#CyberSBC2022

Cybersecurity Summer Bootcamp

LEÓN - 2022

Extracting Malware Indicators of Compromise in Memory Forensics

Ricardo J. Rodríguez
University of Zaragoza

5 to 15 July 2022
León, Spain

incibe.es/summer-bootcamp

Organizers:

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE

Partners:

OAS More rights for more people

Canada

AYUNTAMIENTO DE LEÓN

LEÓN CON GRE SOS

universidad de león

Instructor

- **Ricardo J. Rodríguez**
  - PhD on Computer and Systems Engineering
  - Associate Professor (public servant) at the University of Zaragoza
  - **Researcher in cybersecurity issues**, especially in:
    - Program Binary Analysis
    - Digital forensics (in particular, in memory)
    - Security in systems based on RFID/NFC
  - **DisCo research group**
    - RME-DisCo: https://reversea.me
    - Follow us on Twitter and on Telegram! @reverseame
  - **E-mail**: rjrodriguez@unizar.es
    - Feel free to contact me if you have questions after the workshop!
  - **Personal website**: http://www.ricardojrodriguez.es

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

AGENDA

1. **Introduction**
   - Incident Response
   - Memory forensics
   - Malware
2. **Previous Concepts**
   - Program Structure. Loading Executables into Memory
   - Virtual Memory. Pages and Processes. Issues
3. **Malware Analysis in Memory Forensics**
   - Malware Analysis Phases
   - Malware Analysis Phases in Memory Forensics

AGENDA

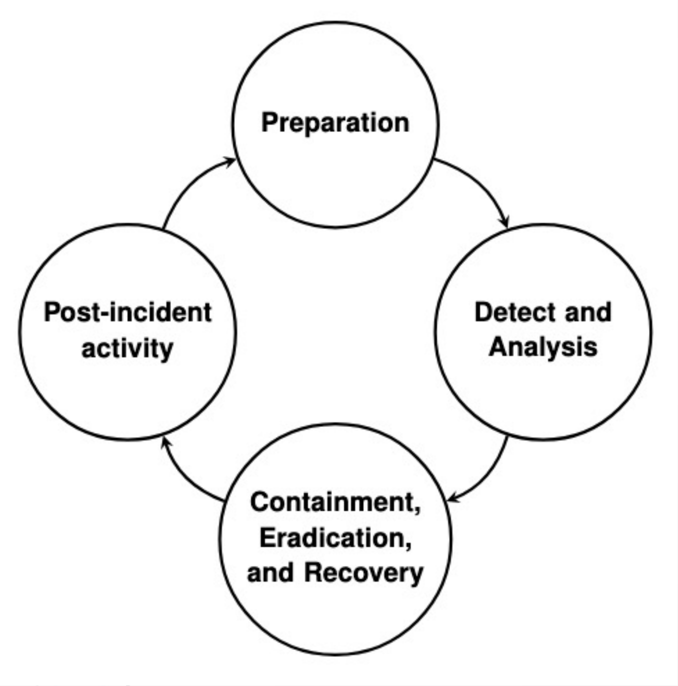
4. **Collection of Memory Evidence**
   - Memory acquisition
   - Memory Dump Analysis: *Volatility*
   - Detection of Indicators of Compromise with *Volatility*
5. **Advanced Detection of Indicators of Compromise**
   - Unofficial Plugins
6. **Development of Own Analysis Tools**
7. **Workflows Design for Evidence Analysis**
   - Workflow Assembly
   - Information Exchange

1. Introduction

1. Introduction

Incident Response

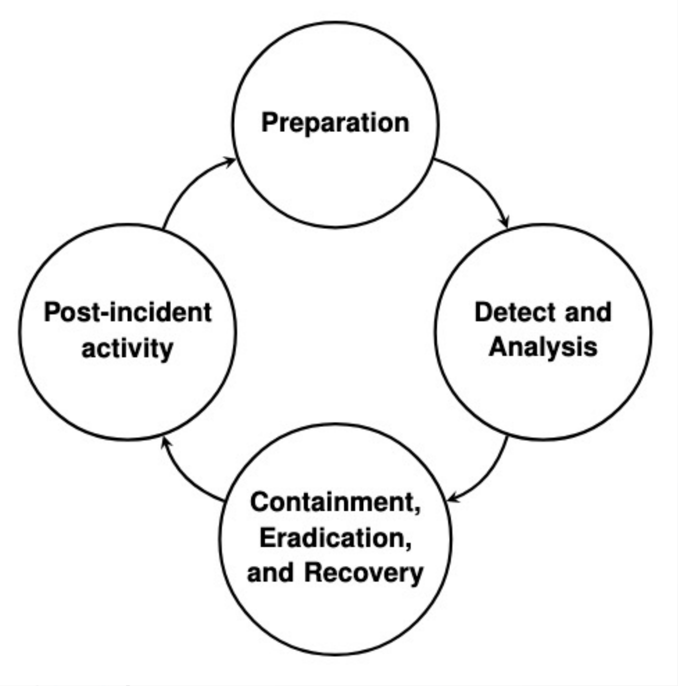
- Incident response phases (NIST SP 800-61)
  1. **Preparation**
     - Preparedness for incident management
     - Incident prevention
  2. **Detect and Analysis**
     - Attack vectors
     - Indicators of incidence
     - Sources of precursors and indicators
     - Incident analysis, documentation, prioritization and notification
  3. **Containment, Eradication, and Recovery**
  4. **Post-incident activity**



1. Introduction

Incident Response

- Incident response phases (NIST SP 800-61)
  1. **Preparation**
  2. **Detect and Analysis**
  3. **Containment, Eradication, and Recovery**
     - Containment strategies
     - Collection and management of evidence
     - Identification of attackers
     - Eradication and recovery
  4. **Post-incident activity**
     - Learned lessons
     - Use of information collected from the incident
     - Evidence retention



1. Introduction

Incident Response

- **Know what has happened, preserving all the information** related to the incident
- Respond to the well-known 6 W's: *what, who, why, how, when*, and *where*
- **Usual incident**: presence of malicious software (*malware*)
- **Various aspects of forensic analysis**:
  - ***Device forensics***
    - Digital media
    - ***Memory***
  - Forensic analysis of communications

1. Introduction

Incident Response

- **Forensic analysis of digital media versus memory**
  - Difficulty of access to digital media
  - Encrypted information
  - Volatile information
  - Excessive amount of information

1. Introduction

Memory Forensics

- Can I use memory forensics to detect malware?
  - **Yes. And no.**
  - **Problems related to the content available in memory**
    - Page swapping
    - Load on demand (also called *lazy loading*)
    - Page smearing
  - The best would be to use the forensic analysis of digital media **as a complement**
    - That is, that memory forensics is not only what we rely on

1. Introduction

Malware

- *Malicious software*
  - **Software specially designed to do some kin**
  - **Different types, depending on their functio**
    - They can have several functionalities at the s
  - **Lifecycle**
    1. Initial compromise (social engineering attac
    2. Persistence
    3. Communication with C&C servers
    4. Lateral movement
    5. Data exfiltration / malicious activity

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.*
‡*Depends on the trigger conditions defined to launch the program.*

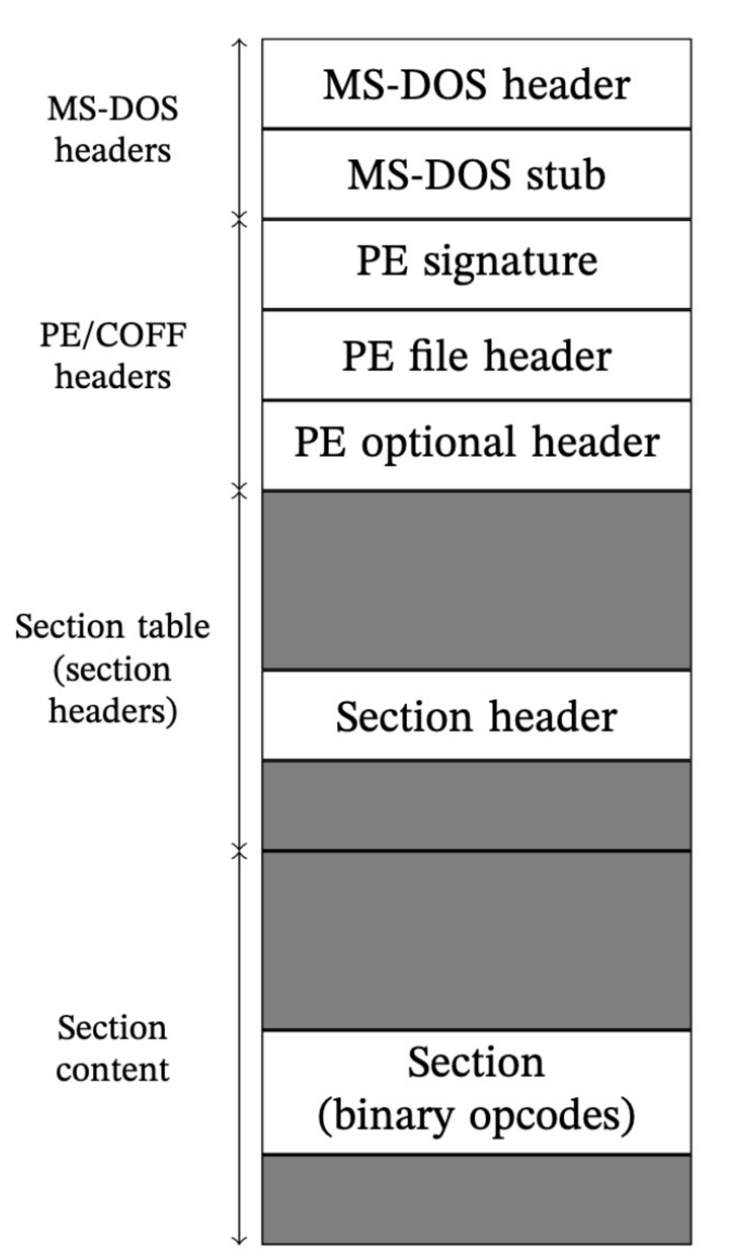
More details: Uroz, D. & Rodríguez, R. J. **Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics**. Digital Investigation, 2019, 28, S95-S104, Elsevier. https://doi.org/10.1016/j.diin.2019.01.026

2. Previous Concepts

2. Previous Concepts

Program Structure

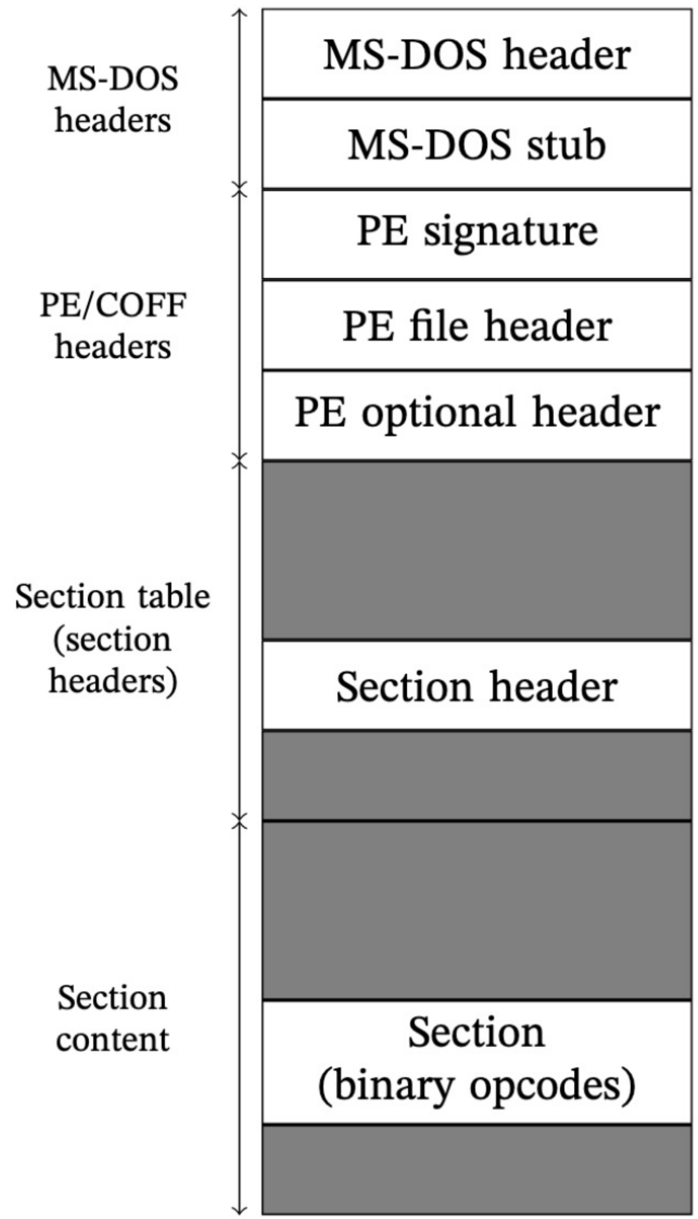
- Since Windows NT 3.1
- **PE: Portable Executable**
  - Data structure defined in WinNT.h (Microsoft Windows SDK)
  - Three parts: MS-DOS headers, PE/COFF headers, Section headers
  - https://docs.microsoft.com/en-us/windows/win32/debug/pe-format
- **MS-DOS headers**
  - First 64 bytes
  - e_magic: MZ (Mark Zbikowski)
  - e_lfanew: offset to PE/COFF headers



2. Previous Concepts

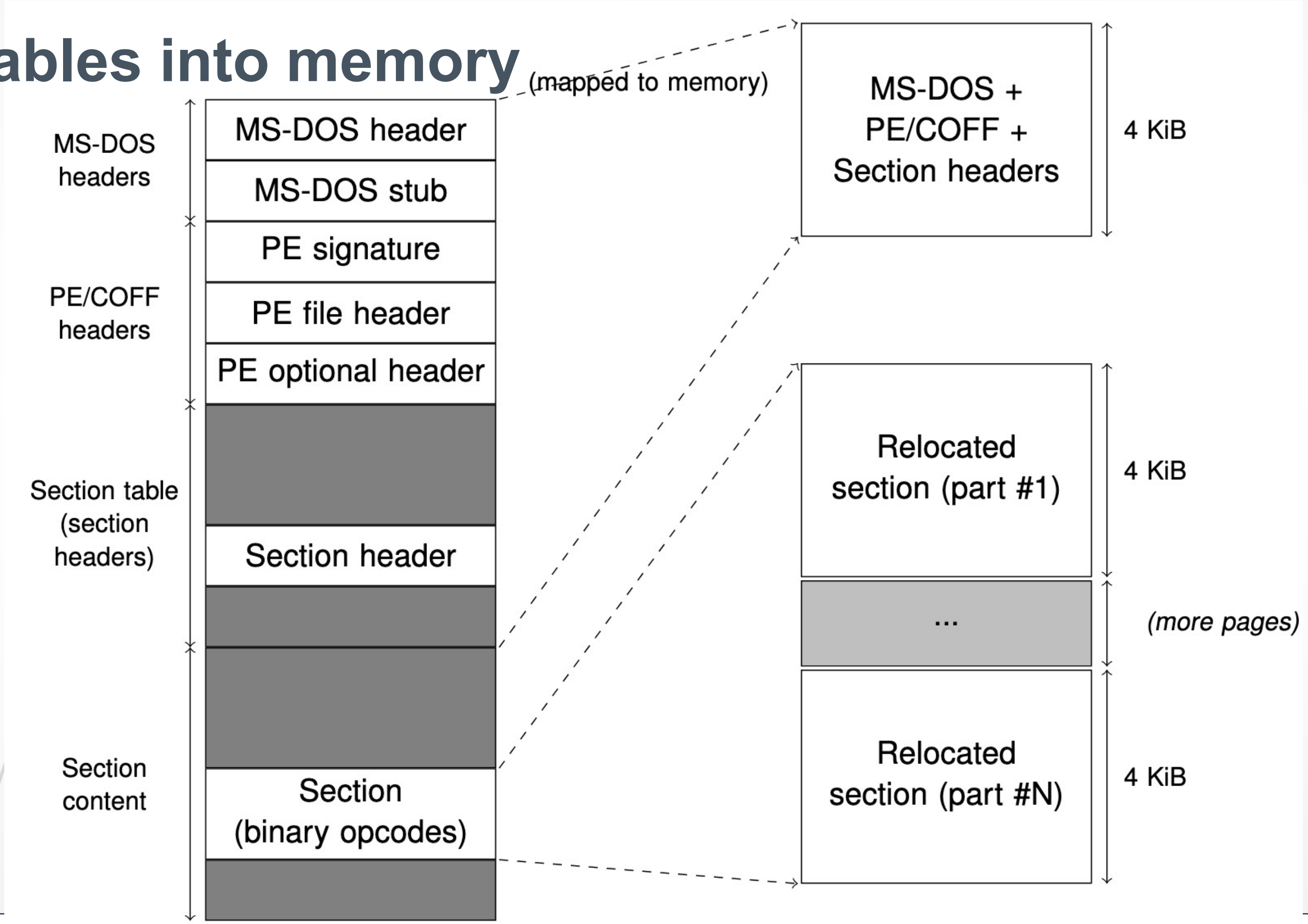
Program Structure

- PE/COFF headers
  - **PE signature** (“PE\0\0”)
  - **PE file header**
    - Define target machine, number of sections, characteristics, etc.
  - **PE optional header**
    - Optional for some object files
    - Fields of interest: ImageBase, BaseOfCode, AddressOfEntryPoint
    - DataDirectory: Directory table. Each entry has a meaning
- Section headers
  - IMAGE_SECTION_HEADER structure
  - Common sections: .text/.code, .rdata/.rodata, .data, .reloc, …



2. Previous Concepts

Loading executables into memory



2. Previous Concepts

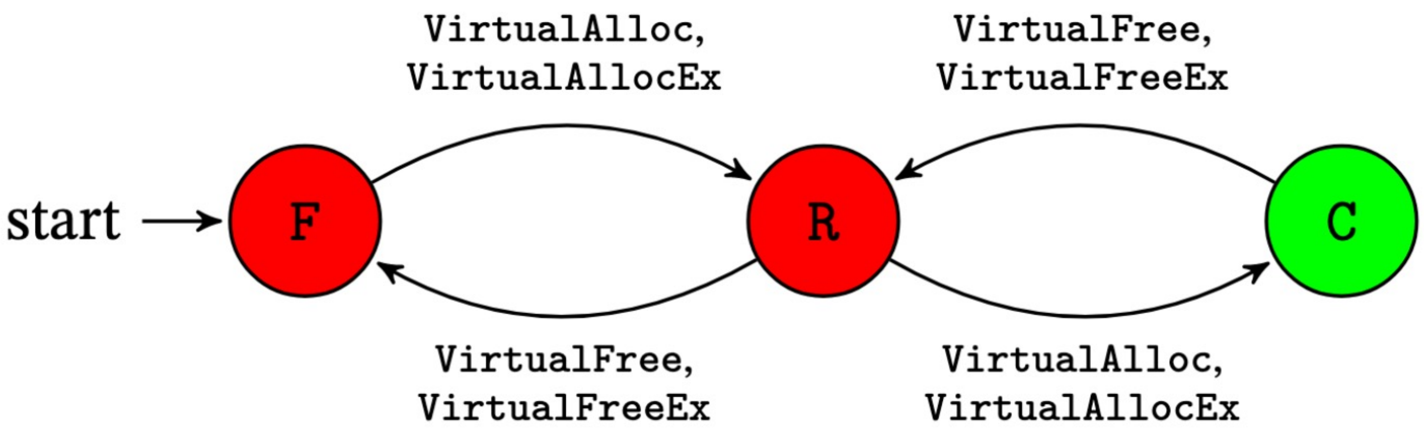
Virtual Memory

- **Physical address** vs. **Virtual address**
  - Translation performed by the memory management unit (MMU)
  - PTE: *page table entries*
    - Each process and the kernel itself have their own page tables
    - **Map virtual address to physical address**
- **Virtual memory space of a process**
  - **Contiguous regions**
  - Different uses: file mapping (disk file backup), unmapped memory
- *Virtual Address Descriptor* (VAD)
  - Kernel structure to represent a contiguous region of memory (can contain multiple pages)
  - Balanced tree
  - Different permissions (we will comment later…)

2. Previous Concepts

Virtual Memory: pages

- Page: minimal memory granularity
  - Contiguous, fixed-size block of virtual memory
  - Small (4KiB) and large (for example, 2MiB on x86 and x64, 4MiB on ARM)
- States:
  - Free: initial state
  - Reserved: for future use
  - Committed (ready to use)



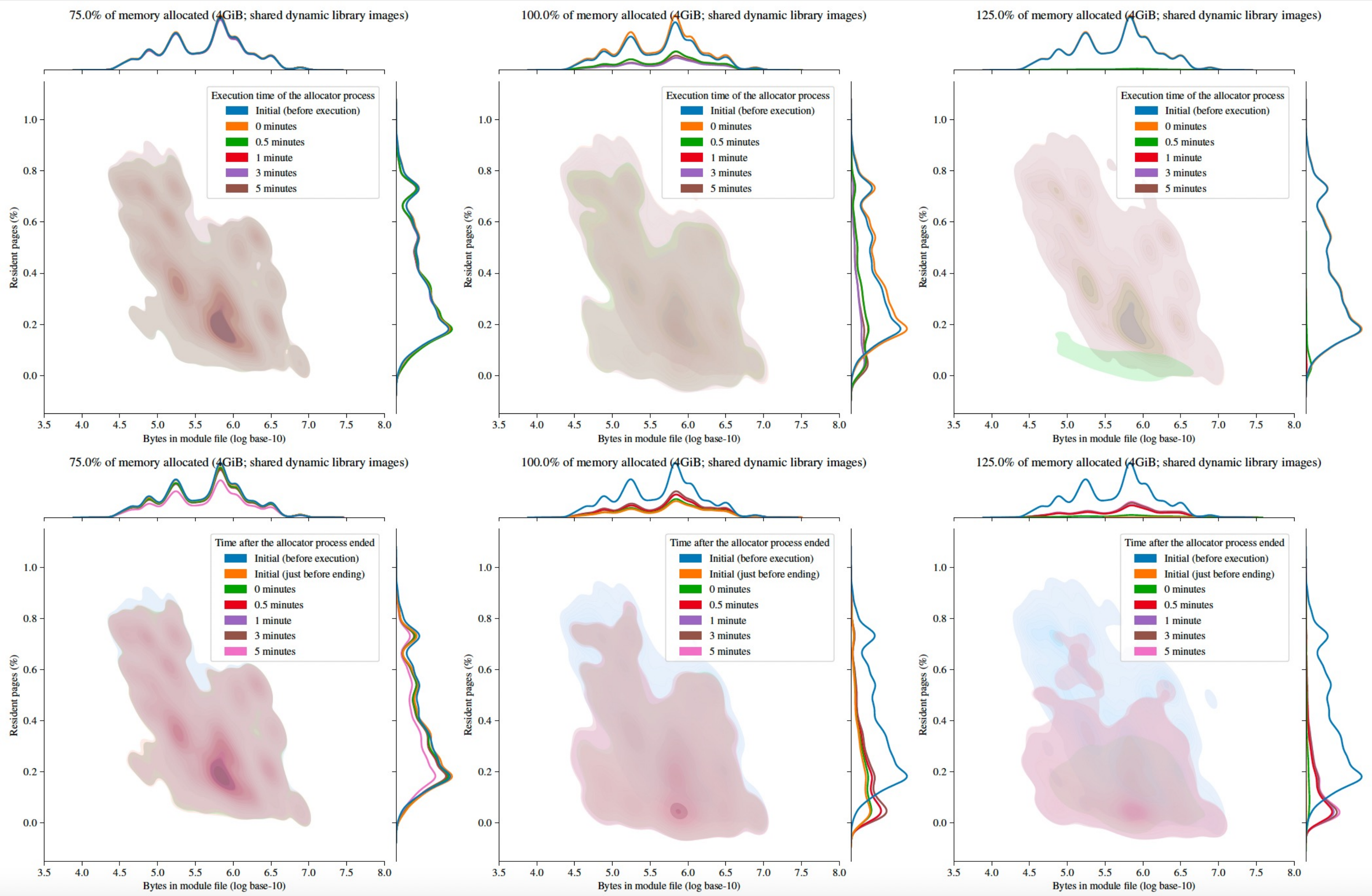
2. Previous Concepts

Virtual Memory: Problems

1. **Page swapping**
   - Memory space available for **a** process in 32 bits: 2GiB
   - Is it physically possible?
   - MMU manages memory pages that are accessed and paged, retrieving them from disk and placing them back into memory
2. **Load on demand**
   - Only the memory pages that are needed are loaded, and when they are needed (lazy loading)
   - Copy-on-Write (CoW) mechanism
3. **Page smearing**
   - Memory is a living entity, continually changing
   - Memory capturing issue on running systems
     - Possible references between very distant memory areas

2. Prev

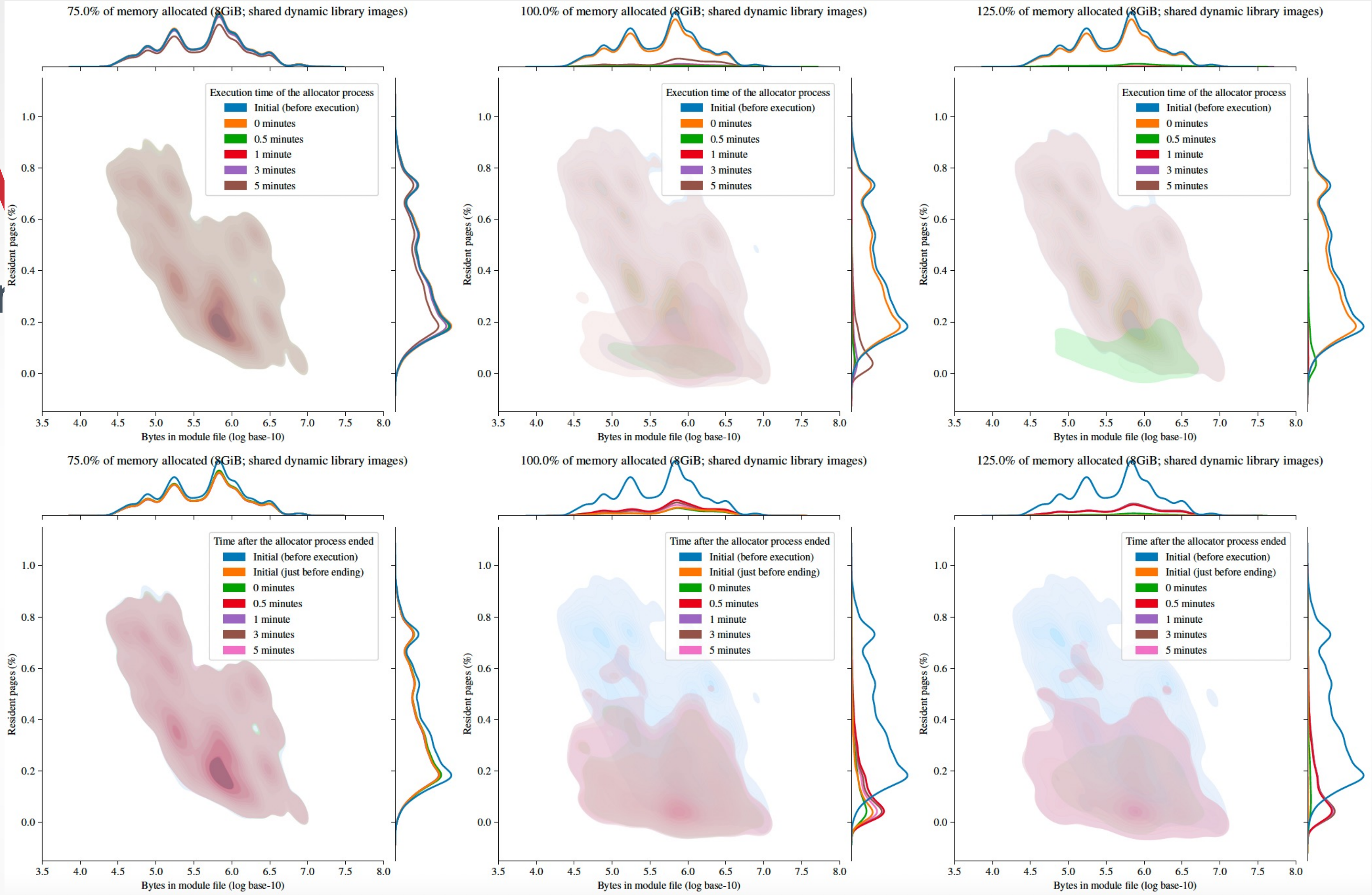
Virtual Mem



More details: Martín-Pérez, M., Rodríguez, R.J. (2022). **Quantifying Paging on Recoverable Data from Windows User-Space Modules**. In: *Digital Forensics and Cyber Crime*. ICDF2C 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 441. Springer. https://doi.org/10.1007/978-3-031-06365-7_

2. Prev

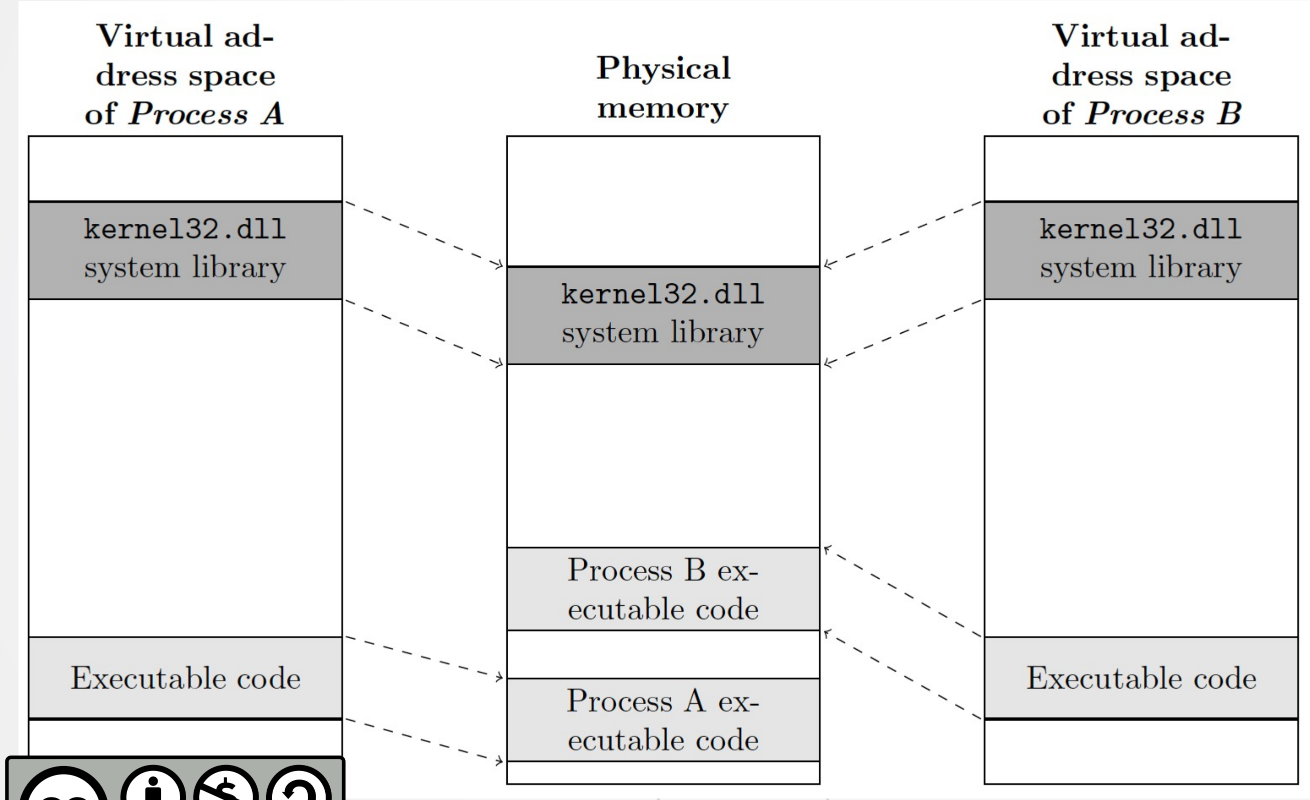
Virtual Mem

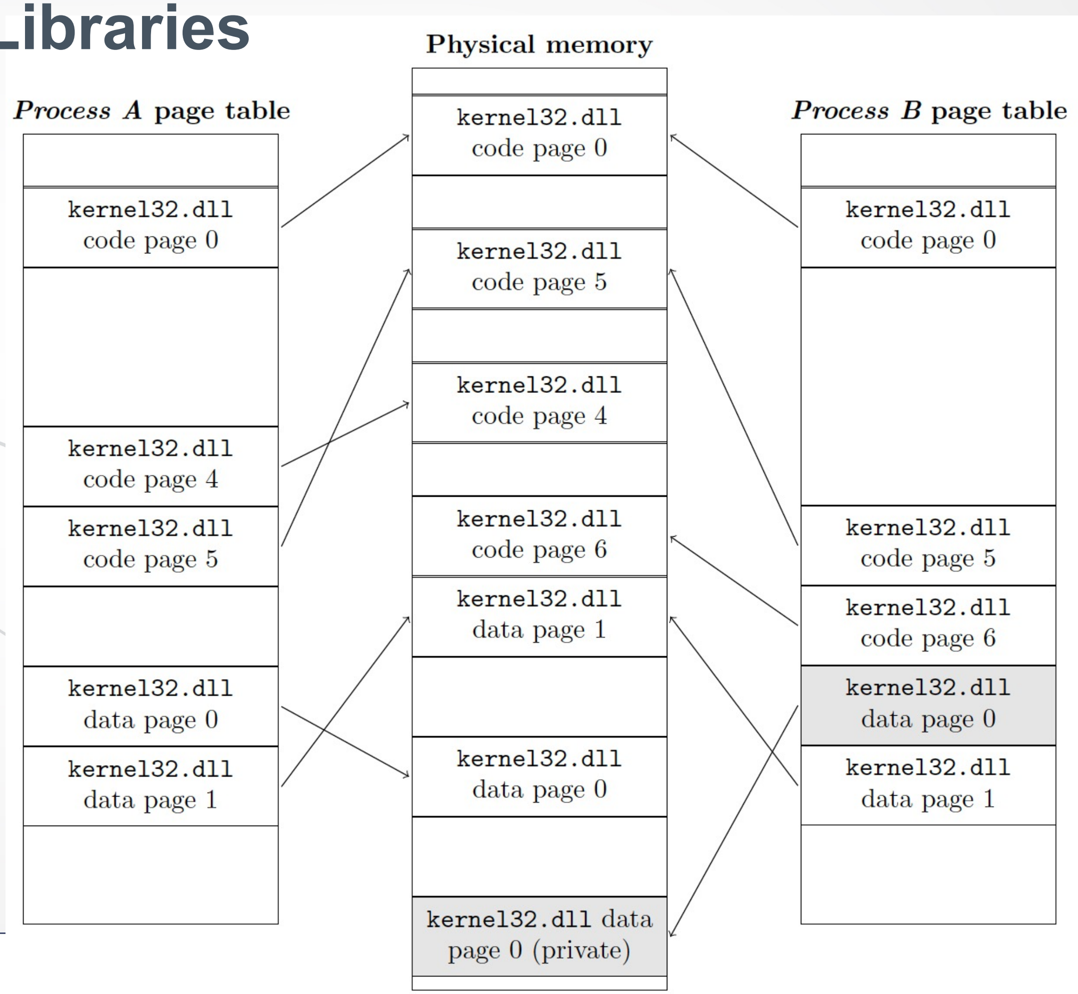


More details: Martín-Pérez, M., Rodríguez, R.J. (2022). **Quantifying Paging on Recoverable Data from Windows User-Space Modules**. In: *Digital Forensics and Cyber Crime*. ICDF2C 2021. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 441. Springer. https://doi.org/10.1007/978-3-031-06365-7_

2. Previous Concepts

Virtual Memory: Processes and Shared Libraries





3. Malware Analysis in Memory Forensics

3. Malware Analysis in Memory Forensics

Malware Analysis Phases

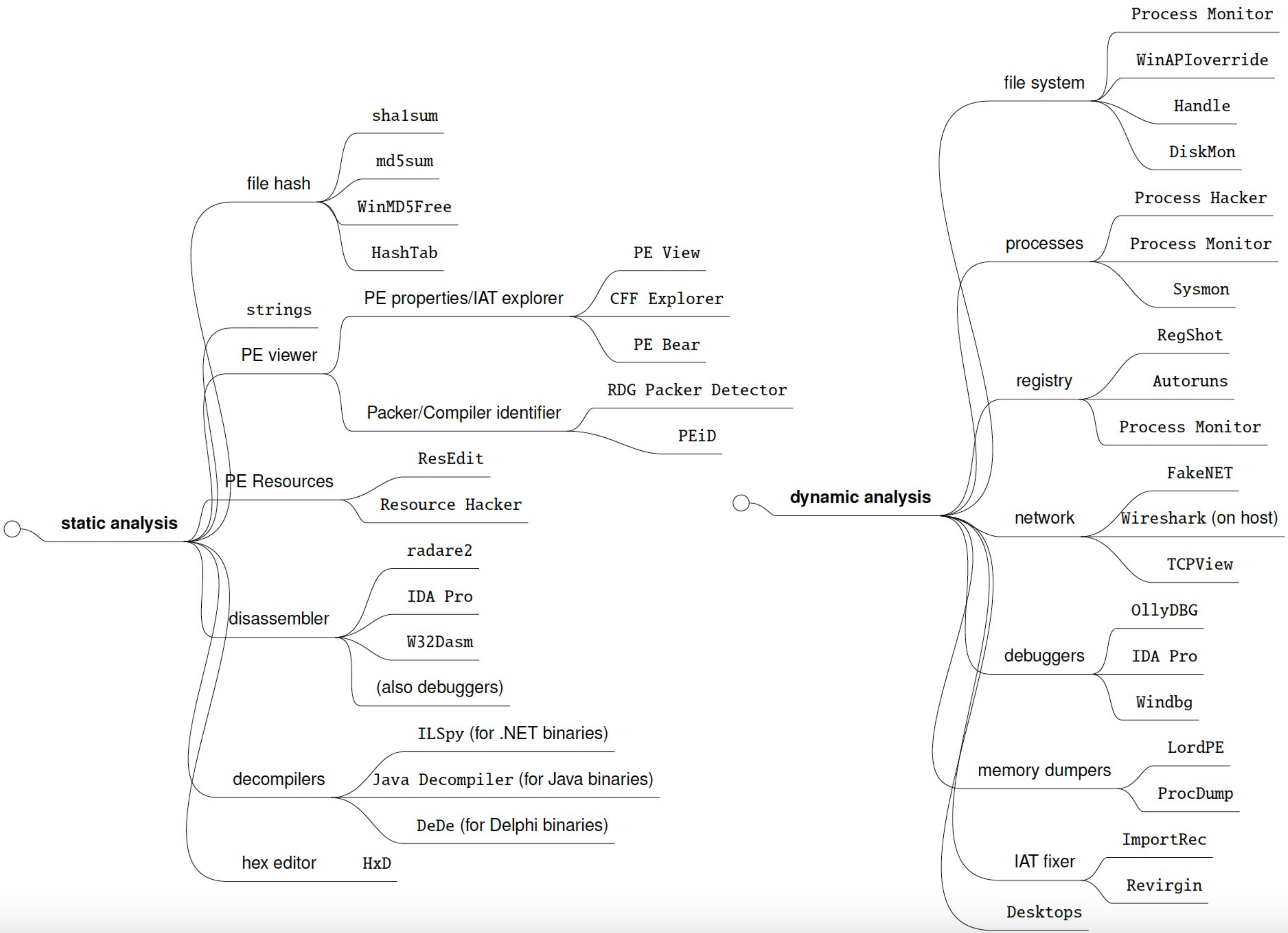
- **Static analysis** (the program does not run)
  - Signatures (MD5, SHA-1, SHA-256…)
    - HashTab, md5sum, sha1sum, WinMD5Free, …
  - Strings
    - strings
  - PE properties
    - Fields of interest (obfuscated? packed?)
    - External functions set in *Import Address Table* (IAT)
    - Resources within the PE

3. Malware Analysis in Memory Forensics

Malware Analysis Phases

- **Dynamic analysis** (the program runs – typically in an isolated environment)
  - OS interaction: files
    - Creation? Access? Modification? Deletion?
  - OS interaction: Windows Registry
    - Creation? Access? Modification? Deletion?
  - OS interaction: processes
    - Creation? Access?
  - Interaction with the outside: network communications
    - IP addresses
    - Domain names

3.


static analysis
file hash
sha1sum
md5sum
WinMD5Free
HashTab
strings
PE viewer
PE properties/IAT explorer
PE View
CFF Explorer
PE Bear
Packer/Compiler identifier
RDG Packer Detector
PEiD
PE Resources
ResEdit
Resource Hacker
disassembler
radare2
IDA Pro
W32Dasm
(also debuggers)
decompilers
ILSpy (for .NET binaries)
Java Decompiler (for Java binaries)
DeDe (for Delphi binaries)
hex editor
HxD
dynamic analysis
file system
Process Monitor
WinAPIoverride
Handle
DiskMon
processes
Process Hacker
Process Monitor
Sysmon
registry
RegShot
Autoruns
Process Monitor
network
FakeNET
Wireshark (on host)
TCPView
debuggers
OllyDBG
IDA Pro
Windbg
memory dumpers
LordPE
ProcDump
IAT fixer
ImportRec
Revirgin
Desktops

3. Malware Analysis in Memory Forensics

Malware Analysis Phases in Memory Forensics

- **Memory dumps**
  - Contains item **artifacts** that were running at the time of acquisition
    - **Running processes, connected users, open sockets, etc.**

❖ **Process: memory representation of a program**

1. **Memory mapped executable file**
   - Page alignment → *inconclusive hash signatures*
2. **Load on demand**
   - **Partial content**: *problem to know the real malicious activity carried out by the sample*
   - The way of acquiring memory can affect
3. **Resolved IAT Function Table**
   - *Difficulty of subsequent execution in the same or other environments*

4. Collection of memory evidence

4. Collection of Memory Evidence

Memory Acquisition

- **Various acquisition techniques**
  - 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, https://doi.org/10.1016/j.diin.2019.01.001
- **Software tools for complete memory dump**
  - WinPmem: https://github.com/Velocidex/WinPmem
    - Apache license
    - Support for Windows XP up to Windows 10, for 32 and 64 bits
    - Example: winpmem_mini_x64.exe physmem.raw
  - Linux Memory Extractor (LiME): https://github.com/504ensicsLabs/LiME
    - GNU/GPLv2 license
    - Support for Linux and Android
    - Extraction via local port connection
  - FTK Imager: https://accessdata.com/product-download/ftk-imager-version-4-2-1
    - Commercial tool
    - Support for Windows

4. Collection of Memory Evidence

Memory Acquisition

- **Acquisition in virtual machines**
  - VirtualBox
    - vboxmanage debugvm "Win7" dumpvmcore --filename test.elf
  - VMWare
    1. Create a snapshot of the virtual machine execution (.vmss and .vmem files are generated)
    2. vmss2core tool: https://flings.vmware.com/vmss2core??src=vmw_so_vex_mraff_549
- **Other tools for extracting processes or modules**
  - ProcDump: https://docs.microsoft.com/en-us/sysinternals/downloads/procdump
    - procdump -ma 4572
    - Single dump (fichero .dmp)
  - Windows Memory Extractor: https://github.com/pedrofdez26/windows-memory-extractor
    - GNU/GPLv3 license
    - WindowsMemoryExtractor_x64.exe --pid 1234
    - Create sectional dump of process memory

4. Collection of Memory Evidence

Memory Dump Analysis: *Volatility*

- **De facto standard** to analyze memory dumps
- FOSS (GNU/GPLv2 license)
- Published in 2007 in BH USA, called *Volatoools*
- Support for Windows, Linux and MacOS, in 32 and 64 bits
- Very extensive API for your own implementations
- Version 2.6 vs. Version 3
  - Python2 vs Python3
  - Version 3 is already stable! https://github.com/volatilityfoundation/volatility3

4. Collection of Memory Evidence

First Steps with *Volatility*

- Virtual machine provided: Debian 10.10
  - Volatility 2.6 and Volatility 3.0 already installed
  - User/password: alumno / alumno
- **Help**:
  - python vol.py –h
- **Memory dump to analyze** :
  - python vol.py --f mem.dmp --profile Win7SP1x86
  - The profile is only necessary in version 2.6. It indicates where are the internal structures of the SO
- *How to know the profile to use?* → imageinfo plugin
  - python vol.py --f mem.dmp imageinfo
- **Plugins are always indicated at the end of the command**

4. Collection of Memory Evidence

Detection of Indicators of Compromise with *Volatility*

- **Processes and DLLs**
  - pslist, pstree (psscan for possible rootkits)
  - dlllist, dlldump
  - handles
  - enumfuncs (list of imported and exported functions, by process/dll)
- **Process memory**
  - memmap, memdump
  - procdump
  - Vadinfo, vadwalk, vadtree, vaddump
  - evtlogs
  - iehistory
- **Network**
  - connections, connscan
  - sockets, sockscan
  - netscan (network artifacts in Win7)

https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

4. Collection of Memory Evidence

Detection of Indicators of Compromise with *Volatility*

- **Kernel memory and other (internal) objects**
  - modules, modscan, moddump
  - driverscan
  - filescan
- **Register**
  - hivescan, hivelist, hivedump
  - printkey
  - lsadump
  - userassist, shellbags, shimcache
  - dumpregistry
- **Filesystem**
  - mbrparser, mftparser
- **Hibernation file analysis or other dumps**

https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

4. Collection of Memory Evidence

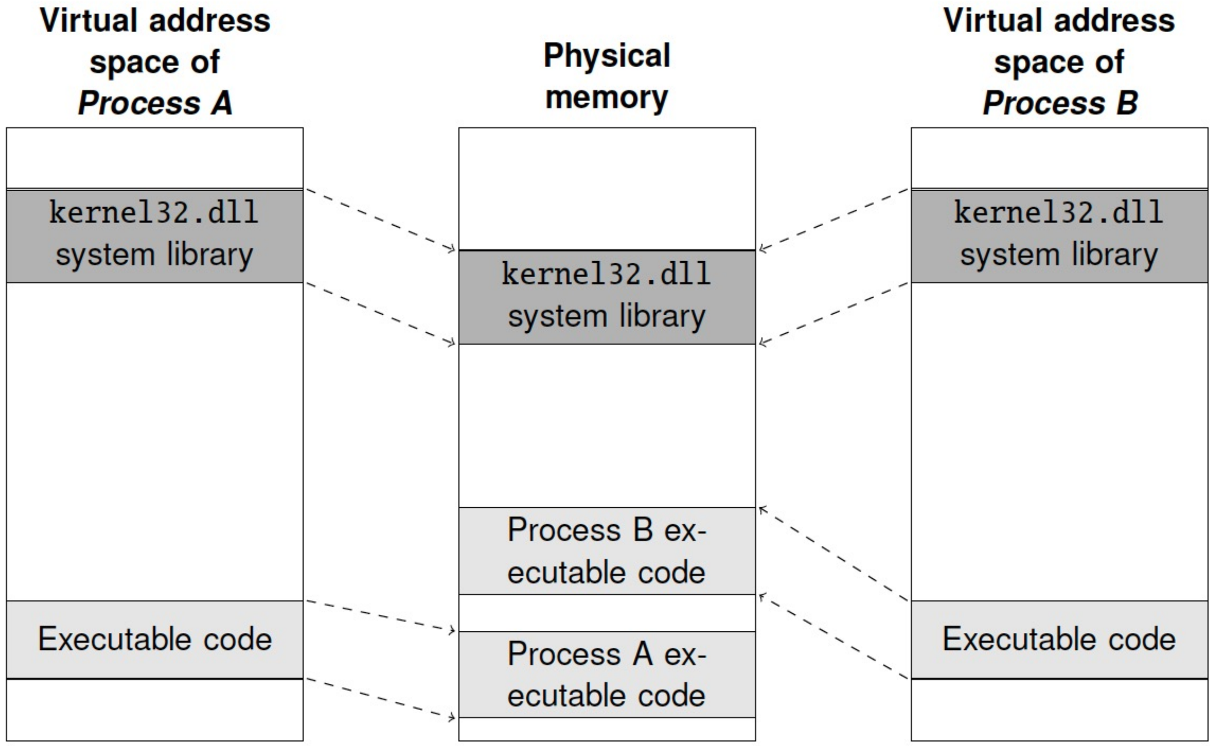
Memory Forensic & Malware Analysis: Related Problems

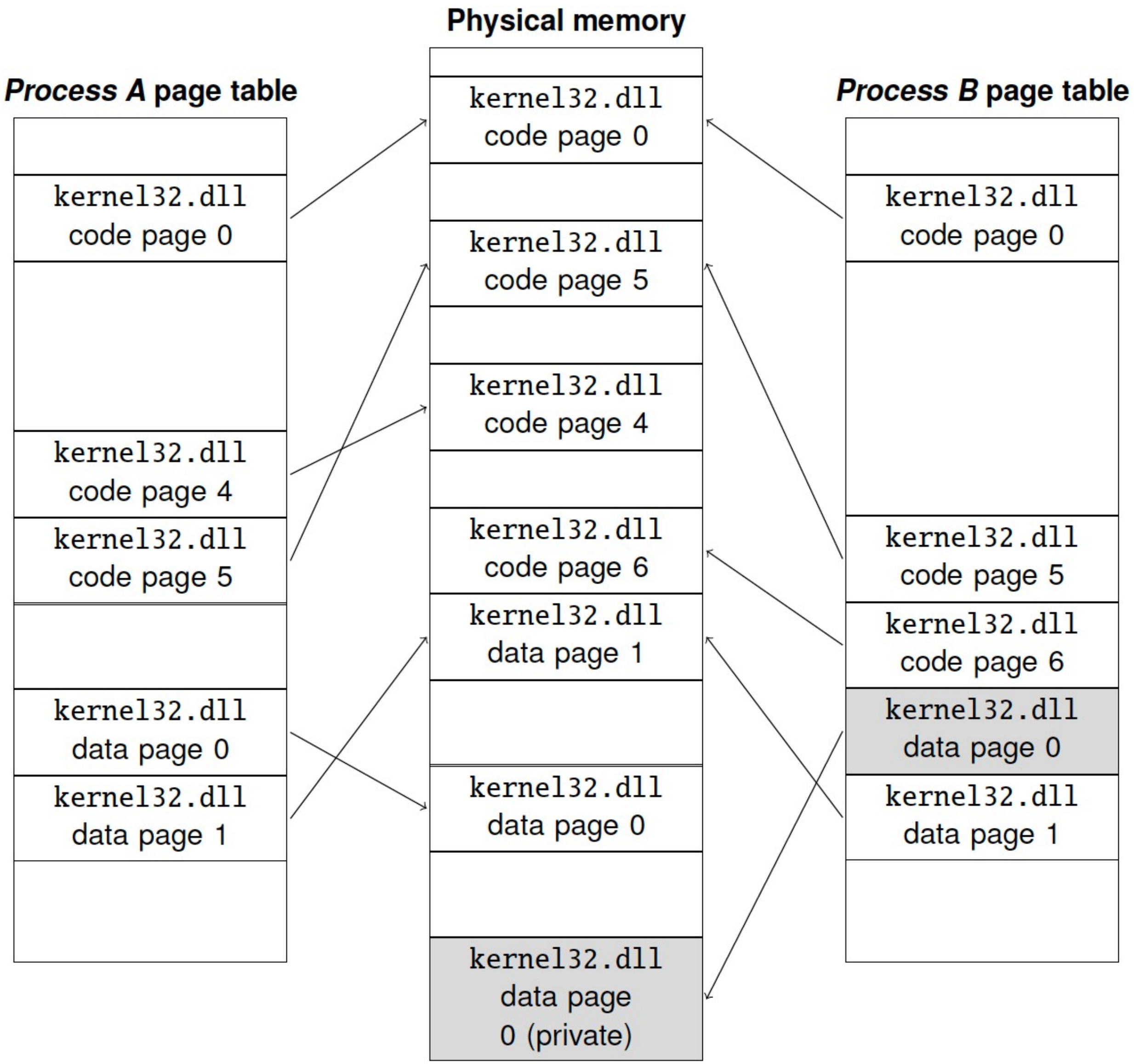
- <u>Imprecision of memory dump content</u>
  - The content of an image is not faithful to its image file
  - Mainly due to:
    - Paginated effect (4kiB alignment causes null bytes filling)
    - Relocation (resolved IAT addresses or lack of some sections)
  - **Solutions?**
    - Use of approximate similarity algorithms (sum plugin)
    - Database construction with allowed hashes

4. Collection of Memory Evidence

Memory Forensic & Malware Analysis: Related Problems

- Lack of completeness of memory dump content
  - The content of an image is not complete, with respect to image file
  - Mainly due to:
    - Swapping effect (if a page is not used, is temporarily saved on disk)
    - Load on demand (only what is going to be used is charged)
  - **Solutions?**
    - Use disk forensic to recover files
    - Combine memory forensics with disk forensics


Virtual address space of Process A
kernel32.dll system library
Executable code
Physical memory
kernel32.dll system library
Process B executable code
Process A executable code
Virtual address space of Process B
kernel32.dll system library
Executable code


Process A page table
kernel32.dll code page 0
kernel32.dll code page 4
kernel32.dll code page 5
kernel32.dll data page 0
kernel32.dll data page 1
Physical memory
kernel32.dll code page 0
kernel32.dll code page 5
kernel32.dll code page 4
kernel32.dll code page 6
kernel32.dll data page 1
kernel32.dll data page 0
kernel32.dll data page 0 (private)
Process B page table
kernel32.dll code page 0
kernel32.dll code page 5
kernel32.dll code page 6
kernel32.dll data page 0
kernel32.dll data page 1

4. Collection of Memory Evidence

Memory Forensic & Malware Analysis: Related Problems

- Imprecision of memory dump
  - Memory is continuously updated and acquired in a non-atomic way
  - Especially relevant when there are acquisitions in living systems
  - Highly probable. Inconsistency due to:
    - Pointers
    - Memory fragmentation
  - Sophisticated malware can force inconsistencies deliberately (DKOM attacks)
  - **Solutions?**
    - Use of other acquisition techniques
    - Check the temporary consistency of the data: *temporal forensics* (*Pagani, F.; Fedorov, O. & Balzarotti, D. Introducing the Temporal Dimension to Memory Forensics. ACM Trans. Priv. Secur., vol. 22, no. 2, pp. 9:1-9:21, ACM, https://doi.org/10.1145/3310355*)

4. Collection of Memory Evidence

Memory Forensic & Malware Analysis: Related Problems

- Stealthy malware
  - VAD are unreliable sources of information
    - Pages permissions are not updated if they are changed after putting the initial permissions
  - You can "swap" pages deliberately
  - *Process hollowing* attacks
  - **Solutions?**
    - Malware signatures (but not based on cryptographic hashes)
    - Robust kernel signatures
    - Volatility Plugins: *malfind*, *malscan*, *impfuzzy*

4. Collection of Memory Evidence

Detection of Indicators of Compromise with *Volatility*: example

LAB SESSION 1

- **“zeus.vmem” memory dump** (from “Malware Analyst’s Cookbook” book)
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2022/malware-memory-forensics/laboratories/lab1_introduction.pdf
  - Details many Volatility plugins of interest for memory dump analysis

5. Advanced Detection of Indicators of Compromise

5. Advanced Detection of IoC

Unofficial Plugins

- There are many additional plugins that expand Volatility functionality
- Mode of use
  1. Plugin installation (for instance, source code repository download)
  2. Execution: volatility --plugins="/path/to/plugin" -f file [OPTIONS] pluginname

5. Advanced Detection of IoC

Unofficial Plugins

- **MalConfScan**: https://github.com/JPCERTCC/MalConfScan
  - Extract configuration, deciphered chains or DGA domains from some malware families
- **Malscan**: https://github.com/reverseame/malscan (for Volatility 2.6)
  - GNU/GPLv3 license
  - Integrates Malfind with ClamAV-daemon (only available in Linux). Less false negatives
  - Operating modes: Normal (regions +WX, any executable module, and VADs-type private memory) and full-scan (regions with +x)
  - VADs without associated executables, beginnings of function and empty pages followed by 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   ................
```


Organizers:
GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL
SPANISH NATIONAL CYBERSECURITY INSTITUTE
OAS
for more people
Canada
AYUNTAMIENTO DE LEÓN
LEÓN CON GRE SOS
universidad de león

5. Advanced Detection of IoC

Unofficial Plugins

- **Similarity Unrelocated Module**: https://github.com/reverseame/similarity-unrelocated-module (for Volatility 2.6)
  - GNU/GPLv3 license
  - Calculate approximate signatures on the modules of a dump:
    - Algorithms: dcfldd, ssdeep, sdhash, TLSH
    - A module is an executable file or library of functions loaded in memory
  - Allows comparison between modules of different memory dumps
  - Undoes the changes made by the operating system (relocation). Two methods :
    - Guided De-relocation
    - Linear Sweep De-relocation
  - More details: 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, https://doi.org/10.1016/j.cose.2020.102119

5. Advanced Detection of IoC

Unofficial Plugins

- **Winesap**: https://github.com/reverseame/winesap (for Volatility 2.6)
  - AGPLv3 license
  - Look for all Windows ASEPs in memory dump
  - Binary or unknown registration keys: they are analyzed as PE
  - Chains related to usual malware file routes (%Appdata%,%TMP%,%Temp%, Appdata), NTFS ADS, Shells commands (e.g., rundll32.exe shell32.dll, Shellexecute_rundll)
  - **More details**: D. Uroz, R. J. Rodríguez, "Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics", Digital Investigation, vol. 28, p. S95-S104, 2019, https://doi.org/10.1016/j.diin.2019.01.026

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.
‡Depends on the trigger conditions defined to launch the program.

5. Advanced Detection of IoC

Unofficial Plugins

- **Sigcheck**: https://github.com/reverseame/sigcheck (for Volatility 2.6)
  - GNU/GPLv3 license
  - Verify PE files digitally signed with Microsoft Authenticode
  - Two signature methods: embedded (in the PE), by catalog (in external file)
  - **IMPORTANT**: Verify that the executable file that began was original
    - If a malware does *process hollowing* would not be detected with this method
  - **More details**: D. Uroz, R. J. Rodríguez, "On Challenges in Verifying Trusted Executable Files in Memory Forensics", Forensic Science International: Digital Investigation, vol. 32, p. 300917, 2020, https://doi.org/10.1016/j.fsidi.2020.300917

5. Advanced Detection of IoC

Example: WannaCry

LAB SESSION 2

- **“wannacry.elf” memory dump**
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2022/malware-memory-forensics/laboratories/lab2_example_wannacry.pdf

6. Development of Own Analysis Tools

6. Development of Own Analysis Tools

LAB SESSION 3

- **“alina1G.elf” memory dump**
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2022/malware-memory-forensics/laboratories/lab3_plugin_development.pdf

7. Workflows design for evidence analysis

7. Workflows design for evidence analysis

Workflow Assembly and Information Exchange

- **Clearly define what you want to obtain**
  - Search in the plugins of the Volatility community if it is already made (the wheel should not be reinvented!)
- **Pipeline development**
  - Python? Bash?
  - Multi-threading
  - Module extraction and analysis
    - Sandbox, VT, pefile
- **Information exchange**
  - Standard formats:
    - JSON, CSV, etc.
- **Final analysis report:**
  - JSON? Markdown?

7. Workflows design for evidence analysis

LAB SESSION 4

- **Any memory dump of interest**
- Follow the laboratory workbook provided on the workshop's website: https://webdiis.unizar.es/~ricardo/sbc-2022/malware-forense-memoria/laboratorios/lab4_automatizacion_analisis.pdf
- With what is described in the workbook, you have to develop an analysis system for:
  - DLLs extraction of a particular process (if provided by parameter) or all DLLS
  - Static analysis to measure entropy of the extracted DLLs (with pefile)
  - ClamAV execution in the DLLs expected

Recommended Bibliography

Recommended Bibliography

- The Art of Memory Forensics
    - Additional material available here
- Practical Malware Analysis. The Hands-On Guide to Dissecting Malicious Software
- Malware Analyst's Cookbook
- Documentación de Volatility 3

Cybersecurity Summer Bootcamp

LEÓN - 2022

#CyberSBC2022

5 to 15 July 2022
León, Spain

incibe.es/summer-bootcamp

Organizers:

GOBIERNO DE ESPAÑA
VICEPRESIDENCIA PRIMERA DEL GOBIERNO
MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE DIGITALIZACIÓN E INTELIGENCIA ARTIFICIAL

incibe_
SPANISH NATIONAL CYBERSECURITY INSTITUTE

Partners:

OAS More rights for more people

Canada

AYUNTAMIENTO DE LEÓN

LEÓN CONGRESOS

universidad de león