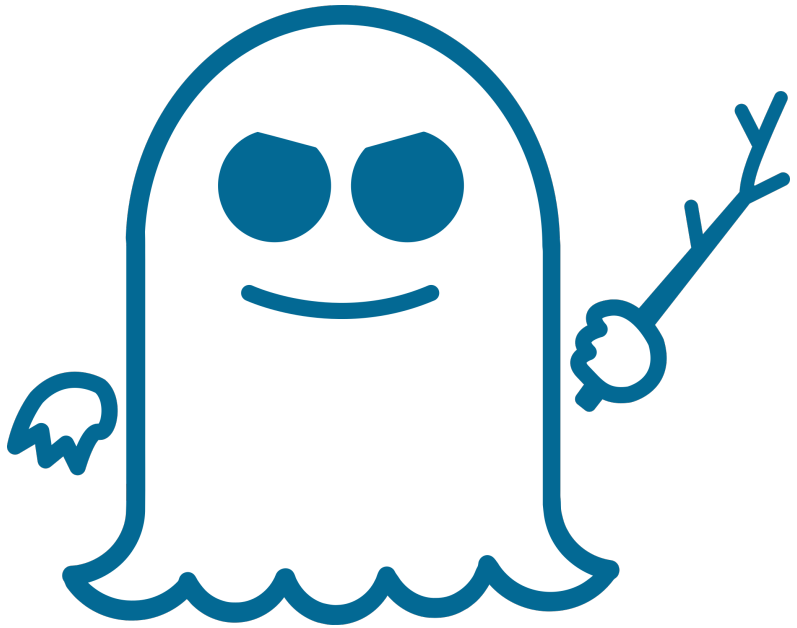
FORTINET

Are there Spectre-based malware on your Android smartphone?

Axelle Apvrille - Fortinet

Pass The Salt, July 2018

All of you have heard of Spectre *



Sophisticated and powerful cache attack on CPUs

* https://spectreattack.com/spectre.pdf

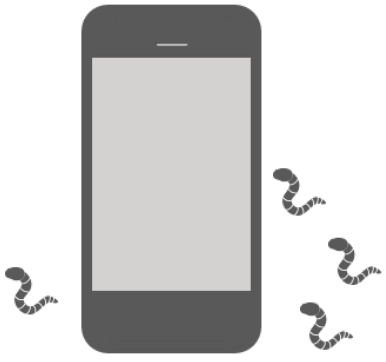
Spectre? *"Please not yet another talk!"*

Good news

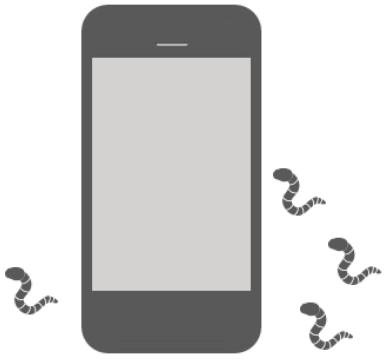
I won't repeat what's already on the web

whether you are Spectre experts or not, you should be able to follow most of it

- Overview: YouTube video
- Tech: `https://gruss.cc/files/cryptacus2018.pdf`

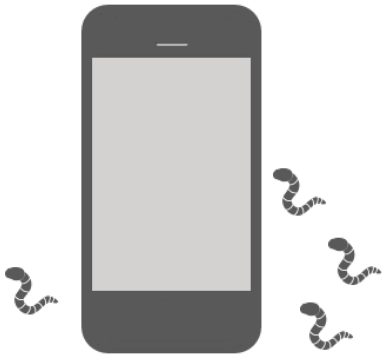


Are there Spectre malware on your Android smartphone?



Are there Spectre malware on your Android smartphone?

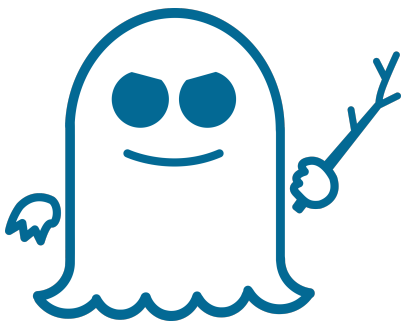
I am an *Anti-Virus* researcher at Fortinet



Are there Spectre malware on your Android smartphone?

I am an *Anti-Virus* researcher at Fortinet

Predestined for a talk on Spectre ☺



official Spectre logo

@cryptax

Are there Spectre malware on your Android smartphone?

We'll rule out *Intel x86* phones:

Are there Spectre malware on your Android smartphone?

We'll rule out *Intel x86* phones:

1. Lots of literature on Spectre for Intel x86 processors

Are there Spectre malware on your Android smartphone?

We'll rule out *Intel x86* phones:

1. Lots of literature on Spectre for Intel x86 processors
2. Most Android smartphones have an ARM processor

Part 1

Are there Spectre malware on my/your Android **ARM**-based smartphone?

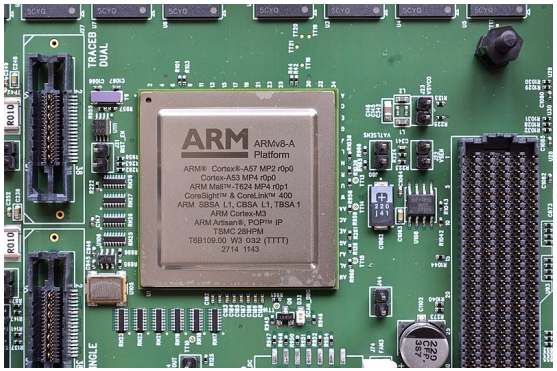
This is going to be soooo simple!

ARM published a **security update** *
Check if our processor is in the list

Processor	Variant 1	Variant 2	Variant 3	Variant 3a	Variant 4
Cortex-R7	Yes*	Yes*	No	No	No
Cortex-R8	Yes*	Yes*	No	No	No
Cortex-A8	Yes	Yes	No	No	No
Cortex-A9	Yes	Yes	No	No	No

* https://developer.arm.com/support/security-update

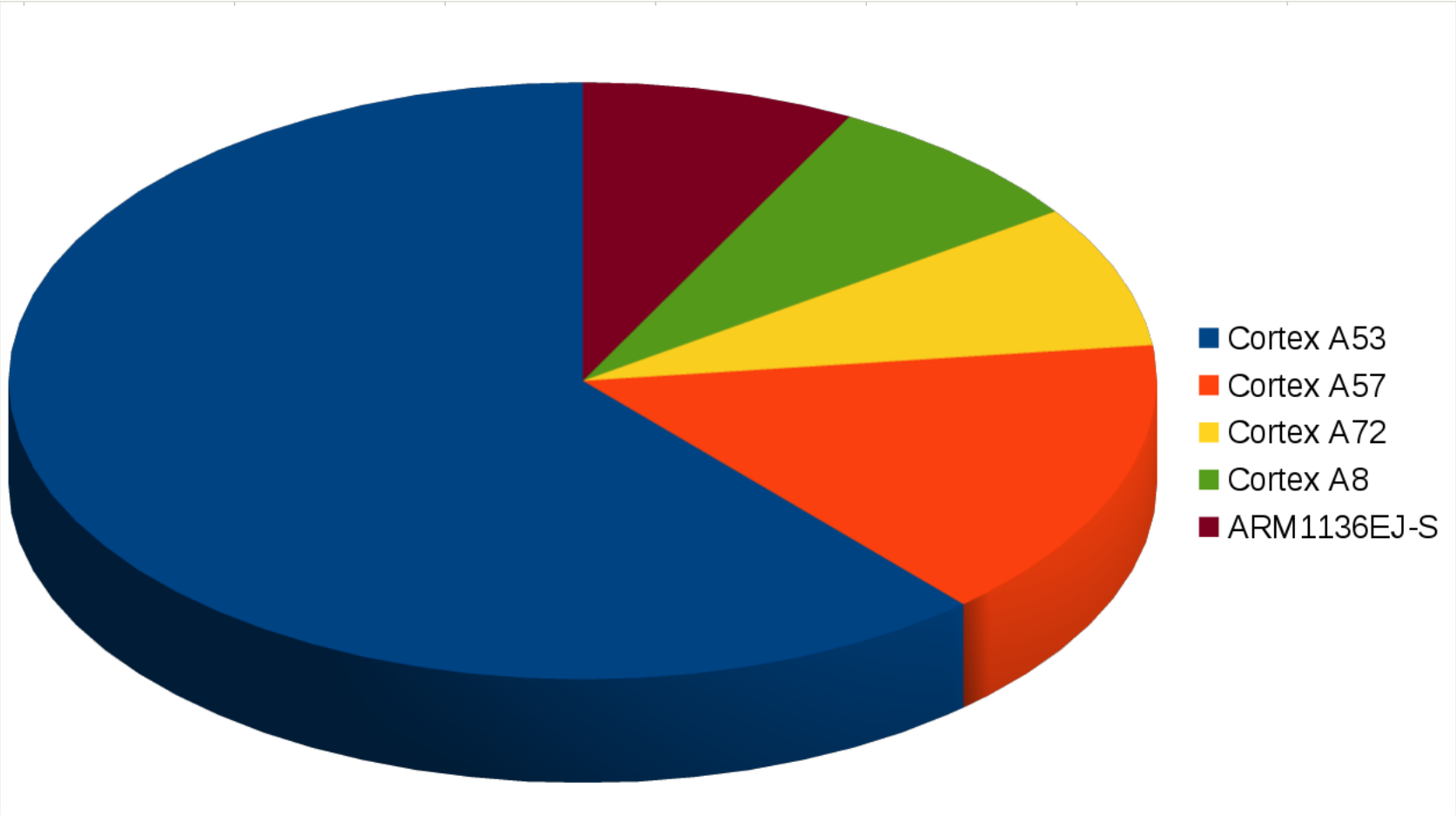
Which ARM processors do we have?



Survey among *colleagues* with an *Android* smartphone

Smartphone	Processor(s)
Huawei Honor	8x **ARM Cortex A53**
Samsung Galaxy S6	1 x ARM Cortex A57 + 1 x **ARM Cortex A53**
Samsung Galaxy J5	4 x **ARM Cortex A53**
Motorola Defy +	ARM Cortex A8
Motorola Moto E 4G	4 x **ARM Cortex A53**
..	..

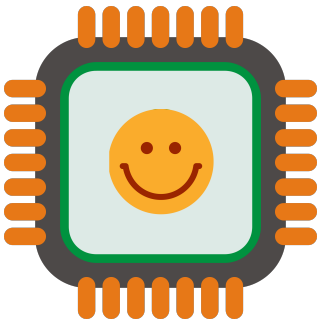
Lots of ARM Cortex A53 processors



Warning

Results among close colleagues at work.
Different from world wide statistics!

Is Cortex A53 vulnerable?



ARM says **it is not vulnerable**:

Cortex-A15	Yes	Yes	No
Cortex-A17	Yes	Yes	No
Cortex-A57	Yes	Yes	No

"Only affected cores are listed,
all other Arm cores are NOT affected."

Why isn't it vulnerable?

Cortex A53: *"in-order pipeline and advanced branch predictor"*

False Idea: in-order processors are immune to Spectre

Wrong. **Spec**tre is for **Spec**ulative Execution.

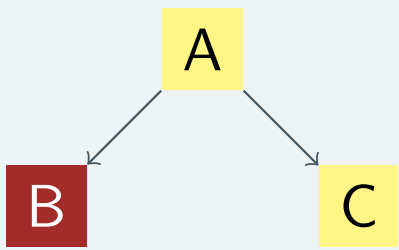
In Order

Out of Order:

"I can do C before B"

Speculative Execution:

"Assume we'll run C"



In Order/Out of Order $\neq$ Speculative Execution

The more we dig, the less we know...

ARM Cortex A53 specs:

0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
0x12_BR_PRED	Predictable branch speculatively executed
0x7A_BR_INDIRECT_SPEC	Predictable branch speculatively executed - indirect branch

The more we dig, the less we know...

ARM Cortex A53 specs:

0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
0x12_BR_PRED	Predictable branch speculatively executed
0x7A_BR_INDIRECT_SPEC	Predictable branch speculatively executed - indirect branch

Sounds like it **is vulnerable** to Spectre!

The more we dig, the less we know...

ARM Cortex A53 specs:

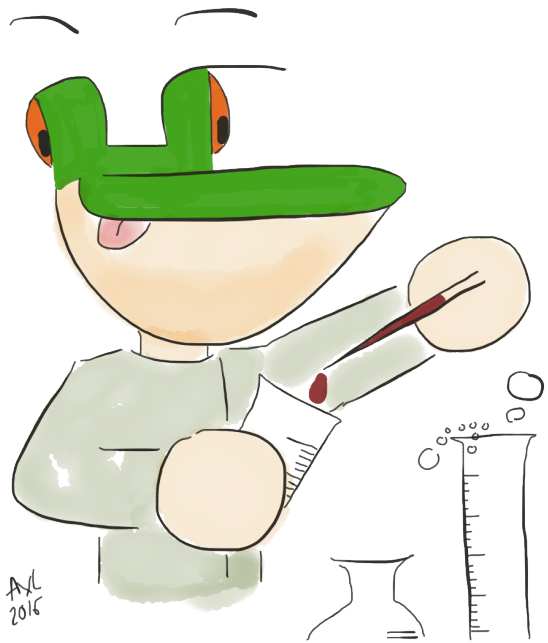
0x10_BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed
0x12_BR_PRED	Predictable branch speculatively executed
0x7A_BR_INDIRECT_SPEC	Predictable branch speculatively executed - indirect branch

Sounds like it **is vulnerable** to Spectre!

Conclusion: is it vulnerable, or not? It's not clear! ☹

Solution: test it!

1. Find an Android smartphone with ARM Cortex A53.
2. Find a PoC of Spectre for that smartphone
3. Test



Step 1: find a smartphone

No problem, I have some in the lab

Step 2: find a PoC

Spectre PoCs

- From the paper, or on github for **Intel x66**: https://github.com/Eugnis/spectre-attack/blob/master/Source.c
- Variant 1 for **Android AArch64** architectures. https://github.com/V-E-O/PoC/tree/master/CVE-2017-5753
- Variant 4 "Spectre-NG". https://www.exploit-db.com/exploits/44695/

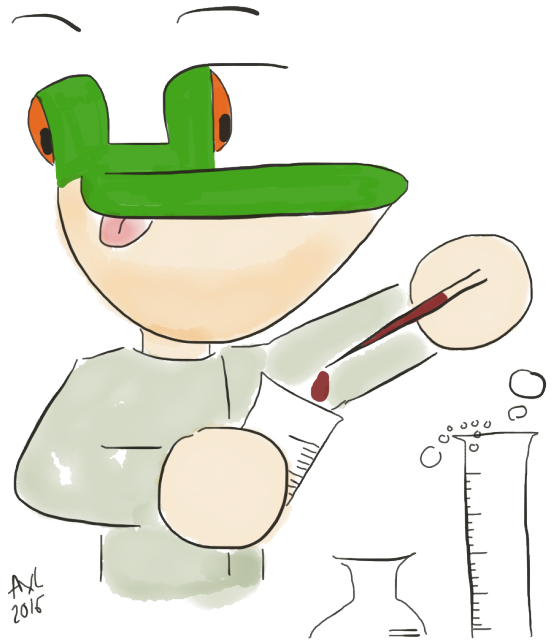
A PoC is not a malware

PoC = Proof of Concept

They recover memory areas from *your own process*!

They are not *malicious*, only a *demo*

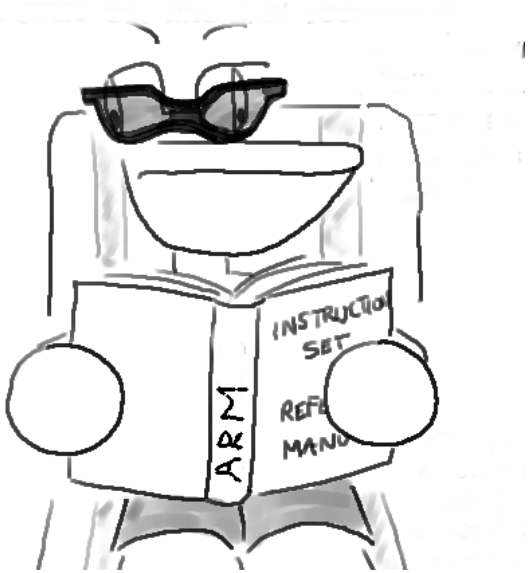
Can we use the PoC for AArch64?



Spectre PoCs

- From the paper
- Variant 1 for Android **AArch64** architectures.
- Variant 4 "Spectre-NG"

Can we use the PoC for AArch64?

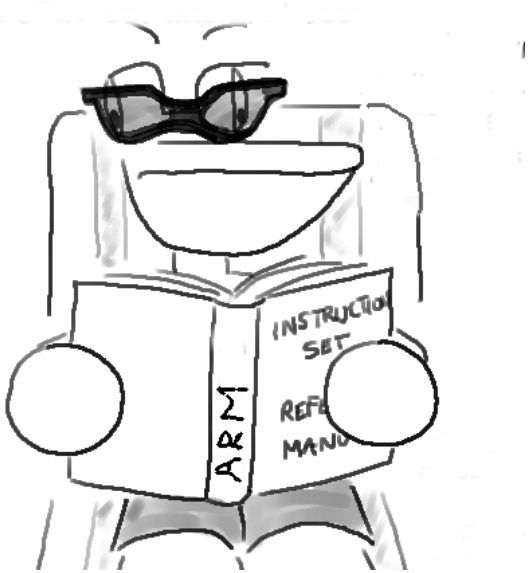


Cortex A53 characteristics

"The Cortex-A53 can be implemented

https://developer.arm.com/products/processors/cortex-a/cortex-a53

Can we use the PoC for AArch64?

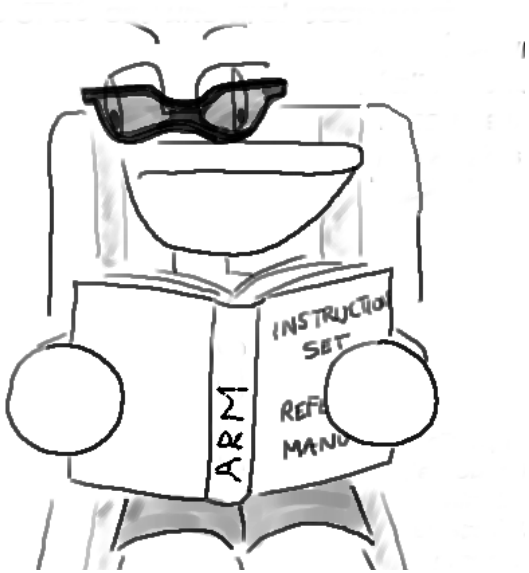


Cortex A53 characteristics

"The Cortex-A53 can be implemented in **two** *execution states:* **AArch32** *and* **AArch64***."*

https://developer.arm.com/products/processors/cortex-a/cortex-a53

Can we use the PoC for AArch64?



Cortex A53 characteristics

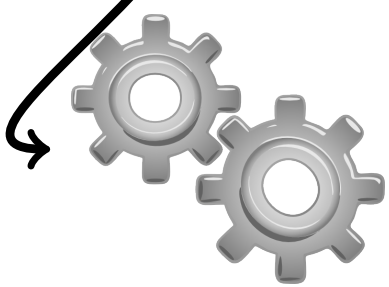
"The Cortex-A53 can be implemented in **two** *execution states:* **AArch32** *and* **AArch64***."*

- AArch32: execute ARMv7 apps - 32 bit
- AArch64: 64 bit

https://developer.arm.com/products/processors/cortex-a/cortex-a53

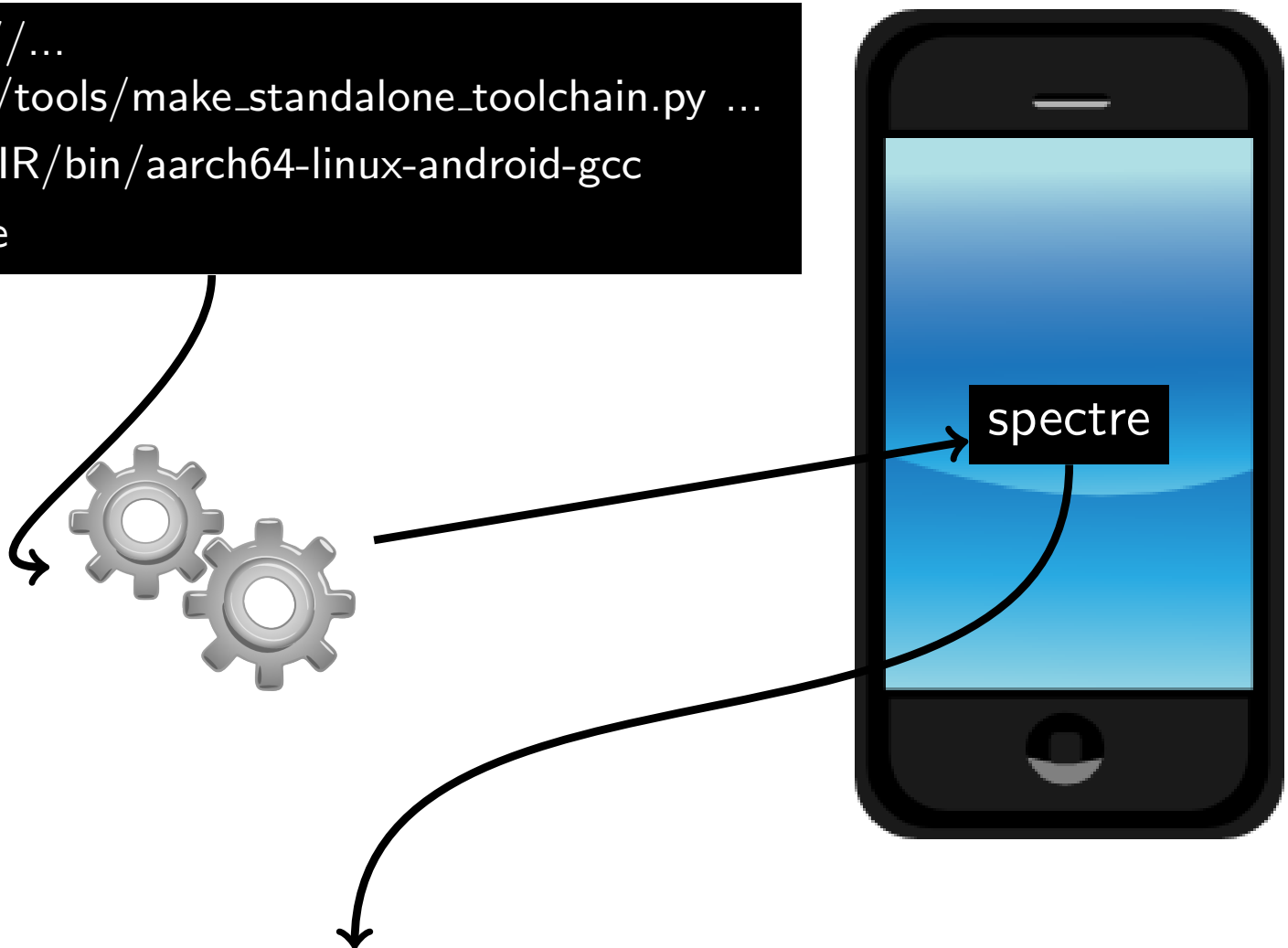
Push it and run on the smartphone

```
$ git clone https://...
$ NDK_DIR/build/tools/make_standalone_toolchain.py ...
$ TOOLCHAIN_DIR/bin/aarch64-linux-android-gcc
source.c -o spectre
```



Push it and run on the smartphone

```
$ git clone https://...
$ NDK_DIR/build/tools/make_standalone_toolchain.py ...
$ TOOLCHAIN_DIR/bin/aarch64-linux-android-gcc
source.c -o spectre
```




```
/system/bin/sh: ./spectre: not executable: 64-bit ELF file
```

Why isn't it working?

```
shell@surnia:/ $ cat /proc/cpuinfo
processor: 0
model name : ARMv7 Processor rev 0 (v7l)
BogoMIPS: 38.00
Features: swp half thumb fastmult vfp edsp ...
CPU implementer : 0x41
CPU architecture: 7
CPU variant : 0x0
CPU part: 0xd03
CPU revision: 0
```

ARMv7 is 32-bit!

64-bit capable processor
but *32-bit* stock kernel !

We need a PoC for **ARMv7** / **AArch32** (32 bit apps)

We need a PoC for **ARMv7** / **AArch32** (32 bit apps)

There are none...

We need a PoC for **ARMv7** / **AArch32** (32 bit apps)

There are none...

Let's implement one!

A PoC is not a malware

Implementation of Flush+Reload

The PoC for **Intel x86** uses:

1. **Flush** the cache: the PoC uses `_mm_clflush`

```
/// \headerfile <x86intrin.h>
///
/// This intrinsic corresponds to the <c> CLFLUSH </c> instruction
///
/// \param __p
/// A pointer to the memory location used to identify the cache li
/// flushed.
void _mm_clflush(void const * __p);
/// \brief Forces strong memory ordering (serialization) between l
/// instructions preceding this instruction and load instructions
/// this instruction, ensuring the system completes all previous l
/// executing subsequent loads.
```

2. **Read time**: the PoC uses `rdtscp`. Returns the value of the Time Stamp Counter (64-bit tick count).

Flush the cache on Android

- No `_mm_clflush`, no `clearcache` ☹
- There is a `__ARM_NR_cacheflush`

In `usr/include/asm/unistd.h`:

```
#define __ARM_NR_BASE (__NR_SYSCALL_BASE+0x0f0000)
...
#define __ARM_NR_cacheflush (__ARM_NR_BASE+2)
```

Measuring Time on Android

No `rdtscp`, no `rdtsc` on Android ☹

Re-use existing work on cache attacks for ARM:

- M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, S. Mangard, ARMageddon: Cache Attacks on Mobile Devices, USENIX Security 2016
- X. Zhang, Y. Xiao, Y. Zhang, Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices, CCS 2016

Solutions to measure time on Android

Strategy	Does it work on our smartphone?
Monitor hardware events via `perf_event_open()` syscall	Hardware counters not available on my smartphone
CPU's **P**erformance **M**onitor **U**nit	Only enabled for kernel space
Dedicated thread timer	Not precise enough
POSIX `clock_gettime()`	**OK**

Results

Run spectre with clock_gettime()

```
 Putting 'The Magic Words are Squeamish Ossifrage.' in memory
MAX_TRIES=999 CACHE_HIT_THRESHOLD=80 len=40
Reading 40 bytes:
Reading at malicious_x = 0xffffe7e4 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7e5 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7e6 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7e7 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7e8 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7e9 Success: 0xFF='?' score=0
Reading at malicious_x = 0xffffe7ea Success: 0xFF='?' score=0
```

$Score = 0$: we have no cache hit!

Tuning...

```
MAX_TRIES=5500 CACHE_HIT_THRESHOLD=364 len=40
Reading 40 bytes:
Reading at malicious_x = 0xffffe7e4 Unclear: 0x6F='o' score=809 (se
Reading at malicious_x = 0xffffe7e5 Unclear: 0xF3='?' score=809 (se
Reading at malicious_x = 0xffffe7e6 Unclear: 0xF0='?' score=877 (se
Reading at malicious_x = 0xffffe7e7 Unclear: 0xF0='?' score=839 (se
```

We still don't recover the secret ☹
Results are different at each run
It's not working ☹

Same results with ARM Cortex A8

- Older ARMv7 processor introduced in 2005
- **ARM says it is vulnerable to Spectre**
- Same results above Android 32-bit ROM: impossible to recover the secret

Conclusion

Possible conclusions:

1. "@cryptax: your implementation is wrong". Don't think so. Getting same results with `libflush` from ARMaggedon...
2. or **ARM Cortex A53 is not vulnerable to Spectre** (but we don't know why)
3. or **POSIX clock_gettime() isn't precise enough**. Option: try Spectre as kernel module.
4. or **__ARM_NR_cacheflush isn't working properly**. To do: don't use Flush+Reload but try *Prime+Probe* or *Evict+Reload*.

https://github.com/cryptax/spectre-armv7

What have we learned? Part 1

Smartphone	Is processor vulnerable?	Is smartphone vulnerable?
Low or middle range Android phones with ARM Cortex A53	Officially no, but unsure	Straight out of the box, no
Old Android phones with ARM Cortex A8	Yes	Straight out of the box, no
High end Android smartphones with 64-bit ROM	Check what ARM security update	Test AArch64 PoC

Spectre on Android

- Can smartphones be affected? **Yes!**
- A *vulnerable processor* is different from a *vulnerable system*

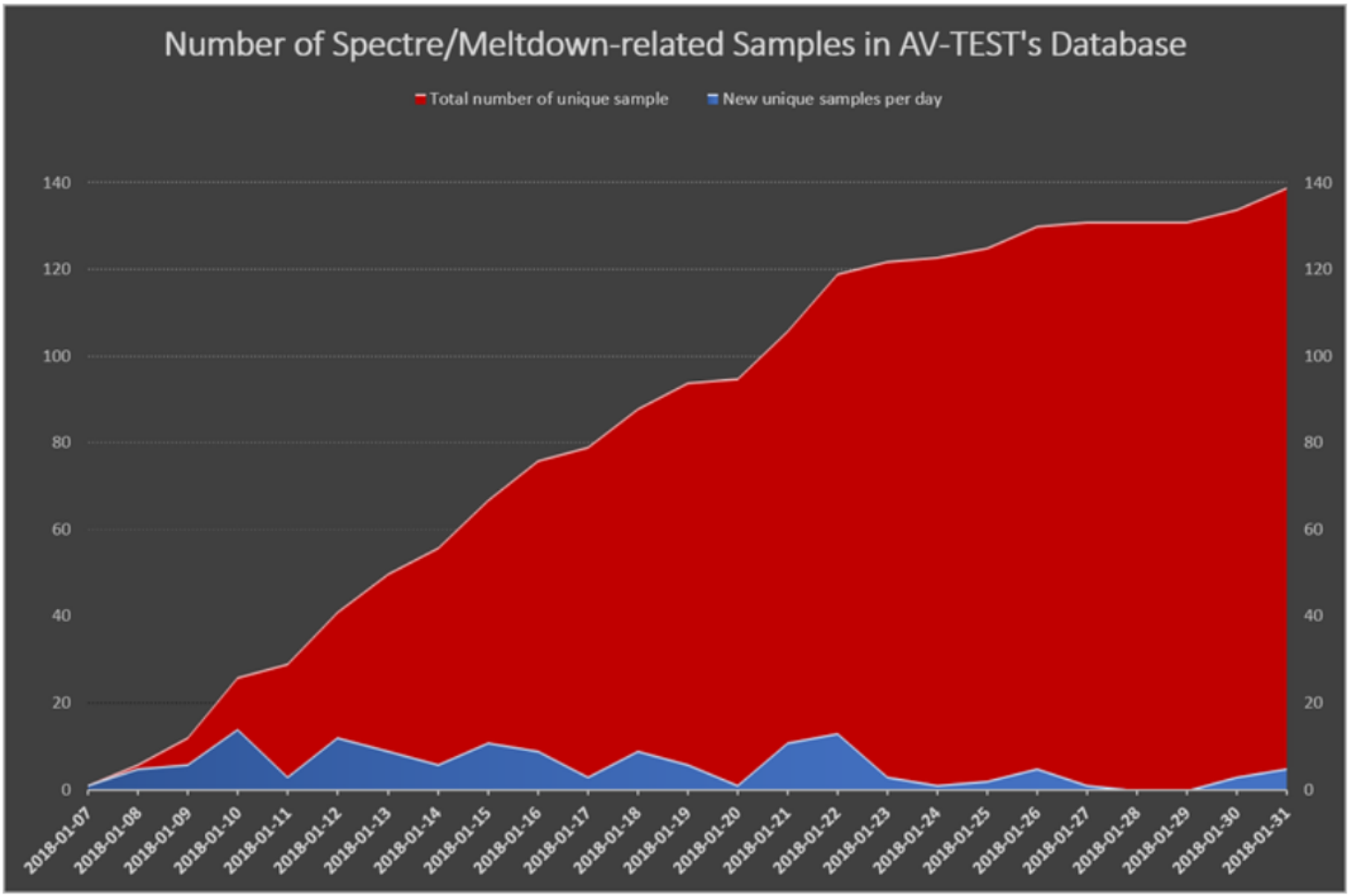
Part 2
Are there malware in the wild?

That's we read in the news (end of January 2018)

Meltdown-Spectre: Malware is already being tested by attackers

Malware makers are experimenting with malware that exploits the Spectre and Meltdown CPU bugs.

That's we read in the news (end of January 2018)



The number of potential Meltdown-Spectre malware samples collected by AV-Test has steadily climbed since the first one was spotted on January 7 to 139 by the end of January.

Image: AV-Test

That's we read in the news (end of January 2018)

Is this **true**?

Checked those samples one by one

At that time, **139** samples:

- W32/Spectre.D!tr
- Riskware/SpectrePOC
- Riskware/POC_Spectre
- Linux/Spectre!tr
- Linux/Spectre.C!tr
- Linux/Spectre.A!exploit 3043151C.vsc

All of them are **Proof of Concepts**

Renamed them to **Riskware/SpectrePOC**

A PoC is **not a malware**

A Proof of Concept demonstrates a concept works

- PoC proves cache attack works by recovering "The Magic Words are Squeamish Ossifrage"
- PoC is not malicious: "The Magic Words are Squeamish Ossifrage" is known from the beginning ☺

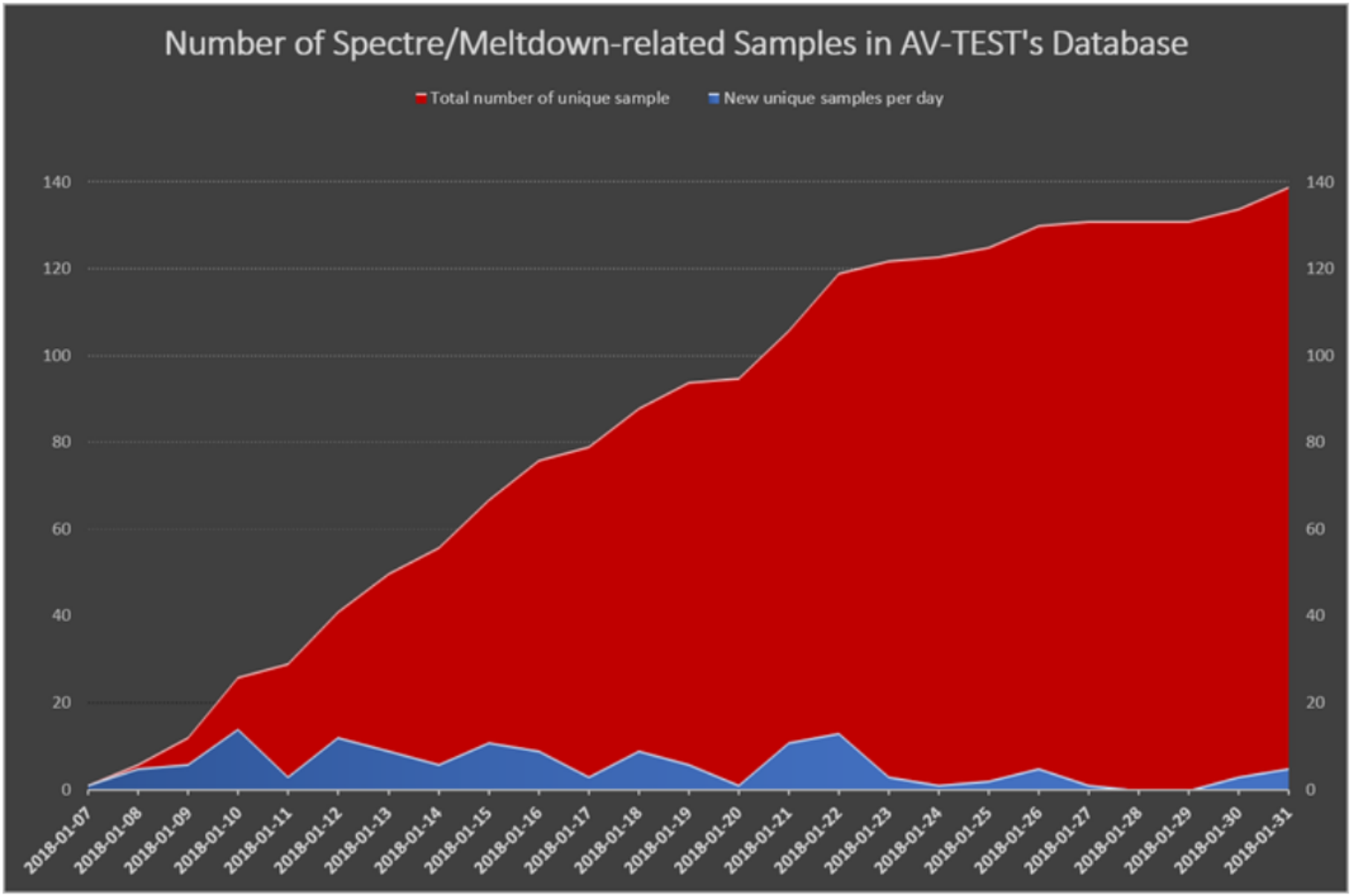
Turning the PoC into malware would require more work

- Identify a vulnerable function in targeted software **potentially long!**
- Access shared memory (inter process communication)
- Compile for given OS and CPU: **cf Android, this can be difficult**

What's true, what's wrong

- There is no **malware** yet in the wild, only **PoCs**
- Attackers are *possibly* testing / experimenting (but we don't have the proof for that)

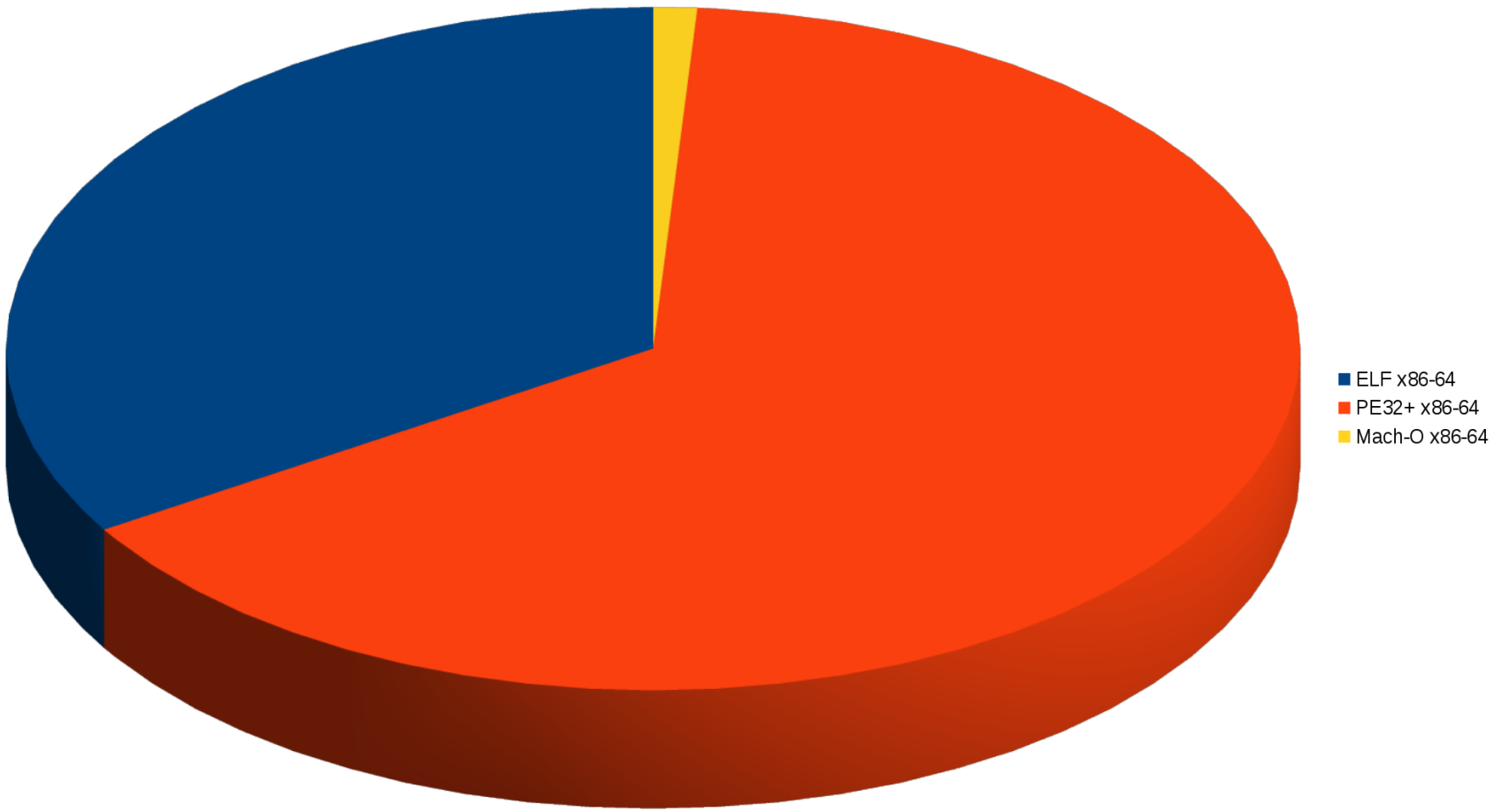
The graph is correct, but the label is wrong



The number of potential Meltdown-Spectre malware samples collected by AV-Test has steadily climbed since the first one was spotted on January 7 to 139 by the end of January.

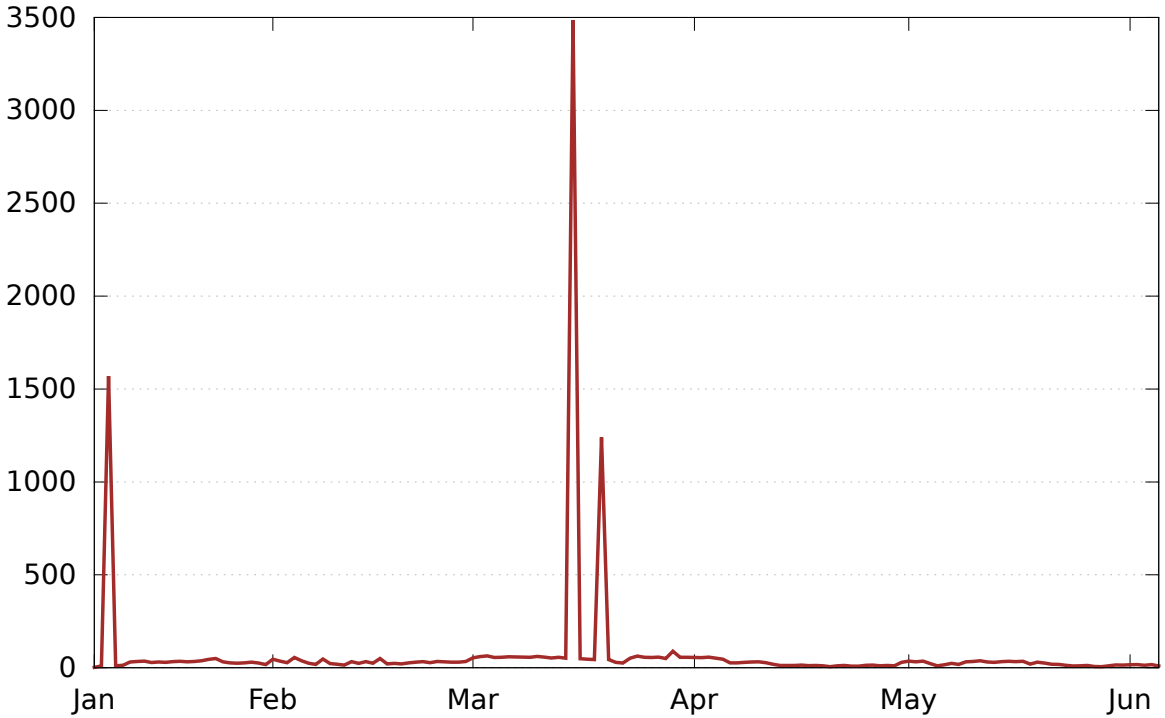
Image: AV-Test

Spectre PoCs status - June 2018



183 PoCs: 119 PE32+, 62 ELF, 2 Mach-O

Detection hits for Spectre Proof of Concepts

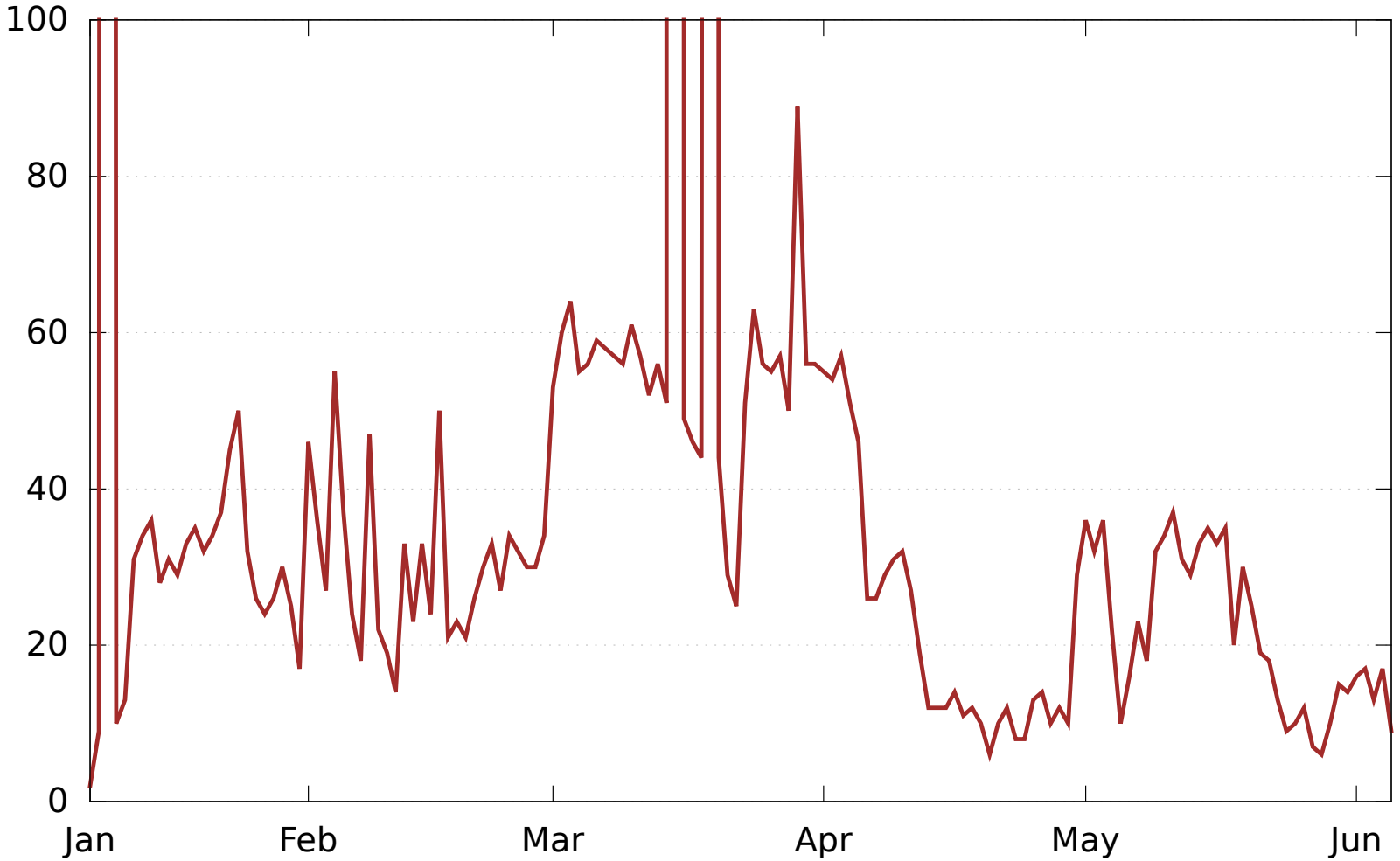


Hits on Riskware/SpectrePoC from Fortinet products (when enabled) in 2018

- Jan 3. Spectre vulnerability publicly disclosed
- Jan 27-29. Patches for Windows
- March 1-13. More patches
- May 3. Spectre-NG

January spike: initial release of signatures
March spikes: customers testing after several patches of Microsoft?

Detection hit details for Proof of Concepts



Apart from spikes, average 40 hits / day
Less starting in April

No Spectre **malware** currently

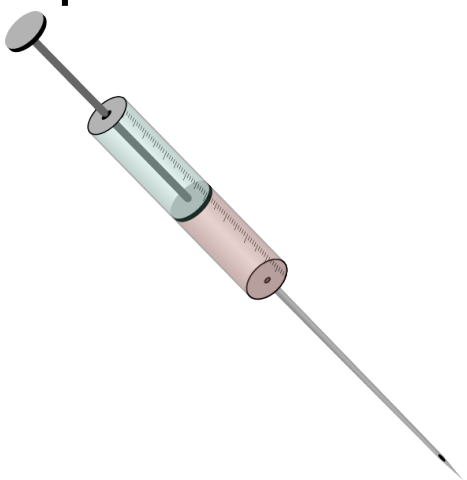
No Spectre **malware** currently

And later?

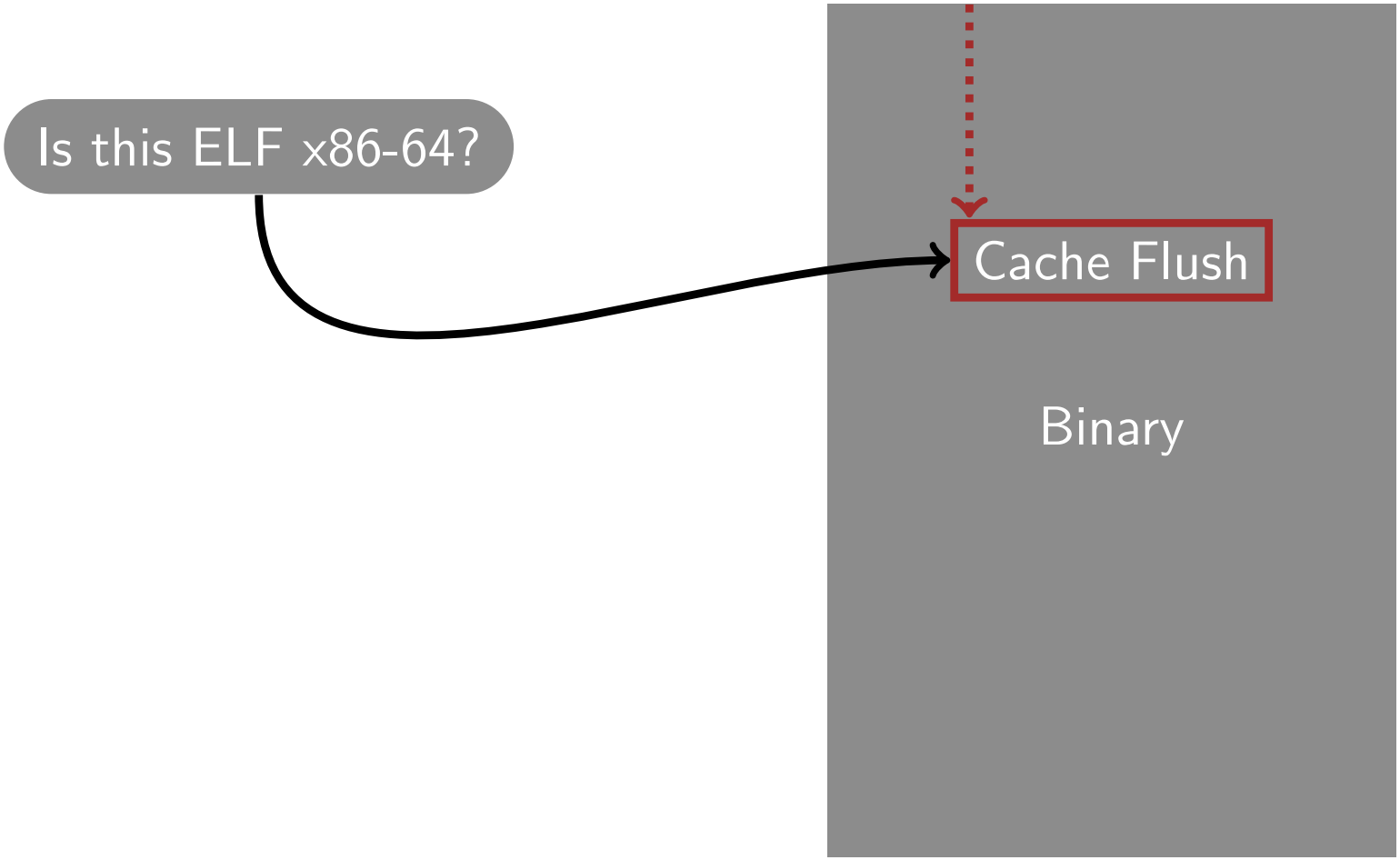
No Spectre **malware** currently

And later?

We need pro-active detection!

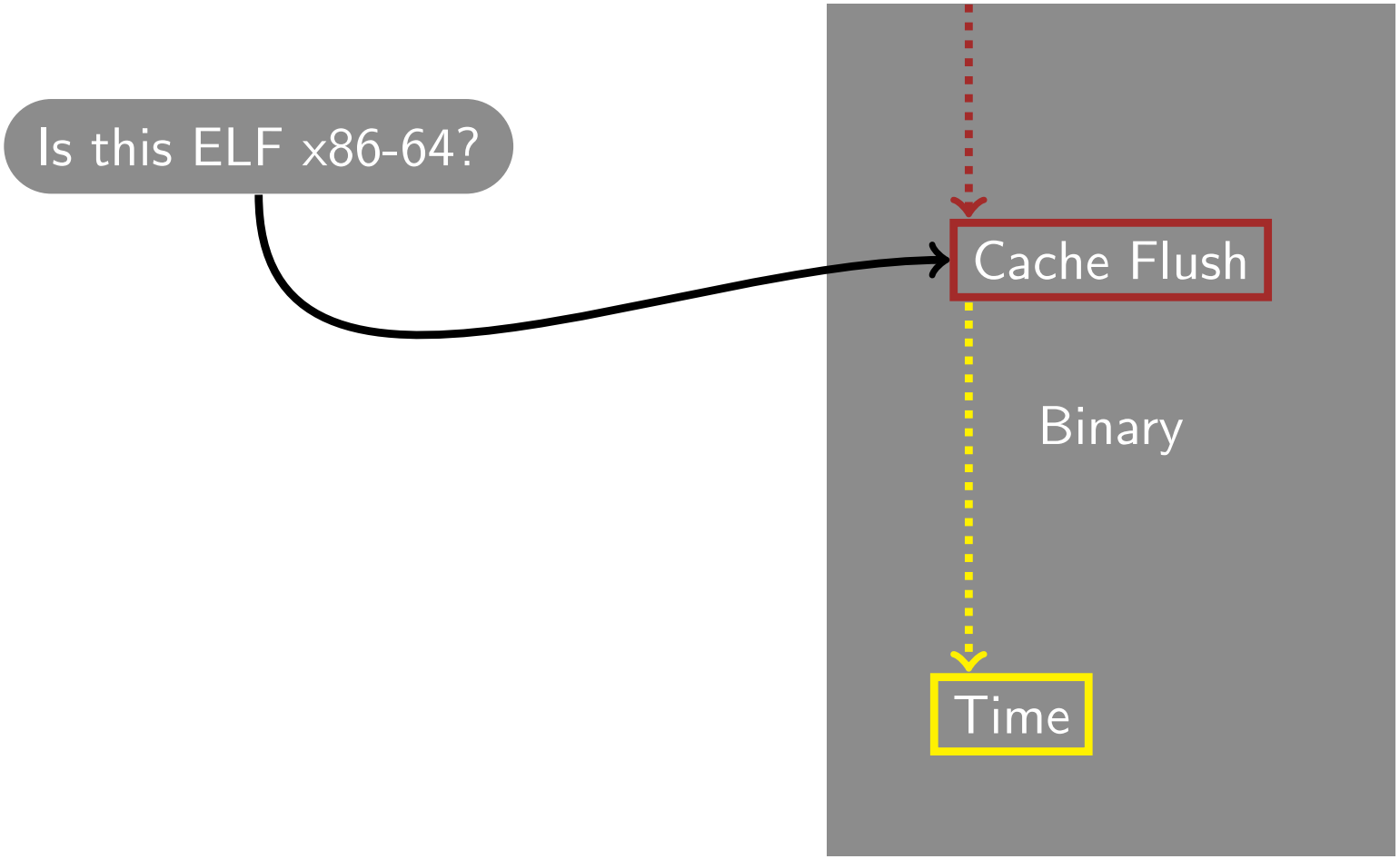


Detect Flush+Reload cache attacks



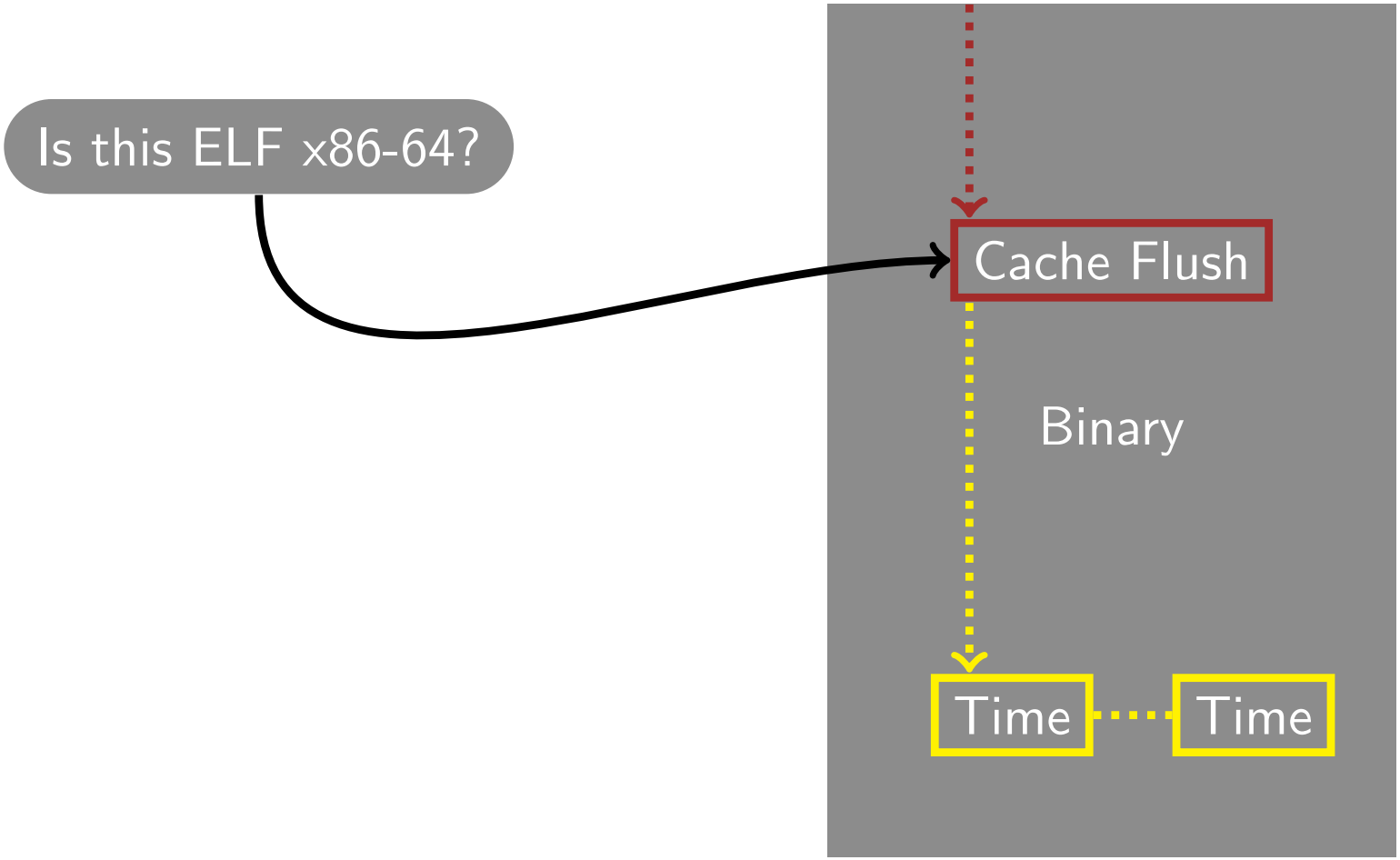
In AV, this is called a *signature*. Though it is **not a cryptographic signature** (nor a hash), rather a *detection pattern*.

Detect Flush+Reload cache attacks



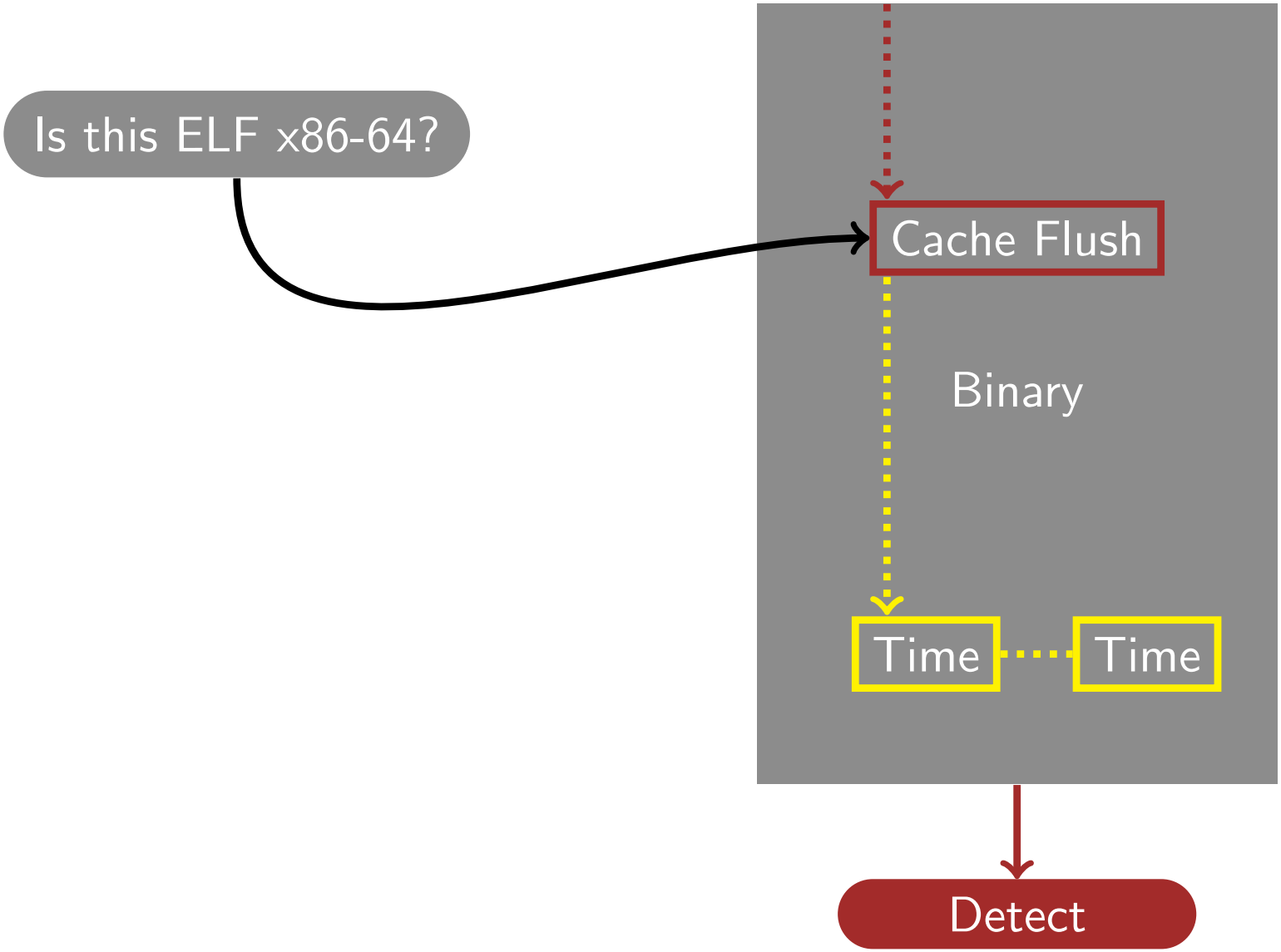
In AV, this is called a *signature*. Though it is **not a cryptographic signature** (nor a hash), rather a *detection pattern*.

Detect Flush+Reload cache attacks



In AV, this is called a *signature*. Though it is **not a cryptographic signature** (nor a hash), rather a *detection pattern*.

Detect Flush+Reload cache attacks



In AV, this is called a *signature*. Though it is **not a cryptographic signature** (nor a hash), rather a *detection pattern*.

This *signature* is far from perfect

This *signature* is far from perfect

Time-consuming (full binary search)
High risk of **False Positives**
Does not detect *Prime+Probe* etc
Always possible to evade

This *signature* is far from perfect

Time-consuming (full binary search)
High risk of **False Positives**
Does not detect *Prime+Probe* etc
Always possible to evade
but let's try it

```
"./2FC4432E.vsc" is infected with the "Linux/FlushReload.A!tr" viru
"./2FC0C6A4.vsc" is infected with the "Linux/FlushReload.A!tr" viru
"./2FC4A10C.vsc" is infected with the "Linux/FlushReload.A!tr" viru
[Summary]  Scanned: 62  Infected: 38  Total bytes: 1.614MiB  Time:
```

Quite good: 38 detections in one shot!

Why are we missing some samples?

We do have 2 rdtscp instructions

```
0x004006af  mov eax, dword [i]

0x004006b2  imul eax, eax, 0xa7
0x004006b8  add eax, 0xd
0x004006bb  and eax, 0xff
0x004006c0  mov dword [mix_i], eax
0x004006c3  mov eax, dword [mix_i]

0x004006c6  shl eax, 9
0x004006c9  cdqe
0x004006cb  add rax, obj.array2
0x004006d1  mov qword [addr], rax
0x004006d5  lea rax, [local_68h]
0x004006d9  mov qword [local_28h], rax
0x004006dd  rdtscp
0x004006e0  mov esi, ecx
0x004006e2  mov rcx, qword [local_28h]
0x004006e6  mov dword [rcx], esi
0x004006e8  shl rdx, 0x20
0x004006ec  or rax, rdx
0x004006ef  mov rbx, rax

0x004006f2  mov rax, qword [addr]
0x004006f6  movzx eax, byte [rax]
0x004006f9  movzx eax, al
0x004006fc  mov dword [local_68h], eax
0x004006ff  lea rax, [local_68h]
0x00400703  mov qword [local_20h], rax
0x00400707  rdtscp
0x0040070a  mov esi, ecx
0x0040070c  mov rcx, qword [local_20h]
0x00400710  mov dword [rcx], esi
0x00400712  shl rdx, 0x20
0x00400716  or rax, rdx
0x00400719  sub rax, rbx

0x0040071c  mov rbx, rax
```

```
while (*(i) <= 0xff) {
    eax = *(i);
    /* ((i*167) + 13) & 255 */
    eax *= eax;
    eax += 0xd;
    eax &= 0xff;
    *(mix_i) = eax;
    eax = *(mix_i);
    /* mix_i * 512 */
    eax <<= 9;
    rax = eax;
    rax += array2;
    *(addr) = rax;
    rax = local_68h;
    *(local_28h) = rax;
    __asm (rdtscp);
    esi = ecx;
    rcx = *(local_28h);
    *(rcx) = esi;
    rdx <<= 0x20;
    rax |= rdx;
    rbx = rax;
    /* memory access to time */
    rax = *(addr);
    eax = rax;
    eax = al;
    *(local_68h) = eax;
    rax = local_68h;
    *(local_20h) = rax;
    __asm (rdtscp);
    esi = ecx;
    rcx = *(local_20h);
    *(rcx) = esi;
    rdx <<= 0x20;
    rax |= rdx;
    rax -= rbx;
    /* get time difference */
    rbx = rax;
```

Missing cache flush! (bad)

```
0x004005ff  jle 0x4005e4

0x00400601  mov dword [tries], 0x3e7
0x00400608  jmp 0x400835
            ????????
0x0040060d  mov eax, dword [tries]

0x00400610  mov ecx, dword [obj.array1_size]
0x00400616  mov edx, 0
0x0040061b  div ecx
0x0040061d  mov eax, edx
0x0040061f  mov eax, eax
0x00400621  mov qword [trainingX], rax

0x00400625  mov dword [j], 0x1d
0x0040062c  jmp 0x40069d
```

```
}

/* init tries=999 */
*(tries) = 0x3e7;

while (*(tries) > 0) {
            ??????????
    eax = *(tries);
    /* computing training_x = tries % array1_size */
    ecx = *(array1_size);
    edx = 0;
    __asm (div ecx);
    eax = edx;
    eax = eax;
    *(trainingX) = rax;
    /* init j=29 */
    *(j) = 0x1d;

    while (*(j) < 0) {
```

This is a *damaged* sample. Won't work.

Good: We don't care our signature does not detect it ☺

Cache attacks are **not** common in malware

Status	Count
Total	661977
DetectionLoss	661941 (99.99%)
Infected	36 (0%)
Crashed	0
Timeout	0
Code injection	0
dynamic memory	0
dump scan	0

Test Detail

Test ID:	1070372
Test Name:	spectre
Test Type:	DetectionTest
Fileset:	msb_ELF_6months
Priority:	normal
Status:	Done [Clients Detail]
Submit Time:	2018-06-17 23:56:18
Start:	2018-06-17 22:43:54
Finish:	2018-06-18 23:56:17
Duration:	1445m
Scan Time:	4m 41s

Signature only caught Spectre PoC samples.
No Linux malware currently using Flush+Reload

Conclusion - Part 2

Spectre malware

- Currently, **no Spectre malware**, only PoCs for W32, Linux and Mac. **Nothing for ARM-based smartphones (or other IoT)**
- **Cache attacks are not common in malware**
- Will there be Spectre malware in the future?

Thanks

@TuxDePoinsisse, Daniel Gruss, Adam Shewchuk, Renaud Pacalet

aapvrille (at) fortinet (dot) com - @cryptax

Smart devices CTF

December 14, 2018 - https://ph0wn.org