Near-memory & In-Memory Detection of Fileless Malware

Marcus Botacin[1], André Grégio[1], Marco Zanata Alves[1]

[1]Federal University of Paraná (UFPR)
{mfbotacin, gregio, mazalves}@inf.ufpr.br

MEMSYS 2020

Agenda

Agenda

What Is Fileless Malware?

Figure: **Source:** https://www.wired.com/2017/02/say-hello-super-stealthy-malware-thats-going-mainstream/

Figure: **Source:** https://www.cyberscoop.com/kaspersky-fileless-malware-memory-attribution-detection/

The Cost of Scanning Memory

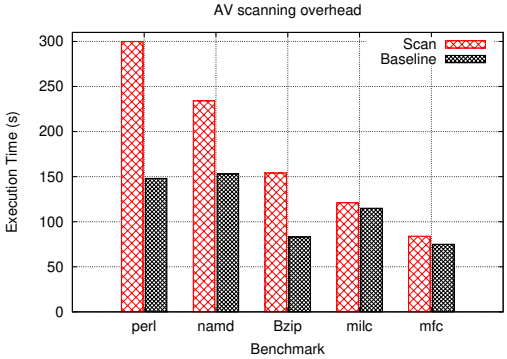

Figure: **In-memory AV scans worst-case and best-case performance penalties.**

A Drawback for Current Security Solutions

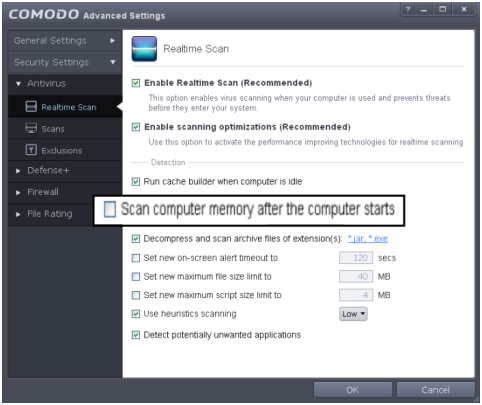

Figure: **Default policy is not to scan memory.**

Why Not New AV implementations?

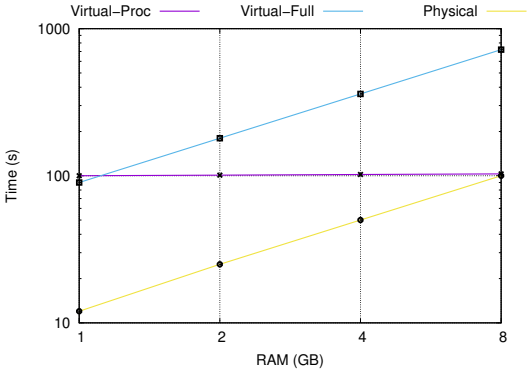

Figure: **Memory dump time for distinct software-based techniques and memory sizes.**

Understanding Malware Detection Tasks

Monitoring

- **You need to know:** When to inspect.

Classifying

- **You need to know:** What to inspect.

Can't We Rely on Page Faults?

Table: **Blocking on Page Faults**. The performance impact is greater as more complex is the applied detection routine.

Benchmark	Cycles	PF	5K	10K	20K	30K
perf	187G	1,8M	4,74%	9,48%	18,96%	28,44%
mcf	69G	375K	2,72%	5,45%	10,89%	16,34%
milc	556G	1,2M	1,05%	2,10%	4,21%	6,31%
bzip	244G	170K	0,35%	0,69%	1,38%	2,08%
namd	491G	325K	0,33%	0,66%	1,32%	1,98%

Agenda

Observing Memory Accesses Patterns

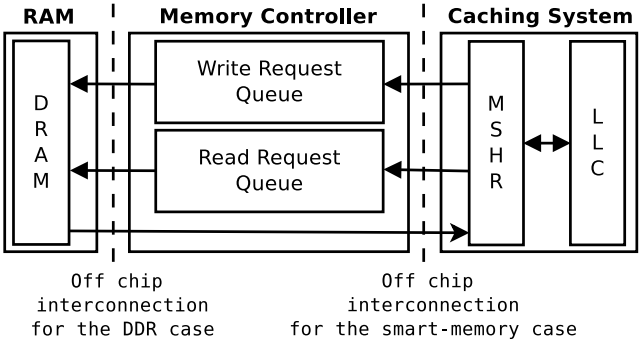

Figure: **Write-to-Read window.** Read requests originated from the MSHR might overlap other memory-buffered read requests for any address, but must not overlap previous memory-buffered write requests for the same address.

Malware Identification based on Near- and In-Memory Evaluation (MINI-ME)

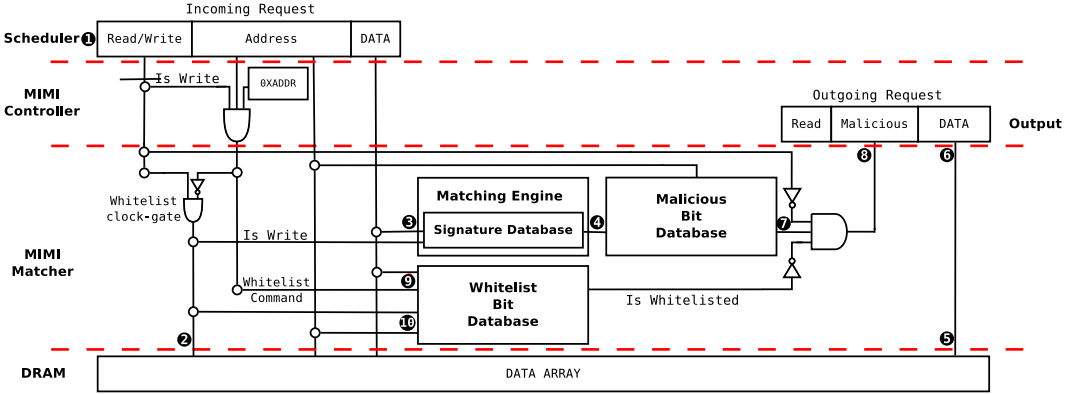

Figure: **MINI-ME Architecture.** MINI-ME is implemented within the memory controller.

Handling Notifications via Page Faults

```
1 void __do_page_fault(...) {
2     // Original Code
3     if (X86_PF_WRITE) ...
4     if (X86_PF_INSTR) ...
5     // Added Code
6     if (X86_MALICIOUS) ...
```

Code 1: **Modified PF handler.** Malicious bit is set when suspicious pages are mapped.

Agenda

How Much Performance Overhead is Acceptable?

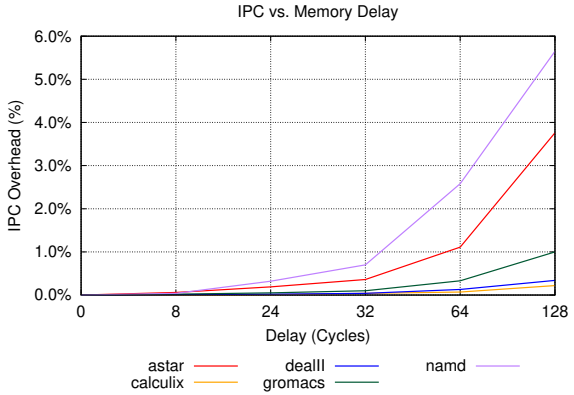

Figure: **MINI-ME database overhead.** Delays of up 32 cycles impose less than 1% of IPC overhead.

Signatures as the Detection Mechanism

```
1 if(IsDebuggerPresent()){
2   evade()
```

Code 2: C code

```
1 mov eax, [fs:0x30]
2 mov eax, [eax+0x2]
3 jne 0 <evade>
```

Code 3: ASM code

```
1 64 8b 04 25 30 00 00
2 67 8b 40 02
3 75 e1
```

Code 4: Instructions Bytes

Signature Size Definition

Table: **Signature Generation**. Signatures (%) detected as false positives for each signature size and memory dump size.

		Memory Size			
		1 GB	**2 GB**	**4 GB**	**8 GB**
Signature Size	**8 B**	8.65%	9.92%	10.18%	11.45%
	16 B	3.06%	3.32%	3.32%	3.32%
	32 B	0.00%	0.00%	0.00%	0.00%
	64 B	0.00%	0.00%	0.00%	0.00%

Matching Mechanism Definition

Table: **Matching Techniques.** FP rates for multiple signature sizes and techniques.

Match. Tech.		Signature size 8 B	16 B	32 B	64 B
	Dir. Mapped Table	8.33%	3.15%	0.00%	0.00%
	Signature Tree	8.33%	3.15%	0.00%	0.00%
	Bloom Filter	8.41%	3.47%	0.00%	0.00%

Matching Policies Definition

Table: **Scan Policies.** FP rate for multiple signature sizes and policies.

Scan Policy	Signature size 8 B	16 B	32 B	64 B
Whole Memory	8.33%	3.15%	0.00%	0.00%
Mapped Pages	0.06%	0.01%	0.00%	0.00%
Whitelist	0.00%	0.00%	0.00%	0.00%
Code-Only	0.01%	0.00%	0.00%	0.00%

MINI-ME in Practice

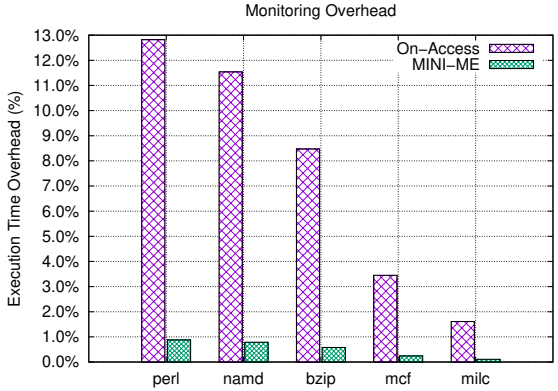

Figure: **Monitoring Overhead.** MINI-ME imposes a smaller overhead while still checking more pages than an on-access solution.

Agenda

Conclusions

Challenges & Lessons

- Fileless malware is a growing hard-to-detect class of threats.
- Traditional AntiViruses (AVs) impose significant performance overhead to perform memory scans.
- In-memory and Near-memory AVs helps reducing AV's performance overheads.
- The more complex the matching mechanism, the greater the performance overhead.
- MINI-ME as platform for future developments.

Questions & Comments.

Contact

- **mfbotacin@inf.ufpr.br**
- **twitter.com/MarcusBotacin**

Additional Material

- `https://github.com/marcusbotacin/In.Memory`