Malware Detection in Memory Forensics: Current Issues and Challenges

Ricardo J. Rodríguez

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

Universidad Zaragoza
1542

Dept. of Computer Science and Systems Engineering
University of Zaragoza, Spain

November 16, 2019

NoConName 2019
Barcelona, Spain

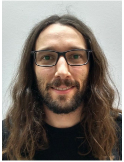
$whoami

- **Assistant Professor at University of Zaragoza**
- **Research lines**:
  - Performance/dependability/security system analysis
  - Program binary analysis / forensics
  - RFID/NFC security
- Speaker and trainer in several security-related conferences (NcN, HackLU, RootedCON, STIC CCN-CERT, HIP, MalCON, HITB…)

Universidad Zaragoza

$whoami

- **Assistant Professor at University of Zaragoza**
- **Research lines**:
  - Performance/dependability/security system analysis
  - Program binary analysis / forensics
  - RFID/NFC security
- Speaker and trainer in several security-related conferences (NcN, HackLU, RootedCON, STIC CCN-CERT, HIP, MalCON, HITB. . .)
- **Research team** – *we make really good stuff!* ☺



Miguel Martín-Pérez
PhD. student

Daniel Uroz
PhD. student

- *Memory forensics*
- *Program binary analysis*
- *Exploiting/reversing*
- *Privacy issues (Tor)*

Open PhD positions, ping me after the talk!

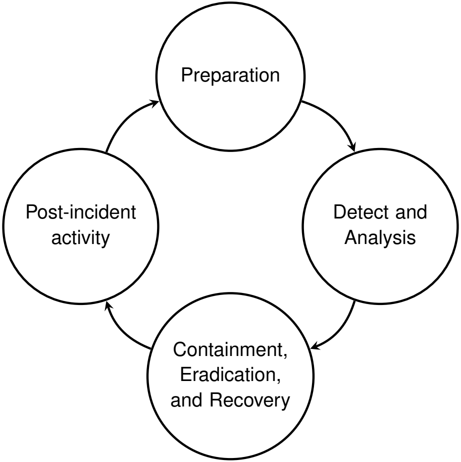
Universidad Zaragoza

Agenda

1. Introduction
2. Current Issues and Challenges
3. Conclusions

Introduction

A little bit of recap...



Incident response as defined by NIST

Introduction

Incident response

- **Figure out *what the heck* happened**, while preserving data related to the incident
- **Ask the well-known 6 W's** (what, who, why, how, when, and where)
- Common incident: **presence of malicious software** (malware)
- **Different types of analysis to get hints**:
  - **Computer forensics: disks + memory**
  - Network forensics

Universidad Zaragoza

Introduction

- **Disk forensics**: analysis of device drives
- **Memory forensics**: analysis of the data contained in the memory of the system under study

Introduction

- **Disk forensics**: analysis of device drives
- **Memory forensics**: analysis of the data contained in the memory of the system under study

Disk vs. memory

- Sometimes, **access to physical device drives are difficult to achieve**
- Think about **current limits of storage capacity versus memory capacity**
  - Terabytes versus gigabytes
  - **Facilitates the initial triage**

Universidad Zaragoza

Introduction

How is memory forensics carried out?

1. **Dump the system's memory into a data file**
   - It stores the current state of the system
   - The output file is known as *memory dump*
2. **Take the file offsite**
3. **Analyze with appropriate tools**
   - For instance, Volatility or Rekall

Universidad Zaragoza

Introduction

What does the memory dump contain?

- **Full of data** to analyze
- **Every element susceptible to analyze is termed as a *memory artifact***
  - Retrieved through appropriate internal OS structures or using a pattern-like search
- Snapshot of the running processes, logged users, open files, or open network connections – **everything running at the time of acquisition**
- It may contain also **recent system resources freed**
  - Normally, memory is not zeroed out when freed

Introduction

How is the memory dump analyzed?

- **Common tools**: Volatility and Rekall

Introduction

How is the memory dump analyzed?

- **Common tools**: Volatility and Rekall

Volatility

- **De facto standard** for analyzing memory dumps in computer forensics
- Released in 2007 at BH USA, *Volatools*. **Open source under GNU GPLv2**
- Currently maintained by The Volatility Foundation. **Implemented in Python**
- **Supports the analysis of memory dumps from Windows, Linux, and Mac OS, in both 32-bit and 64-bit**
- **Provides a rich, scriptable API to implement your own analysis plugins**

Stay tuned for Volatility version 3!

Universidad Zaragoza

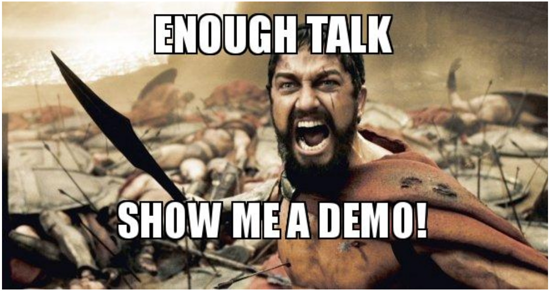
Introduction

A little more of recap...

Malicious software (malware) analysis

- Determine *what the heck* the malware does as harmful activities
- **Static analysis** (or *cold analysis*)
  - Executable files are analyzed without being executed
  - Every possible execution path is considered. **Undecidable problem**
- **Dynamic analysis**
  - Executable files are analyzed when they are executed
  - Only an execution path is considered – depends on inputs, current environment, etc.

Introduction



Talk guided by a demo

- Windows 7 x86 machine
- Alina malware (slightly modified for local connection) + system files

Agenda

1 Introduction

2 Current Issues and Challenges

3 Conclusions

Current Issues and Challenges

Issue #1

A process file DOES NOT match its executable file counterpart! 😕

- *A process is a memory representation of an executable file*
  - Let me recap you some terminology here: **executable file means the binary file as resides in disk**

Current Issues and Challenges

Issue #1

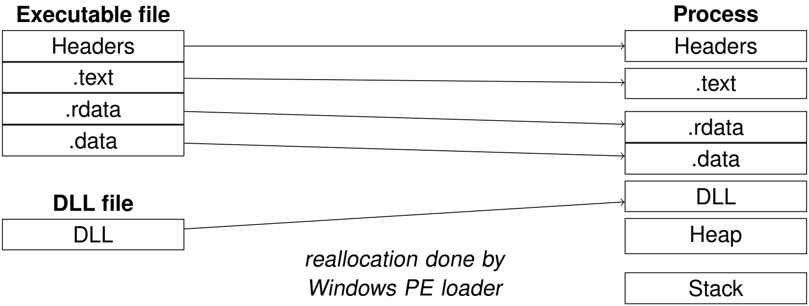
A process file DOES NOT match its executable file counterpart! ☹

- *A process is a memory representation of an executable file*
  - Let me recap you some terminology here: **executable file means the binary file as resides in disk**
- **Why is it possible?**
  - **Windows PE loader pays his debts**. IAT resolved, PE sections removed when mapped into memory (e.g., `.reloc` or Auhtenticode signatures)
  - **Pagination issues** (pages are 4K-byte length, by default)

Universidad Zaragoza

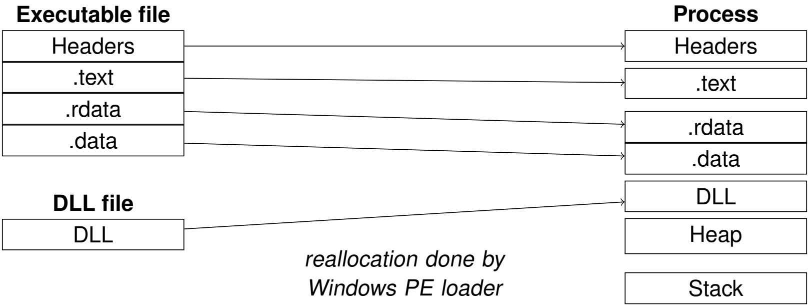
Current Issues and Challenges

Issue #1



Current Issues and Challenges

Issue #1



Our solutions so far

- **Plugin** `ProcessFuzzyHash`: rely on approximation matching algorithms (instead of cryptographic hashes) [RMA18]
- **Plugin** `pefile` **(Python) adapted for undoing the work done by Windows PE loader** (*will be released soon!*)

Current Issues and Challenges
Issue #1

Introducing Approximation Matching Algorithms

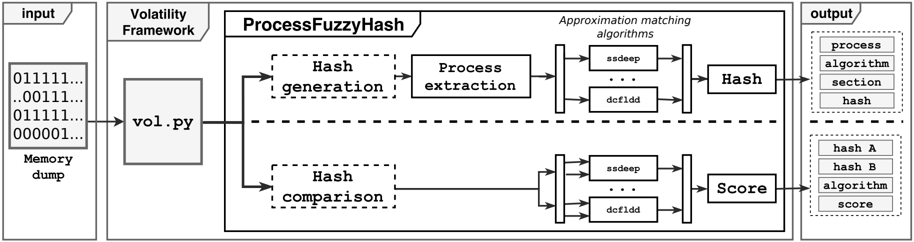
- **Identify similarities between different digital artifacts**
- **Level of granularity:**
  - **Bytewise**: Rely on byte stream
  - **Syntactic**: Rely on internal structure
  - **Semantic**: Use contextual attributes to interpret the artifact
- **Type of similarity:**
  - **Containment**: Identify an object inside an artifact
  - **Resemblance**: Similarity of similar size objects
- **Similarity measure:** $m \in [0, 1]$ $(m \in \mathbb{R})$
  - Versus $m \in \{0, 1\}$ $(m \in \mathbb{Z})$, from cryptographic hashes

Current Issues and Challenges

Issue #1

Plugin `ProcessFuzzyHash`

- Integrates 4 different algorithms for approximate matching hash computation
- Bytewise granularity and resemblance
  - `dcfldd`, `ssdeep`, `SDhash`, and `TLSH`
- Allows (easy) extension to support other algorithms
- Included in the official Volatility Framework (under GNU GPLv3 license)



Current Issues and Challenges

Issue #1

Plugin `ProcessFuzzyHash hashing example`

```
$ python vol.py --plugins=ProcessFuzzyHash/  -f Win7.elf \
> --profile=Win7SP1x86 processfuzzyhash -A ssdeep,SDHash \
> -S pe,.text -N winlogon,services
```

```
Volatility Foundation Volatility Framework 2.6
Name         PID  Create Time Sec   Algori Hash
winlogon.exe 500  131483892000 pe    ssdeep 6144:pzP/qv...
winlogon.exe 500  131483892000 .text ssdeep 768:U+ucmmy...
winlogon.exe 500  131483892000 pe    SDHash sdbf:03:0::...
winlogon.exe 500  131483892000 .text SDHash sdbf:03:0::...
services.exe 544  131483892003 pe    ssdeep 6144:Q/6kXE...
services.exe 544  131483892003 .text ssdeep 1536:9RbbyD...
services.exe 544  131483892003 pe    SDHash sdbf:03:0::...
services.exe 544  131483892003 .text SDHash sdbf:03:0::...
```

Current Issues and Challenges

Issue #1

Plugin `ProcessFuzzyHash comparison example`

```
$ python vol.py --plugins=ProcessFuzzyHash/  -f Win7.elf \
>--profile=Win7SP1x86 processfuzzyhash -A ssdeep -S .text\
> -N svchost -c '768:9n3SsSfvrOtOHW4CO5LTiMRMxVKPhPDjRWWm\
> :d3BGrOtO2NO5LTiqUVKP5/zm'
```

```
Volatility Foundation Volatility Framework 2.6
Hash A            Hash B              Algorithm Score
768:9n3Ss...P5/zm 768:9n3SsS...P5/0m   ssdeep    94
768:9n3Ss...P5/zm 768:9n3SsS...P5/0m   ssdeep    94
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    100
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    97
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    100
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    97
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    97
768:9n3Ss...P5/zm 768:9n3SsS...P5/zm   ssdeep    97
```

Current Issues and Challenges
Issue #2

My process file is missing some pages! 😕

- **Page swapping**
  - The OS stores unused memory pages in a secondary source until those pages are needed again
  - Allows us to use more memory than the actually available in RAM
- **Demand paging** (or *lazy page loading*)
  - The OS does not bring data from files on disk into memory until they are absolutely needed
  - Optimization issue

Current Issues and Challenges

Issue #2

My process file is missing some pages! ☹

- **Page swapping**
  - The OS stores unused memory pages in a secondary source until those pages are needed again
  - Allows us to use more memory than the actually available in RAM
- **Demand paging** (or *lazy page loading*)
  - The OS does not bring data from files on disk into memory until they are absolutely needed
  - Optimization issue

Our solutions so far

- *We have some ideas, but it's still an ongoing work* (next NCN,maybe?)

Universidad Zaragoza

Current Issues and Challenges

Issue #3

Is the extracted data from a dump accurate enough? 😕

- **Page smearing**
  - Memory inconsistency due to the acquired page tables referencing physical pages whose contents changed during the acquisition process
  - **Commonly found on systems with +8GB of RAM or under heavy load**
  - Of course, it **only occurs in acquisitions done in live systems**

Solutions (*we are not facing with this at the moment*)

- Freeze the memory
- Provoke a crash dump
- Check the temporal consistency of data acquired: temporal forensics!

Current Issues and Challenges

Issue #3

Introducing Temporal forensics

- Idea from by Pagani et al. [PFB19]
  - *“we argue that memory forensics should also consider the time in which each piece of data was acquired. This new temporal dimension provides a preliminary way to assess the reliability of a given result and opens the door to new research directions that can minimize the effect of the acquisition time or detect inconsistencies”*
- **Volatility is modified to precisely record time data in a memory dump**
- Currently submitted to Volatility Plugin Contest’19
  - Publicly available at `https://github.com/pagabuc/atomicity_tops`

Output example (extracted from [PFB19])

```
$ ./vol.py -f dump.raw --profile=... --pagetime pslist
<original pslist output>
Accessed physical pages: 171
Acquisition time window: 72s
[XX-------------XxX---xXXX--xX-xX---Xxx-xx-X-XxxX-XXX]
```

Universidad Zaragoza

Current Issues and Challenges

Issue #4

Persistence by means of registry-based Windows 🙁

- **Windows Registry contains volatile hives**
- Furthermore, **not all registry keys are in memory** [D08]
  - Do you remember demand paging?
  - Some on-disk hives are mapped into the memory during Windows start-up

Our solution so far

- **Plugin** `winesap`: detection of Windows registry keys commonly used by malware for persistence [UR19]

Universidad Zaragoza

Current Issues and Challenges
Issue #4

Plugin `winesap`

- **Available under GNU GPLv3**

  `https://gitlab.unizar.es/rrodrigu/winesap`

- Marks suspicious activity depending on Windows registry value:
  - `REG_BINARY` or `REG_NONE` when contains a PE header
  - `REG_SZ`, `REG_EXPAND_SZ`, or `REG_LINK` when contains:
    - Suspicious paths
    - Well-known shell commands that indirectly launch programs (e.g: `rundll32.exe shell32.dll,ShellExecute_RunDLL <filepath>`)

Output example

```
WARNING:
Suspicious path file
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\firefox.exe
Debugger: REG_SZ: C:\Users\me\AppData\Roaming\Yztrpxpt\cmd.exe
------------------------------
WARNING:
Suspicious path file
HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows
AppInit_DLLs: REG_SZ: C:\Users\me\AppData\Roaming\Uxkgoeaoqbf\autoplay.dll
```

Universidad Zaragoza

Current Issues and Challenges

Issue #5

Initial triage for malware detection 🙁

- Help to *separate the sheep from the goats*
- **Provide hits for malware analysts**
  - Binary analysis is a really **tedious and time-consuming task**
- **Common signature methods can be applied**
- *What if the malware code is injected in a process? And if the memory page containing such a code was swapped out of memory?*

Current Issues and Challenges
Issue #5

Initial triage for malware detection 🙁

- Help to *separate the sheep from the goats*
- **Provide hits for malware analysts**
  - Binary analysis is a really **tedious and time-consuming task**
- **Common signature methods can be applied**
- *What if the malware code is injected in a process? And if the memory page containing such a code was swapped out of memory?*

Our solution so far

- **Plugin** `malscan`: warns about suspicious parts of processes, relying on Virtual Address Descriptors (VADs) [D07]

Universidad Zaragoza

Current Issues and Challenges
Issue #5

Plugin `malscan`

- **Integrated with** `clamav-daemon`
  - Limitation: only works for Linux

Universidad Zaragoza

Current Issues and Challenges

Issue #5

Plugin `malscan`

- **Integrated with** `clamav-daemon`
  - Limitation: only works for Linux
- **Two working modes**:
  - **Normal mode**: it analyzes every memory region with W+X permission, every executable module (to detect process hollowing), and private memory regions of type VadS
  - **Full-scan mode**: it analyzes every memory region with +X permission

Universidad Zaragoza

Current Issues and Challenges

Issue #5

Plugin `malscan`

- **Integrated with** `clamav-daemon`
  - Limitation: only works for Linux
- **Two working modes**:
  - **Normal mode**: it analyzes every memory region with W+X permission, every executable module (to detect process hollowing), and private memory regions of type VadS
  - **Full-scan mode**: it analyzes every memory region with +X permission
- **Additional detection mechanisms**:
  - When a VAD exists without an associated executable module
  - Common function prologues (e.g., `push ebp;mov ebp, esp`)
  - Empty page followed by a function prologue (e.g., a process which has intentionally stripped its header)

Let's see an example in a demo...

Universidad Zaragoza

Current Issues and Challenges

More issues ahead

Our problems have not finished yet... [CR17] 😕

- **Window hibernation file analysis**
- **Windows 10**: swap files, Device Guard, Powershell
- **Definition of profiles for memory acquisition in Linux and Android**
- **iOS, Chromebooks, IoT devices** ¿?

Universidad Zaragoza

Agenda

1 Introduction

2 Current Issues and Challenges

3 Conclusions

Universidad Zaragoza

Conclusions

- **Memory forensics brings several issues**
  - Mismatch between executable files in-disk and in-memory
  - Incompleteness (page swapping and demand paging)
  - Inaccurate data on dumps from live systems (page smearing)
  - Windows Registry contains volatile data
  - Lot of memory artifacts to consider
  - And other that are unknown to us at the moment...
- **We can face these challenges: time and human resources** ☺
- **Develop your own plugins to overcome these issues** (or at least to mitigate their effect)
  - `ProcessFuzzyHash`, `winesap`, `malscan`, ...
  - Temporal forensics
  - (and much more other works from many people working in this area, providing good tools and ideas)

Universidad Zaragoza

Conclusions

- **Memory forensics brings several issues**
  - Mismatch between executable files in-disk and in-memory
  - Incompleteness (page swapping and demand paging)
  - Inaccurate data on dumps from live systems (page smearing)
  - Windows Registry contains volatile data
  - Lot of memory artifacts to consider
  - And other that are unknown to us at the moment...
- **We can face these challenges: time and human resources** ☺
- **Develop your own plugins to overcome these issues** (or at least to mitigate their effect)
  - `ProcessFuzzyHash`, `winesap`, `malscan`, ...
  - Temporal forensics
  - (and much more other works from many people working in this area, providing good tools and ideas)

IMPORTANT: do not forget to share your efforts!

Universidad Zaragoza

Conclusions
Some useful references

D07 Dolan-Gavitt, B. *The VAD tree: A process-eye view of physical memory*. Digital Investigation, 2007, 4, 62-64

D08 Dolan-Gavitt, B.*Forensic analysis of the Windows registry in memory*. Digital Investigation, 2008, 5, S26-S32

CR17 Case, A. & Richard, G. G. *Memory forensics: The path forward*. Digital Investigation, 2017, 20, 23-33

RMA18 Rodríguez, R. J.; Martín-Pérez, M. & Abadía, I. *A Tool to Compute Approximation Matching between Windows Processes*. Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018, 313-318

PFB19 Pagani, F.; Fedorov, O. & Balzarotti, D. *Introducing the Temporal Dimension to Memory Forensics*. ACM Trans. Priv. Secur., ACM, 2019, 22 , 9:1-9:21

UR19 Uroz, D. & Rodríguez, R. J. *Characteristics and Detectability of Windows Auto-Start Extensibility Points in Memory Forensics*. Digital Investigation, 2019, 28, S95-S104

Malware Detection in Memory Forensics: Current Issues and Challenges

Ricardo J. Rodríguez

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

Universidad Zaragoza 1542

Dept. of Computer Science and Systems Engineering
University of Zaragoza, Spain

November 16, 2019

NoConName 2019
Barcelona, Spain