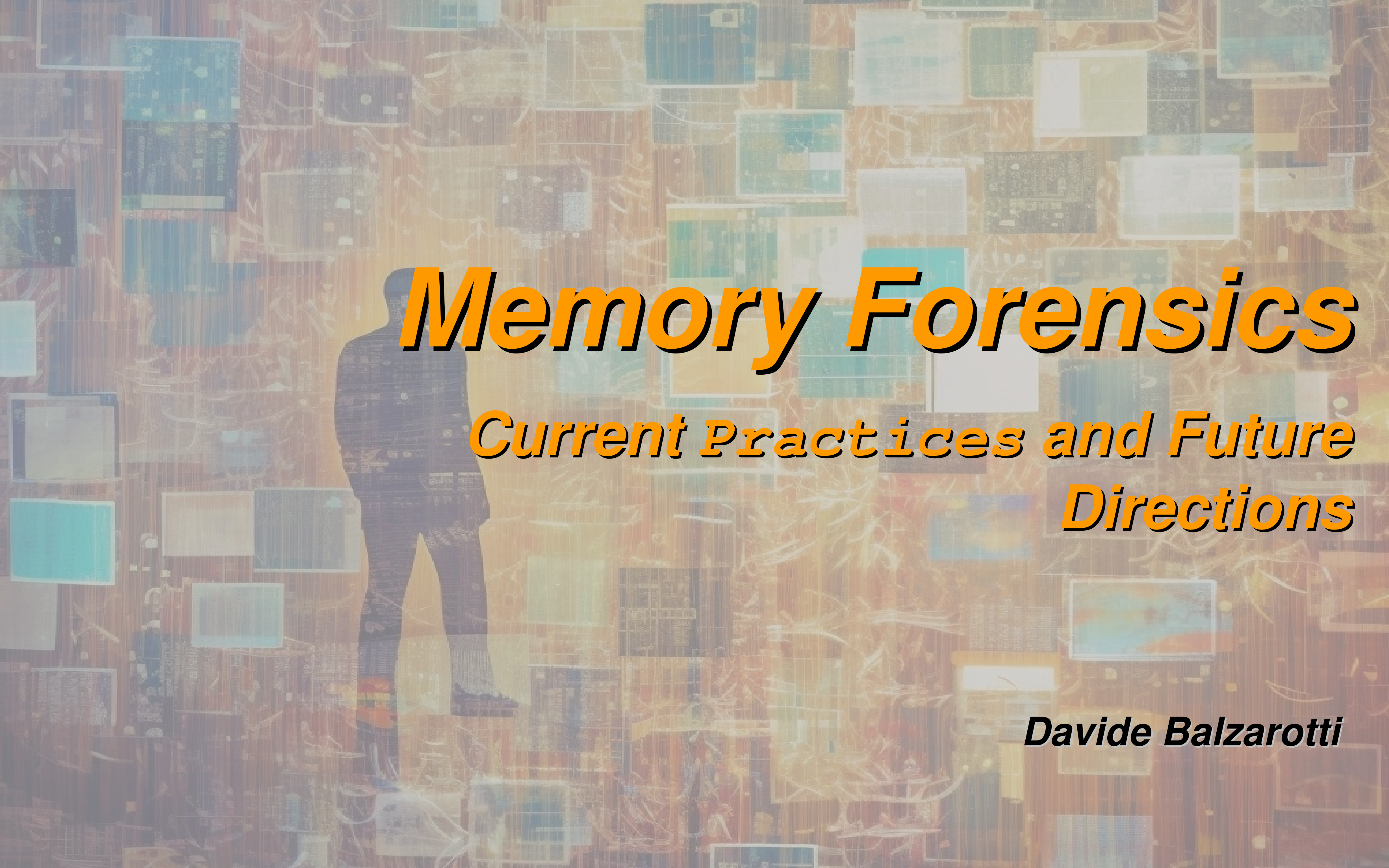


Memory Forensics

Current Practices and Future Directions

Davide Balzarotti

http://s3.eurecom.fr/~balzarot/

@balzarot

Professor of Computer Security @ EURECOM

...
Binary Analysis
Malware
Web Security
Fuzzing
Memory Forensics
....

http://s3.eurecom.fr/~balzarot/

@balzarot

Professor of Computer Security @ EURECOM

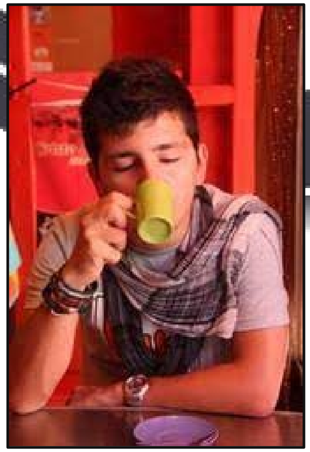
Security CIRCUS

https://www.s3.eurecom.fr/~balzarot/security-circus

...
Binary Analysis
Malware
Web Security
Fuzzing
Memory Forensics
....

SHELLPHISH

ORDER OF THE OVERFLOW



Mariano Graziano

Fabio Pagani

Andrea Oliveri

fo·ren·sic

Adjective: Of, relating to, or denoting the application of *scientific methods and techniques* to the *investigation* of crime

Memory forensics

The *preservation*, *collection*, *validation*, *identification*, *analysis*, *interpretation*, *documentation*, and *presentation* of digital evidences extracted from the memory

(my definition)

Memory forensics:

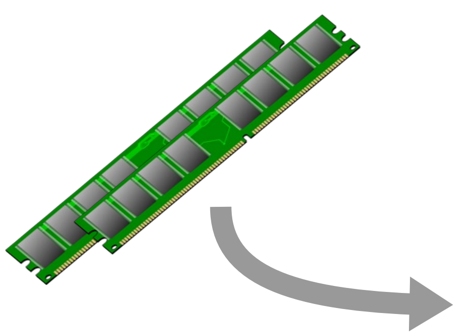
Reverse Engineering on Steroids

Pros

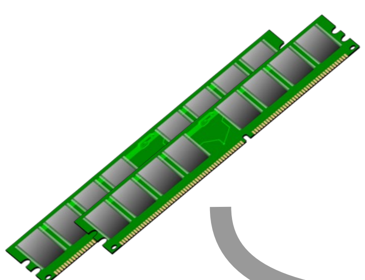
- Attackers often overlook their memory footprint
- Many of the kernel artifacts can be used for forensics
- Even rootkits designed to hide data in a running system need to be located somewhere in memory
- Certain information (loaded kernel modules, open sockets, ...) may be difficult to extract otherwise
- Some malware samples only reside in memory

Cons

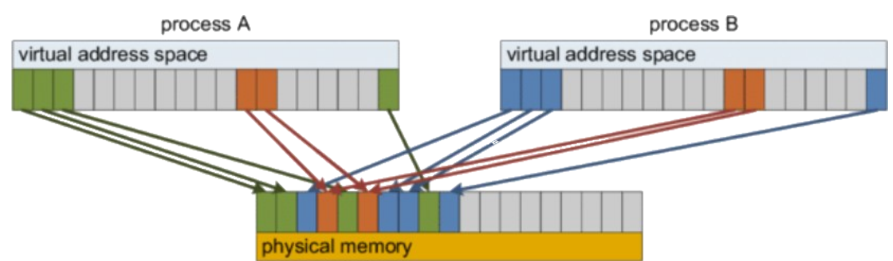
- Memory is difficult to acquire
- The content of the memory keeps changing so even consecutive image acquisitions give different results
- Data collection requires an efficient approach with a small footprint
- Data structures change among different OSs and OS versions

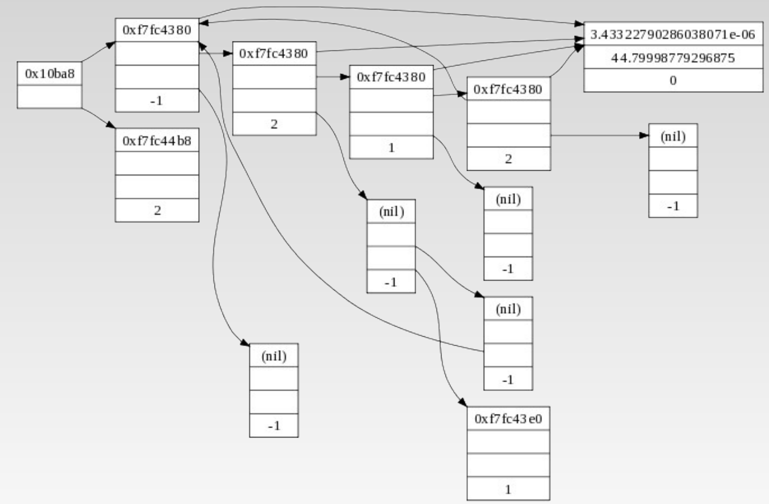


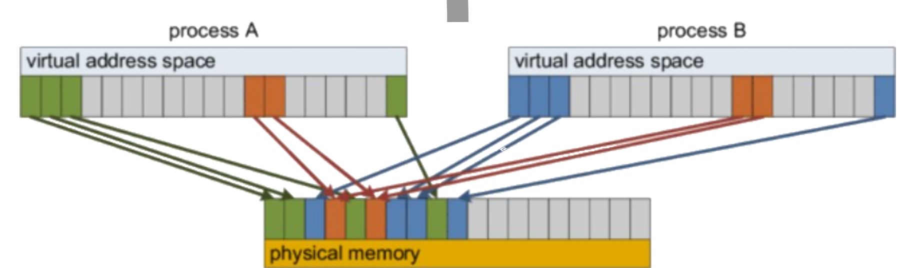
```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```

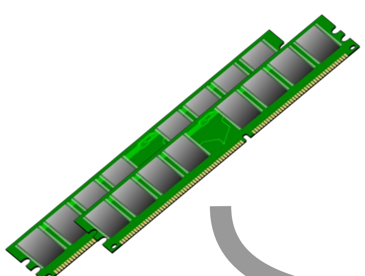


```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```

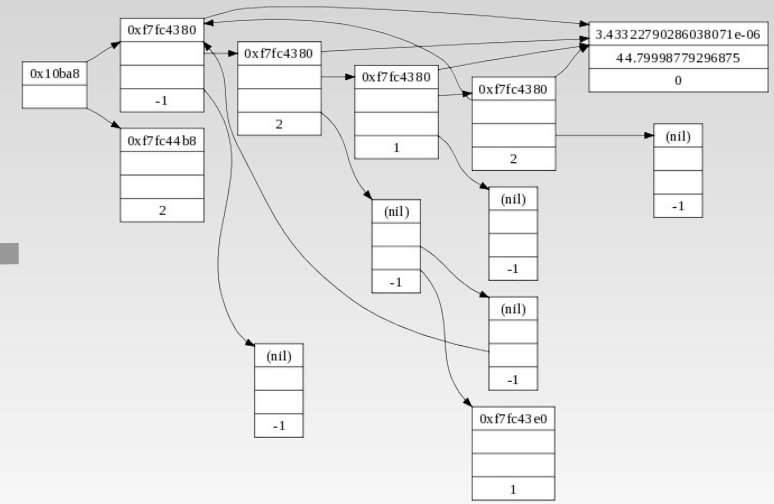


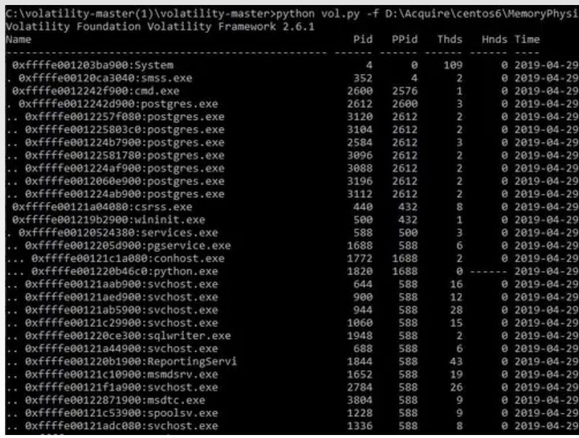




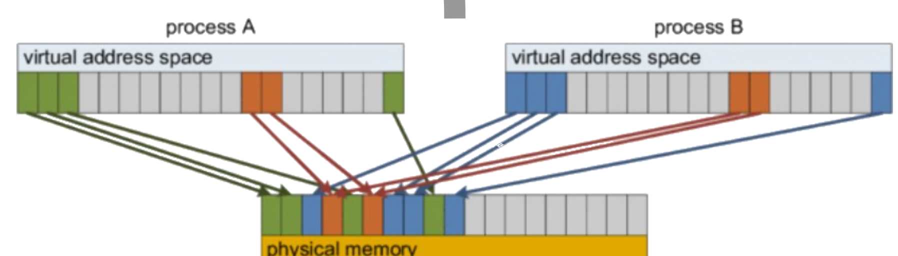


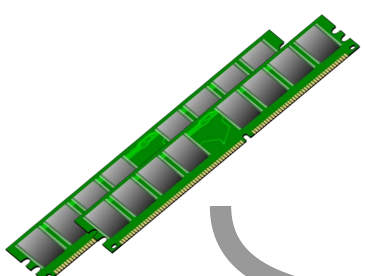
```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```





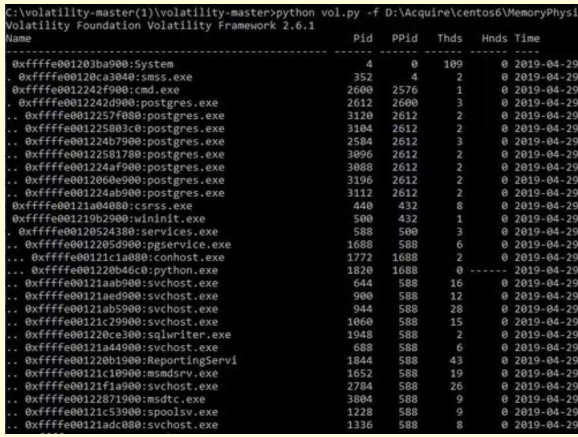
```
C:\volatility-master(1)\volatility-master>python vol.py -f D:\Acquire\centos6\MemoryPhysic
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xffffe001203ba900:System                              4      0    109      0 2019-04-29
. 0xffffe00120ca3040:smss.exe                         352      4      2      0 2019-04-29
 0xffffe0012242f900:cmd.exe                          2600   2576      1      0 2019-04-29
. 0xffffe0012242d900:postgres.exe                    2612   2600      3      0 2019-04-29
.. 0xffffe0012257f080:postgres.exe                   3120   2612      2      0 2019-04-29
.. 0xffffe001225803c0:postgres.exe                   3104   2612      2      0 2019-04-29
.. 0xffffe001224b7900:postgres.exe                   2584   2612      3      0 2019-04-29
.. 0xffffe00122581780:postgres.exe                   3096   2612      2      0 2019-04-29
.. 0xffffe001224af900:postgres.exe                   3088   2612      2      0 2019-04-29
.. 0xffffe0012060e900:postgres.exe                   3196   2612      2      0 2019-04-29
.. 0xffffe001224ab900:postgres.exe                   3112   2612      2      0 2019-04-29
 0xffffe00121a04080:csrss.exe                         440    432      8      0 2019-04-29
 0xffffe001219b2900:wininit.exe                       500    432      1      0 2019-04-29
. 0xffffe00120524380:services.exe                     588    500      3      0 2019-04-29
.. 0xffffe0012205d900:pgservice.exe                  1688    588      6      0 2019-04-29
... 0xffffe00121c1a080:conhost.exe                   1772   1688      2      0 2019-04-29
... 0xffffe001220b46c0:python.exe                    1820   1688      0 ------ 2019-04-29
.. 0xffffe00121aab900:svchost.exe                     644    588     16      0 2019-04-29
.. 0xffffe00121aed900:svchost.exe                     900    588     12      0 2019-04-29
.. 0xffffe00121ab5900:svchost.exe                     944    588     28      0 2019-04-29
.. 0xffffe00121c29900:svchost.exe                    1060    588     15      0 2019-04-29
.. 0xffffe001220ce300:sqlwriter.exe                  1948    588      2      0 2019-04-29
.. 0xffffe00121a44900:svchost.exe                     688    588      6      0 2019-04-29
.. 0xffffe001220b1900:ReportingServi                 1844    588     43      0 2019-04-29
.. 0xffffe00121c10900:msmdsrv.exe                    1652    588     19      0 2019-04-29
.. 0xffffe00121f1a900:svchost.exe                    2784    588     26      0 2019-04-29
.. 0xffffe00122871900:msdtc.exe                      3804    588      9      0 2019-04-29
.. 0xffffe00121c53900:spoolsv.exe                    1228    588      9      0 2019-04-29
.. 0xffffe00121adc080:svchost.exe                    1336    588      8      0 2019-04-29
```



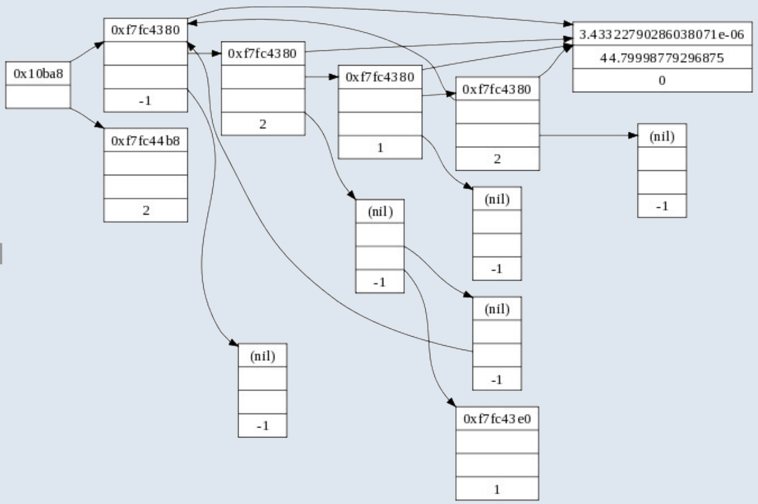


```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```

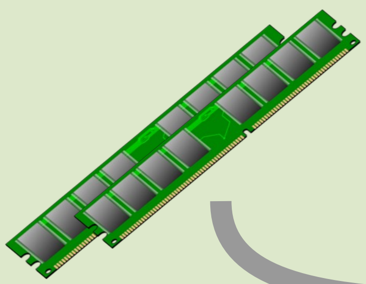
Investigation



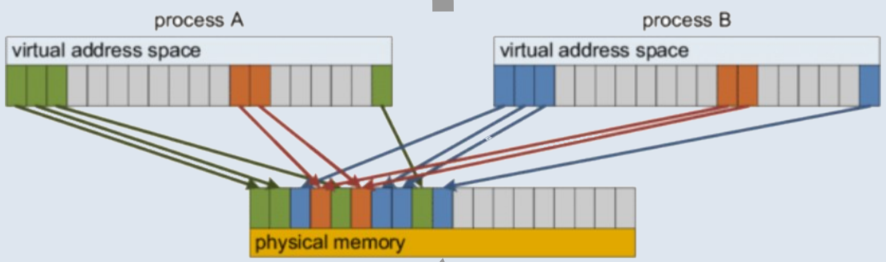
```
C:\volatility-master(1)\volatility-master>python vol.py -f D:\Acquire\centos6\MemoryPhysi
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xffffe001203ba900:System                              4      0    109      0 2019-04-29
. 0xffffe00120ca3040:smss.exe                         352      4      2      0 2019-04-29
 0xffffe0012242f900:cmd.exe                          2600   2576      1      0 2019-04-29
. 0xffffe0012242d900:postgres.exe                    2612   2600      3      0 2019-04-29
.. 0xffffe0012257f080:postgres.exe                   3120   2612      2      0 2019-04-29
.. 0xffffe001225803c0:postgres.exe                   3104   2612      2      0 2019-04-29
.. 0xffffe001224b7900:postgres.exe                   2584   2612      3      0 2019-04-29
.. 0xffffe00122581780:postgres.exe                   3096   2612      2      0 2019-04-29
.. 0xffffe001224af900:postgres.exe                   3088   2612      2      0 2019-04-29
.. 0xffffe0012060e900:postgres.exe                   3196   2612      2      0 2019-04-29
.. 0xffffe001224ab900:postgres.exe                   3112   2612      2      0 2019-04-29
 0xffffe00121a04080:csrss.exe                         440    432      8      0 2019-04-29
 0xffffe001219b2900:wininit.exe                       500    432      1      0 2019-04-29
. 0xffffe00120524380:services.exe                     588    500      3      0 2019-04-29
.. 0xffffe0012205d900:pgservice.exe                  1688    588      6      0 2019-04-29
... 0xffffe00121c1a080:conhost.exe                   1772   1688      2      0 2019-04-29
... 0xffffe001220b46c0:python.exe                    1820   1688      0 ------ 2019-04-29
.. 0xffffe00121aab900:svchost.exe                     644    588     16      0 2019-04-29
.. 0xffffe00121aed900:svchost.exe                     900    588     12      0 2019-04-29
.. 0xffffe00121ab5900:svchost.exe                     944    588     28      0 2019-04-29
.. 0xffffe00121c29900:svchost.exe                    1060    588     15      0 2019-04-29
.. 0xffffe001220ce300:sqlwriter.exe                  1948    588      2      0 2019-04-29
.. 0xffffe00121a44900:svchost.exe                     688    588      6      0 2019-04-29
.. 0xffffe001220b1900:ReportingServi                 1844    588     43      0 2019-04-29
.. 0xffffe00121c10900:msmdsrv.exe                    1652    588     19      0 2019-04-29
.. 0xffffe00121f1a900:svchost.exe                    2784    588     26      0 2019-04-29
.. 0xffffe00122871900:msdtc.exe                      3804    588      9      0 2019-04-29
.. 0xffffe00121c53900:spoolsv.exe                    1228    588      9      0 2019-04-29
.. 0xffffe00121adc080:svchost.exe                    1336    588      8      0 2019-04-29
```



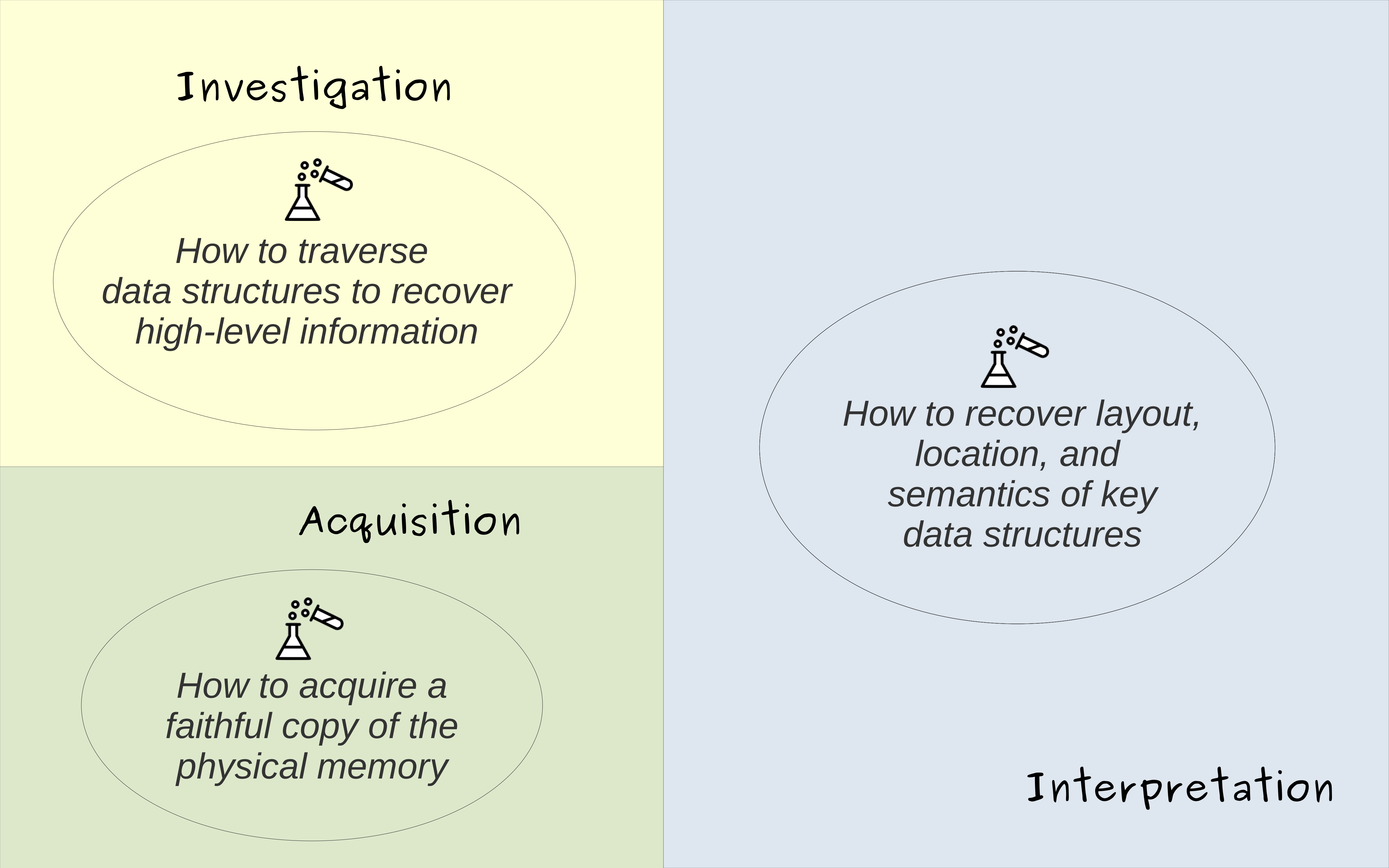
Acquisition



```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```



Interpretation


Investigation
How to traverse data structures to recover high-level information
Acquisition
How to acquire a faithful copy of the physical memory
How to recover layout, location, and semantics of key data structures
Interpretation

(pre-2005)
Carving

Memory Forensics 0.1

Looking for something you do not know in something you know

Looking for something you know in something you do not know

Looking for something you do not know in something you know

– Structured Data Analysis –

Looking for something you know in something you do not know

– Carving –

Rules/Heuristics
Manually Written

Memory Forensics 1.0

Very Time Consuming

Ad-Hoc

Rules/Heuristics

Manually Written

Difficult to Port to other Systems

Ad-Hoc

Very Time Consuming

Rules/Heuristics Manually Written

Difficult to Port to other Systems

Ad-Hoc

Very Time Consuming

Rules/Heuristics
Manually Written

Alternatives?

Lack of metrics to assess
Precision, Reliability, Robustness…

Difficult to Port to other Systems

Very Time Consuming

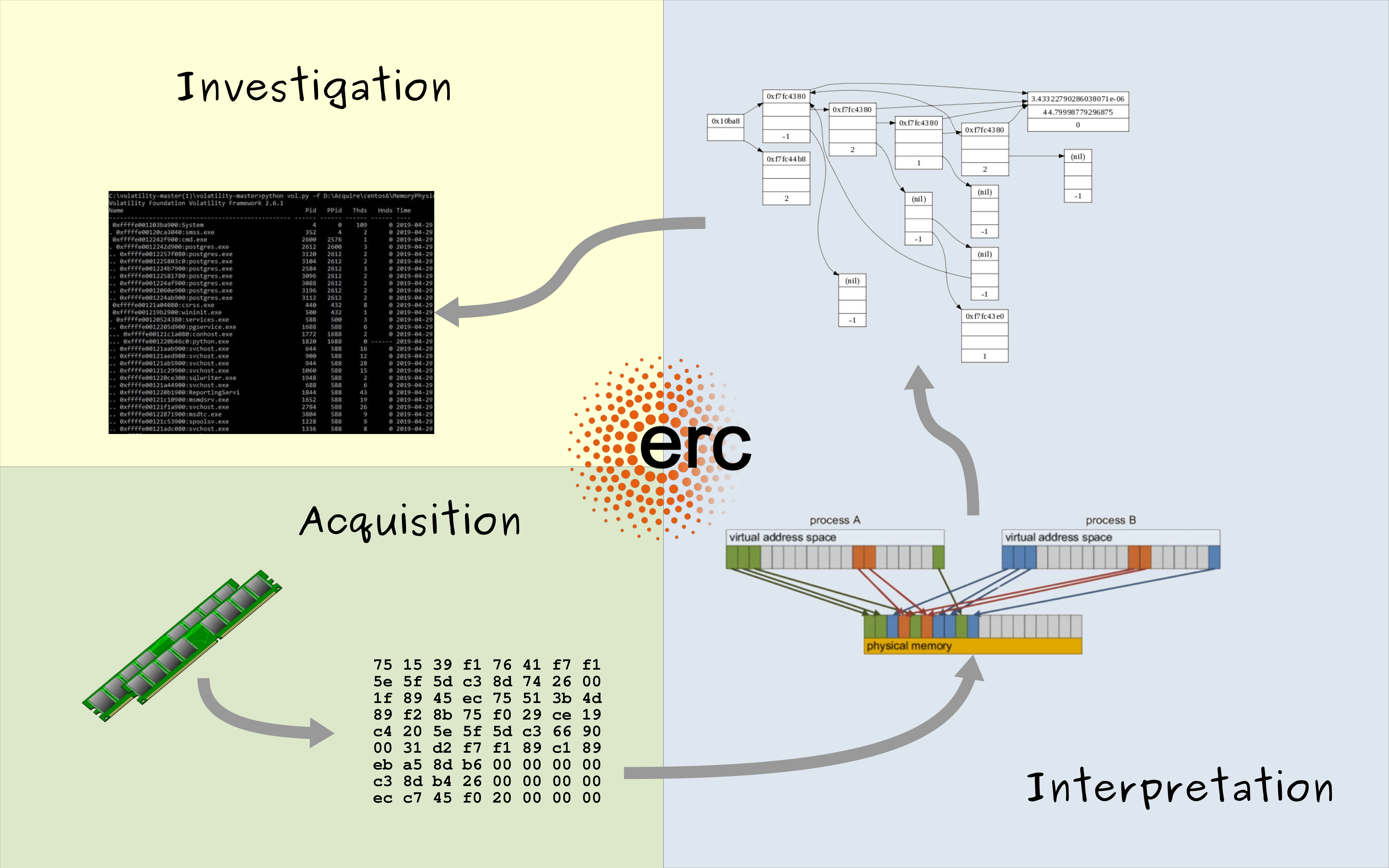
Ad-Hoc

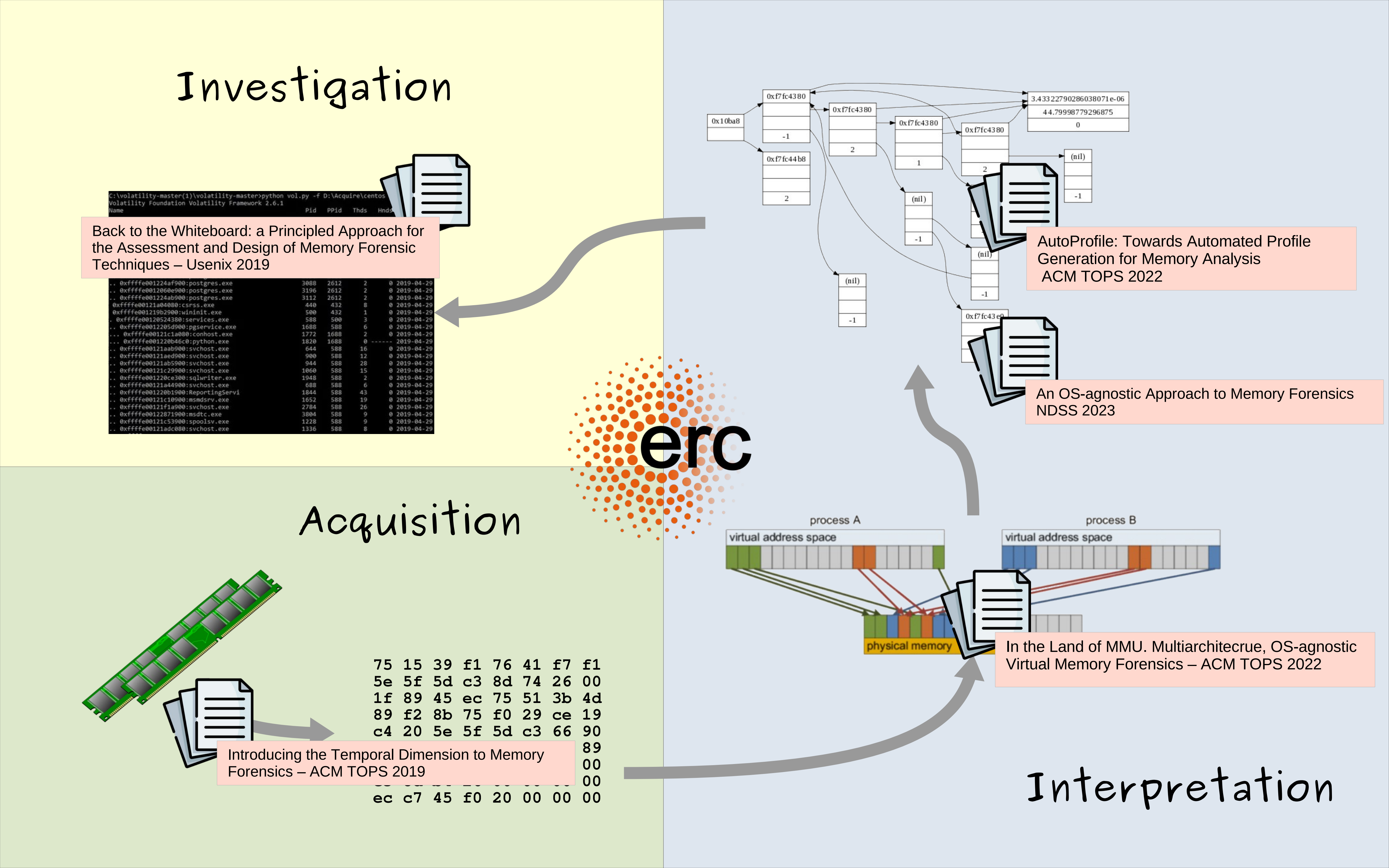
Rules/Heuristics
Manually Written

Alternatives?

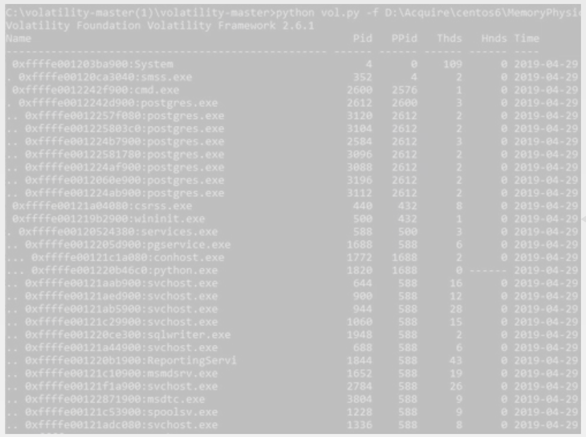
Lack of metrics to assess
Precision, Reliability, Robustness...

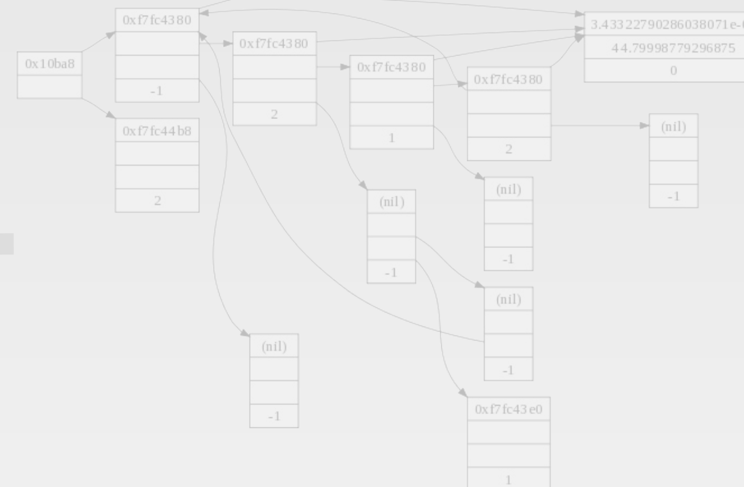
Hard to compare different options


Investigation
Acquisition
Interpretation
erc
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
process A
process B
virtual address space
virtual address space
physical memory
0x10ba8
0xf7fc4380
0xf7fc44b8
0xf7fc43e0
(nil)
3.43322790286038071e-06
44.79998779296875
0
-1
2
1
Volatility Foundation Volatility Framework 2.6.1

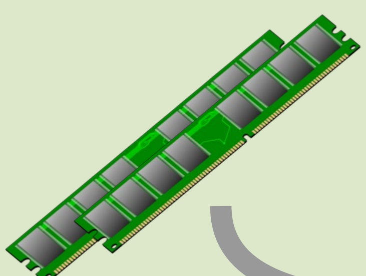

Investigation
Back to the Whiteboard: a Principled Approach for the Assessment and Design of Memory Forensic Techniques – Usenix 2019
AutoProfile: Towards Automated Profile Generation for Memory Analysis ACM TOPS 2022
An OS-agnostic Approach to Memory Forensics NDSS 2023
erc
process A
virtual address space
process B
virtual address space
physical memory
In the Land of MMU. Multiarchitecrue, OS-agnostic Virtual Memory Forensics – ACM TOPS 2022
Acquisition
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
ec c7 45 f0 20 00 00 00
Introducing the Temporal Dimension to Memory Forensics – ACM TOPS 2019
Interpretation

Investigation





Acquisition



```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```

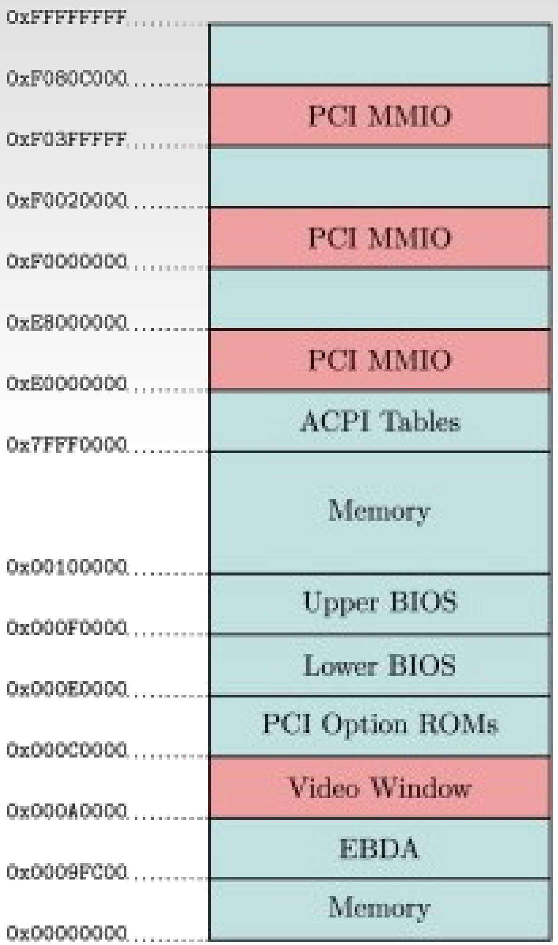
Interpretation

The physical address space is NOT contiguous:

> sudo cat /proc/iomem

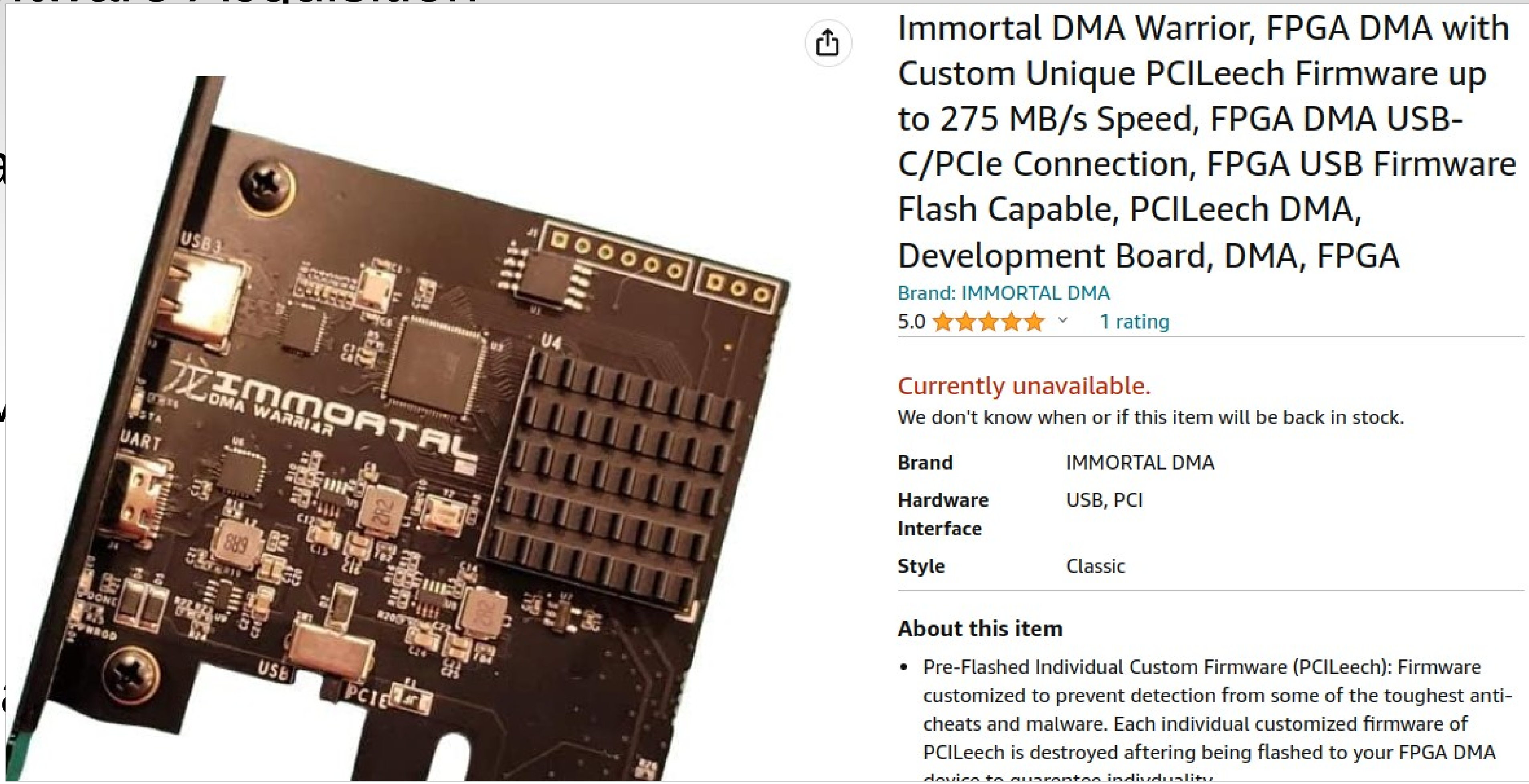
Hardware peripherals map registers or parts of their integrated memory into the physical address space via **Memory Mapped I/O**

Any attempt to read the memory mapped to a device would probably crash the system



- Software Acquisition
    - Use software to read and dump the memory from within the system
- Hardware Acquisition
    - Access memory from DMA
    - Firewire, PCI-Express, USB 4, Intel DCI, Jtag
- VM Acquisition
    - Atomic acquisition
    - New technologies like AMD Secure Encrypted Virtualization can block any type of memory dump from the hypervisor
- Crash dumps, hybernation files, ..
- Cold boot attacks

- Software Acquisition
- Ha
- VM
- Cr
- Cold boot attacks



The Problem of (lack of) Atomicity

A complete memory acquisition takes several minutes, during which the OS is running

The Problem of (lack of) Atomicity

A complete memory acquisition takes several minutes, during which the OS is running

+

When idle, the Linux kernel performs over 300K write operations per second

The Problem of (lack of) Atomicity

A complete memory acquisition takes several minutes, during which the OS is running

+

When idle, the Linux kernel performs over 300K write operations per second

=

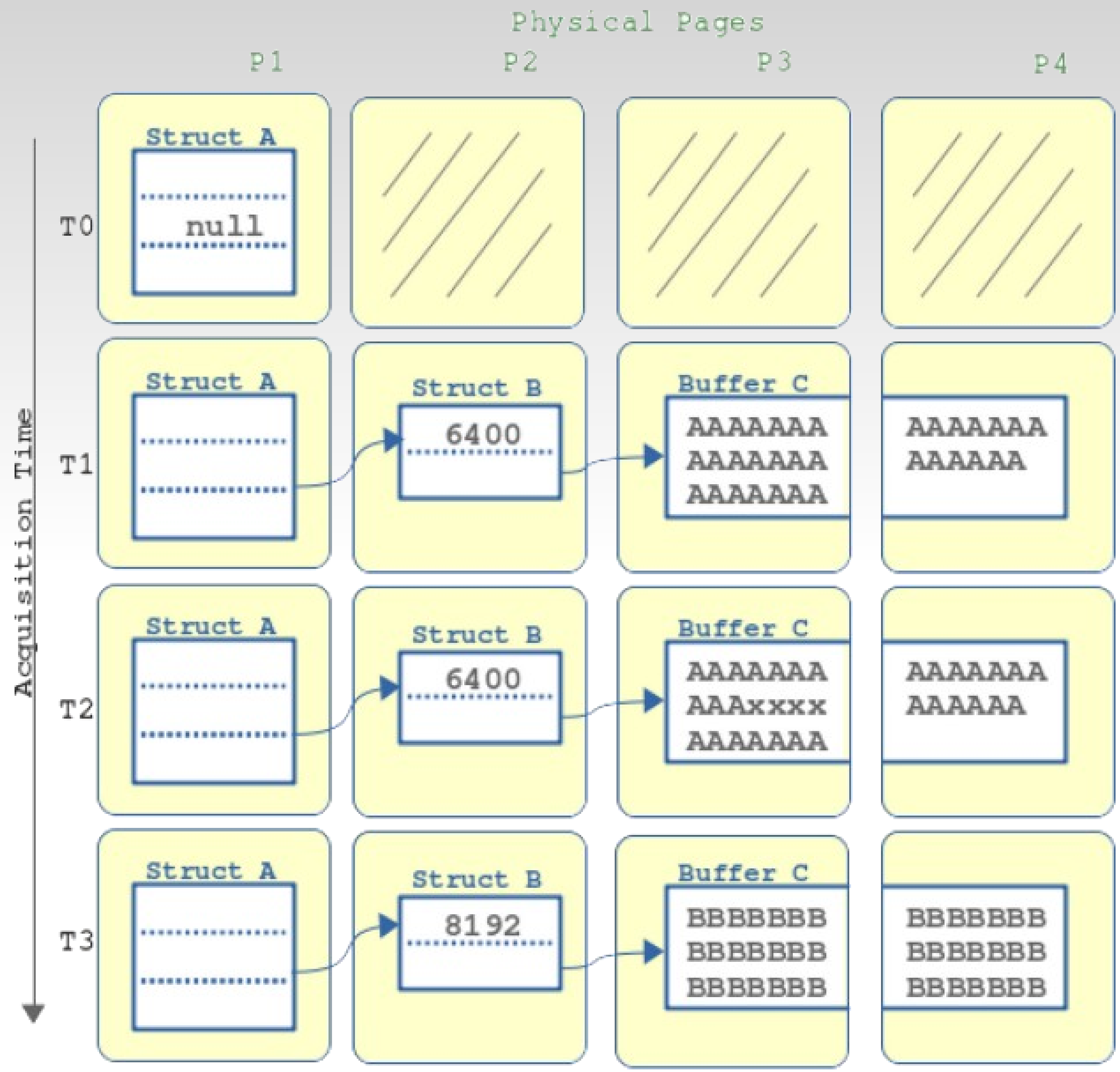
Mode	Writes on kernel address space (Millions)		Writes on MMIO regions		Total size (GiB)		Unique physical pages		Time required (ratio)	
	USB	SATA	USB	SATA	USB	SATA	USB	SATA	USB	SATA
Btrfs	874	811	59824	37778	6.01	5.58	249340	249204	1.81x	1.53x
exFAT	1005	938	96112	61772	6.89	6.43	251074	250728	1.79x	1.33x
Ext4	818	757	60692	35696	5.61	5.20	249421	248873	1.76x	1.16x
Ext4 no journal	776	719	61744	36443	5.33	4.95	249439	248864	1.78x	1.10x
F2FS	951	910	61329	36743	6.51	6.23	249406	249379	2.04x	1.33x
NTFS	796	739	61329	38711	5.48	5.09	249411	249129	1.75x	1.31x
FAT32	1404	1317	84456	89542	9.65	9.06	250328	250908	2.39x	1.82x
XFS	632	569	57605	34255	4.41	3.97	249405	249041	1.62x	1x
Btrfs D. I/O	49137	37708	9147061	6707022	344.04	265.19	75037	78885	255.76x	73.00x
exFAT D. I/O	10698	4713	5204034	3950081	73.20	32.11	466	497	109.34x	15.85x
FAT32 D. I/O	16657	6000	9138277	5773820	114.15	40.88	1125	1127	236.71x	19.58x
Network	1336	-	1000453	-	8.64	-	488	-	2.73x	-

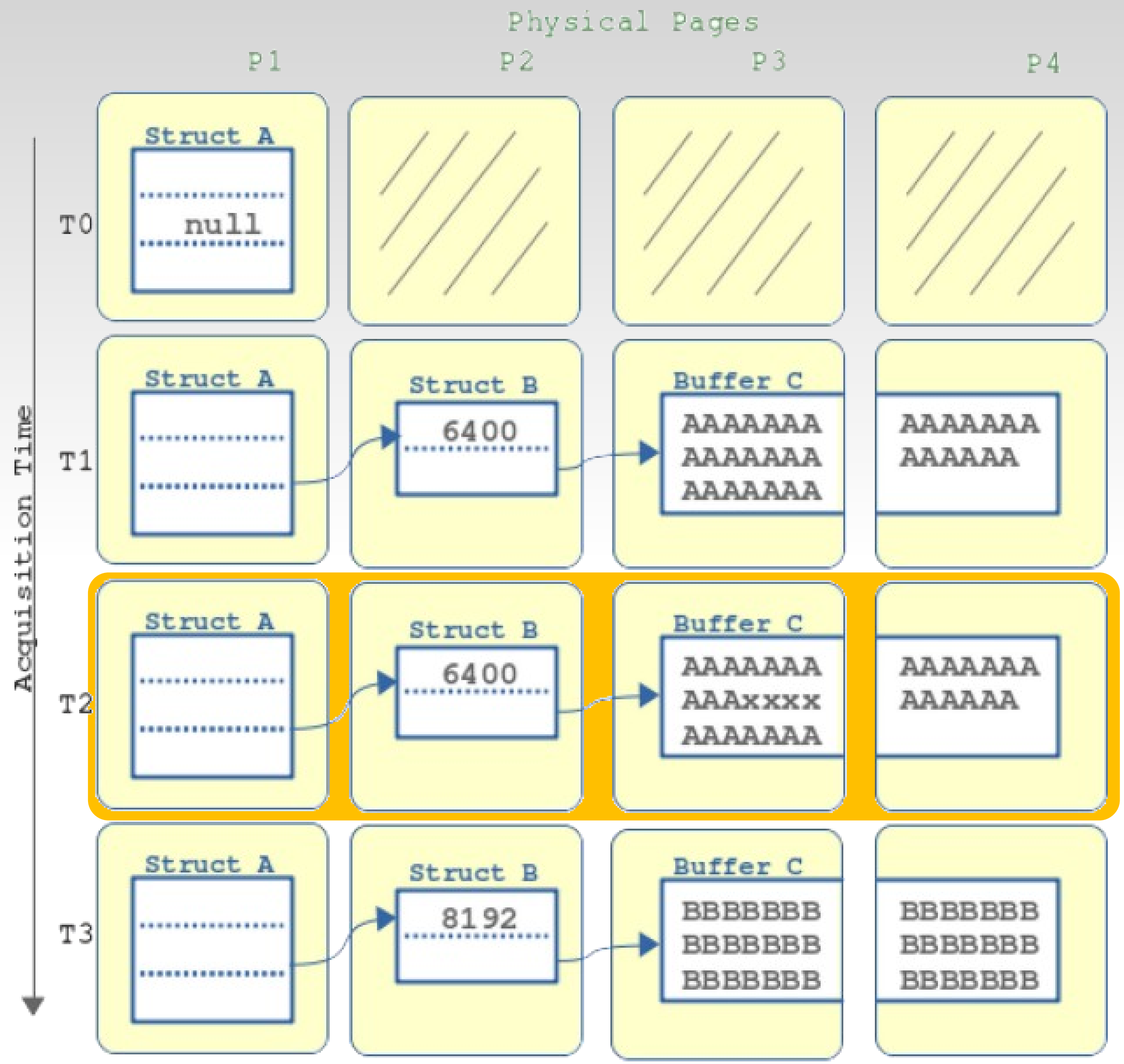
Fresh from the Lab

Re: Digital forensics of the physical memory

From: Harlan Carvey <keydet89 () yahoo com>
Date: Fri, 17 Jun 2005 09:35:16 -0700 (PDT)

```
One of the issues in particular is that he starts off
by mentioning the FU rootkit and the SQL Slammer worm,
both of which are specific to Windows...and then
presents examples using only a Linux system.  He
states in the paper that similar work can be done on
Windows systems, but never provided any information to
that effect.

Based on entries I made to my blog the other day, I
ended up having a conversation w/ someone from MS
about this very issue.  The issue of using dd.exe to
image Physical Memory goes beyond the fact that there
don't seem to be any maps describing how physical
memory is used by Windows systems, and that memory
used by processes consists of both RAM and the
pagefile.  Additional issues include, as you pointed
out, that while the imaging process is occurring, the
kernel memory (and even user-mode memory) is
changing...so what you end up with is a smear, for
want of a better term.

Even tools like pmdump.exe and LiveKD
(SysInternals.com) are not sufficient for collecting
user-mode memory, b/c they do not lock or suspend
memory.
```


Physical Pages
P1
P2
P3
P4
Acquisition Time
T0
T1
T2
T3
Struct A
null
Struct B
6400
8192
Buffer C
AAAAAAA
AAAxxxx
AAAAAA
BBBBBBB



Physical Pages
P1
P2
P3
P4
Acquisition Time
T0
T1
T2
T3
Struct A
null
Struct B
6400
8192
Buffer C
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAA
AAAxxxx
BBBBBBB


Physical Pages
P1
P2
P3
P4
Acquisition Time
T0
T1
T2
T3
Struct A
null
Struct B
6400
8192
Buffer C
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAA
AAAxxxx
BBBBBBB
BBBBBBB
BBBBBBB


Physical Pages
P1
P2
P3
P4
Acquisition Time
T0
T1
T2
T3
Struct A
null
Struct A
Struct B
6400
Buffer C
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAAA
AAAAAA
Struct A
Struct B
6400
Buffer C
AAAAAAA
AAAxxxx
AAAAAAA
AAAAAAA
AAAAAA
Struct A
Struct B
8192
Buffer C
BBBBBBB
BBBBBBB
BBBBBBB
BBBBBBB
BBBBBBB
BBBBBBB

Introducing the Temporal Dimension to Memory Forensics

```
$ ./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]
```

Ongoing experiments repeated on 10 dumps

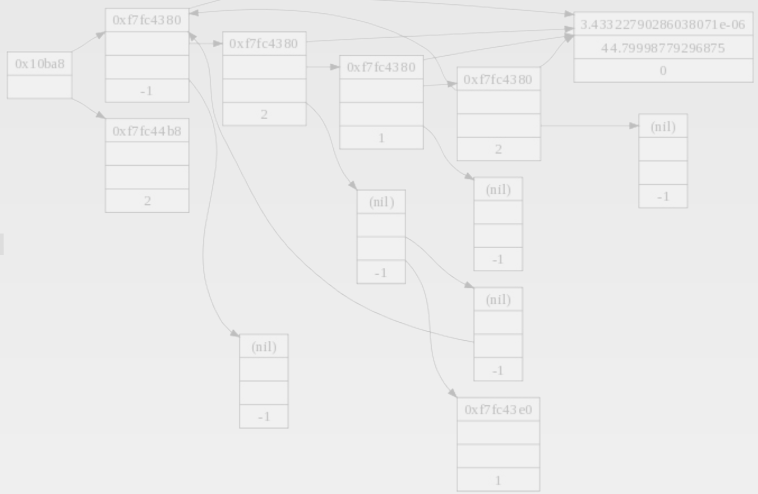
ALL contain inconsistencies in page tables

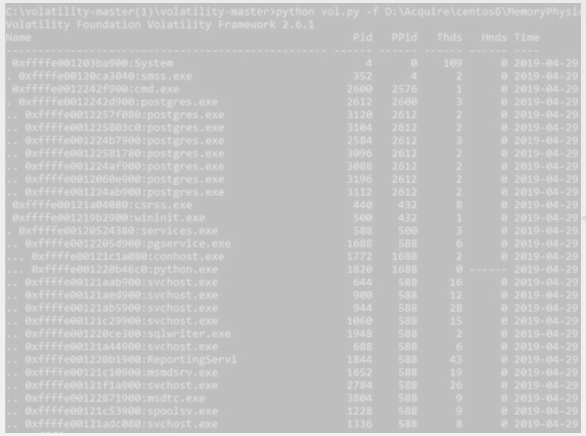
The kernel is ALWAYS affected

Dozens of processes with corrupted address spaces

Two cases in which the pages of one process get attributed to another

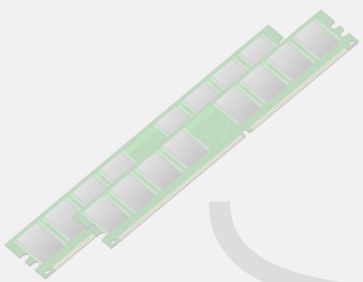
Investigation



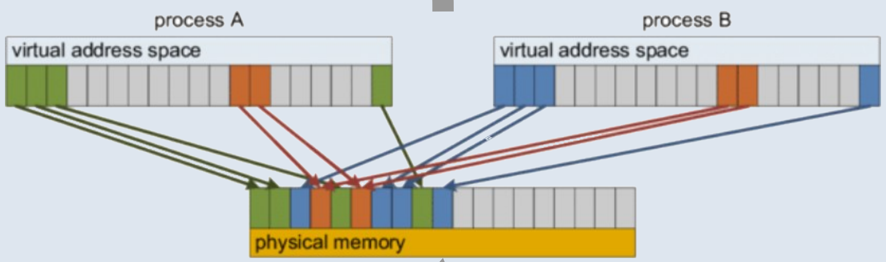


```
C:\volatility-master(1)\volatility-master>python vol.py -f D:\Acquire\centos6\MemoryPhysi
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xffffe001203ba900:System                              4      0    109      0 2019-04-29
. 0xffffe00120ca3040:smss.exe                          352      4      2      0 2019-04-29
 0xffffe0012242f900:cmd.exe                           2600   2576      1      0 2019-04-29
. 0xffffe0012242d900:postgres.exe                     2612   2600      3      0 2019-04-29
.. 0xffffe0012257f080:postgres.exe                    3120   2612      2      0 2019-04-29
.. 0xffffe001225803c0:postgres.exe                    3104   2612      2      0 2019-04-29
.. 0xffffe001224b7900:postgres.exe                    2584   2612      3      0 2019-04-29
.. 0xffffe00122581780:postgres.exe                    3096   2612      2      0 2019-04-29
.. 0xffffe001224af900:postgres.exe                    3088   2612      2      0 2019-04-29
.. 0xffffe0012060e900:postgres.exe                    3196   2612      2      0 2019-04-29
.. 0xffffe001224ab900:postgres.exe                    3112   2612      2      0 2019-04-29
 0xffffe00121a04080:csrss.exe                          440    432      8      0 2019-04-29
 0xffffe001219b2900:wininit.exe                        500    432      1      0 2019-04-29
. 0xffffe00120524380:services.exe                      588    500      3      0 2019-04-29
.. 0xffffe0012205d900:pgservice.exe                   1688    588      6      0 2019-04-29
... 0xffffe00121c1a080:conhost.exe                    1772   1688      2      0 2019-04-29
... 0xffffe001220b46c0:python.exe                     1820   1688      0 ------ 2019-04-29
.. 0xffffe00121aab900:svchost.exe                      644    588     16      0 2019-04-29
.. 0xffffe00121aed900:svchost.exe                      900    588     12      0 2019-04-29
.. 0xffffe00121ab5900:svchost.exe                      944    588     28      0 2019-04-29
.. 0xffffe00121c29900:svchost.exe                     1060    588     15      0 2019-04-29
.. 0xffffe001220ce300:sqlwriter.exe                   1948    588      2      0 2019-04-29
.. 0xffffe00121a44900:svchost.exe                      688    588      6      0 2019-04-29
.. 0xffffe001220b1900:ReportingServi                  1844    588     43      0 2019-04-29
.. 0xffffe00121c10900:msmdsrv.exe                     1652    588     19      0 2019-04-29
.. 0xffffe00121f1a900:svchost.exe                     2784    588     26      0 2019-04-29
.. 0xffffe00122871900:msdtc.exe                       3804    588      9      0 2019-04-29
.. 0xffffe00121c53900:spoolsv.exe                     1228    588      9      0 2019-04-29
.. 0xffffe00121adc080:svchost.exe                     1336    588      8      0 2019-04-29
```

Acquisition



```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```



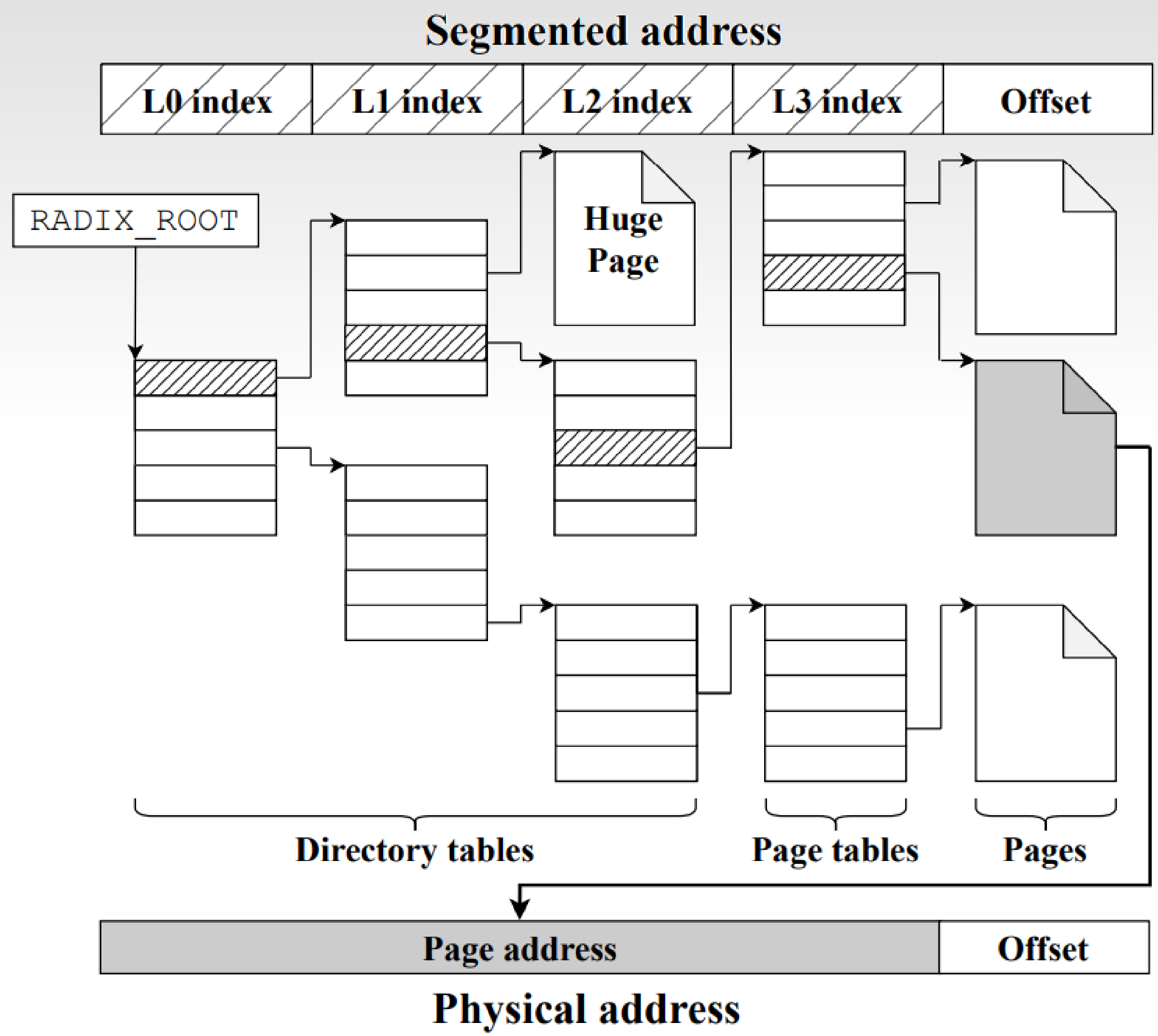
Interpretation

Virtual-to-Physical Memory Translation

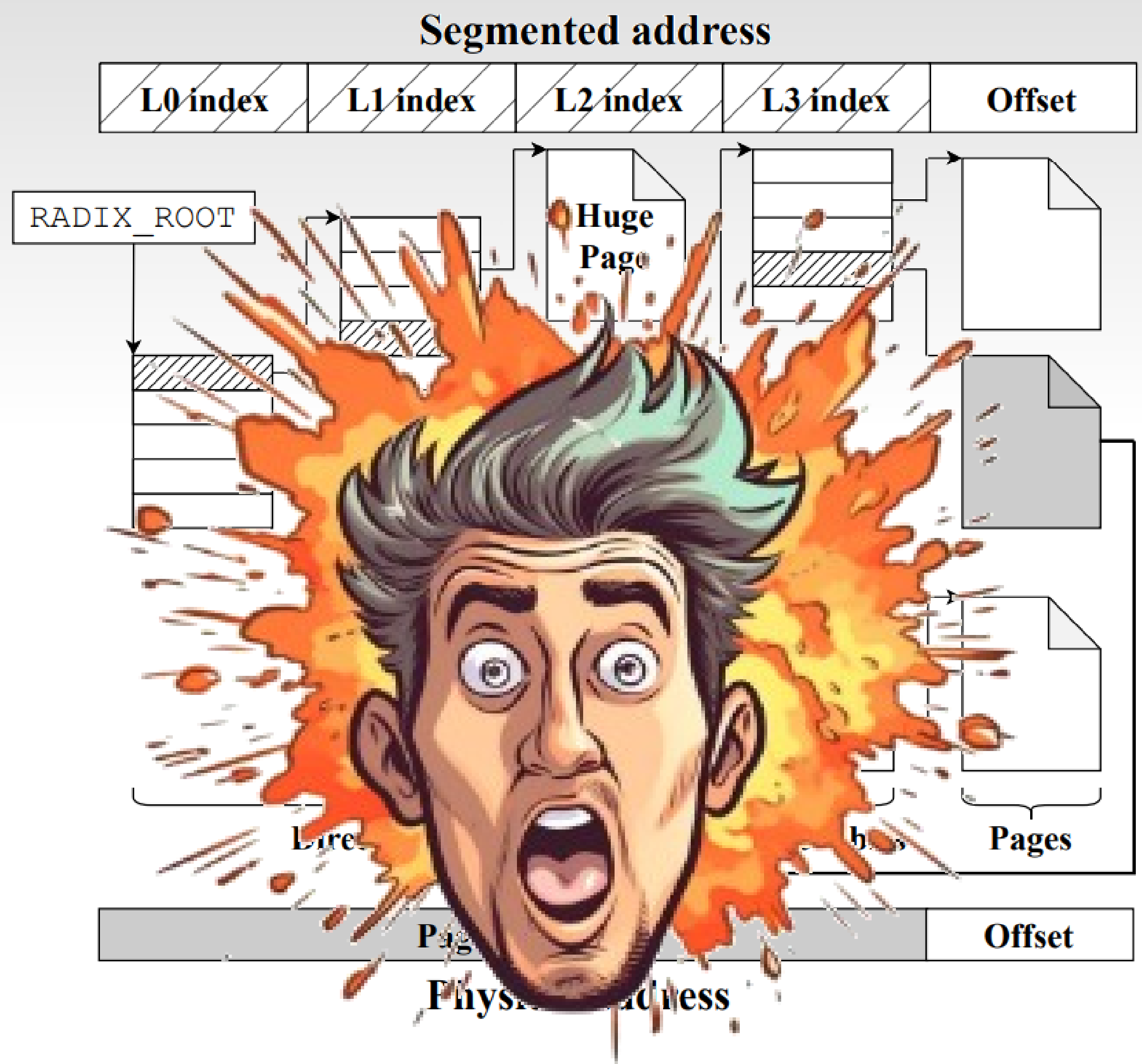
The V2P translation is performed in hardware by the Memory Management Unit (**MMU**) based on in-memory data structures and dedicated CPU registers that are configured by the OS

The translation process can involve **segmentation** and **paging**. Some architectures use one or the other, some use both.

Virtual-to-Physical Memory Translation



Virtual-to-Physical Memory Translation



Multiarchitecture OS-Agnostic Virtual Memory Forensics

radix-tree

RISC-V, intel, ARM

hash table

PowerPC, Power

fully customizable

MIPS

Multiarchitecture OS-Agnostic Virtual Memory Forensics

Want to know more?

In the Land of MMUs: Multiarchitecture OS-Agnostic Virtual Memory Forensics

ANDREA OLIVERI, Eurecom, France
DAVIDE BALZAROTTI, Eurecom, France

The first step required to perform any analysis of a physical memory image is the reconstruction of the virtual address spaces, which allows translating virtual addresses to their corresponding physical offsets. However, this phase is often overlooked and the challenges related to it are rarely discussed in the literature. Practical tools solve the problem by using a set of custom heuristics tailored on a very small number of well-known operating systems running on few architectures.

In this paper, we look for the first time at all the different ways the virtual to physical translation can be operated in 10 different CPU architectures. In each case, we study the inviolable constraints imposed by the MMU that can be used to build signatures to recover the required data structures from memory without any knowledge about the running operating system. We build a proof-of-concept tool to experiment with the extraction of virtual address spaces showing the challenges of performing an OS-agnostic virtual to physical address translation in real-world scenarios. We conduct experiments on a large set of 26 different OSs and a use case on a real hardware device. Finally, we show a possible usage of our technique to retrieve information about user space processes running on an unknown OS without any knowledge of its internals.

CCS Concepts: • **Applied computing** → **System forensics**; • **Security and privacy** → *Operating systems security*.

Additional Key Words and Phrases: memory forensics, OS-agnostic forensics, virtual memory, MMU

ACM Reference Format:
Andrea Oliveri and Davide Balzarotti. 2022. In the Land of MMUs: Multiarchitecture OS-Agnostic Virtual Memory Forensics. 1, 1 (April 2022), 33 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

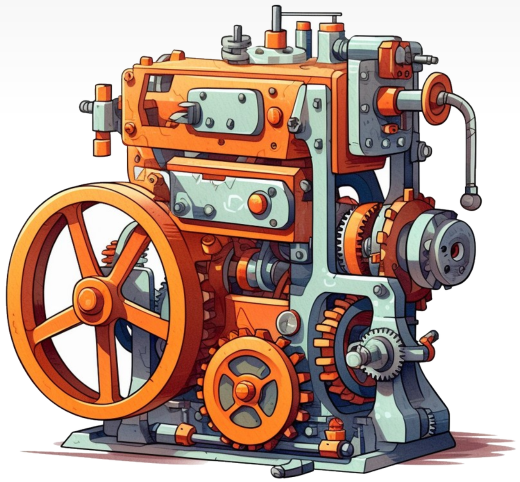
The problem of recovering semantic information from low-level data is common to many areas of computer security. In particular, this is the main obstacle when performing a physical memory analysis—a task that is key for both memory forensics and virtual machine introspection. The problem, often called the *semantic gap*, captures the challenge of "interpreting low level bits and bytes into a high level semantic state of an in-guest operating system" [35]. However, at a closer look, the semantic gap can be further divided into two different aspects: the reconstruction of the virtual address spaces (which deal with translating pointers expressed as virtual addresses to their physical position in the

1. **Structural Signatures** derived by inviolable MMU constraints

2. **Validation Rules** based on inviolable constraints imposed by other CPU subsystems
   (e.g., pages containing the Interrupt Address Table should be mapped in all VASs)

3. **Binary code analysis** to recover MMU-related CPU registers

Physical to Virtual Memory Translation

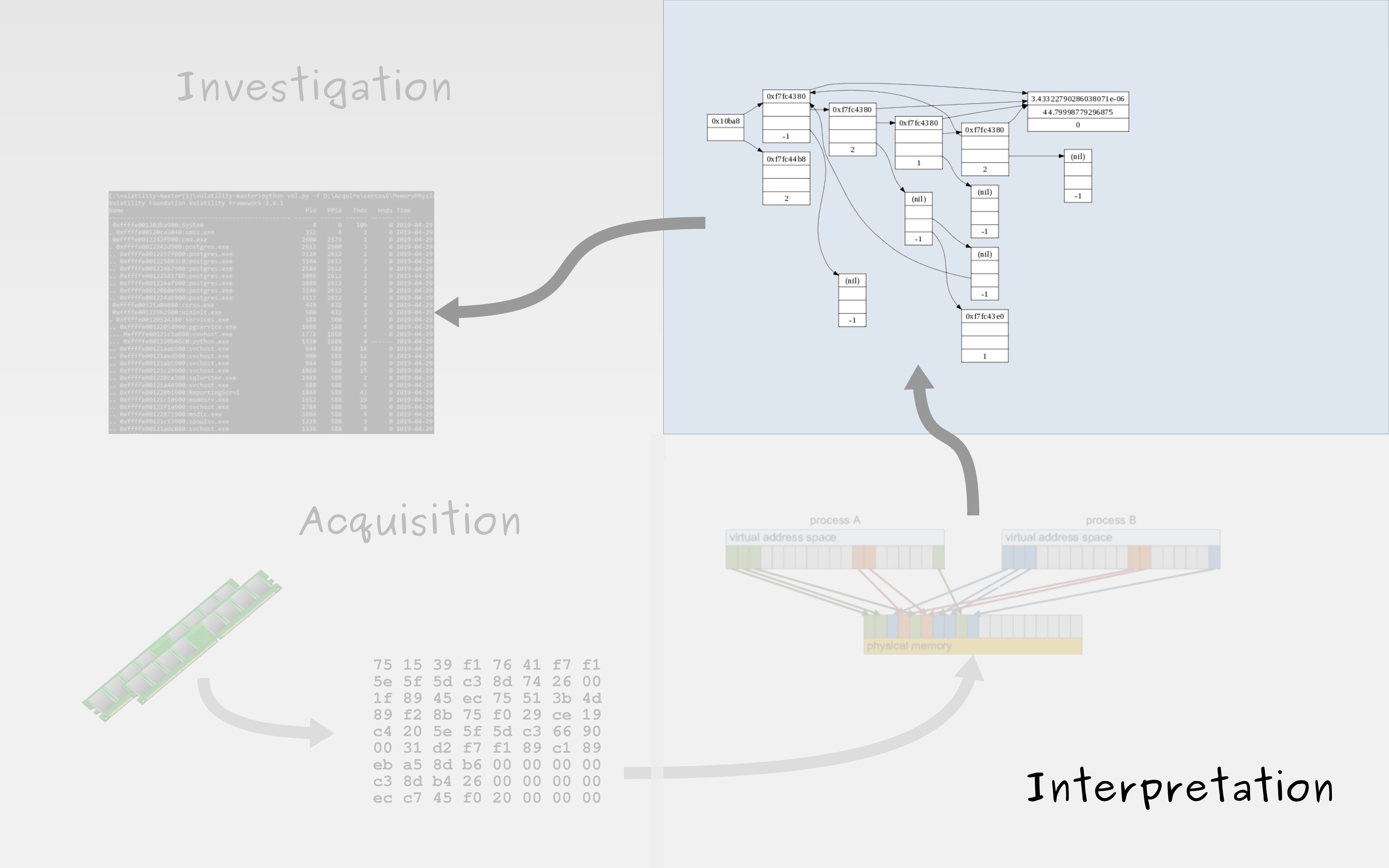


MMUShell

https://github.com/eurecom-s3/mmushell

nory Translation

OS	Kernel type[1]	Open-source	AArch64 Long	AMD64	ARM32 Short	MIPS32 TLBs	MIPS32 Radix	PowerPC	RISC-V SV32	RISC-V SV48	x86 IA32	x86 PAE
9Front[24]	H	●									🟢	
Barrelfish[17]	U	●	🟢	🟢	🟢							
Darwin[4]	H	●		🟢								
Embox[5]	R	●	🟡		🟡	🟢					🔴	
FreeBSD	M	●	🟢	🟢						🟢	🟢	
GenodeOS[6]	m	●		🟡								
HaikuOS[7]	H	●		🟢							🟢	
HelenOS[8]	m	●	🟢	🟢	🟡	🟢		🟡				
Linux Buildroot[3]	M	●	🟢	🟢	🟡	🟡	🟡	🟢	🟢	🟢	🟢	🟢
Linux Debian	M	●	🟢	🟢	🟡			🟢		🟢	🟢	
MacOS 9	n	○						🟡				
MacOS X	H	○						🟡				
Minix3[9]	m	●									🟢	
MorphOS[10]	m	○						🟡				
NetBSD	M	●	🟢	🟢	🟢			🟡			🟢	
Illumos[29]	M	●		🟢								
QNX[11]	R	○									🟢	
rCore[13]	M	●	🟢	🔴					🟢	🟢		
ReactOS[14]	m	●									🟢	
RedoxOS[15]	m	●		🟡								
vxWorks[19]	R	○		🟡								
Windows 10	H	○	🟡	🟢								🟢
Windows 95	M	○									🟢	
Windows NT	H	○									🟢	
Windows XP	H	○		🟢							🟢	🟢
XV6[20]	M	●								🟢		


Investigation
Acquisition
Interpretation
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
process A
process B
virtual address space
virtual address space
physical memory
0x10ba8
0xf7fc4380
0xf7fc44b8
0xf7fc43e0
(nil)
3.43322790286038071e-06
44.79998779296875

The Problem with Profiles

Memory Forensics is based on **PROFILES,** which contain precise descriptions of all the kernel data structures necessary to perform the analysis.

Q1: Can we automatically generate profiles starting from the dump itself?

Q2: Can we perform some analysis also *<u>without any</u>* profile?

The important is NOT how much kernel structures change across kernels

But how much they change **within** a single version – because of user configurations or compiler options.
E.g., The layout of `task_struct` is shaped by more than 60 different `#ifdef`

Modern kernels also support *structure layout randomization* as a form of protection against exploitation

While the `struct` definitions are lost during the compilation process, they are “reflected’’ in the code itself.

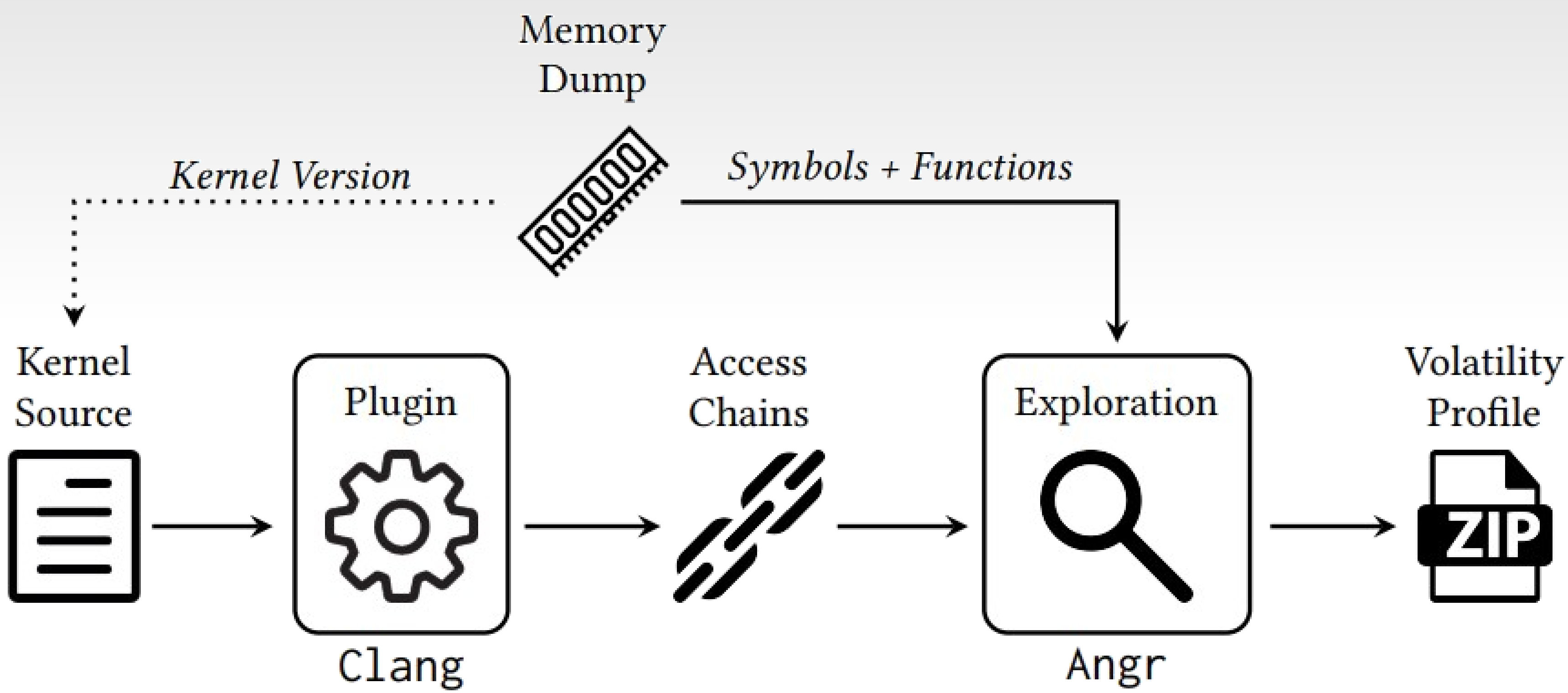
Automated Profile Generation

```
struct creds{
  uint32_t uid;
  uint32_t gid;
};

struct task{
  struct task *next;
  struct creds cred;
#ifdef CONFIG_TIME
  uint64_t start_time;
#endif
  char *name;
};

void setup_task(struct task *t,
                char *new_name,
                int gid)
{
  t->name = new_name;
  t->cred.gid = gid;
#ifdef CONFIG_TIME
  t->start_time = time(NULL);
#endif
}
```

```
mov     QWORD PTR [rdi+0x10],rsi
mov     DWORD PTR [rdi+0xc],edx
ret
```

```
push    rbx
mov     rbx,rdi
mov     QWORD PTR [rdi+0x18],rsi

mov     DWORD PTR [rdi+0xc],edx
xor     edi,edi
call    0x1030 <time@plt>
mov     QWORD PTR [rbx+0x10],rax
pop     rbx
ret
```

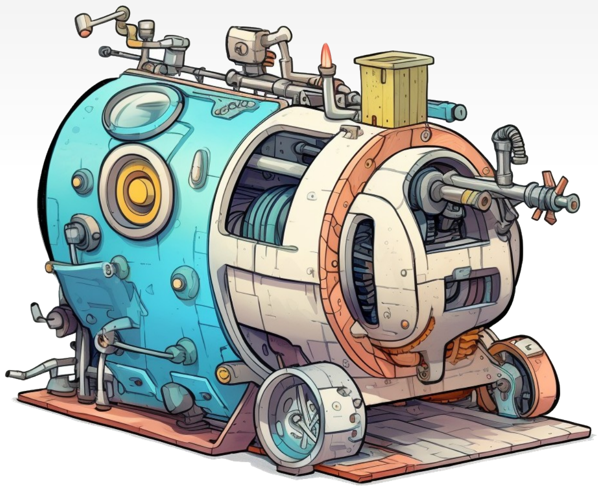
Automated Profile Generation



*AutoProfile: Towards Automated Profile Generation for Memory Analysis

Automated Profile Generation

Version	Release Date	Configuration	Used Fields	Extracted Fields
4.19.37	04/2019	Debian	234	220 (94%)
4.19.37	04/2019	Debian + RANDSTRUCT	234	194 (83%)
5.6.19	03/2020	Raspberry Pi	227	217 (95%)
4.4.71	06/2017	OpenWrt	236	216 (92%)
3.18.94	05/2018	Goldfish (Android)	239	220 (92%)
2.6.38	03/2011	Ubuntu	226	213 (94%)

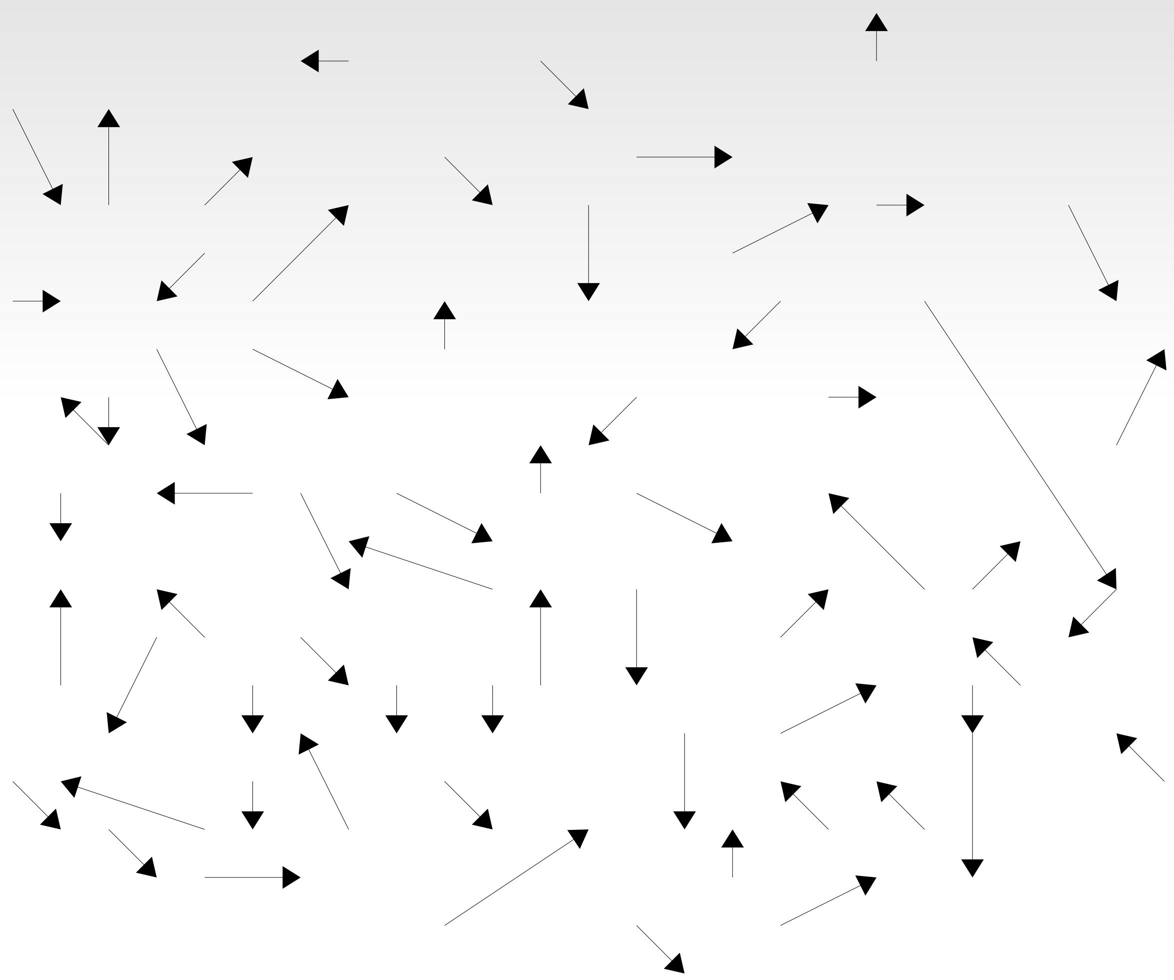


Katana
(very very similar solution published one year later)

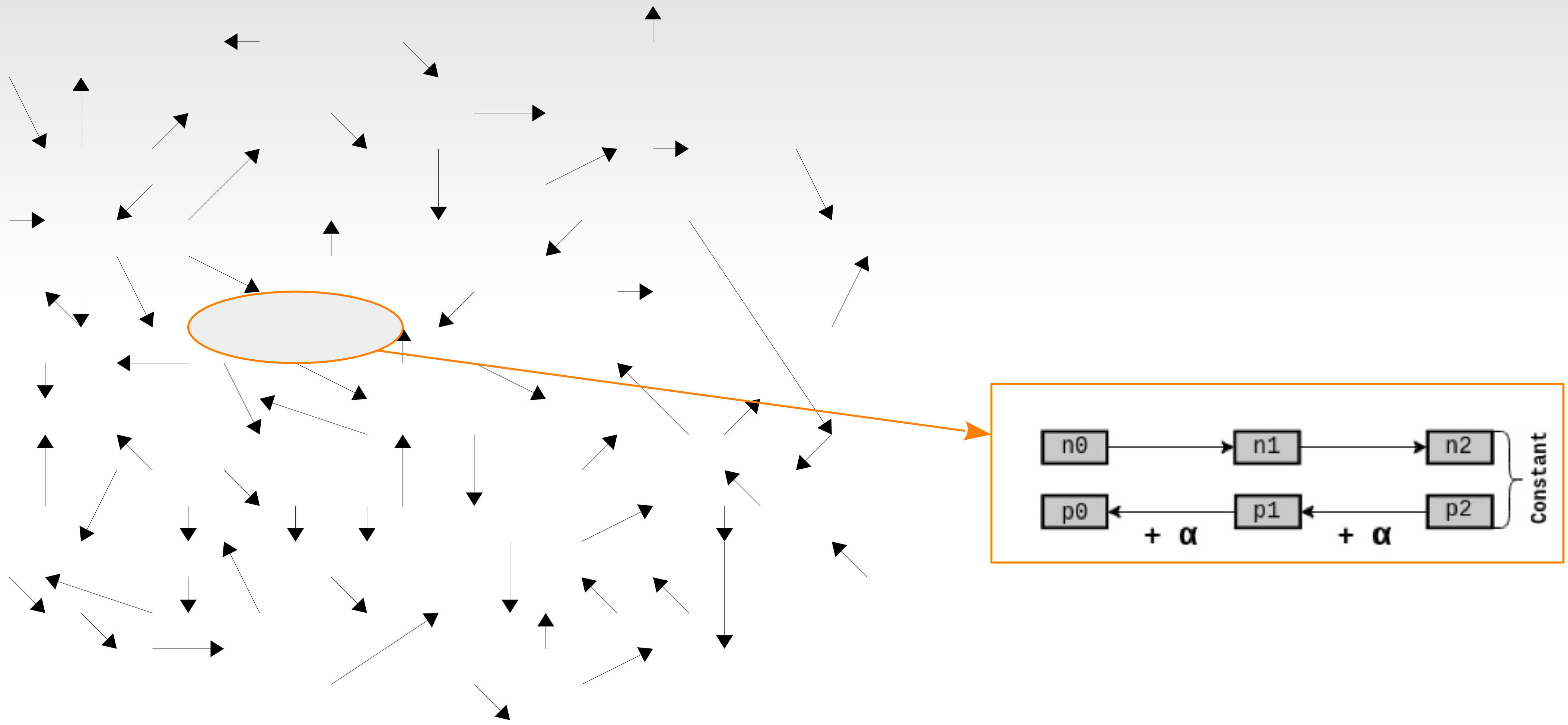
https://github.com/tum-itsec/katana

*Katana: Robust, Automated, Binary-Only Forensic Analysis of Linux Memory Snapshots

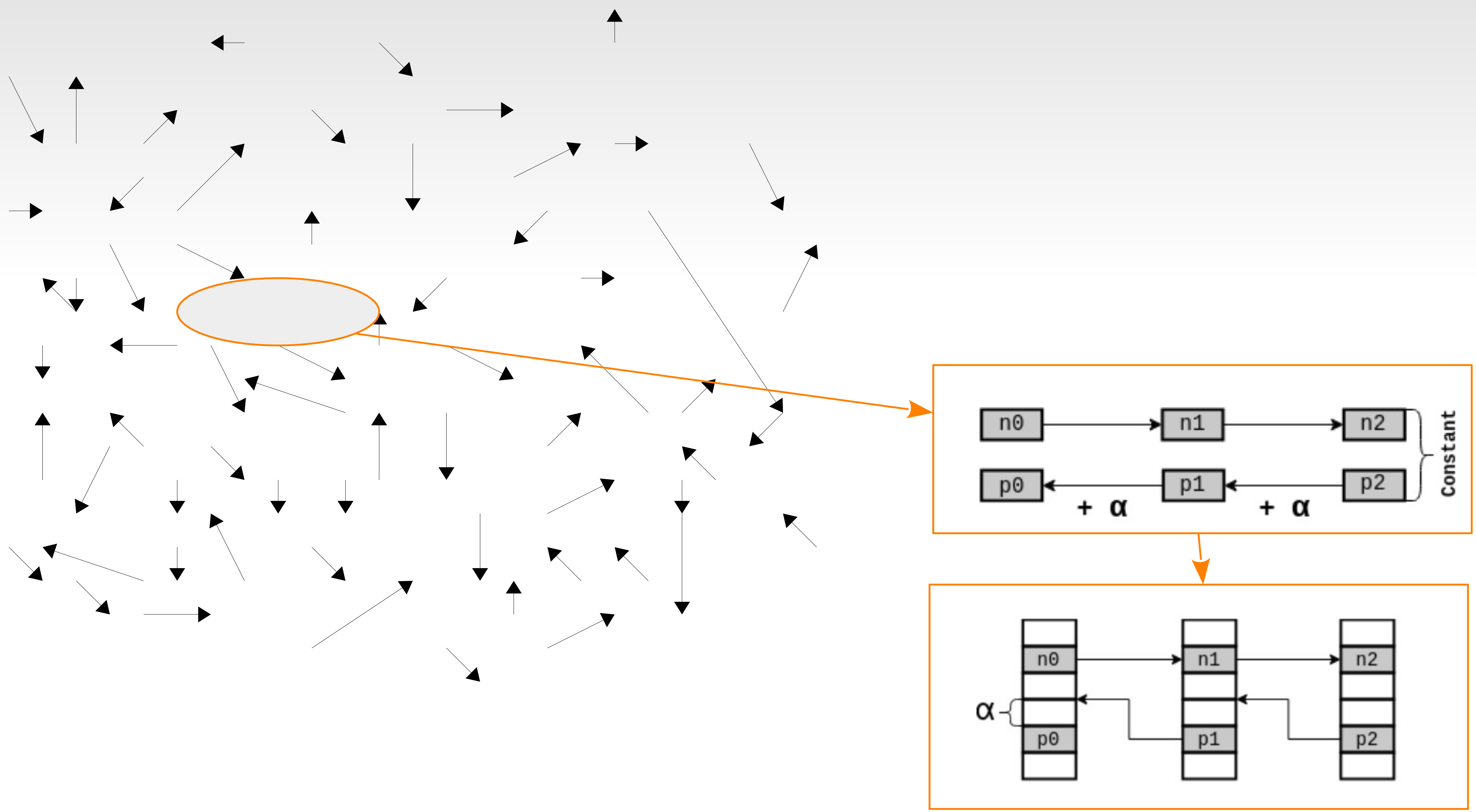
Look Mum,
no Profiles!



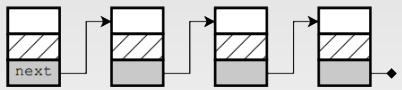
OS-Agnostic Memory Forensics

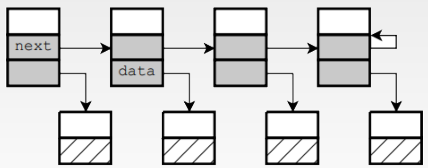


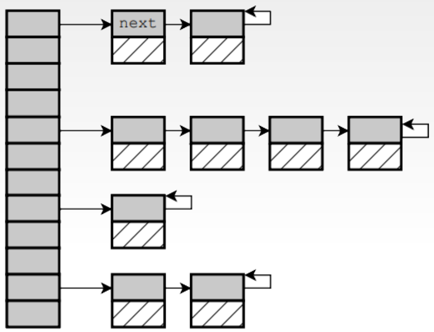
OS-Agnostic Memory Forensics

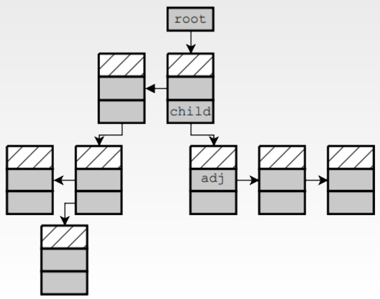


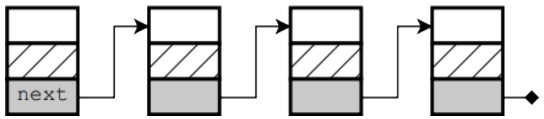
OS-Agnostic Memory Forensics

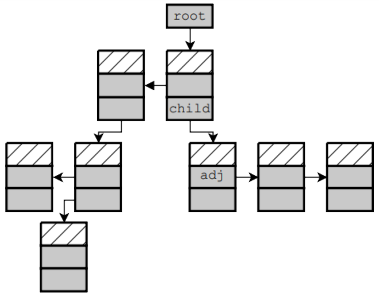


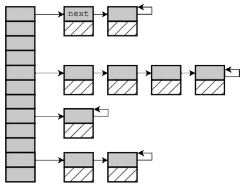


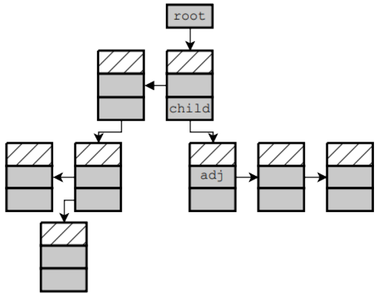




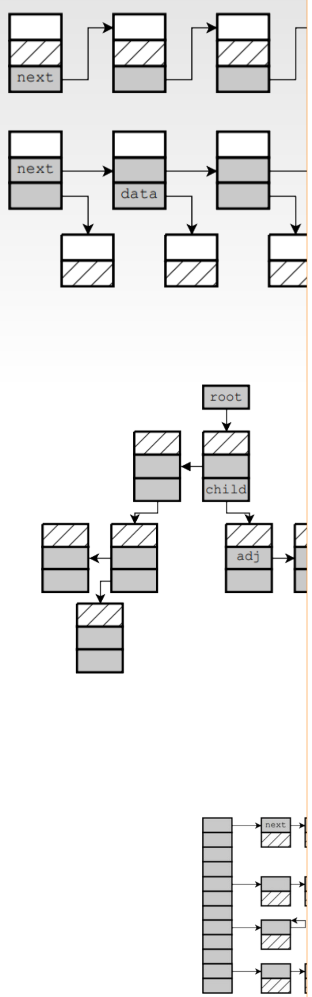




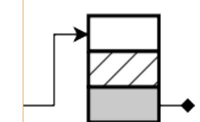




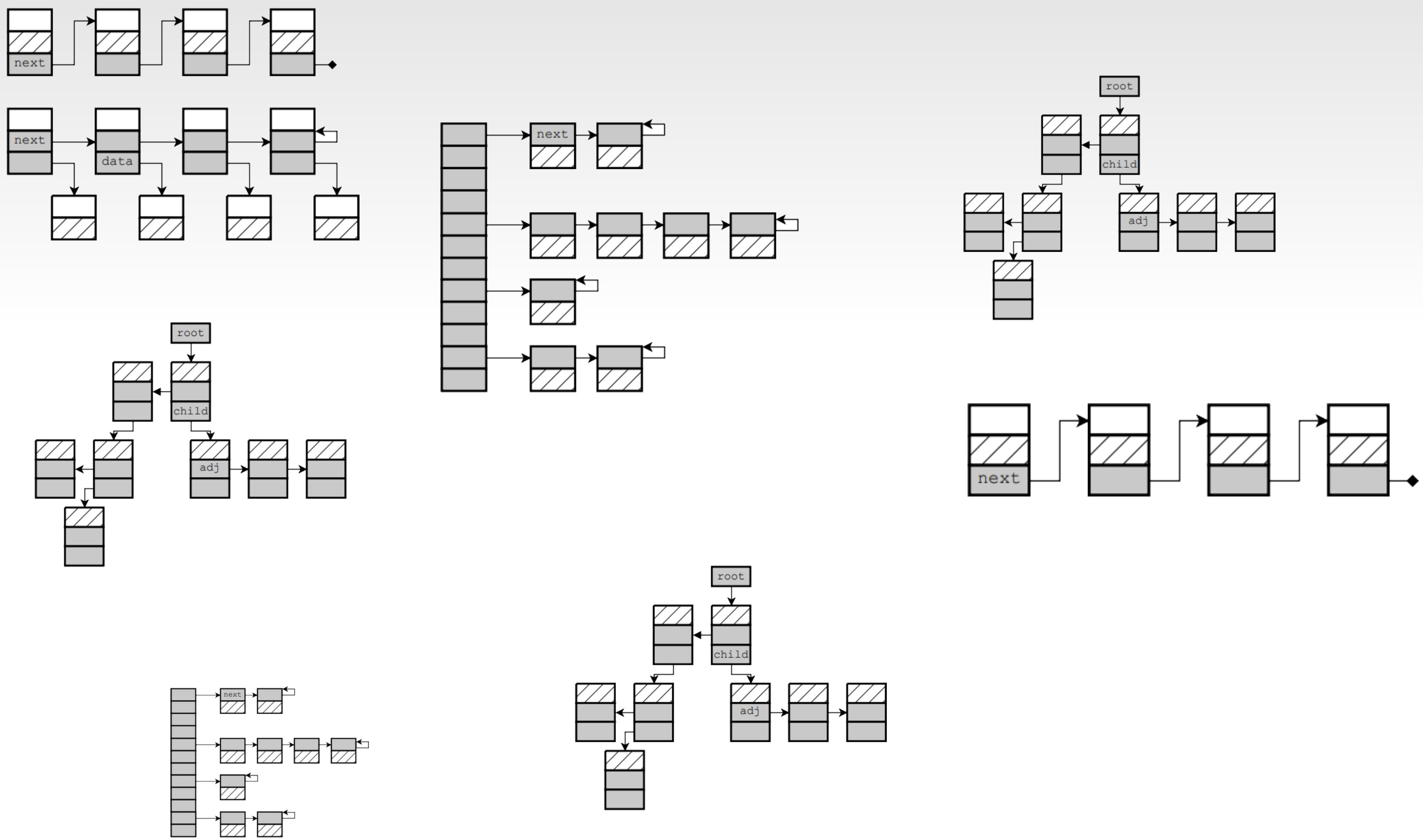
OS-Agnostic Memory Forensics



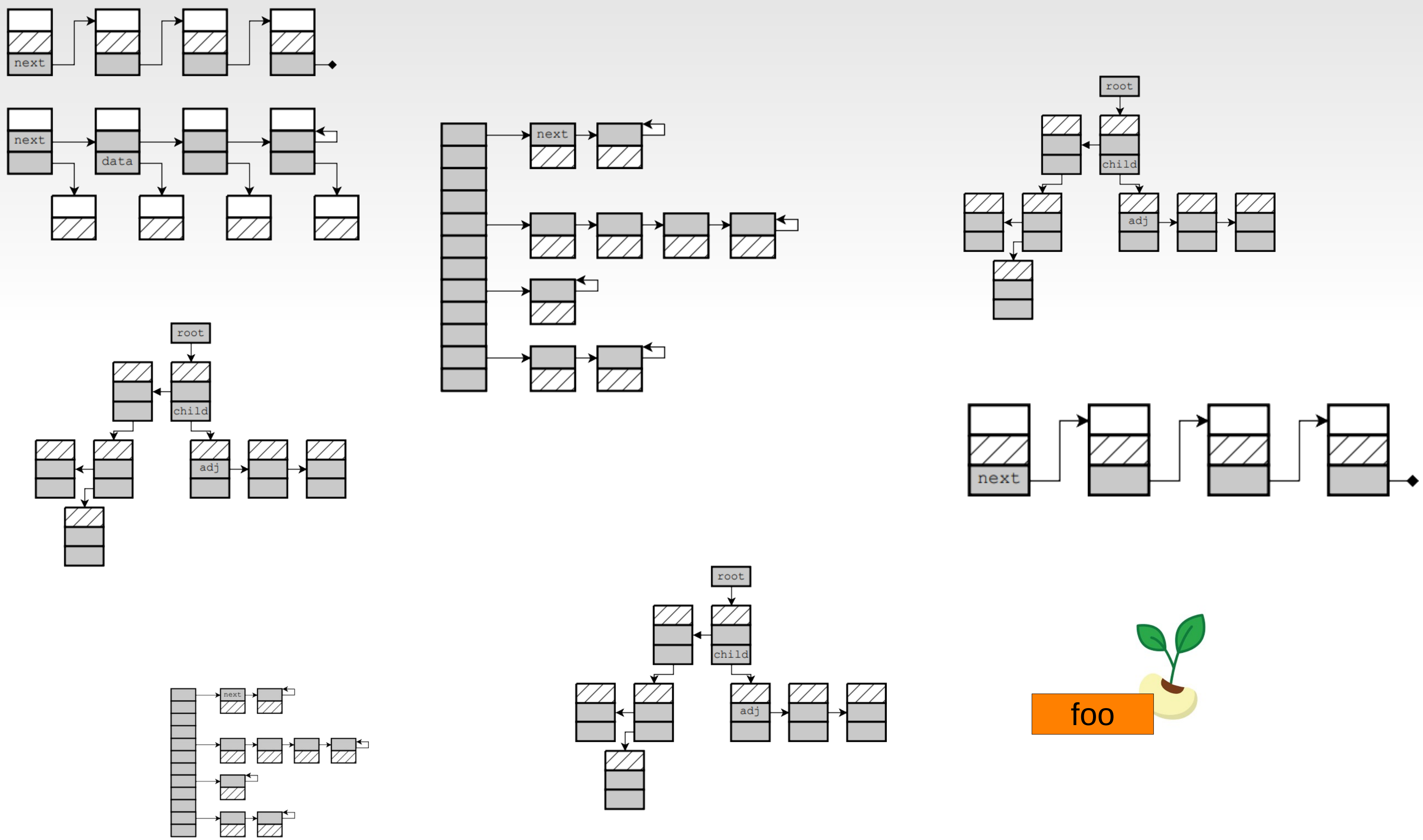
OS	Linear D.-L. L.	Circular D.-L. L.	Trees	Arrays of *structs	Arrays of *strings	Linked Lists
Darwin	11	385	127	1214	1801	35
Embox	0	22	35	1131	795	6
FreeBSD	86	0	993	1008	895	41
HaikuOS	4117	64	0	305	232	1184
HelenOS	25	1173	127	41	45	1
iOS	20	256	192	5234	229	36
Linux	120	3632	1034	693	5947	46
Linux (Aarch64)	110	3362	936	229	4985	43
NetBSD	41	18	1218	1482	406	45
ReactOS	7	200	49	492	325	12
ToaruOS	101	0	14	62	229	15
vxWorks	51	14	199	349	416	13
Windows XP	38	889	228	463	206	20
Windows 10	145	6639	36	0	282	0



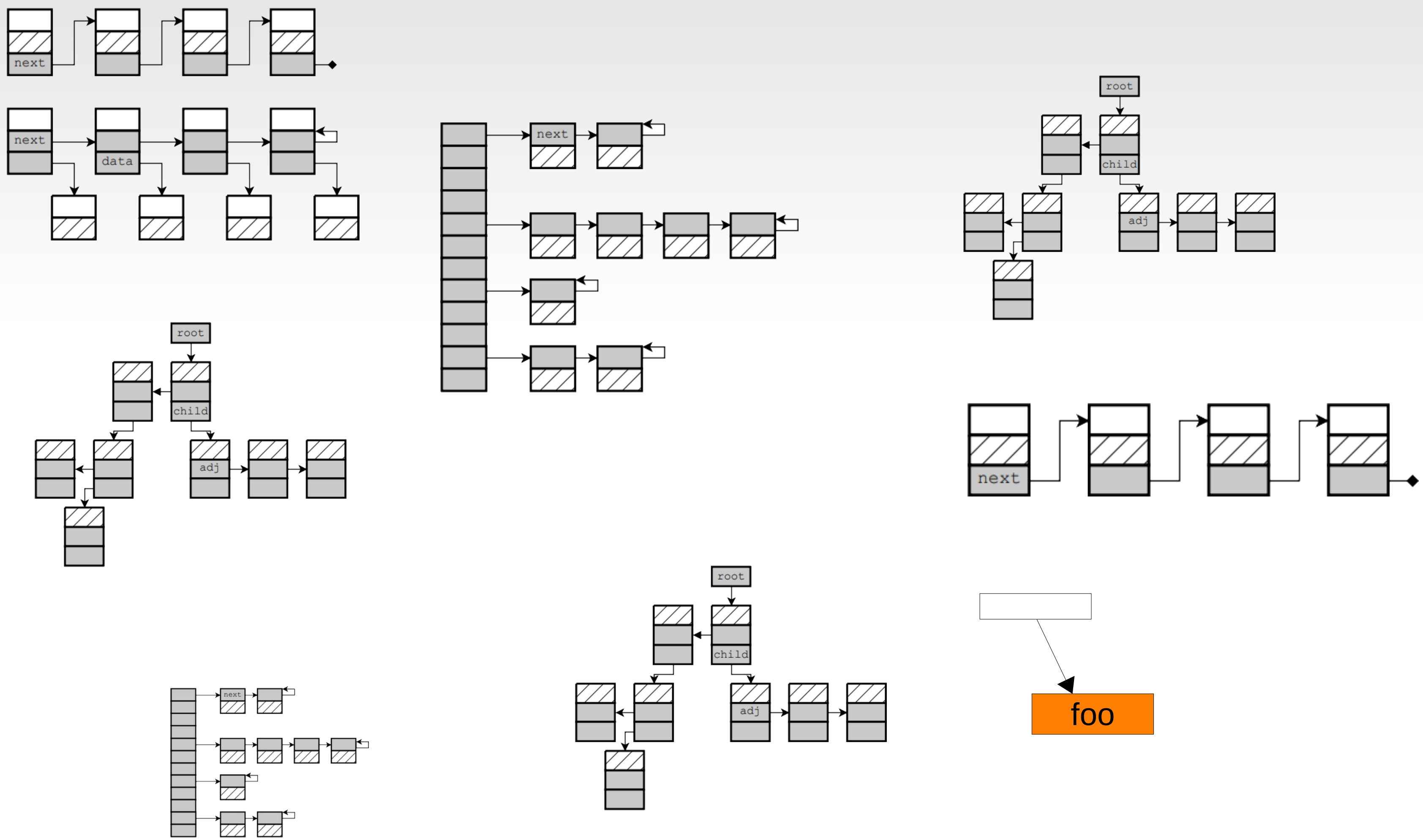
OS-Agnostic Memory Forensics



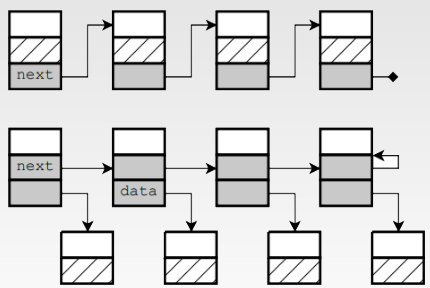
OS-Agnostic Memory Forensics

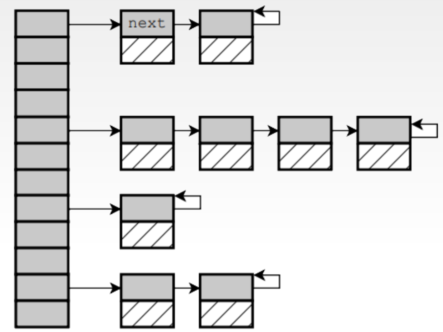


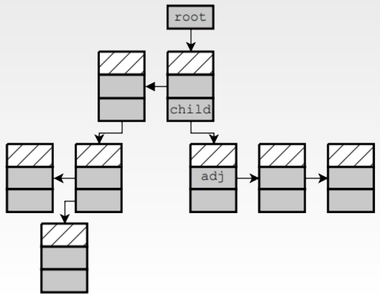
OS-Agnostic Memory Forensics

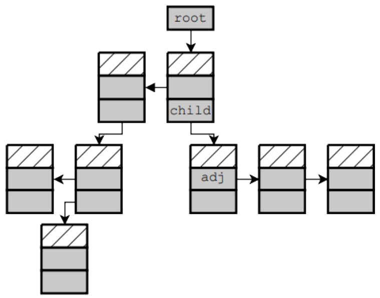


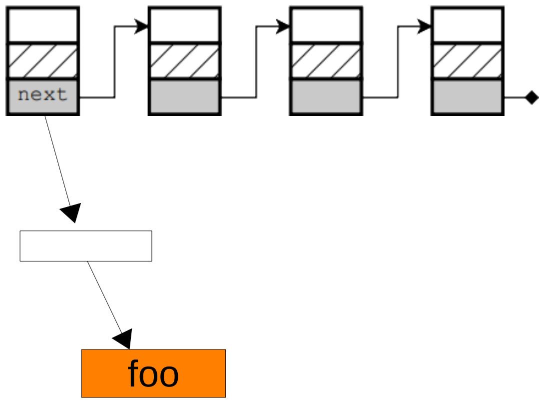
OS-Agnostic Memory Forensics

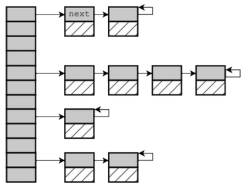


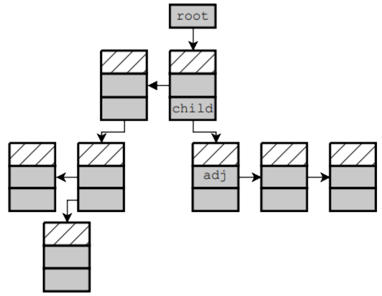




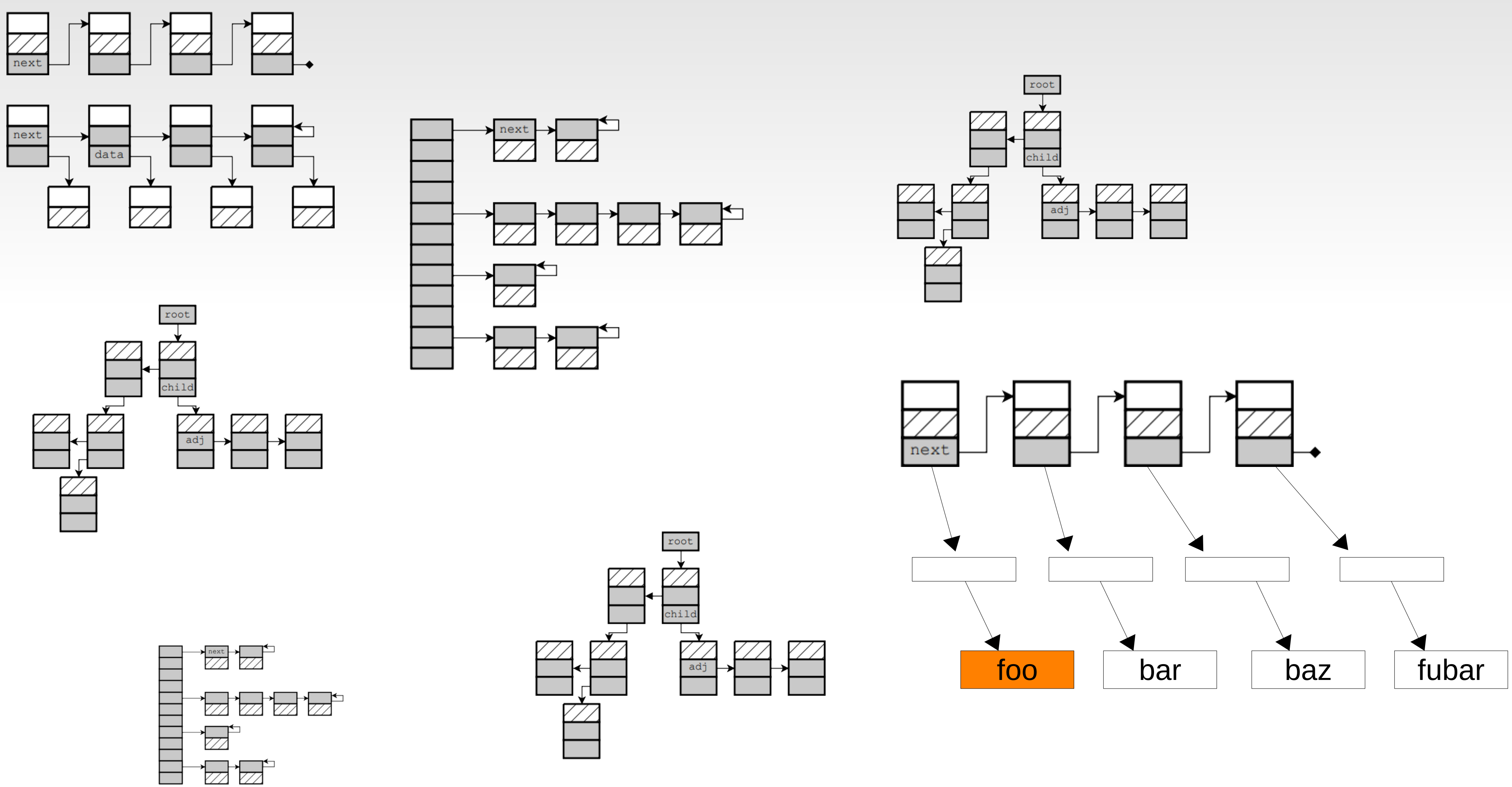


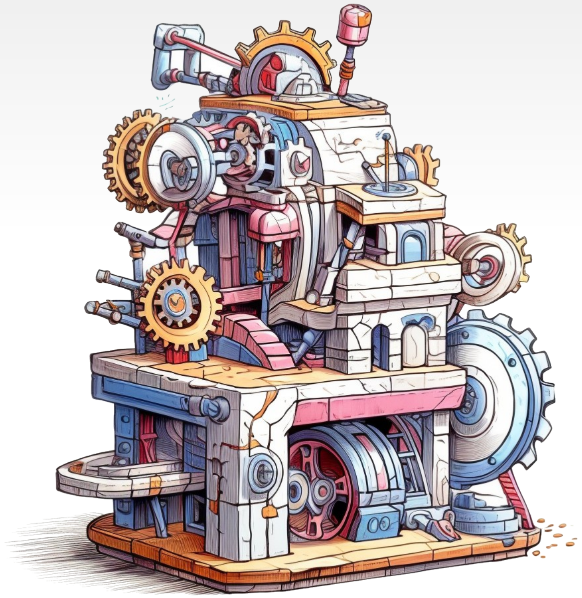






OS-Agnostic Memory Forensics





Fossil

https://github.com/eurecom-s3/fossil

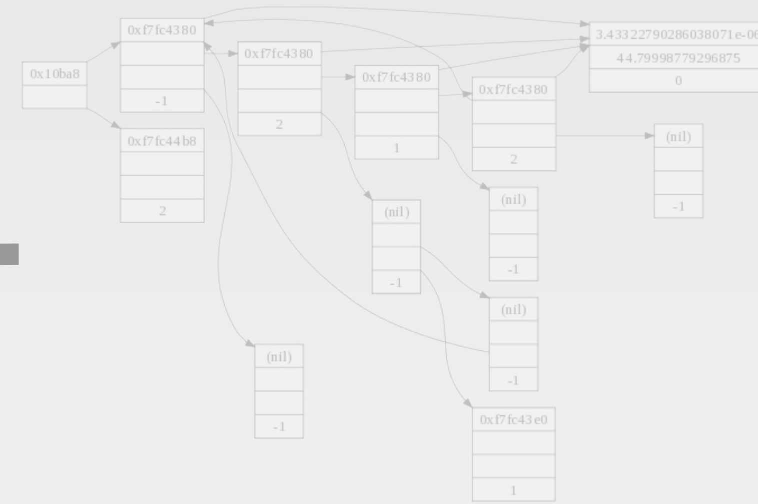
OS-Agnostic Memory Forensics

OS	Kernel modules	Kernel pools	File systems	Other structures
Darwin	●	●	●	• List of network devices • System locks • Kernel/user pipes • Kernel parameters
Embox	●		○	• List of commands
FreeBSD	●		●	
HaikuOS	●	●	○	• Executable libraries • Kernel/user pipes • Semaphores
HelenOS	●	●	●	
iOS	○	●	●	• List of network devices • System locks • Kernel/user pipes • Kernel parameters
Linux	●	●	●	• Files in `sysfs` • Network protocols
Linux (AArch64)	●	●	●	• Files in `sysfs` • Network protocols
NetBSD	●	●	●	• Kernel tasks
ReactOS	○	●	○	
ToaruOS	●		●	• Devices' list • Processes' environment
vxWorks	○	●	○	• Devices' list • Open sockets
Windows XP	●	●	○	
Windows 10	●	●	●	

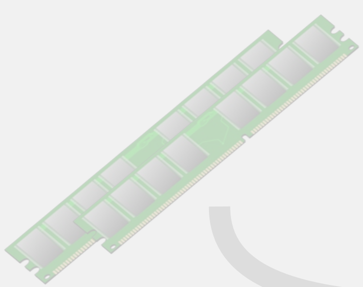
OS	Process list	Kernel modules	Kernel pools	Filesystem
Darwin	2	10	11	7
Embox	17	○		
FreeBSD	24	31		26
HaikuOS	6	1	11	
HelenOS	4	2	1	1
iOS	2		2	15
Linux	5	28	26	15
Linux (AArch64)	4	22	19	24
NetBSD	2	6	18	○
ReactOS	5		12	
ToaruOS	3	2	3	
vxWorks	4		2	
Windows XP	5	1	2	
Windows 10	41	○	○	○

Investigation

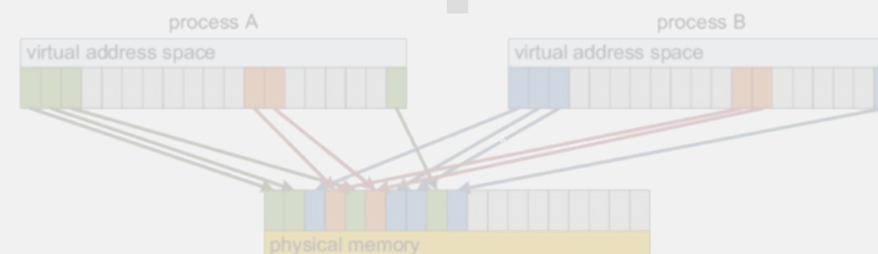
```
C:\volatility-master(1)\volatility-master>python vol.py -f D:\Acquire\centos6\MemoryPhysi
Volatility Foundation Volatility Framework 2.6.1
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xffffe001203ba900:System                              4      0    109      0 2019-04-29
. 0xffffe00120ca3040:smss.exe                         352      4      2      0 2019-04-29
 0xffffe0012242f900:cmd.exe                          2600   2576      1      0 2019-04-29
. 0xffffe0012242d900:postgres.exe                    2612   2600      3      0 2019-04-29
.. 0xffffe0012257f080:postgres.exe                   3120   2612      2      0 2019-04-29
.. 0xffffe001225803c0:postgres.exe                   3104   2612      2      0 2019-04-29
.. 0xffffe001224b7900:postgres.exe                   2584   2612      3      0 2019-04-29
.. 0xffffe00122581780:postgres.exe                   3096   2612      2      0 2019-04-29
.. 0xffffe001224af900:postgres.exe                   3088   2612      2      0 2019-04-29
.. 0xffffe0012060e900:postgres.exe                   3196   2612      2      0 2019-04-29
.. 0xffffe001224ab900:postgres.exe                   3112   2612      2      0 2019-04-29
 0xffffe00121a04080:csrss.exe                         440    432      8      0 2019-04-29
 0xffffe001219b2900:wininit.exe                       500    432      1      0 2019-04-29
. 0xffffe00120524380:services.exe                     588    500      3      0 2019-04-29
.. 0xffffe0012205d900:pgservice.exe                  1688    588      6      0 2019-04-29
... 0xffffe00121c1a080:conhost.exe                   1772   1688      2      0 2019-04-29
... 0xffffe001220b46c0:python.exe                    1820   1688      0 ------ 2019-04-29
.. 0xffffe00121aab900:svchost.exe                     644    588     16      0 2019-04-29
.. 0xffffe00121aed900:svchost.exe                     900    588     12      0 2019-04-29
.. 0xffffe00121ab5900:svchost.exe                     944    588     28      0 2019-04-29
.. 0xffffe00121c29900:svchost.exe                    1060    588     15      0 2019-04-29
.. 0xffffe001220ce300:sqlwriter.exe                  1948    588      2      0 2019-04-29
.. 0xffffe00121a44900:svchost.exe                     688    588      6      0 2019-04-29
.. 0xffffe001220b1900:ReportingServi                 1844    588     43      0 2019-04-29
.. 0xffffe00121c10900:msmdsrv.exe                    1652    588     19      0 2019-04-29
.. 0xffffe00121f1a900:svchost.exe                    2784    588     26      0 2019-04-29
.. 0xffffe00122871900:msdtc.exe                      3804    588      9      0 2019-04-29
.. 0xffffe00121c53900:spoolsv.exe                    1228    588      9      0 2019-04-29
.. 0xffffe00121adc080:svchost.exe                    1336    588      8      0 2019-04-29
```



Acquisition



```
75 15 39 f1 76 41 f7 f1
5e 5f 5d c3 8d 74 26 00
1f 89 45 ec 75 51 3b 4d
89 f2 8b 75 f0 29 ce 19
c4 20 5e 5f 5d c3 66 90
00 31 d2 f7 f1 89 c1 89
eb a5 8d b6 00 00 00 00
c3 8d b4 26 00 00 00 00
ec c7 45 f0 20 00 00 00
```

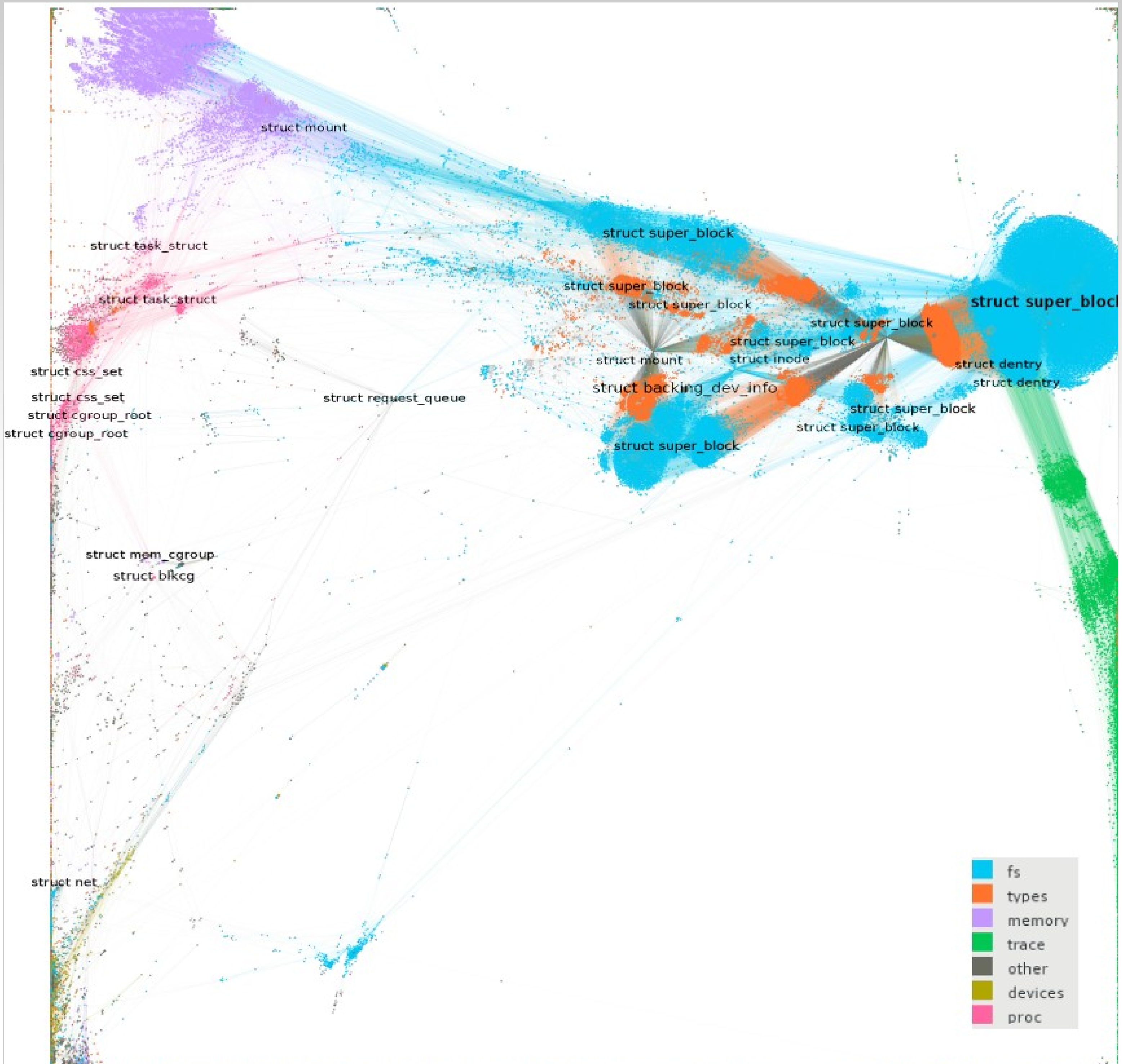


Interpretation

Memory Forensics as a Graph Exploration Problem

The goal of the analyst is to traverse the graph of kernel data structures to locate the information she needs.

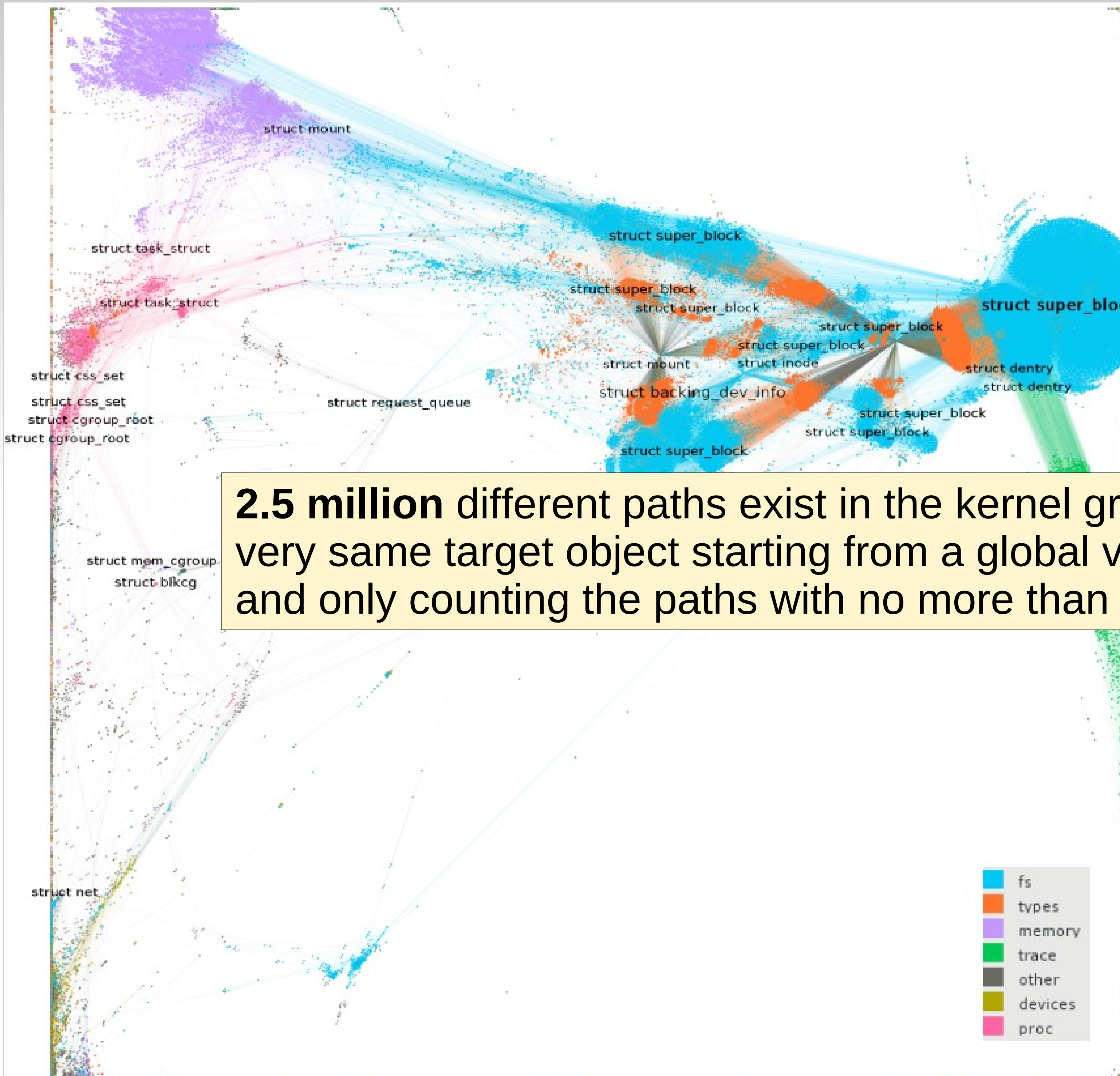
Each rule (e.g., a plugin to list processes) corresponds to a set of paths on the graph.



6K Unique Structures
40K Fields

100K Kernel Objects (nodes)
840K Pointers (edges)

53% of nodes (**and 96% of those used by Volatility**) are part of a single strongly-connected component



6K Unique Structures
40K Fields

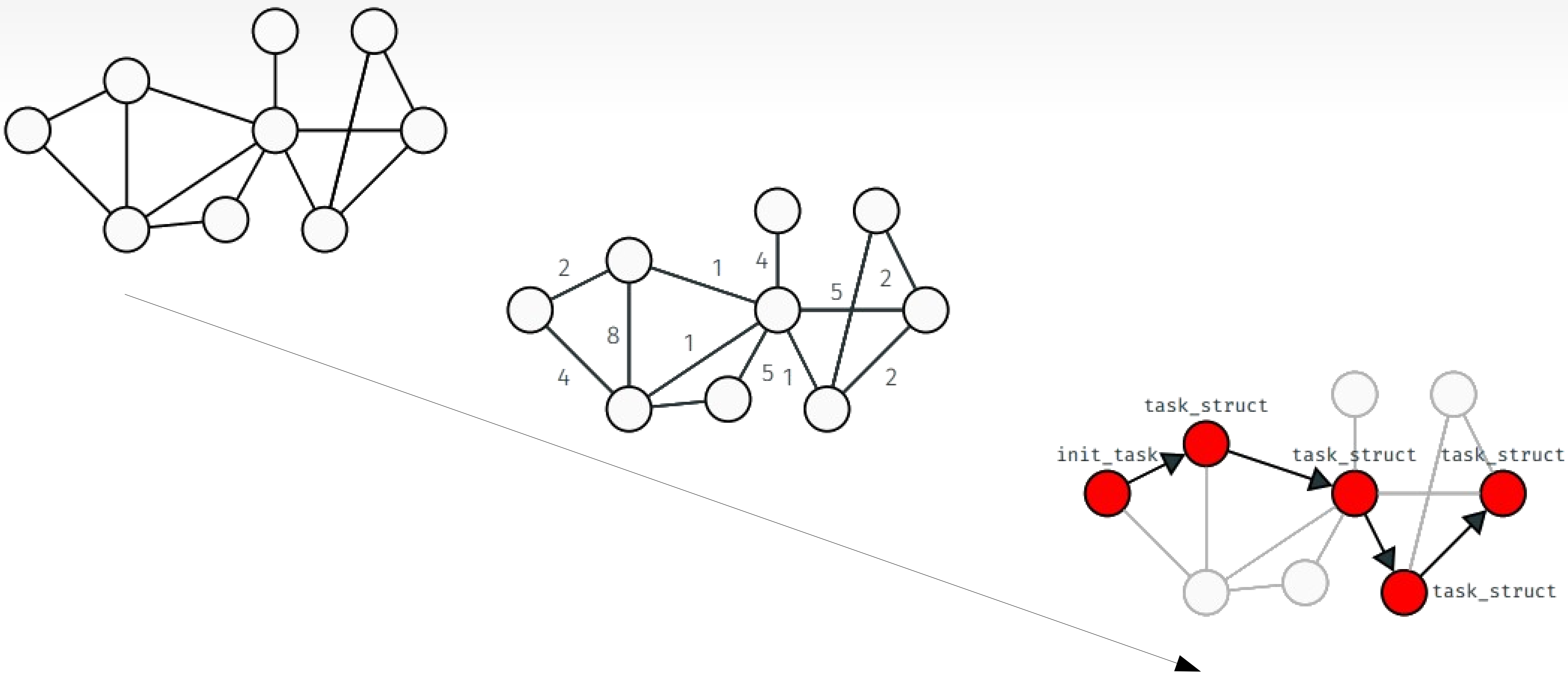
2.5 million different paths exist in the kernel graph to reach the very same target object starting from a global variable,
and only counting the paths with no more than 10 edges!!

jects (nodes)
dges)

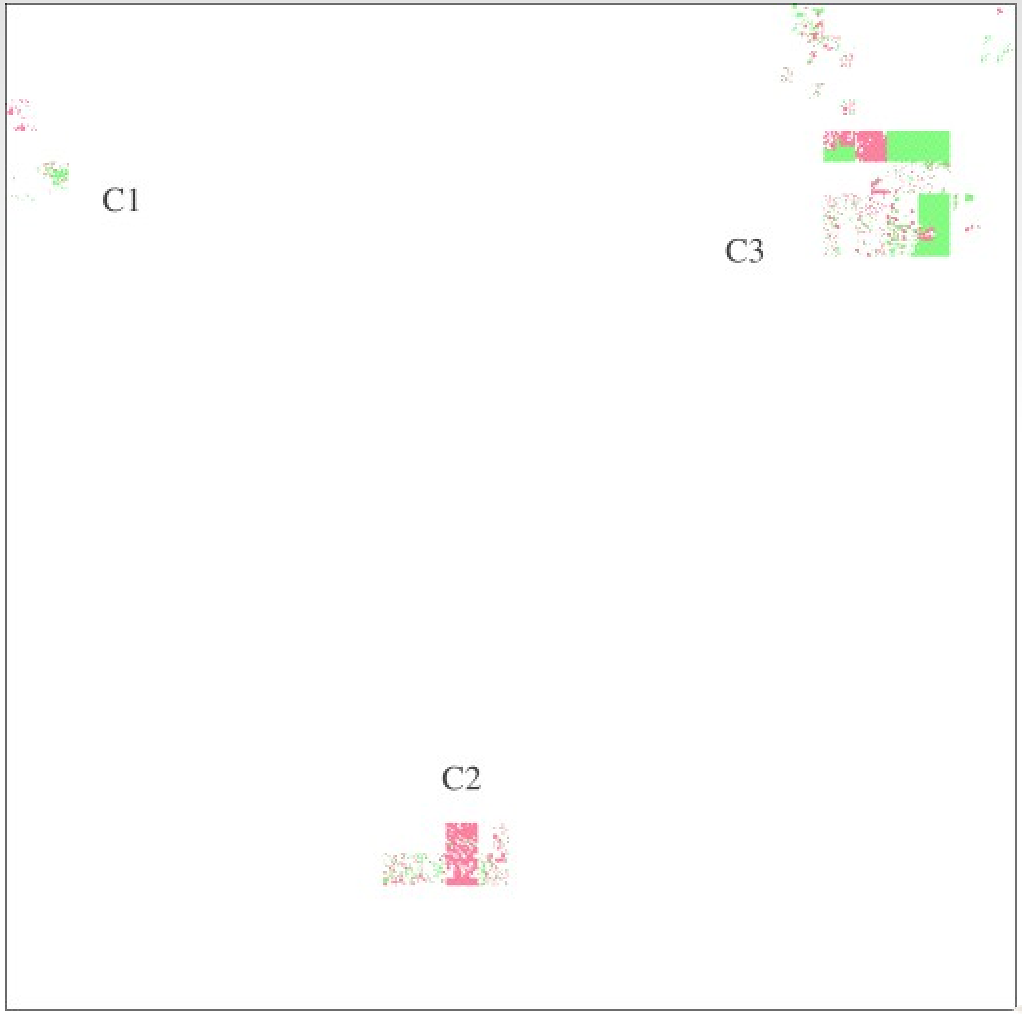
53% of nodes (**and 96% of those used by Volatility**) are part of a single strongly-connected component

Memory Forensics as a Graph Exploration Problem

Path comparison based on different **metrics**



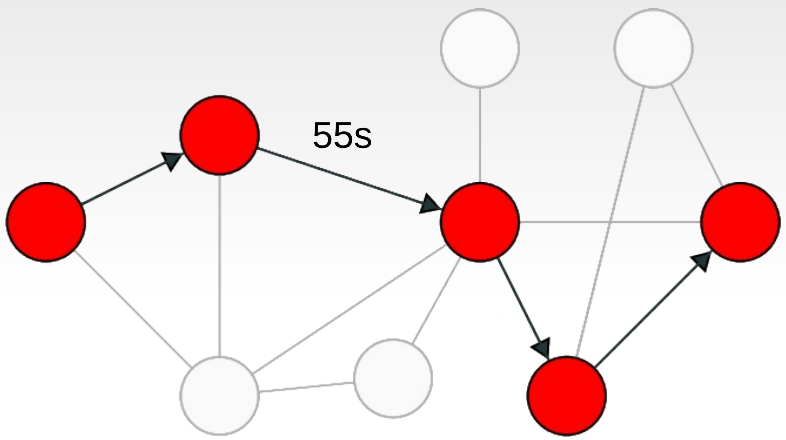
Memory Forensics as a Graph Exploration Problem



Memory Forensics as a Graph Exploration Problem

Atomicity

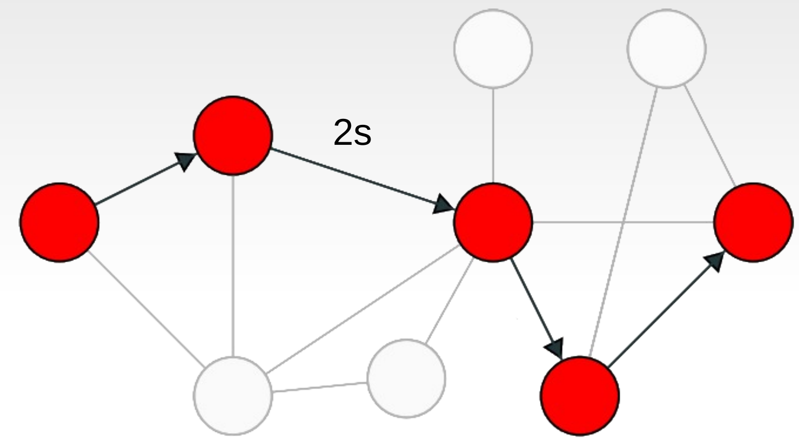
(distance between two structures)

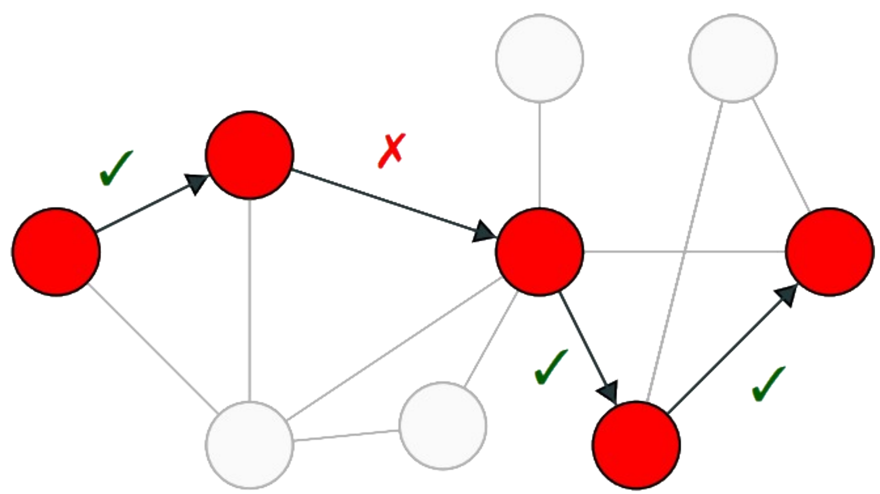


+

Stability

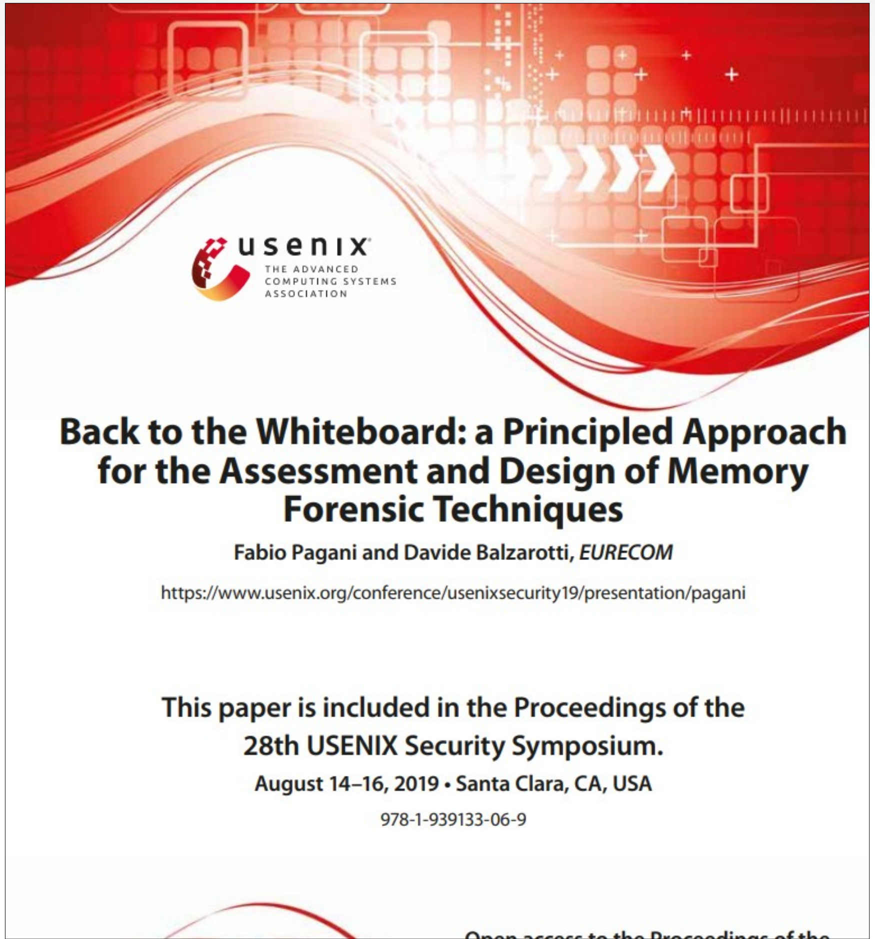
(how often the pointers are updated)





RESULTS

want to know more?


usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Back to the Whiteboard: a Principled Approach for the Assessment and Design of Memory Forensic Techniques

Fabio Pagani and Davide Balzarotti, *EURECOM*

https://www.usenix.org/conference/usenixsecurity19/presentation/pagani

This paper is included in the Proceedings of the 28th USENIX Security Symposium.

August 14–16, 2019 • Santa Clara, CA, USA

978-1-939133-06-9

Open access to the Proceedings of the

Memory Forensics 2.0 ?

Manual → Automated

Manual → Automated

Knowledge-based → Zero-Knowledge

Quantitative Measurements → Optimal Solutions
(for a given metric)

Heuristics → Optimal Solutions

Manual → Automated

Knowledge-based → Zero-Knowledge

“Wet the Appetite” by Midjourney

davide.balzarotti@eurecom.fr

@balzarot

http://s3.eurecom.fr/~balzarot