Per-process private data and code**
- Stored in kernel-mode-only accessible pages
  - Prevents user-mode threads from modifying their own address space layout
- **Default virtual size of 32-bit Windows processes**: 2 GiB (before Win8)
  - Can be extended to 3 GiB (or 4 GiB on 64-bit Windows) if the program file is specifically marked as a large address space and the system starts up with a special option
  - On 64-bit Windows 8.1 (and later): 128TiB (although the maximum amount of physical memory currently supported by Windows is less than 24 TiB ¯_(ツ)_/¯)

Universidad Zaragoza

Background

Virtual Memory Translation (in x86)



Credits: *Windows Internals, 7th Ed., Part 1*. P. Yosifovich et al. Microsoft Press, ISBN 978-0735684188

Background

More Terminology

Page

- **Contiguous fixed-length block of memory**
- Small (4 KiB) and large pages (2 MiB [x86 & x64] to 4 MiB [ARM])
- *Page frame*: how physical memory (RAM) is divided
- In a *paged system*, virtual memory is divided into *virtual pages* of the same size as the page frames

Universidad Zaragoza

Background

More Terminology

Page

- **Contiguous fixed-length block of memory**
- Small (4 KiB) and large pages (2 MiB [x86 & x64] to 4 MiB [ARM])
- *Page frame*: how physical memory (RAM) is divided
- In a *paged system*, virtual memory is divided into *virtual pages* of the same size as the page frames

Page Table Entry (PTE)

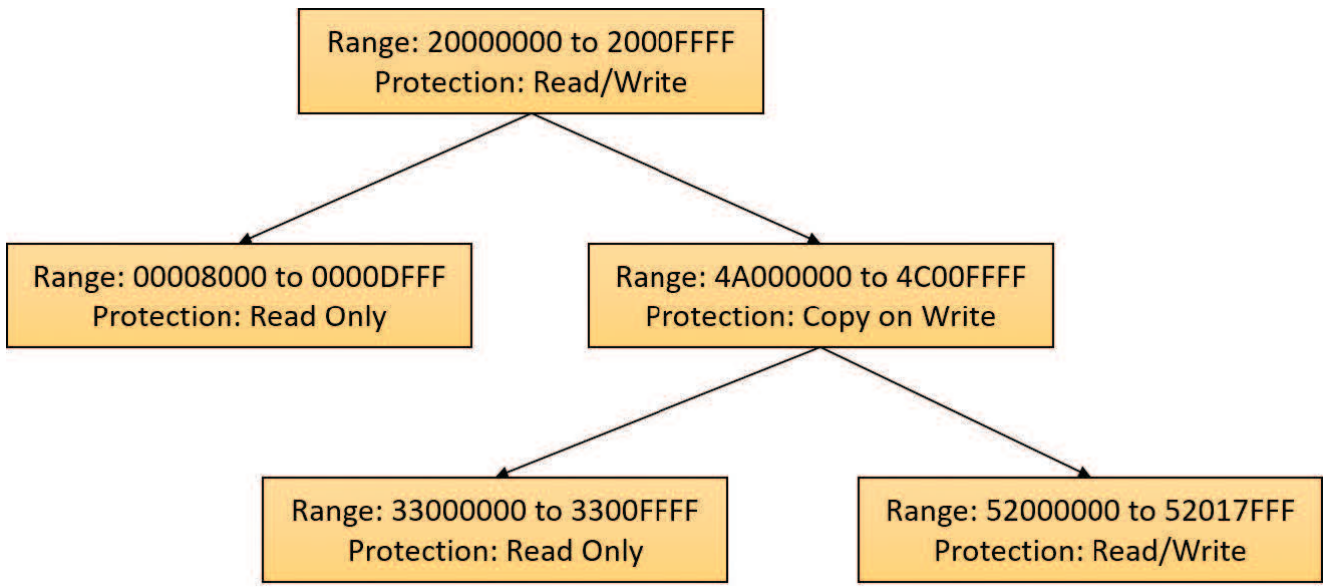
- **Maps each virtual page with its corresponding physical address**
- Stored in *page tables*
  - The set of page tables of a process defines its virtual address space
  - **512 entries**
- Every entry value is also called *page frame number* (PFN)

Universidad Zaragoza

Background
More Terminology

Virtual Address Descriptors (VADs)

- **Keep track of which virtual addresses have been reserved in the process's address space (and which have not)**
- Stored in a self-balancing AVL tree



Credits: *Windows Internals, 7th Ed., Part 1*. P. Yosifovich et al. Microsoft Press, ISBN 978-0735684188

Agenda

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Memory Acquisition

- Various acquisition techniques
  - **Recommended Reading**: Tobias Latzo, Ralph Palutke, Felix Freiling, "*A universal taxonomy and survey of forensic memory acquisition techniques,*" Digital Investigation, Volume 28, 2019, pp. 56–69, ISSN 1742-2876, doi: 10.1016/j.diin.2019.01.001
- Software tools for full memory dumping
  - `WinPmem`: https://github.com/Velocidex/WinPmem
    - Apache License
    - Supports Windows XP to Windows 10 (32/64-bit)
    - Example: `winpmem_mini_x64.exe physmem.raw`
  - `Linux Memory Extractor` (LiME): https://github.com/504ensicsLabs/LiME
    - GNU/GPLv2 License
    - Supports Linux and Android
    - Extraction via local port connection
  - `FTK Imager`: https://accessdata.com/product-download/ftk-imager-version-4-2-1
    - Commercial tool
    - Windows support

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Memory Acquisition

- Acquisition in virtual machines
  - `VirtualBox`
    - `vboxmanage debugvm "Win7" dumpvmcore --filename test.elf`
  - `VMWare`
    - Create a VM snapshot (generates `.vmss` and `.vmem` files)
    - Tool `vmss2core`: https://archive.org/download/flings.vmware.com/Flings/Vmss2core
- Other tools for extracting processes or modules
  - `ProcDump`: https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
    - `procdump -ma 4572`
    - Single dump (`.dmp` file)
  - `Windows Memory Extractor`: https://github.com/pedrofdez26/windows-memory-extractor
    - GNU/GPLv3 License
    - `WindowsMemoryExtractor_x64.exe --pid 1234`
    - Creates section-based process memory dumps

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Memory Dump Analysis

`Volatility`

- De facto standard for memory dump analysis
- Open source license GNU/GPLv2
- Released in 2007 at BH USA, called `Volatools`
- Supports Windows, Linux, and macOS (32/64-bit)
- Rich API for custom implementations
- Version 2.6 vs. Version 3
  - Python2 vs. Python3
  - Version 3 is now stable

`https://github.com/volatilityfoundation/volatility3`

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Getting Started with Volatility

- Help: `python vol.py -h`
- Memory dump to analyze: `python vol.py --f mem.dmp --profile Win7SP1x86`
  - The profile is only needed in version 2.6
  - Indicates where the OS internal structures are located
- *How to determine the correct profile?*: use the `imageinfo` plugin
  `python vol.py --f mem.dmp imageinfo`
  - Plugins are always specified at the end of the command

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Analysis – Official Plugins (Volatility 2)

- **Processes and DLLs**
  - `pslist`, `pstree` (`psscan` for potential rootkits)
  - `dlllist`, `dlldump`
  - `handles`
  - `enumfuncs` (list of imported/exported functions per process/DLL)
- **Process Memory**
  - `memmap`, `memdump`
  - `procdump`
  - `vadinfo`, `vadwalk`, `vadtree`, `vaddump`
  - `evtlogs`
  - `iehistory`
- **Networking**
  - `connections`, `connscan`
  - `sockets`, `sockscan`
  - `netscan` (network artifacts in Win7)

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Analysis – Official Plugins (Volatility 2)

- **Kernel Memory and Other Objects**
  - `modules`, `modscan`, `moddump`
  - `driverscan`
  - `filescan`
- **Registry**
  - `hivescan`, `hivelist`, `hivedump`
  - `printkey`
  - `lsadump`
  - `userassist`, `shellbags`, `shimcache`
  - `dumpregistry`
- **File System**
  - `mbrparser`, `mftparser`
- **Analysis of hibernation files or other dump types**

Universidad Zaragoza

Memory Acquisition & Forensics with Volatility

Analysis – Unofficial Plugins

- Many additional plugins extend Volatility's capabilities
- How to use:
  1. Install the plugin (e.g., clone from a repository)
  2. Execute with:
     `volatility --plugins="/path/to/plugin" -f file [OPTIONS] pluginname`

Memory Acquisition & Forensics with Volatility

Volatility 2 vs. Volatility 3

Summary table		
Feature	**Volatility 2**	**Volatility 3**
Language	Python 2	Python 3
Plugin Structure	Flat, less modular	Fully modular and object-oriented
Profiling System	Requires OS profiling	No profiling (auto-detects kernel symbols)
Cross-platform	Limited	Improved Linux/macOS support
Performance	Slower, single-threaded	More efficient, supports streaming
Extensibility	More difficult to extend	Easier to create custom plugins
Development Status	Legacy/Frozen	Active maintenance

Command syntax comparison		
Task	**Volatility 2 Syntax**	**Volatility 3 Syntax**
Specify Memory Dump	`-f mem.raw --profile Win7SP1x86`	`-f mem.raw` (no profile)
List Processes	`vol2.py -f mem.raw pslist`	`vol3.py -f mem.raw windows.pslist.PsList`
Process Dump	`procdump -p <PID>`	`windows.pslist.PsList --pid <PID> --dump`
View Loaded DLLs	`dlllist -p <PID>`	`windows.dlllist.DllList --pid <PID>`

Volatility 3 uses full plugin paths (`namespace.plugin.ClassName`); tab completion helps!

Universidad Zaragoza

Laboratory Session

Lab 1: Getting Started with Volatility

Goals

- Explore the Volatility framework
- Become familiar with its basic commands

Steps

1. Download the `zeus.vmem` memory dump from the workshop webpage
2. Explore initial plugins such as `imageinfo`, `pslist`, etcetera

Tip:
Focus on understanding the structure and output format of each command

Universidad Zaragoza

Laboratory Session

Lab 1: Getting Started with Volatility

Key takeaways

- Volatility is a powerful framework for memory forensics, supporting modular plugin development
- You learned how to execute basic plugins to extract system and process-level information from a memory dump
- The memory dump structure reveals valuable artifacts such as process lists, loaded DLLs, and system configuration
- Volatility's plugin output can be redirected, filtered, or piped for automation and further analysis.
- Understanding the basics of memory layout and plugin outputs is necessary for deeper malware investigation in later labs

Universidad Zaragoza

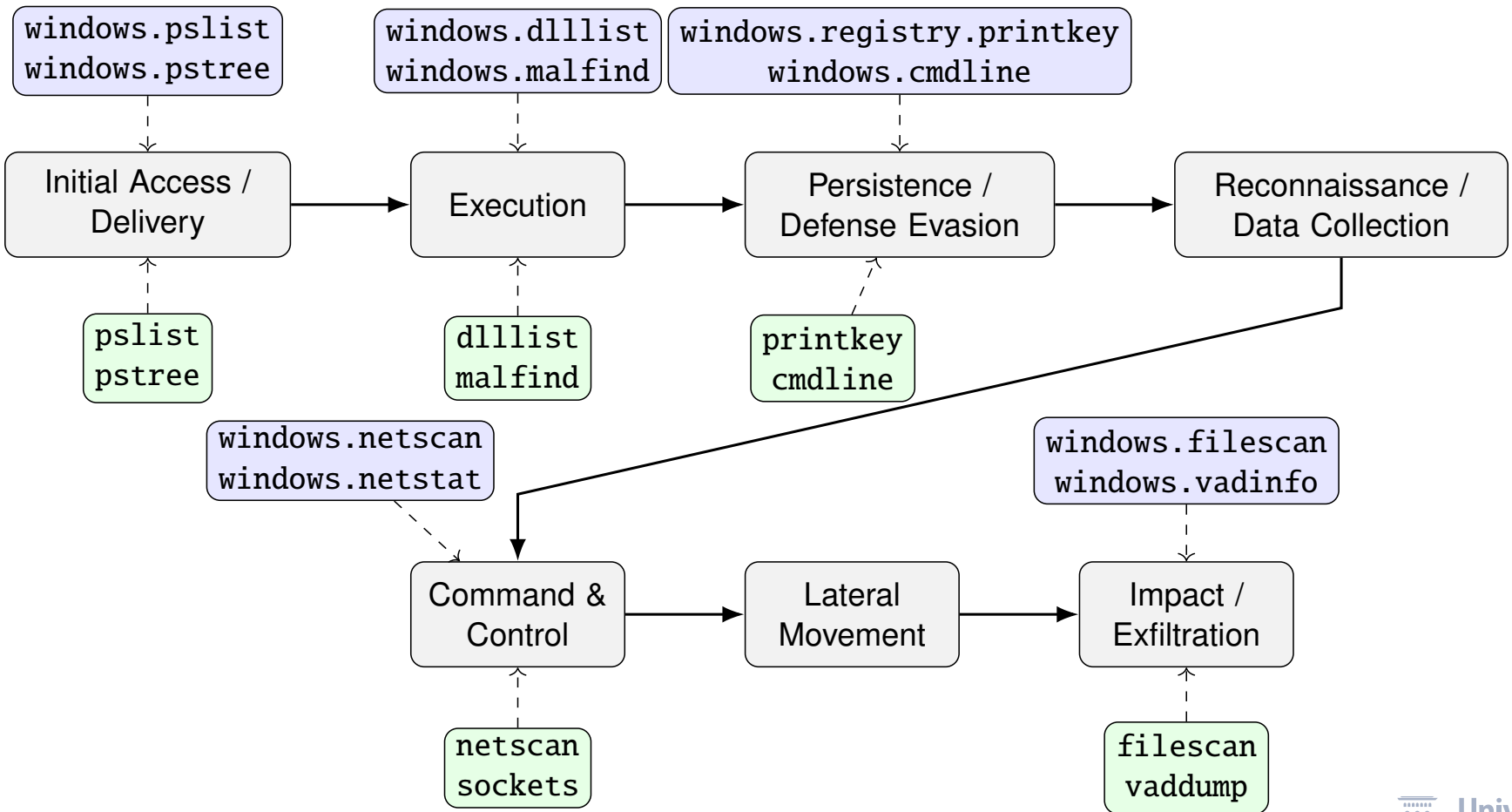
Agenda

Universidad Zaragoza

Analyzing Malware Artifacts in Memory

Malware attack lifecycle

- Typical stages that malware follows during an intrusion



What might we find in memory after a malware infection?

Take a moment to think:

If a system is infected with malware, what kinds of traces or artifacts might we be able to find in a memory dump?

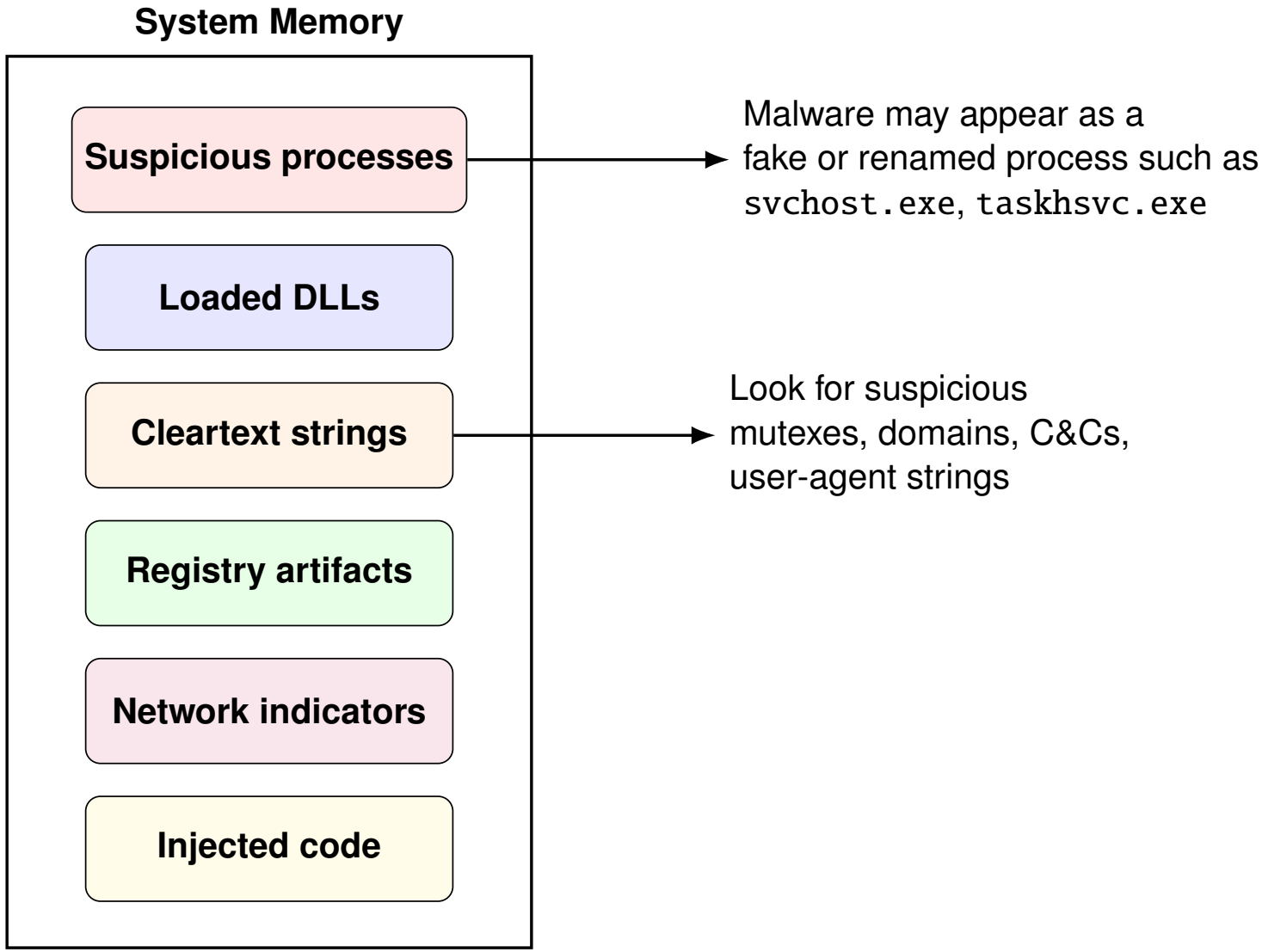
What might we find in memory after a malware infection?

Take a moment to think:
If a system is infected with malware, what kinds of traces or artifacts might we be able to find in a memory dump?

- **Specific behaviors**
- **Artifacts** (e.g., files, processes, keys, etc.)
- **System interactions**

Analyzing Malware Artifacts in Memory

Key Artifacts in Memory After Malware Execution



These indicators can later be mapped to ATT&CK techniques for attribution

Analyzing Malware Artifacts in Memory

Memory Forensics – Unofficial Plugins

MalConfScan (https://github.com/JPCERTCC/MalConfScan)

- Extracts configurations, decrypted strings, or DGA domains from known malware families

Malscan (https://github.com/reverseame/malscan; for Volatility 2.6)

- GNU/GPLv3 License
- Combines `malfind` + `clamav-daemon` (Linux only). Fewer false negatives
- Modes: *normal* (regions +WX, executable modules, private VadS) and *full-scan* (+X regions)
- Detects VADs without associated executables, function prologues, and empty pages before code

```
➜  ~ python2 volatility/vol.py --plugins=/home/alumno/malscan -f /mnt/volcados/alina1G.elf
--profile=Win7SP1x86 malscan
Volatility Foundation Volatility Framework 2.6.1

Process: ALINA_CJLXYJ.e Pid: 1828 Space Address: 0xc20000-0xc44fff
Vad Tag: Vad  Protection: PAGE_EXECUTE_WRITECOPY
Flags: CommitCharge: 5, Protection: 7, VadType: 2
Scan result: Win.Trojan.Alina-4

0x00c20000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00   MZ..............
0x00c20010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ........@.......
0x00c20020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0x00c20030  00 00 00 00 00 00 00 00 00 00 00 00 f0 00 00 00   ................
```

Universidad Zaragoza

Analyzing Malware Artifacts in Memory

Memory Forensics – Unofficial Plugins

Similarity Unrelocated Module

(https://github.com/reverseame/similarity-unrelocated-module; for Volatility 2.6)

- GNU/GPLv3 License
- Computes approximate signatures of modules found in a memory dump
- SDA algorithms: `dcfldd`, `ssdeep`, `sdhash`, `TLSH`
- A *module* is an executable or DLL loaded in memory
- Allows comparison of modules across different dumps
- Reverses OS relocations (relocation undoing) using:
  - Guided De-relocation
  - Linear Sweep De-relocation
- *More info*: M. Martín-Pérez, R. J. Rodríguez, D. Balzarotti, "*Pre-processing Memory Dumps to Improve Similarity Score of Windows Modules*," Computers & Security, vol. 101, p. 102119, 2021, doi: 10.1016/j.cose.2020.102119

Universidad Zaragoza

Analyzing Malware Artifacts in Memory

Memory Forensics – Unofficial Plugins

Winesap (https://github.com/reverseame/winesap; for Volatility 2.6)

- AGPLv3 License
- Searches for all Windows auto-start locations in the memory dump
- Binary or unknown registry keys are treated as PE files
- *More info*: D. Uroz, R. J. Rodríguez, "*Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics*," Digital Investigation, vol. 28, pp. S95–S104, 2019, doi: 10.1016/j.diin.2019.01.026

Universidad Zaragoza

Analyzing Malware Artifacts in Memory

Memory Forensics – Unofficial Plugins

Winesap

Windows Auto-Start Extensibility Points	*Characteristics*					
	Write permissions	**Execution privileges**	**Tracked down in memory forensics†**	**Freshness of system**	**Execution scope**	**Configuration scope**
System persistence mechanisms						
Run keys (HKLM root key)	yes	user	yes	user session	application	system
Run keys (HKCU root key)	no	user	yes	user session	application	user
Startup folder (%ALLUSERSPROFILE%)	yes	user	no	user session	application	system
Startup folder (%APPDATA%)	no	user	no	user session	application	user
Scheduled tasks	yes	any	no	not needed‡	application	system
Services	yes	system	yes	not needed‡	application	system
Program loader abuse						
Image File Execution Options	yes	user	yes	not needed	application	system
Extension hijacking (HKLM root key)	yes	user	yes	not needed	application	system
Extension hijacking (HKCU root key)	no	user	yes	not needed	application	user
Shortcut manipulation	no	user	no	not needed	application	user
COM hijacking (HKLM root key)	yes	any	yes	not needed	system	system
COM hijacking (HKCU root key)	no	user	yes	not needed	system	user
Shim databases	yes	any	yes	not needed	application	system
Application abuse						
Trojanized system binaries	yes	any	no	not needed	system	system
Office add-ins	yes	user	yes	not needed	application	user
Browser helper objects	yes	user	yes	not needed	application	system
System behavior abuse						
Winlogon	yes	user	yes	user session	application	system
DLL hijacking	yes	any	no	not needed	system	system
AppInit DLLs	yes	any	yes	not needed	system	system
Active setup (HKML root key)	yes	user	yes	user session	application	system
Active setup (HKCU root key)	no	user	yes	user session	application	application

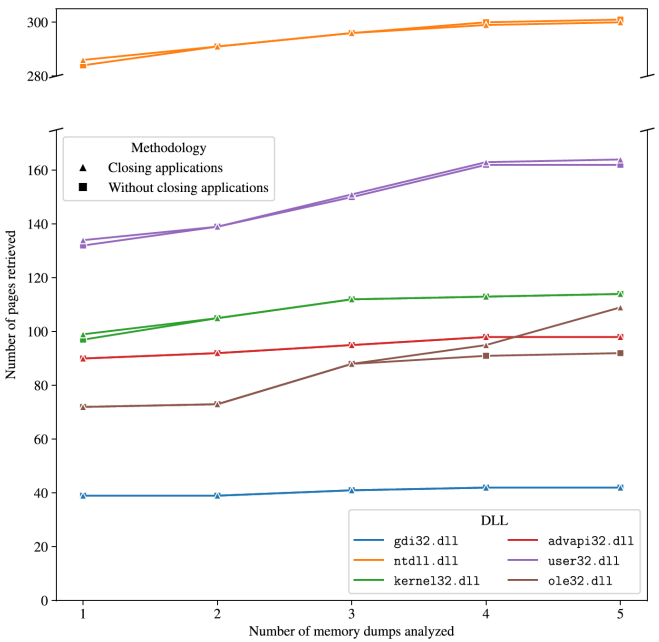
†*If the memory is paging to disk, it would be not possible to track down these ASEPs in memory forensics.*

Analyzing Malware Artifacts in Memory

Memory Forensics – Unofficial Plugins

Modex / Intermodex (https://github.com/reverseame/modex; for Volatility 3)

- GNU/GPLv3 License
- Enables more complete extraction of modules (from one or more dumps)
- Remark: each process only holds the memory pages it uses from a module
- Capable of detecting DLL hijacking attacks



- *More info*: P. Fernández-Álvarez, R. J. Rodríguez, "*Module Extraction and DLL Hijacking Detection via Single or Multiple Memory Dumps*," Forensic Science International: Digital Investigation, vol. 44, pp. 301505, 2023, doi: 10.1016/j.fsidi.2023.301505

Universidad Zaragoza 1474

Laboratory Session

Lab 2: Memory Dump Analysis with Volatility 3

Goal

- Analyze a real-world memory dump of a Windows system infected with WannaCry ransomware using Volatility 3

Steps

1. Download the `wannacry.elf` memory dump from the workshop webpage
2. Identify suspicious processes and malware indicators
3. Examine loaded DLLs and handles to discover artifacts such as Tor components or mutexes
4. Dump binaries and DLLs from memory for offline analysis

Tip:
Document each step and justify how you identified malware-related activity

Laboratory Session

Lab 2: Memory Dump Analysis with Volatility 3

Key takeaways

- Memory analysis reveals runtime artifacts that may not exist on disk (processes, handles, DLLs, mutexes, etcétera)
- You identified the presence of WannaCry through suspicious process names and hierarchy (`@WanaDecryptor`, `taskhsvc.exe`)
- The `dlllist` and `handles` plugins exposed key indicators such as embedded Tor libraries and mutexes used for anti-reinfection
- Dumping suspicious binaries from memory allows follow-up with static and dynamic analysis (next lab)
- Volatility 3 provides a plugin-based approach for forensic investigation, letting you extract specific evidence to support threat detection and reporting

Universidad Zaragoza 1474

Agenda

Universidad Zaragoza

From Malware to Attribution

What's Next?

In the last lab, you identified suspicious processes and extracted binaries from a real infected system

From Malware to Attribution

What's Next?

In the last lab, you identified suspicious processes and extracted binaries from a real infected system

Now the question is:

- *What do these binaries do?*
- *How do they persist or communicate?*
- *Can we classify or attribute them?*

In this last part: We move from forensic analysis of alerts to malware analysis and threat attribution of alerts

Universidad Zaragoza

From Malware to Attribution

Triage vs. Analysis vs. Attribution

Phase	Goal	Outcome
Triage	Rapid risk assessment	Suspicious binary, hash, IoCs
Static/Dynamic Analysis	Behavioral understanding	Registry keys, API calls, persistence method, ...
Attribution	Link to known threats or actors TTP	MITRE techniques, campaign family, YARA rules

The three stages build on each other, based on what the memory reveals

Universidad Zaragoza

From Malware to Attribution

What makes a binary suspicious?

Flags to look out for

- Unusual or obfuscated process names (e.g., `taskhsvc.exe`)
- Hardcoded URLs, IPs, or domains
- Registry manipulation APIs (`RegSetValueEx`, etc.)
- Networking functions (`WinHttpOpen`, `connect`)
- Unsigned or altered PE headers
- Suspicious compile timestamps
- Memory leaks or mutex creation

Universidad Zaragoza

From Malware to Attribution

MITRE ATT&CK & Threat Attribution

MITRE ATT&CK is a curated knowledge base of adversary behavior, based on real-world observations

Why use MITRE ATT&CK?

- Provides a standardized way to describe adversary behavior
- Helps map observable activity to known techniques
- Facilitates threat hunting, detection engineering, and reporting

Examples:

- Using `RegSetValueA` → `T1112` (Modify Registry)
- Command line persistence → `T1059` (Command and Scripting Interpreter)
- Mutex creation → `T1497.001` (Virtualization/Sandbox Evasion: System Checks)

Universidad Zaragoza 1474

From Malware to Attribution

MITRE ATT&CK – Stages of an Intrusion

- **Reconnaissance**
- **Resource Development**
- **Initial Access**
- **Execution**
- **Persistence**
- **Privilege Escalation**
- **Defense Evasion**
- **Credential Access**
- **Discovery**
- **Lateral Movement**
- **Collection**
- **Command and Control**
- **Exfiltration**
- **Impact**

Universidad Zaragoza

From Malware to Attribution

Understanding MITRE ATT&CK and TTP

TTPs = Tactics + Techniques + Procedures

- **Tactic:** The adversary's *goal* (e.g., *persistence*, *credential access*)
- **Technique:** The *meaning* of achieving that goal (e.g., *modifying the registry*)
- **Procedure:** The *specific implementation* (e.g., *using* `RegSetValueA` *to change an execution key*)

Universidad Zaragoza

From Malware to Attribution

Understanding MITRE ATT&CK and TTP

TTPs = Tactics + Techniques + Procedures

- **Tactic:** The adversary's *goal* (e.g., *persistence*, *credential access*)
- **Technique:** The *meaning* of achieving that goal (e.g., *modifying the registry*)
- **Procedure:** The *specific implementation* (e.g., *using* `RegSetValueA` *to change an execution key*)

Why it's important in malware analysis:

- Helps classify behavior in a structured and repeatable way
- Links findings to known threat actors and campaigns
- Facilitates defensive mapping (e.g., detection coverage, SIEM correlation)

Universidad Zaragoza

From Malware to Attribution

Malware Analysis Phases

- **Static analysis**
  - *Examine the binary without executing it*
  - Hashes, strings, imports, PE structure
  - **Main goal**: Reveal potential behaviors
- **Dynamic analysis**
  - Execute the binary in a controlled environment (e.g., sandbox)
  - OS (file/registry/process) + external (network) interaction
  - **Main goal**: Observe real interactions
- **Hybrid analysis**
  - Combine both approaches
  - **Main goal**: Improve coverage and correlation

Universidad Zaragoza

From Malware to Attribution

Malware Analysis Phases

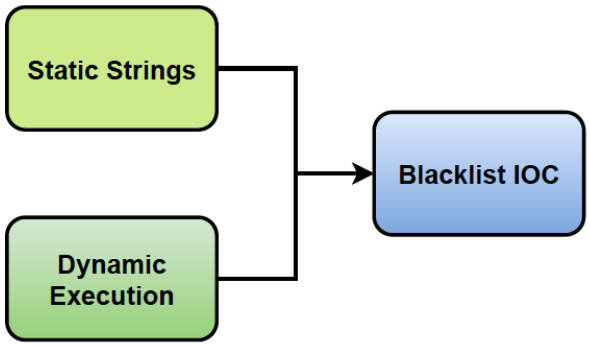
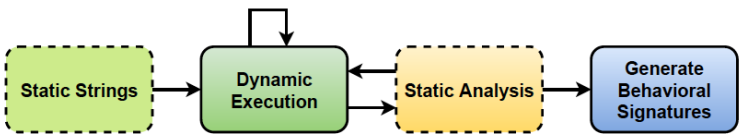
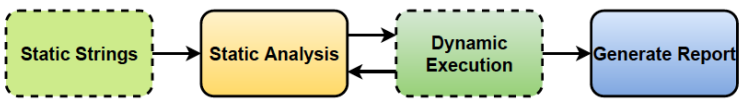


Figure 2: Tier 1 Participant Workflow; P3

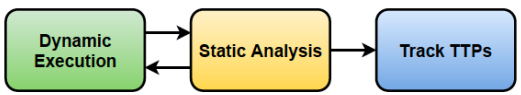


(a) Tier 2A Workflow; P1, P2, P8, P10, P11, P15, P16, P17

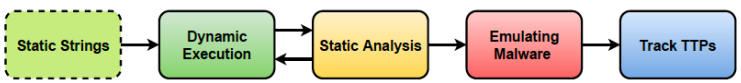


(b) Tier 2B Workflow; P4, P6, P20

Figure 3: Tier 2 Participant Workflows. The dashed boxes represent optional steps in the workflow process.



(a) 3A workflow; P9, P12, P13



(b) 3B workflow; P7, P14, P18, P19

Figure 4: Tier 3 Participant Workflows. The dashed boxes represent optional steps in the workflow process.

Credits: *An Inside Look into the Practice of Malware Analysis*. Yong Wong et al., doi: 10.1145/3460120.34847599

Universidad Zaragoza

From Malware to Attribution

Memory Forensics – Unofficial Plugins

Sigcheck (https://github.com/reverseame/sigcheck; for Volatility 2.6)

- Useful for initial triage
- GNU/GPLv3 License
- Verifies PE files digitally signed with Microsoft Authenticode
- Two types of signatures: embedded (in PE), catalog-based (external file)
- **IMPORTANT**: verifies if the original executable was legit
  - If malware performs process hollowing, this won't detect it
- *More info*: D. Uroz, R. J. Rodríguez, "*On Challenges in Verifying Trusted Executable Files in Memory Forensics*," Forensic Science Int. Digital Investigation, vol. 32, p. 300917, 2020, doi: 10.1016/j.fsidi.2020.300917

Universidad Zaragoza

Laboratory Session

Lab 3: Practical Malware Analysis – From Memory Forensics to Threat Attribution

Goal

- Perform basic malware analysis on binaries extracted from memory to understand their behavior and support threat attribution

Steps

1. Extract binaries from a compromised system (e.g., WannaCry or ALINA)
2. Apply static analysis techniques: hash generation, string extraction, PE header inspection, and API import review
3. Identify behavioral indicators such as mutex names, embedded URLs, registry usage, or memory scraping patterns
4. Map findings to MITRE ATT&CK techniques to assist in attribution

Tip:
Treat this as a real-world triage scenario – build your analyst mindset and document your findings like a threat report

Universidad Zaragoza 1474

Laboratory Session

Lab 3: Practical Malware Analysis – From Memory Forensics to Threat Attribution

Key takeaways

- You practiced end-to-end malware analysis starting from memory forensics to static analysis of extracted binaries
- Key insights were gathered using hash checks, string extraction, PE header inspection, and analysis of imported functions
- Behavioral indicators such as mutexes, registry paths, or suspicious strings helped characterize the malware's purpose
- Mapping to MITRE ATT&CK techniques enables structured understanding of adversary behavior and supports reporting, correlation, and defensive actions

Universidad Zaragoza 1474

Agenda

Universidad Zaragoza

Practical Takeaways

Just like in Inception, where every dream leaves a subtle trace – malware leaves footprints in memory. Our job is to find them before they fade

Universidad Zaragoza

Practical Takeaways

1. **Conduct quick memory-based investigations:**
   - Use tools such as `WinPmem` or `LiME` to capture volatile memory
   - Load it into Volatility 3 for immediate inspection of suspicious activity
2. **Perform triage on unknown/suspicious binaries:**
   - Apply *lightweight* static analysis techniques (hashing, strings, imports, YARA) to quickly evaluate malware samples
   - Even without Internet access or a full testing environment, static analysis can provide valuable insights of the possible behavior
   - Note that more advanced analysis would be needed to further confirm the behavior
3. **Mapping behavior with MITRE ATT&CK to improve your reporting:**
   - Use your observations (e.g., registry key usage, mutexes, API calls) to map ATT&CK techniques
   - Facilitates attribution and detection engineering

Start small: *one dump, one binary, one insight at a time*

Universidad Zaragoza

Practical Malware Analysis and Memory Forensics for Incident Response

Ricardo J. Rodríguez

rjrodriguez@unizar.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

April 1, 2025

DFRWS EU 2025

Brno, Czech Republic