Meltdown, Spectre, ZombieLoad

Daniel Gruss
May 16, 2019

Graz University of Technology
You realize it is something big when...

• it is in the news, all over the world
• you get a Wikipedia article in multiple languages
• there are comics, including xkcd
• you get a lot of Twitter followers after Snowden mentioned you
You realize it is something big when...

- it is in the news, all over the world
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
- you get a lot of Twitter followers after Snowden mentioned you
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
- you get a lot of Twitter followers after Snowden mentioned you
You realize it is something big when...

• it is in the news, all over the world
• you get a Wikipedia article in multiple languages
• there are comics, including xkcd
• you get a lot of Twitter followers after Snowden mentioned you
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
- you get a lot of Twitter followers after Snowden mentioned you

SECURITY FLAW REVEALED

<table>
<thead>
<tr>
<th>Intel (Prev)</th>
<th>45.26</th>
<th>-1.59</th>
<th>[-3.39%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel (After Hours)</td>
<td>44.85</td>
<td>-0.41</td>
<td>[-0.91%]</td>
</tr>
</tbody>
</table>

SHROUT: ISSUE NOT UNIQUE TO INTEL, BUT IT'S AFFECTED THE MOST

Daniel Gruss — Graz University of Technology
Der Kernschmelzer

Daniel Gruss hat eine schwere Sicherheitslücke in Computerchips entdeckt. Warum gelingt dem Informatiker, woran die Hersteller scheitern?

Von Jens Tönnesmann

You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
Kernel page-table isolation

From Wikipedia, the free encyclopedia

"KPTI" redirects here. For other uses, see KPTI (disambiguation).

Kernel page-table isolation (KPTI or PTI,[1] previously called KAISER)[2][3] is a Linux kernel feature that mitigates the Meltdown security vulnerability (affecting mainly Intel's x86 CPUs)[4] and improves kernel hardening against attempts to bypass kernel address space layout randomization (KASLR). It works by better isolating user space and kernel space memory.[5][6] KPTI was merged into Linux kernel version 4.15,[7] and backported to Linux kernels 4.14.11, 4.9.75, 4.4.110.[8][9][10] Windows[11] and macOS[12] released similar updates. KPTI does not address the related Spectre vulnerability.[13]

Contents [hide]
1 Background on KAISER
2 Meltdown vulnerability and KPTI
3 Implementation
4 References

One set of page table for use in kernel mode includes both kernel-space and user-space. The second set of page table for use in user mode...
You realize it is something big when...

• it is in the news, all over the world
• you get a Wikipedia article in multiple languages
• there are comics, including xkcd
• you get a lot of Twitter follower after Snowden mentioned you

Meltdown is a hardware vulnerability affecting Intel x86 microprocessors and some ARM-based microprocessors. It allows a rogue process to read all memory, even when it is not authorized to do so.

Meltdown affects a wide range of systems. At the time of disclosure, this included all devices running any but the most recent and patched versions of iOS, Linux, macOS, or Windows. Accordingly, many servers and cloud services were impacted, as well as a potential majority of smart devices and embedded devices using ARM-based processors (mobile devices, smart TVs and others), including a wide range of networking equipment. A purely software workaround to Meltdown has been assessed as slowing computers between 5 and 30 percent in certain specialized workloads, although companies responsible for software correction of the exploit are reporting minimal impact from general benchmark testing.

Meltdown was issued a Common Vulnerabilities and Exposures ID of CVE-2017-5754, also known as Rogue Data Cache Load, in January 2018. It was disclosed in conjunction with another exploit, Spectre, with which it shares some, but not all characteristics. The Meltdown and Spectre vulnerabilities are considered "catastrophic"
Spectre (security vulnerability)

Spectre is a vulnerability that affects modern microprocessors that perform branch prediction.[1][2][3] On most processors, the speculative execution resulting from a branch misprediction may leave observable side effects that may reveal private data to attackers. For example, if the pattern of memory accesses performed by such speculative execution depends on private data, the resulting state of the data cache constitutes a side channel through which an attacker may be able to extract information about the private data using a timing attack.[4][5][6]

Two Common Vulnerabilities and Exposures IDs related to Spectre, CVE-2017-5753 and CVE-2017-5715 (bounds check bypass) and CVE-2017-5715 (branch target injection), have been issued.[7] JIT engines used for JavaScript were found vulnerable. A website can read data stored in the browser for another website, or the browser’s memory itself.[8]

Several procedures to help protect home computers and related devices from the Spectre (and Meltdown) security vulnerabilities have been published.[9][10][11][12] Spectre patches have been reported to significantly slow down performance, especially on older computers; on the newer 8th generation Core platforms, benchmark performance drops of 2–14 percent have been measured.[13] Meltdown patches may also produce performance loss.[5][14][15] On January 18, 2018, unwanted reboots, even for newer Intel chips, due to
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
You realize it is something big when...

• it is in the news, all over the world
• you get a Wikipedia article in multiple languages
• there are comics, including xkcd
• you get a lot of Twitter followers after Snowden mentioned you

The MELTDOWN and SPECTRE exploits use "speculative execution." What's that?

YOU KNOW THE TROLLEY PROBLEM? WELL, FOR A WHILE NOW, CPU's have basically been sending trolleys down BOTH paths, quantum-style, while awaiting your choice. Then the unneeded "phantom" trolley disappears.

That sounds bad. Honestly, I've been assuming we were doomed ever since I learned about rowhammer.

WHAT'S THAT?

If you toggle a row of memory cells on and off really fast, you can use electrical interference to flip nearby bits and—do lie just suck at computers?

Yup. Especially shared ones.

So you're saying the cloud is full of phantom trolleys armed with hammers.

...Yes, that is exactly right.

Okay. Ill, uh... install updates?

Good idea.
The Fallout

You realize it is something big when...

• it is in the news, all over the world
• you get a Wikipedia article in multiple languages
• there are comics, including xkcd
• you get a lot of Twitter followers after Snowden mentioned you
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
- you get a lot of Twitter follower after Snowden mentioned you
You realize it is something big when...

- it is in the news, all over the world
- you get a Wikipedia article in multiple languages
- there are comics, including xkcd
- you get a lot of Twitter followers after Snowden mentioned you

Edward Snowden @Snowden

You may have heard about @Intel's horrific #Meltdown bug. But have you watched it in action? When your computer asks you to apply updates this month, don't click "not now." (via spectreattack.com & @misc0110)


152 views | 6.547 favorites | 6.512 comments
FANTASTIC TIMERS
AND WHERE TO FIND THEM
HIGH-RESOLUTION MICROARCHITECTURAL ATTACKS IN JAVASCRIPT
Stealing Bitcoins?
Stealing Bitcoins?

Daniel Gruss — Graz University of Technology
Stealing Bitcoins?

Daniel Gruss — Graz University of Technology
Stealing Bitcoins?
Stealing Bitcoins?
Application

Untrusted part

Operating System
Application

Untrusted part

Create Enclave

Operating System
Application

Untrusted part

- Create Enclave

Trusted part

- Trusted Fnc.

Call Gate

Operating System

Daniel Gruss — Graz University of Technology
SGX

Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Operating System

Daniel Gruss — Graz University of Technology
Application

Untrusted part
- Create Enclave
- Call Trusted Fnc.

Trusted part
- Trusted Fnc.

Call Gate

Operating System
SGX

Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Operating System

Daniel Gruss — Graz University of Technology
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System

SGX
Application

<table>
<thead>
<tr>
<th>Untrusted part</th>
<th>Trusted part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Enclave</td>
<td>Trusted Fnc.</td>
</tr>
<tr>
<td>Call Trusted Fnc.</td>
<td>Return</td>
</tr>
</tbody>
</table>

Operating System

Daniel Gruss — Graz University of Technology
Application

Untrusted part

Create Enclave

Call Trusted Fnc.

...

Trusted part

Call Gate

Trusted Fnc.

Return

Operating System

---

Daniel Gruss — Graz University of Technology
Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks. It is the enclave developer's responsibility to address side-channel attack concerns.
Protection from Side-Channel Attacks
Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks.
Protection from Side-Channel Attacks

Intel SGX does not provide explicit protection from side-channel attacks. It is the enclave developer’s responsibility to address side-channel attack concerns.
CAN'T BREAK YOUR SIDE-CHANNEL PROTECTIONS

IF YOU DON'T HAVE ANY
SGX Wallets

- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX
SGX Wallets

- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

Teechain

[...] We assume the TEE guarantees to hold
**SGX Wallets**

- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

---

**Teechain**

[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE.
SGX Wallets

- Ledger SGX Enclave for blockchain applications
- BitPay Copay Bitcoin wallet
- Teechain payment channel using SGX

**Teechain**

[...] We assume the TEE guarantees to hold and do not consider side-channel attacks [5, 35, 46] on the TEE. Such attacks and their mitigations [36, 43] are outside the scope of this work. [...]
Attacking a weak RSA implementation inside SGX

Raw Prime+Probe trace...
Attacking a weak RSA implementation inside SGX

...processed with a simple moving average...
Attacking a weak RSA implementation inside SGX

...allows to clearly see the bits of the exponent
YOU CAN'T DO THAT!

THAT'S AGAINST THE RULES!
WANT TO DISCUSS THREAT MODELS NOW?
1337 4242

FOOD CACHE

**Revolutionary** concept!

Store your food at home, never go to the grocery store during cooking.

Can store **ALL** kinds of food.

ONLY TODAY INSTEAD OF $1,300

$1,299

ORDER VIA PHONE: +555 12345
printf("%d", i);
printf("%d", i);
printf("%d", i);
Cache miss
printf("%d", i);
printf("%d", i);
printf("%d", i);
```c
printf("%d", i);
printf("%d", i);
```
printf("%d", i);
printf("%d", i);
printf("%d", i);
printf("%d", i);

Cache miss
Request
Response

Cache hit

Daniel Gruss — Graz University of Technology
CPU Cache

printf("%d", i);

Cache miss
DRAM access, slow

printf("%d", i);

Cache hit

Daniel Gruss — Graz University of Technology
CPU Cache

printf("%d", i);

Cache miss

DRAM access,
slow

Cache hit

No DRAM access,
much faster

printf("%d", i);

Request

Response

Daniel Gruss — Graz University of Technology
Flush+Reload

Shared Memory

ATTACKER

flush

access

VICTIM

access
Flush+Reload

ATTACKER

Shared Memory

VICTIM

flush

access

cached

cached

flush

access

cached

cached

Shared Memory

Daniel Gruss — Graz University of Technology
Flush+Reload

ATTACKER

flush
access

Shared Memory

VICTIM

access
Flush+Reload

ATTACKER
flush
access

Shared Memory

VICTIM
access

Daniel Gruss — Graz University of Technology
Flush+Reload

Shared Memory

ATTACKER
flush
access

VICTIM
access
Flush+Reload

ATTACKER

flush
access

Shared Memory

VICTIM

access
Flush+Reload

ATTACKER

flush

access

Shared Memory

VICTIM

access

Daniel Gruss — Graz University of Technology
Flush+Reload

Shared Memory

ATTACKER
flush
access

fast if victim accessed data, slow otherwise

VICTIM
access

Daniel Gruss — Graz University of Technology
7. Serve with cooked and peeled potatoes
Wait for an hour
Wait for an hour
LATENCY
1. Wash and cut vegetables

2. Pick the basil leaves and set aside

3. Heat 2 tablespoons of oil in a pan

4. Fry vegetables until golden and softened
1. Wash and cut vegetables
2. Pick the basil leaves and set aside
3. Heat 2 tablespoons of oil in a pan
4. Fry vegetables until golden and softened
int width = 10, height = 5;

float diagonal = sqrt(width * width + height * height);
int area = width * height;

printf("Area %d x %d = %d\n", width, height, area);
int width = 10, height = 5;

float diagonal = sqrt(width * width + height * height);

int area = width * height;

printf("Area %d x %d = %d\n", width, height, area);
```c
char data = *(char*)0xffffffff81a000e0;
printf("%c\n", data);
```
char data = *(char*)0xfffffffff81a000e0;
printf("%c\n", data);

segfault at ffffffff81a000e0 ip
0000000000400535
sp 00007ffce4a80610 error 5 in reader
Adapted code

*(volatile char*) 0;
array[84 * 4096] = 0;  // unreachable
Flush+Reload over all pages of the array
Flush+Reload over all pages of the array

This also works on AMD and ARM!
• Combine the two things

```c
char data = *(char*)0xfffffffff81a000e0;
array[data * 4096] = 0;
```
• Combine the two things

```c
char data = *(char*)0xfffffffff81a000e0;
array[data * 4096] = 0;
```
Flush+Reload again...

... Meltdown actually works.
I SHIT YOU NOT

THERE WAS KERNEL MEMORY ALL OVER THE TERMINAL
CAN YOU ENHANCE THAT
meltdown@meltdown ~/ppm2 % taskset 1 ./imgdump 0x375a00000 14919 > output.flif
Reading from 0xffff880375a00000
Leaking Passwords from your Password Manager

Daniel Gruss — Graz University of Technology
AND IN OTHER NEWS...

WE'RE ALL DOOMED, SANDRA. HOW ABOUT THE WEATHER?
Not so fast...
• Kernel addresses in user space are a problem
Take the kernel addresses...

- Kernel addresses in user space are a problem
- Why don’t we take the kernel addresses...
...and remove them

- ...and remove them if not needed?
...and remove them

- ...and remove them if not needed?
- User accessible check in hardware is not reliable
Kernel Address Isolation to have Side channels Efficiently Removed
Kernel Address Isolation to have Side channels Efficiently Removed
Our patch
Adopted in Linux
Adopted in Windows
Adopted in OSX/iOS
now in every computer

Daniel Gruss — Graz University of Technology
• Our patch
• Adopted in Linux
KAISER Patches

- Our patch
- Adopted in Linux

- Adopted in Windows
KAISER Patches

- Our patch
- Adopted in Linux
- Adopted in Windows
- Adopted in OSX/iOS

now in every computer

Daniel Gruss — Graz University of Technology
Our patch
Adopted in Linux
Adopted in Windows
Adopted in OSX/iOS
→ now in every computer
Problem solved?
Foreshadow / Foreshadow-NG

CPU micro-architecture

1. PT walk? (vadrs)
   - fail: Page fault
   - ok: EPT walk?

2. EPT walk? (guest pads)
   - fail: Page fault
   - ok: SGX?

3. SGX? (host pads)
   - fail: EPCM fail
   - ok: Abort page

Allow
Boot from ROM...
early console in extract_kernel
input_data: 0x00000000001e0a276
input_len: 0x00000000003d48f8
output: 0x0000000000100000
output_len: 0x000000000011bc258
kernel_total_size: 0x000000000000dec00
booted via startup_32()
Physical KASLR using RDTSC...
Virtual KASLR using RDTSC...

Decompressing Linux... Parsing ELF... Performing relocations... done.
Booting the kernel.

L1 Terminal Fault

Run `reader <pfn> [<cache miss threshold>]` to leak hypervisor data from the L1
SO YOU ARE TELLING ME
YOU CAN DUMP THE MEMORY STORED IN L1?
WHAT IF I TOLD YOU
YOU CAN LEAK THE ENTIRE MEMORY
ZOMBIELOAD ATTACK

Processors leak sensitive data and keys while accessing them.

After Meltdown, Spectre, and Foreshadow we discovered more critical vulnerabilities in modern processors. The ZombieLoad attack allows stealing sensitive data and keys while the computer accesses them. While programs normally only see their own data, a malicious program can exploit the fill buffers to get hold of secrets currently processed by other running programs.

The attack does not only work on personal computers, but can also be exploited in the cloud.
ZombieLoad: Neue Sicherheitslücken in Intel-Prozessoren

Bei vielen bisherigen Core-i- und Xeon-Prozessoren kann Malware Daten laufender Prozesse belauschen, wenn sie auf demselben Kern läuft.

Von Christof Windeck
<table>
<thead>
<tr>
<th>Physical Page Number</th>
<th>Virtual Page Number</th>
<th>Page Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**ZombieLoad Comparison**

Daniel Gruss — Graz University of Technology
## ZombieLoad Comparison

<table>
<thead>
<tr>
<th>Page Number</th>
<th>Page Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Virtual</td>
</tr>
<tr>
<td>Meltdown</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>12</td>
</tr>
<tr>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>Foreshadow</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>12</td>
</tr>
<tr>
<td>47</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

Daniel Gruss — Graz University of Technology
<table>
<thead>
<tr>
<th>ZombieLoad Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Page Number</strong></td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>Meltdown</td>
</tr>
<tr>
<td>Physical</td>
</tr>
<tr>
<td>Virtual</td>
</tr>
<tr>
<td>Foreshadow</td>
</tr>
<tr>
<td>Physical</td>
</tr>
<tr>
<td>Virtual</td>
</tr>
<tr>
<td>Foreshadow</td>
</tr>
<tr>
<td>Physical</td>
</tr>
<tr>
<td>Virtual</td>
</tr>
</tbody>
</table>

Daniel Gruss — Graz University of Technology
Privacy for anyone anywhere

Tails is a live operating system that you can start on almost any computer from a USB stick or a DVD.

It aims at preserving your privacy and anonymity, and helps you to:
- use the Internet anonymously and circumvent censorship.
- all connections to the Internet are forced to go through the Tor network.
- leave no trace on the computer you are using unless you ask it explicitly.
- use state-of-the-art cryptographic tools to encrypt your files, emails and instant messaging.

Learn more about Tails

Install Tails 3.13.2
2019-09-09

About
Getting started...
Documentation
Help & Support
Contribute

News
Security
Zombieload: Grazer Forscher entdeckten gravierende Lücken bei Intel-Prozessoren

14. Mai 2019, 19:00

Prozessoren der Jahre 2012 bis 2018 betroffen – Neue Updates werden notwendig


Patches gegen Meltdown und Spectre schützen nicht

"ZombieLoad" und "Store-to-Leak Forwarding" haben die
A table for 6 please
Speculative Cooking
A table for 6 please
index = 0;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
index = 0;

char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    0
index = 0;
char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    Speculate
    Prediction
    0
index = 0;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
index = 1;

char* data = "textKEY";

if (index < 4)
then
    LUT[data[index] * 4096]
else
    0

Prediction
index = 1;

char* data = "textKEY";

if (index < 4)
  LUT[data[index] * 4096]
else
  0
index = 1;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

Prediction

0
index = 1;

char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    0
index = 2;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
index = 2;

char* data = "textKEY";

if (index < 4)
    Prediction
    then
    LUT[data[index] * 4096]
    else
    0
index = 2;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
Spectre-PHT (aka Spectre Variant 1)

index = 2;

char* data = "textKEY";

if (index < 4)

then

LUT[data[index] * 4096]

else

0

Daniel Gruss — Graz University of Technology
index = 3;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
index = 3;

char* data = "textKEY";

if (index < 4)

then

LUT[data[index] * 4096]

else

0
index = 3;

char* data = "textKEY";

if (index < 4)
    Speculate
    LUT[data[index] * 4096]
else
    Prediction
    0
index = 3;

char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    0
index = 4;

char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    0
index = 4;

char* data = "textKEY";

if (index < 4)
  then
    LUT[data[index] * 4096]
  else
    0

Prediction
Spectre-PHT (aka Spectre Variant 1)

Speculate

index = 4;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
Spectre-PHT (aka Spectre Variant 1)

index = 4;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

Prediction

Execute

0
index = 5;

char* data = "textKEY";

if (index < 4)
    then
        LUT[data[index] * 4096]
    else
        Prediction
            0
Spectre-PHT (aka Spectre Variant 1)

\[
\text{index} = 5;
\]

\[
\text{char* data} = "textKEY";
\]

\[
\text{if} \ (\text{index} < 4)
\]

\[
\text{then} \quad \text{LUT}[\text{data[index]} \times 4096]
\]

\[
\text{else} \quad 0
\]

Daniel Gruss — Graz University of Technology
`index = 5;`

`char* data = "textKEY";`

```
if (index < 4)
    LUT[data[index] * 4096]
else
    0
```
index = 5;

char* data = "textKEY";

if (index < 4)
    LUT[data[index] * 4096]
else
    0
index = 6;

char* data = "textKEY";

if (index < 4)
then
LUT[data[index] * 4096]
else
Prediction

index = 6;

char* data = "textKEY";

if (index < 4)
then
LUT[data[index] * 4096]
else
Prediction

0
index = 6;

char* data = "textKEY";

if (index < 4) {
    LUT[data[index] * 4096]
} else {
    0
}
index = 6;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]

else

0
NetSpectre:

- completely remote - we just send network requests
- leak around 15\text{60 bit per second}
- no attacker code on target system
NetSpectre:

- completely remote - we just send network requests
NetSpectre:

- completely remote - we just send network requests
- leak around 15–60 bit per second
NetSpectre:

- completely remote - we just send network requests
- leak around 15–60 bit per second
- no attacker code on target system
Classification Tree

Transient cause?

Spectre-type microarchitectural buffer

Meltdown-type fault type

Spectre-PHT
Spectre-BTB
Spectre-RSB
Spectre-STL [32]

Cross-address-space
Same-address-space

Cross-address-space
Same-address-space

Meltdown-NM [86]
Meltdown-AC
Meltdown-DE
Meltdown-PF
Meltdown-UD
Meltdown-SS
Meltdown-BR
Meltdown-GP [10, 41]

Meltdown-AC
Meltdown-DE
Meltdown-PF
Meltdown-UD
Meltdown-SS
Meltdown-BR
Meltdown-GP [10, 41]

Meltdown-US [61]
Meltdown-P [93, 96]
Meltdown-RW [52]
Meltdown-PK
Meltdown-XD
Meltdown-SM
Meltdown-MPX [44]
Meltdown-BND

in-place (IP) vs., out-of-place (OP)
mistreating strategy

PHT-CA-IP ★
PHT-CA-OP ★
PHT-SA-IP [54, 52]
PHT-SA-OP ★
BTB-CA-IP [54, 18]
BTB-CA-OP [54]
BTB-SA-IP ★
BTB-SA-OP [18]
RSB-CA-IP [64, 56]
RSB-CA-OP [56]
RSB-SA-IP [64]
RSB-SA-OP [64, 56]
Computer Architecture Today

Informing the broad computing community about current activities, advances and future directions in computer architecture.

Let's Keep it to Ourselves: Don't Disclose Vulnerabilities

by Gus Uht on Jan 31, 2019 | Tags: Opinion, Security
Ignorance is bliss?
Where in Computer Science are we?

Computer science:
- computer engineering
- philosophy
- artificial science

Daniel Gruss — Graz University of Technology
Where in Computer Science are we?
Where in Computer Science are we?

Computer science:
Computer science:
- computer engineering
Computer science:
• computer engineering
• philosophy
Where in Computer Science are we?

Computer science:
- computer engineering
- philosophy
- artificial science
The Sciences of the Artificial” (Herbert A. Simon, 1969)

- Natural sciences: studying something natural
- Artificial science: studying something artificial (something man-made) as if it was something natural

A consequence of complexity
Microarchitectural Attacks ⊆ Artificial Science

Artificial Science

Natural sciences: studying something natural
Artificial science: studying something artificial (something man-made) as if it was something natural

A consequence of complexity

Daniel Gruss — Graz University of Technology
“The Sciences of the Artificial” (Herbert A. Simon, 1969)
“The Sciences of the Artificial” (Herbert A. Simon, 1969)

- Natural sciences: studying something natural
“The Sciences of the Artificial” (Herbert A. Simon, 1969)

- Natural sciences: studying something natural
- Artificial science: studying something artificial (something man-made) as if it was something natural
"The Sciences of the Artificial" (Herbert A. Simon, 1969)

- Natural sciences: studying something natural
- Artificial science: studying something artificial (something man-made) as if it was something natural

→ A consequence of complexity
What do we learn from it?

We have ignored microarchitectural attacks for many many years:

• attacks on crypto should be fixed
• attacks on ASLR is broken anyway
• attacks on SGX and TrustZone not part of the threat model
• Rowhammer attacks only affects cheap sub-standard modules

for years we solely optimized for performance
What do we learn from it?

We have ignored microarchitectural attacks for many many years:

- attacks on crypto
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone
What do we learn from it?

We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
- Rowhammer attacks
We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
- Rowhammer attacks → “only affects cheap sub-standard modules”
What do we learn from it?

We have ignored microarchitectural attacks for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”
- attacks on SGX and TrustZone → “not part of the threat model”
- Rowhammer attacks → “only affects cheap sub-standard modules”

→ for years we solely optimized for performance
Conclusions

- The complexity of the systems we built is too large to fully understand them
- We need to study man-made systems like nature to find flaws
- We need good and adjustable trade-offs between security and performance, efficiency, and complexity
- Learn from nature, Learn to cope with diseases
Meltdown, Spectre, ZombieLoad

Daniel Gruss
May 16, 2019
Graz University of Technology