Quantifying Paging on Recoverable Data from Windows User-Space Modules

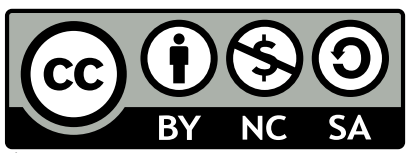
Miguel Martín-Pérez, **Ricardo J. Rodríguez***

Universidad Zaragoza

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

December 7, 2021

12th EAI International Conference on Digital Forensics & Cyber Crime
Singapur



*Corresponding author: rjrodriguez@unizar.es

Outline

Universidad Zaragoza

Outline

Universidad Zaragoza

Introduction

Memory forensics

Memory dump

- **Full of data** to analyze
- **Each item that can be analyzed is called *memory* 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**
- 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 vs. Python3

Universidad Zaragoza

Introduction

A little more of recap...

Malicious software (malware) analysis

- Determine *what the heck* the malware does as harmful activities
- **Static analysis**: executable files are analyzed without being executed
- **Dynamic analysis**: executable files are analyzed when run

Malware in memory

- Unless memory hardware-protection mechanisms are in place, running malware will leave traces of its nefarious activity in memory
- More likely evidence of activity from sophisticated or fileless malware
- We focus on Windows, as it is still the most predominant target of attacks

Universidad Zaragoza

Introduction

The Windows memory subsystem

- **Maps a process virtual address space into physical memory**
- **Manages memory paging**: memory pages are...
  - Paged to disk when the demanding memory of running threads exceeds the available physical memory; and
  - Returned to physical memory when needed

Universidad Zaragoza

Introduction

The Windows memory subsystem

- **Maps a process virtual address space into physical memory**
- **Manages memory paging**: memory pages are...
  - Paged to disk when the demanding memory of running threads exceeds the available physical memory; and
  - Returned to physical memory when needed

Memory page

- **Contiguous fixed-length block of virtual memory**
- Small (4 KiB) and large pages (2 MiB [x86 & x64] to 4 MiB [ARM])
- **Different states**: free, reserved, and committed

Universidad Zaragoza

Introduction

Terminology

- **Image**: executable, shared library, or driver file loaded as part of a process
- **Image file**: the corresponding (on-disk) file
- Processes and images are internally represented by a **module**

Universidad Zaragoza

Introduction

Terminology

- **Image**: executable, shared library, or driver file loaded as part of a process
- **Image file**: the corresponding (on-disk) file
- Processes and images are internally represented by a **module**

Contribution

- Analysis of how paging issues affect the user-mode Windows modules
- Discussion on the issues to detect malware artifacts in memory forensics
- As a side product, `residentmem` Volatility plugin
  - Provides information on the amount of binary data that cannot be analyzed correctly

Universidad Zaragoza

Outline

Universidad Zaragoza

Related work

- **Identify all user allocations and determine their purpose** using kernel & user-space metadata sources via Virtual Address Descriptors (VAD)
- **Parse of PTE in Linux via a kernel module** (`PageDumper`)
- **Malware detection in memory forensics**
  - VMI, YARA, machine learning approaches
  - Detection of hidden malware via PTE or GPU memory

Universidad Zaragoza
1542

Related work

- **Identify all user allocations and determine their purpose** using kernel & user-space metadata sources via Virtual Address Descriptors (VAD)
- **Parse of PTE in Linux via a kernel module** (`PageDumper`)
- **Malware detection in memory forensics**
  - VMI, YARA, machine learning approaches
  - Detection of hidden malware via PTE or GPU memory

- **First study that quantifies the effect of paging in Windows user-space modules**
- Our work is **complementary to all those**

Universidad Zaragoza

Outline

Universidad Zaragoza

Characterization of the Windows paging mechanism

- **Paging issues in user-space modules** on a Windows 10 64-bit system (build 19041) with 4GiB and 8GiB RAM memory
- **Different memory workloads**: 25%, 50%, 75%, 100%, 125%, and 150%
  - We developed a naif tool that allocates memory and writes a random byte every 4KiB

Universidad Zaragoza
1542

Characterization of the Windows paging mechanism

- **Paging issues in user-space modules** on a Windows 10 64-bit system (build 19041) with 4GiB and 8GiB RAM memory
- **Different memory workloads**: 25%, 50%, 75%, 100%, 125%, and 150%
  - We developed a naif tool that allocates memory and writes a random byte every 4KiB
- System memory acquired at various runtimes for each memory workload
  - *First observation moment*: every 15 seconds for the first minute, every minute for 4 more minutes, while allocating memory
  - *Second observation moment*: same pattern, after stopping the memory allocator tool

Universidad Zaragoza

Characterization of the Windows paging mechanism

- **Paging issues in user-space modules** on a Windows 10 64-bit system (build 19041) with 4GiB and 8GiB RAM memory
- **Different memory workloads**: 25%, 50%, 75%, 100%, 125%, and 150%
  - We developed a naif tool that allocates memory and writes a random byte every 4KiB
- System memory acquired at various runtimes for each memory workload
  - *First observation moment*: every 15 seconds for the first minute, every minute for 4 more minutes, while allocating memory
  - *Second observation moment*: same pattern, after stopping the memory allocator tool
- **Side product of our research**: `residentmem`
  - Volatility2 plugin, GNU/GPLv3. `https://github.com/reverseame/residentmem`
  - Extracts the number of resident pages (that is, in memory) of each image and each process within a memory dump
  - Provides forensic analysts with information on the amount of binary data that cannot be analyzed correctly

Universidad Zaragoza
1542

Characterization of the Windows paging mechanism

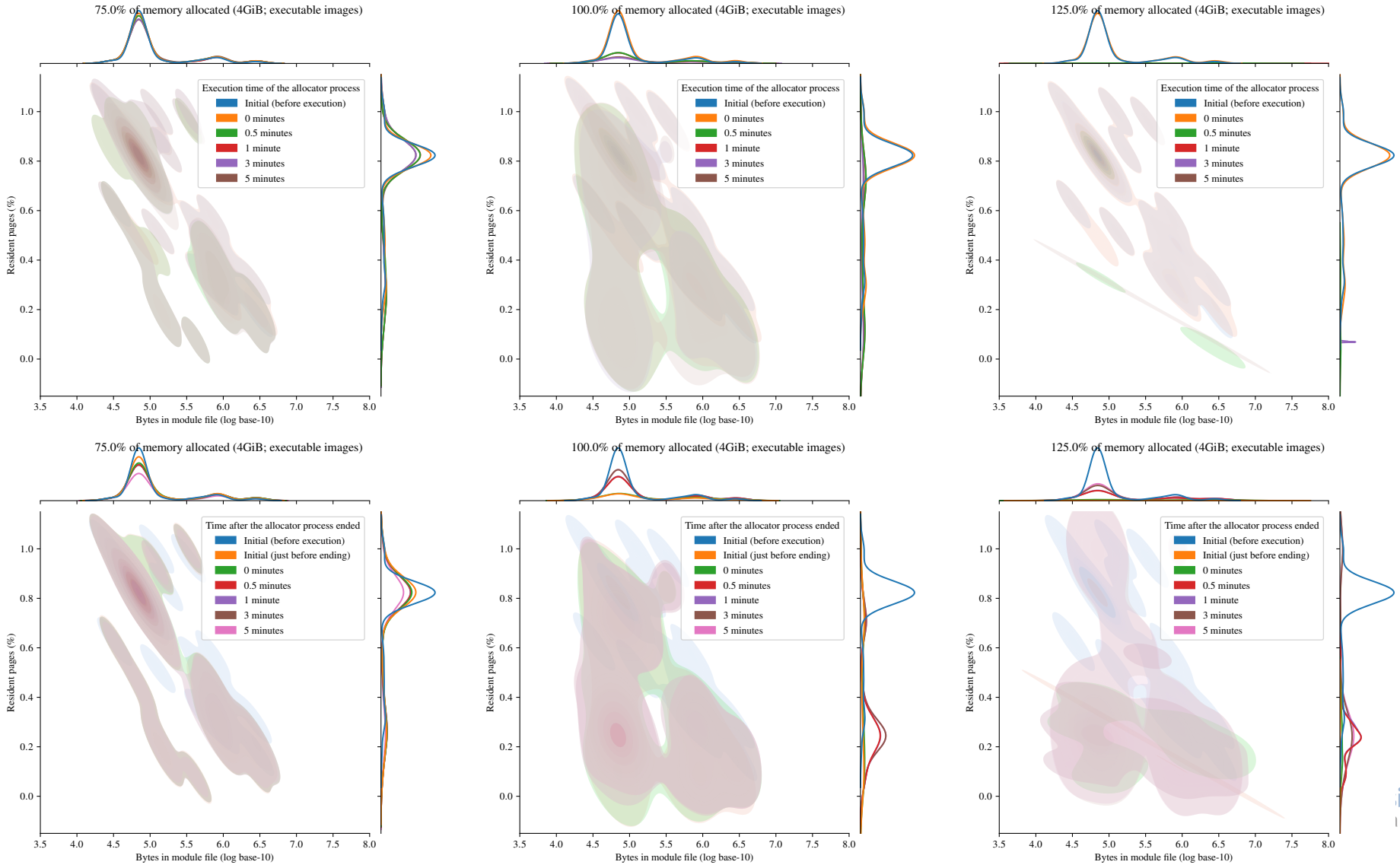
Description of the plots

- **Distributions of two variables**
  - Size of a module file in log-base 10 (x-axis)
  - Percentage of resident pages (y-axis)
- Color intensity: **the darker the region, the more data is in that region**
- **Subplots reveal the distribution** of resident pages and module file sizes

Universidad Zaragoza

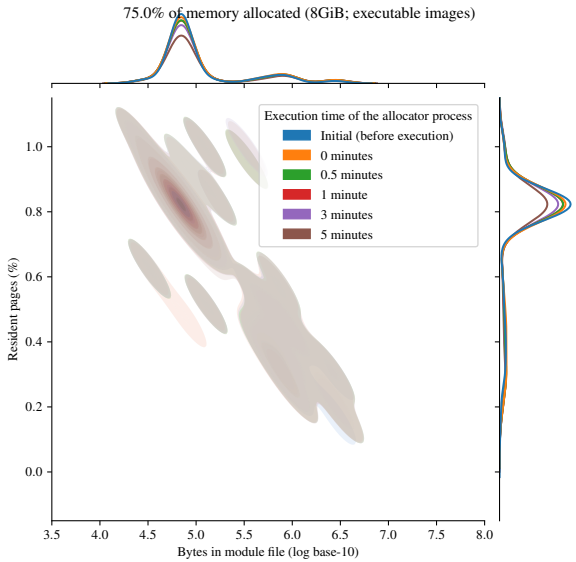
Characterization of the Windows paging mechanism

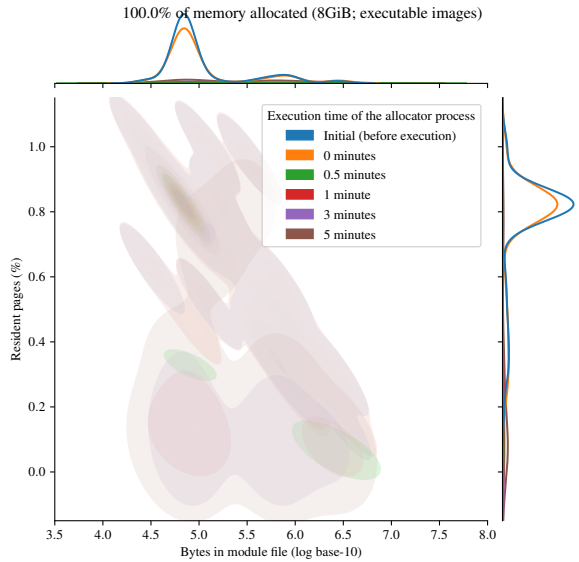
Results of quantification (4GiB; `exe`)

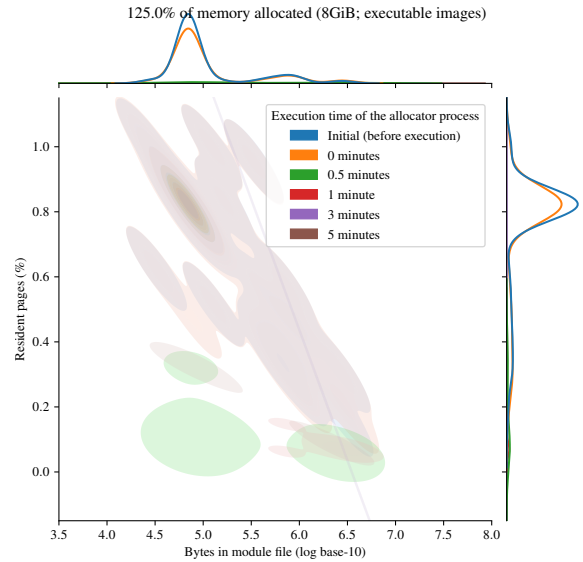


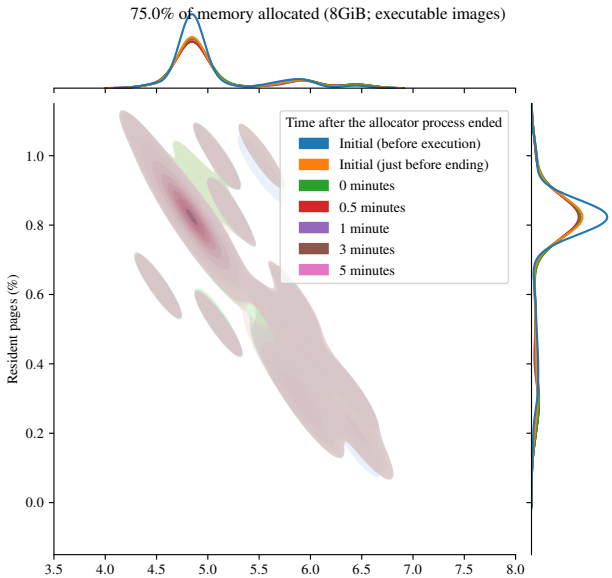
Characterization of the Windows paging mechanism

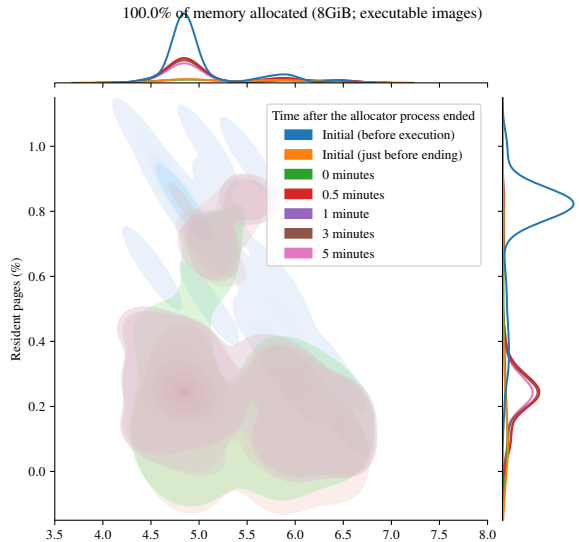
Results of quantification (8GiB; `exe`)

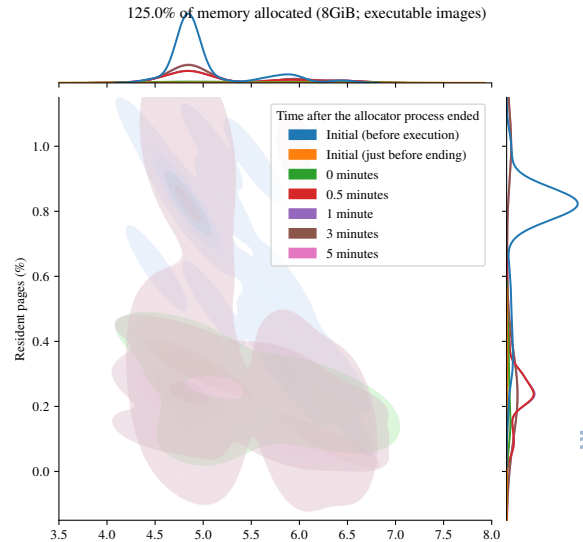












Characterization of the Windows paging mechanism

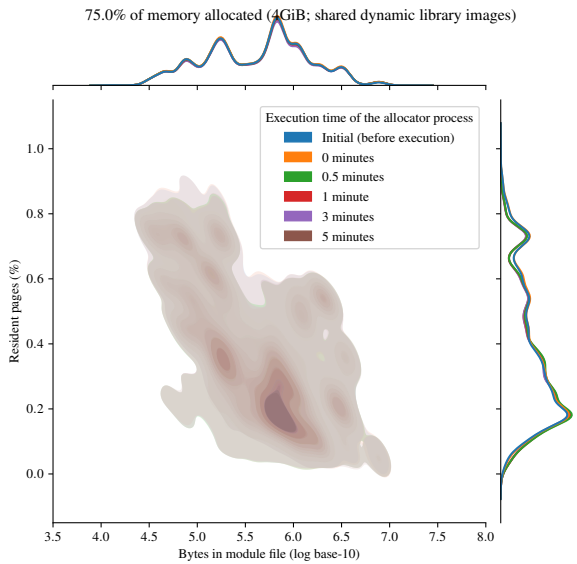
Discussion on executable modules

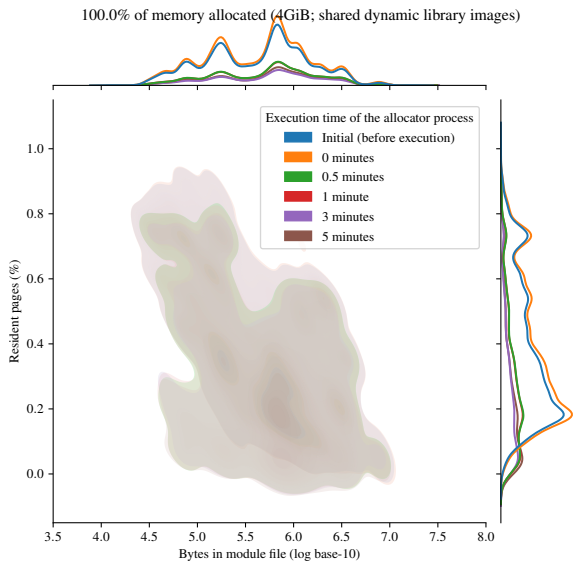
- **Almost 80% of the executable module pages are resident** in memory
- With 100% and 125%, in 0.5 minutes:
  - **Most modules are expelled**
  - The number of resident pages for retrievable modules is drastically reduced
- Modules progressively come back to memory, after memory exhaustion
  - **Ratio of resident pages for retrievable modules** $\leq 25\%$
  - Significant increases in 0.5 minutes and in 3 minutes are observed

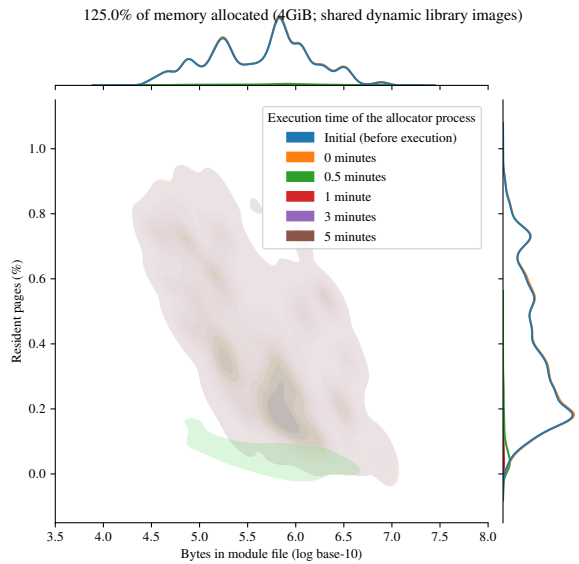
Universidad Zaragoza

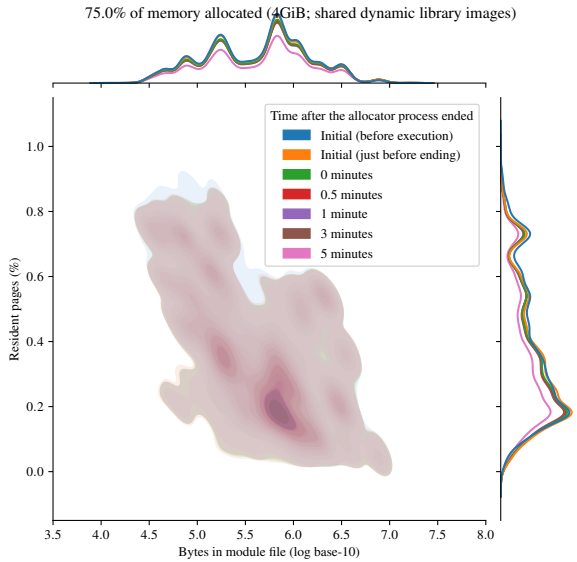
Characterization of the Windows paging mechanism

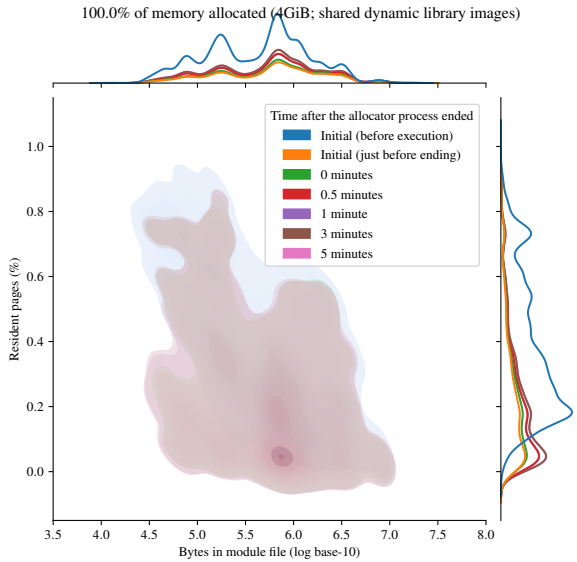
Results of quantification (4GiB; `dll`)

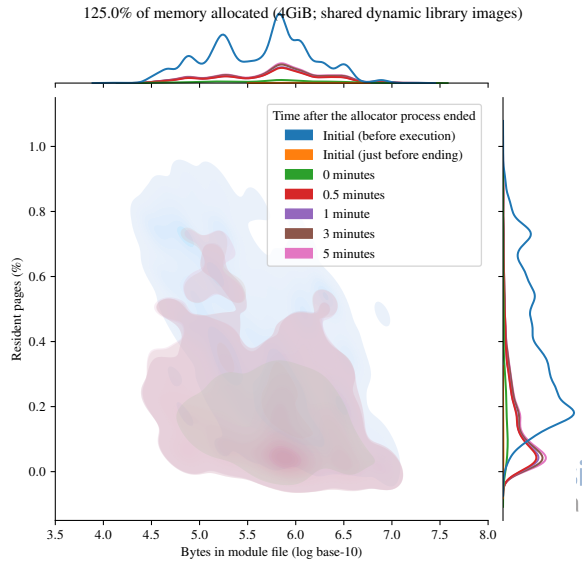






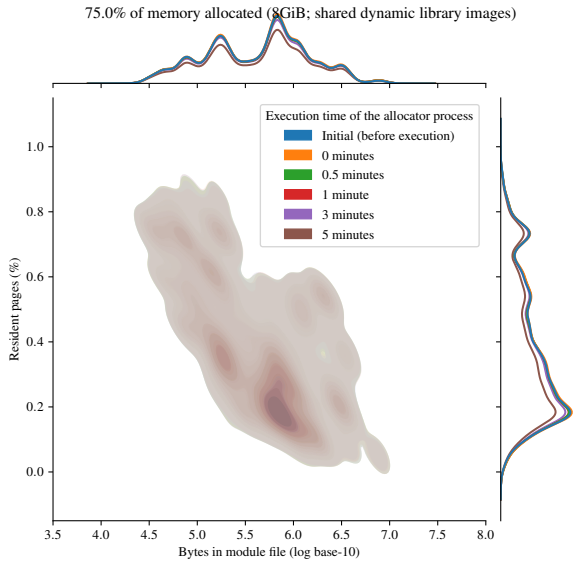


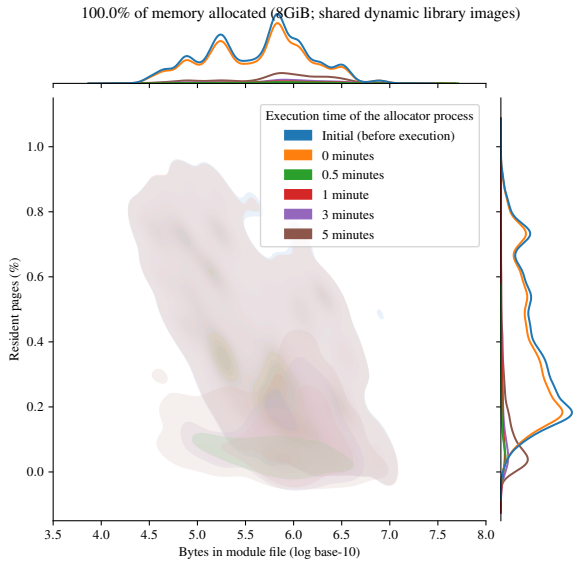


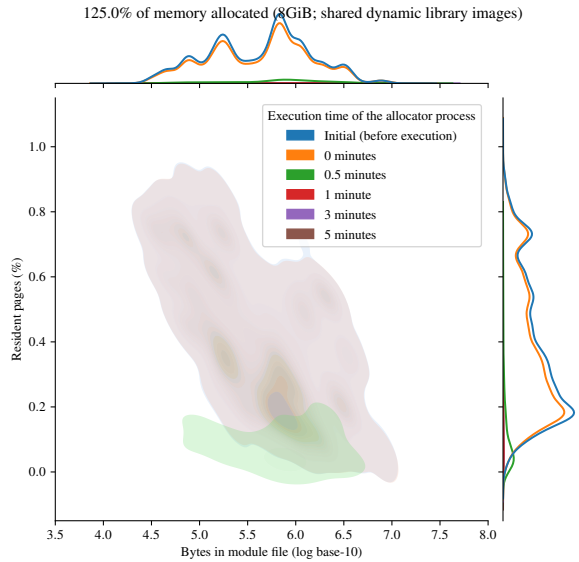


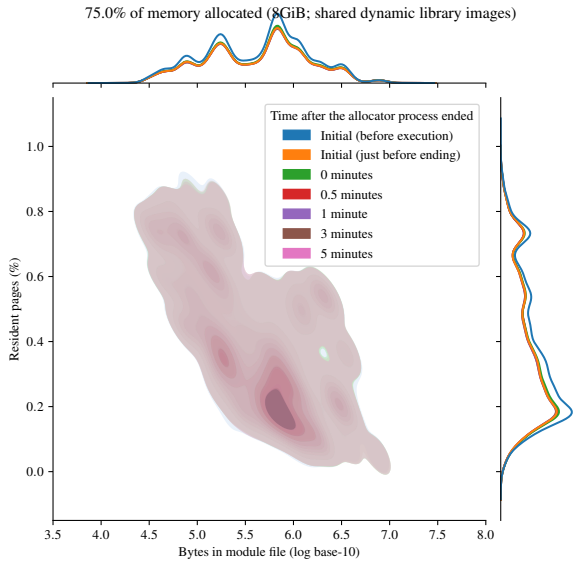
Characterization of the Windows paging mechanism

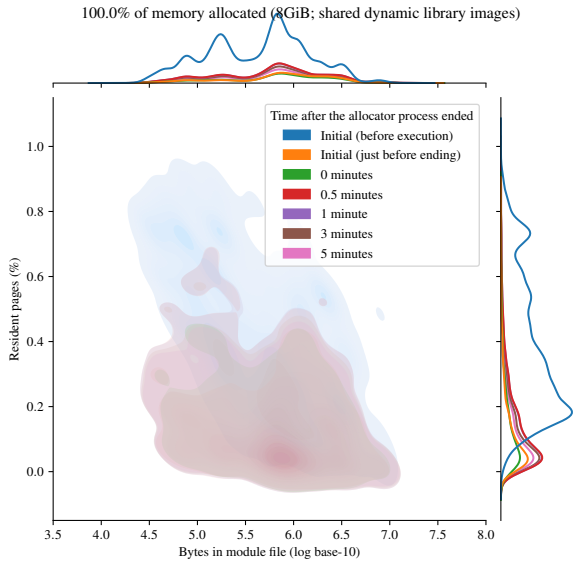
Results of quantification (8GiB; `dll`)

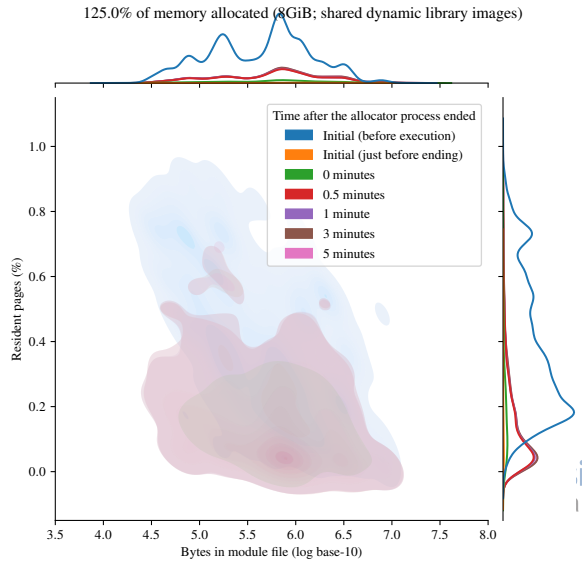












Characterization of the Windows paging mechanism

Discussion on shared library modules

- Modules only have **20% of their pages resident**, with a maximum percentage observed of 75%
- With 100% and 125%, **in 0.5 minutes the system starts expelling them**
  - Distribution shape is similar in both memory configurations
  - Aggressive expelling of modules is observed in 8GiB
- **Most modules have only less than 5% of their pages resident, after memory exhaustion**

Universidad Zaragoza

Outline

Universidad Zaragoza

Detection of Malware in Memory Forensics

Inaccuracy of the content of memory artifacts

The content of an image is inaccurate
(relative to its image file)

Everything happens for a reason...

- **Paging effect**
  - Image file mapped into 4KiB aligned memory regions (assuming small pages)
  - As a consequence, a zero padding may appear
- **Relocation**
  - Addresses of external functions resolved (e.g., IAT functions)
  - PE sections removed (e.g., `.reloc` or Authenticode signatures)

Universidad Zaragoza
1542

Detection of Malware in Memory Forensics

Inaccuracy of the content of memory artifacts

The content of an image is inaccurate
(relative to its image file)

Everything happens for a reason...

- **Paging effect**
  - Image file mapped into 4KiB aligned memory regions (assuming small pages)
  - As a consequence, a zero padding may appear
- **Relocation**
  - Addresses of external functions resolved (e.g., IAT functions)
  - PE sections removed (e.g., `.reloc` or Authenticode signatures)

Feasible solutions

- Use approximate matching algorithms to calculate similarity
- Constructions of allow-list hash databases

Universidad Zaragoza
1542

Detection of Malware in Memory Forensics

Incompleteness of images

The content of an image is incomplete

(relative to its image file)

Everything happens for a reason...

- **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 is actually available in RAM
- **Demand paging** (or *lazy page loading*)
  - The OS does not bring data from files on disk to memory until it is absolutely necessary
  - Optimization issue

Universidad Zaragoza

Detection of Malware in Memory Forensics

Incompleteness of images

The content of an image is incomplete
(relative to its image file)

Everything happens for a reason...

- **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 is actually available in RAM
- **Demand paging** (or *lazy page loading*)
  - The OS does not bring data from files on disk to memory until it is absolutely necessary
  - Optimization issue

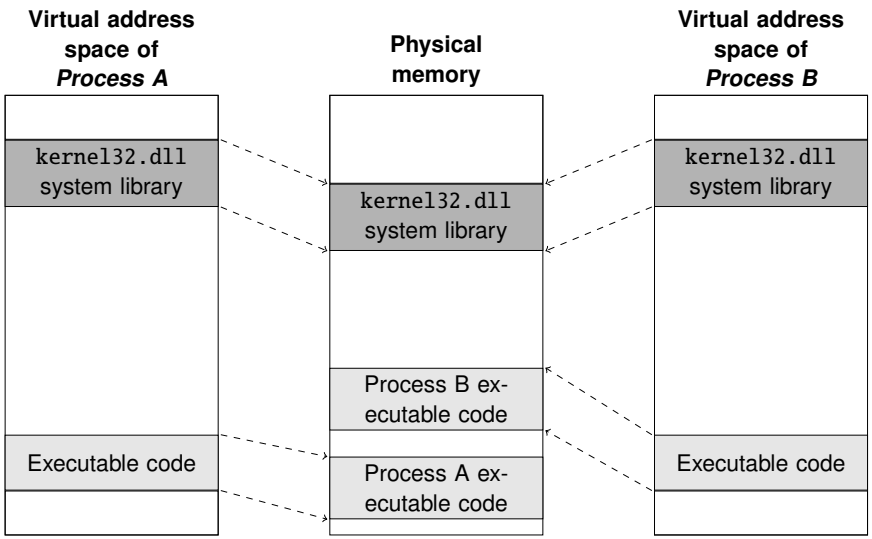
Feasible solutions

- Use disk forensics to first recover the page files and then use them together with the memory dump
- Combine memory forensics with disk forensics

Universidad Zaragoza
1542

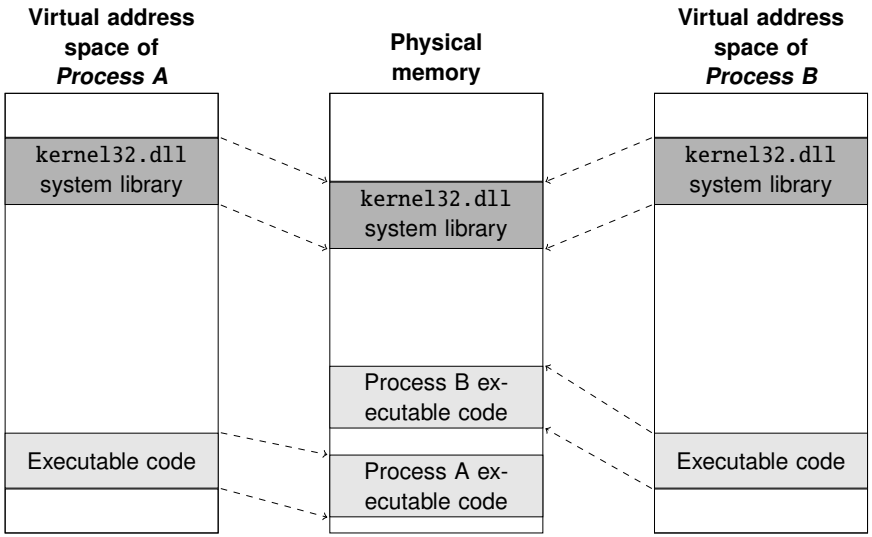
Detection of Malware in Memory Forensics

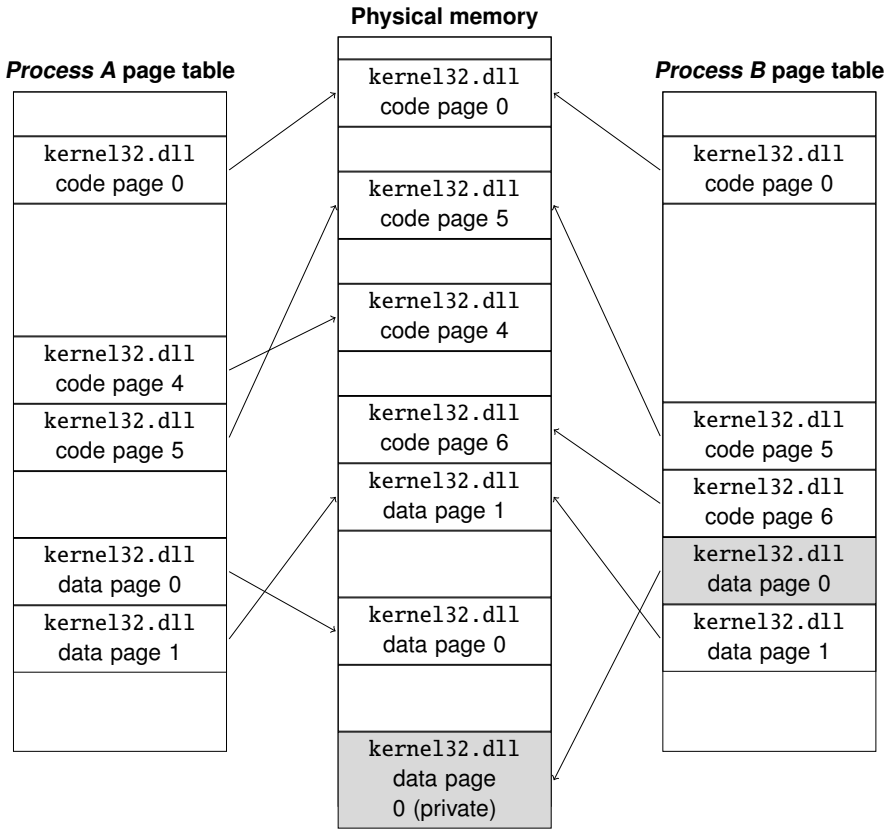
Incompleteness of images



Detection of Malware in Memory Forensics

Incompleteness of images





Detection of Malware in Memory Forensics

Inaccuracy of a memory dump

- Memory is **continually updated and acquired non-atomic**: *page smearing*
  - Particularly relevant when the memory is acquired in a live system
- Highly likely to occur:
  - **Pointer inconsistency**
  - **Fragment inconsistency**
- **Sophisticated malware can deliberately produce these inconsistencies** (DKOM attacks)

Universidad Zaragoza

Detection of Malware in Memory Forensics

Inaccuracy of a memory dump

- Memory is **continually updated and acquired non-atomic**: *page smearing*
  - Particularly relevant when the memory is acquired in a live system
- Highly likely to occur:
  - **Pointer inconsistency**
  - **Fragment inconsistency**
- **Sophisticated malware can deliberately produce these inconsistencies** (DKOM attacks)

Feasible solutions

- Use other acquisition techniques
- Check the temporal consistency of data in a memory dump: *temporal forensics*

Universidad Zaragoza

Detection of Malware in Memory Forensics

Stealthy malware

- VAD are **unreliable source of information**
  - Page permissions are not updated when changed after initial permissions
- **Deliberately triggering of the paging process** for as many pages as possible
- **Process hollowing**

Universidad Zaragoza

Detection of Malware in Memory Forensics

Stealthy malware

- VAD are **unreliable source of information**
  - Page permissions are not updated when changed after initial permissions
- **Deliberately triggering of the paging process** for as many pages as possible
- **Process hollowing**

Feasible solutions

- Malware signatures (but not with cryptohashes!)
- Robust kernel signatures
- Volatility plugins: `malfind`, `malscan`, `impfuzzy`

Universidad Zaragoza

Outline

Universidad Zaragoza

Conclusions

- **Memory dumps are unreliable and partial sources of evidence**
- **Effect of paging in Windows modules of the user-space processes**
  - At first, almost 80% of the executable module pages and 20% of the shared dynamic library module pages are resident. Drastically reduced when the OS needs memory
  - Once the memory load is no longer high, the system recovers some of the paged modules but very slowly, never returning to the initial conditions (25% and 5% for executable and shared library image files, respectively)
- **Problems for malware detection in memory forensics**
  - Data in an image differs from its image file and is incomplete, inaccurate, and unreliable
  - Malware can incorporate features to remain stealthy and hidden from memory forensics

Universidad Zaragoza
1542

Conclusions

- **Memory dumps are unreliable and partial sources of evidence**
- **Effect of paging in Windows modules of the user-space processes**
  - At first, almost 80% of the executable module pages and 20% of the shared dynamic library module pages are resident. Drastically reduced when the OS needs memory
  - Once the memory load is no longer high, the system recovers some of the paged modules but very slowly, never returning to the initial conditions (25% and 5% for executable and shared library image files, respectively)
- **Problems for malware detection in memory forensics**
  - Data in an image differs from its image file and is incomplete, inaccurate, and unreliable
  - Malware can incorporate features to remain stealthy and hidden from memory forensics

Future work

- Study other versions of Windows, apart from Windows 10 (build 19041)
- Better characterize paging distributions under different system workloads
- Quantify the effects of paging on the kernel space
- Investigate new methods to detect stealthy malware in memory forensics

Quantifying Paging on Recoverable Data from Windows User-Space Modules

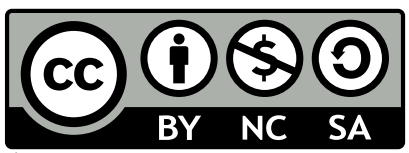
Miguel Martín-Pérez, **Ricardo J. Rodríguez***

Universidad Zaragoza

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

December 7, 2021

12th EAI International Conference on Digital Forensics & Cyber Crime
Singapur



*Corresponding author: rjrodriguez@unizar.es