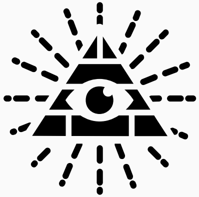
Beyond Belief: Spectre and Meltdown

with Daniel Gruss, Moritz Lipp, Michael Schwarz

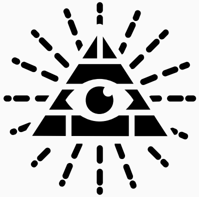
Timeline of a Vulnerability



Is this all a conspiracy?

- Vulnerability existed for many years

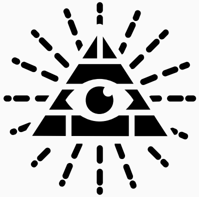
Timeline of a Vulnerability



Is this all a conspiracy?

- Vulnerability existed for many years
- No one discovered it before

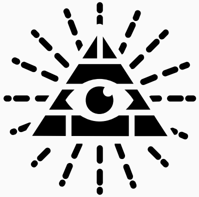
Timeline of a Vulnerability



Is this all a conspiracy?

- Vulnerability existed for many years
- No one discovered it before
- Suddenly, 4 independent teams discover it within 6 months

Timeline of a Vulnerability



Is this all a conspiracy?

- Vulnerability existed for many years
- No one discovered it before
- Suddenly, 4 independent teams discover it within 6 months
- Let’s create an evidence board

MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz

FOGH
CyberWTF

MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz

Academic?
Hacker?

Academic?
Hacker?

black hat

Academic?
Hacker?

black hat

ACM CCS 2016

Academic?
Hacker?

black hat

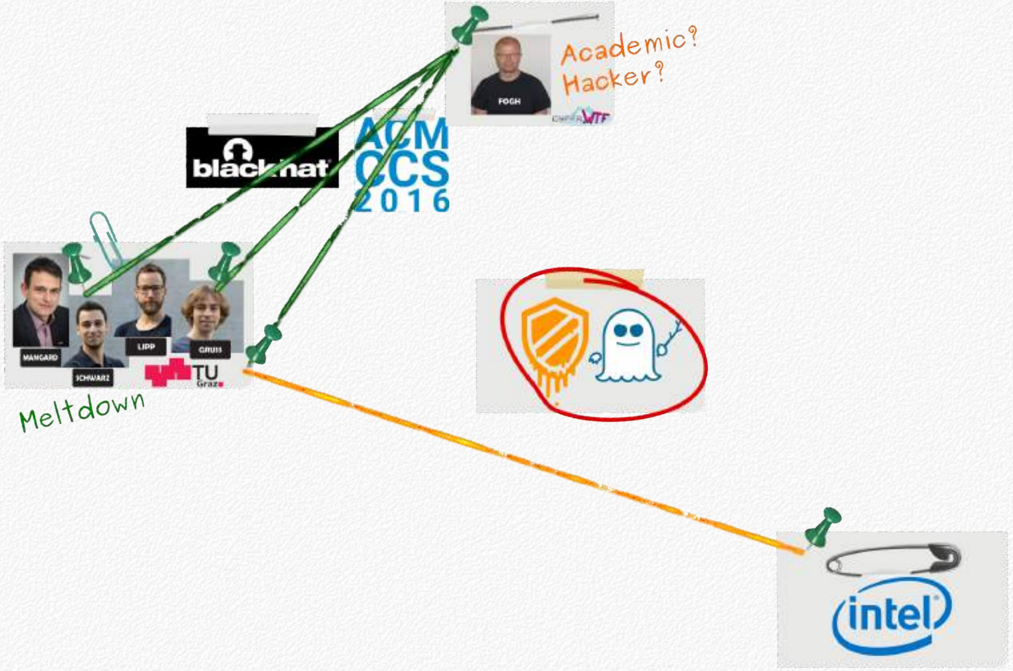
ACM
CCS
2016

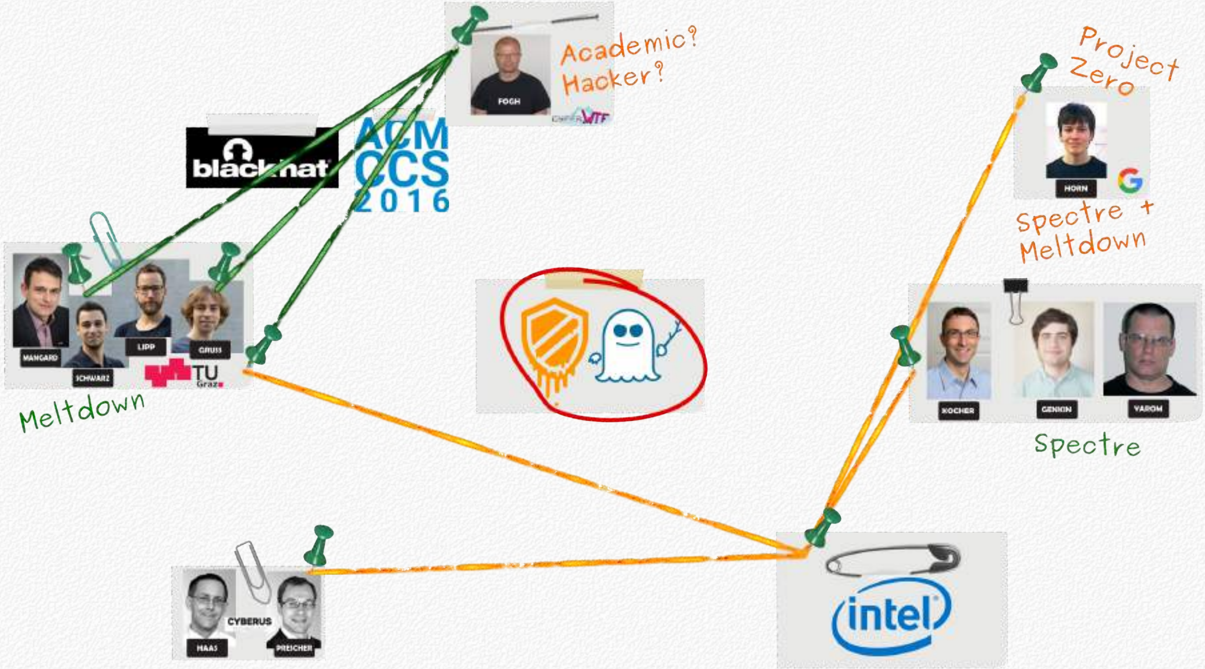
Academic?
Hacker?

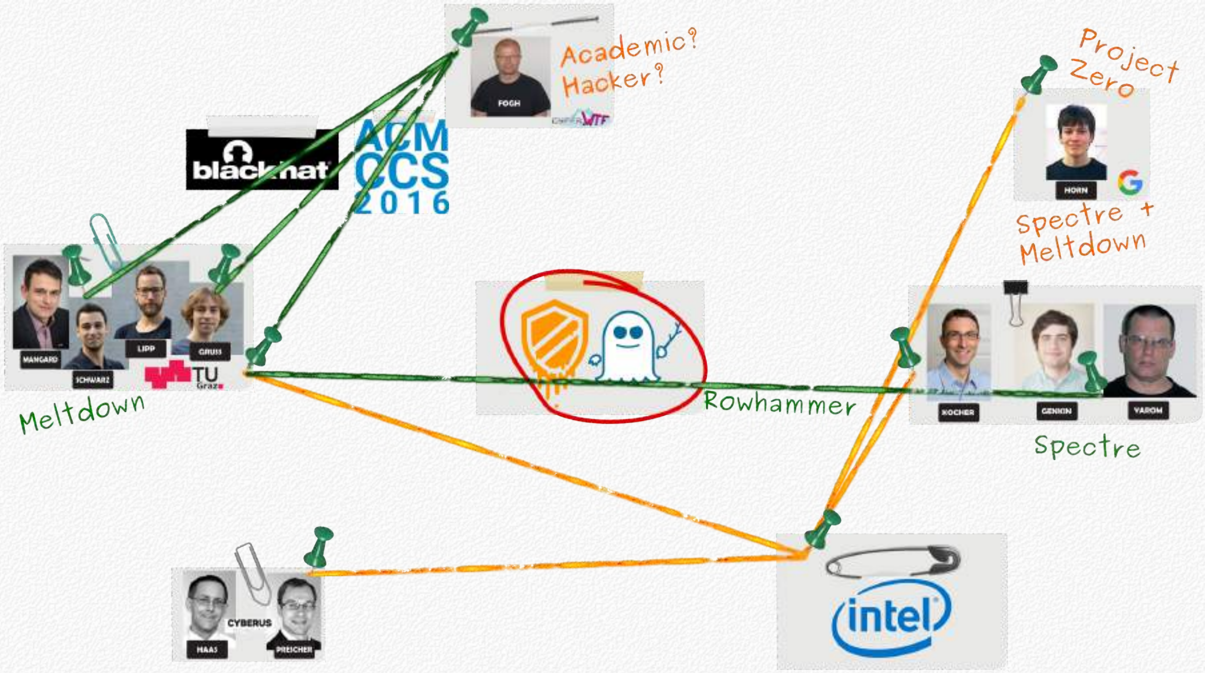
blackhat

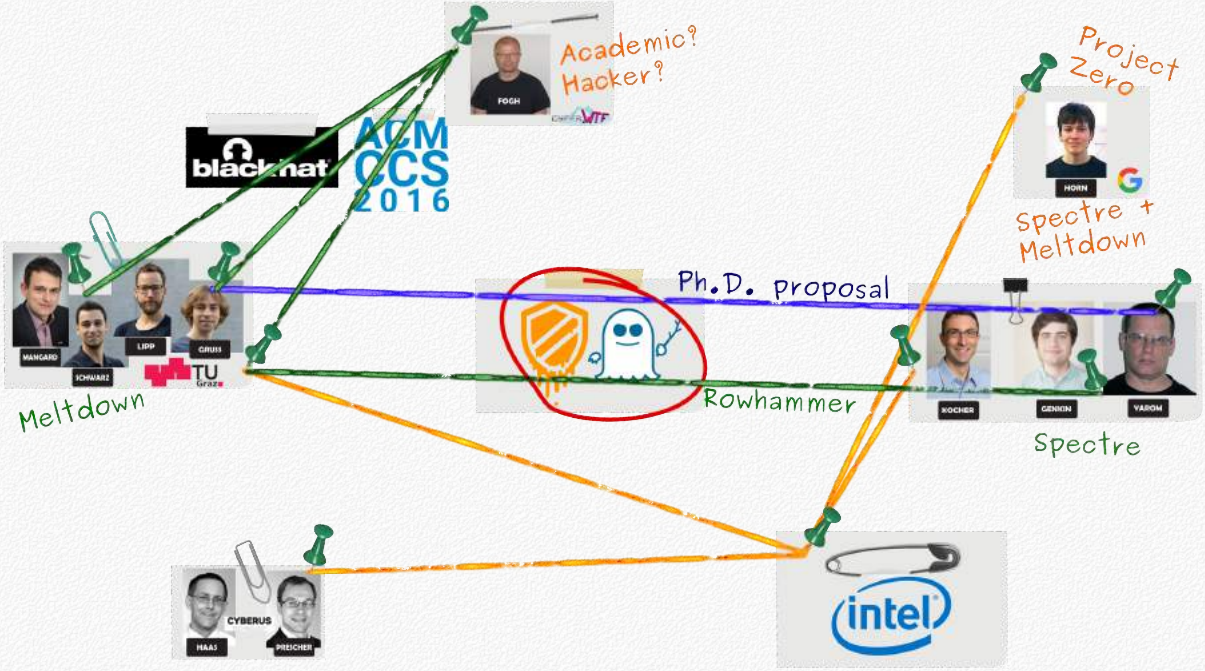
ACM
CCS
2016

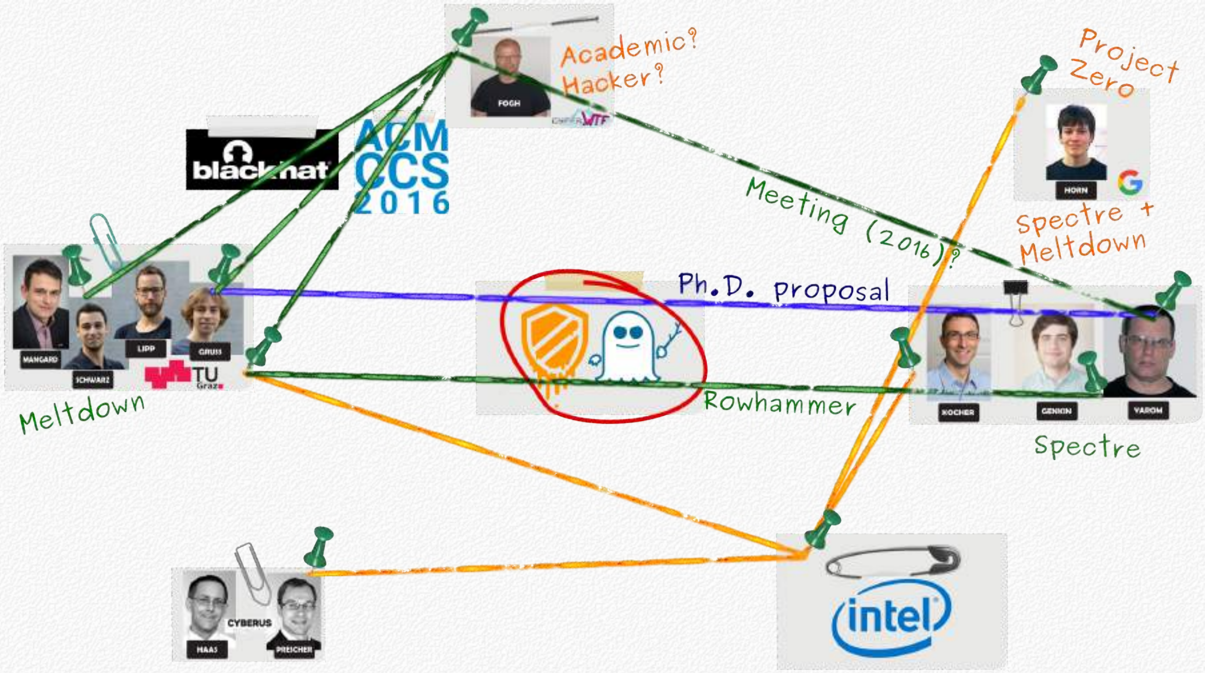
Meltdown


FOGH
Academic?
Hacker?
blackhat
ACM CCS 2016
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
Meltdown
intel

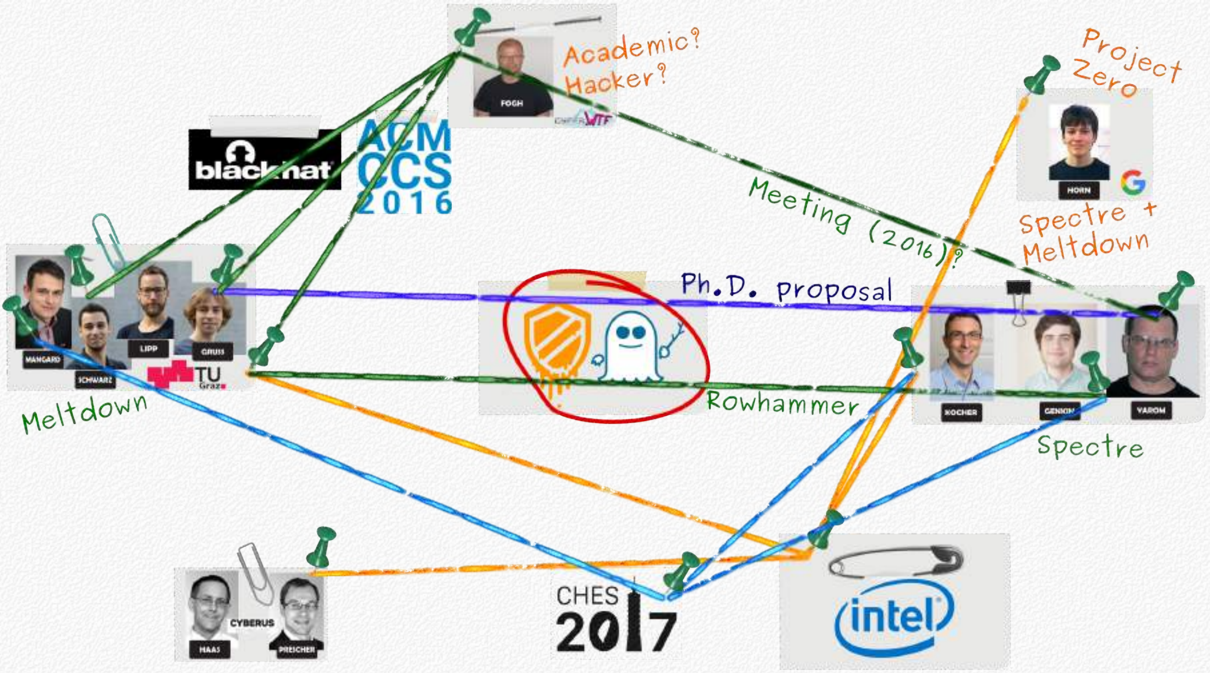

Academic?
Hacker?
FOGH
Project Zero
HORN
Spectre + Meltdown
black hat
ACM CCS 2016
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
Meltdown
KOCHER
GENKIN
YAROM
Spectre
HAAS
CYBERUS
PRESCHER
intel

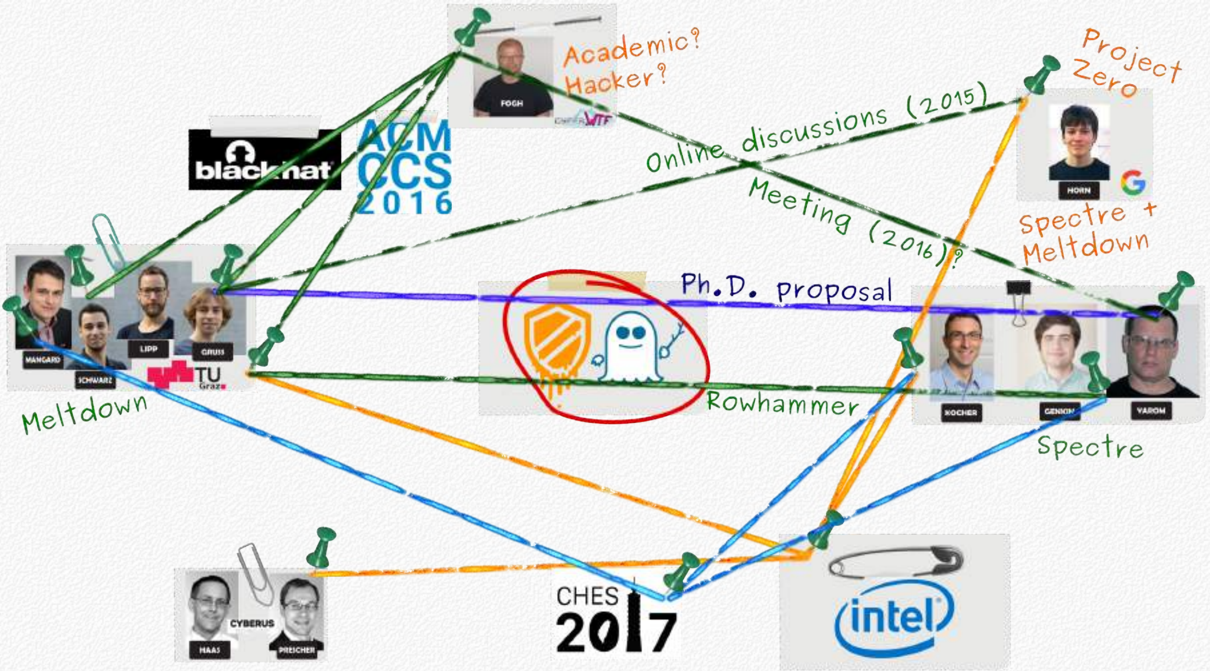

FOGH
Academic?
Hacker?
Project Zero
HORN
Spectre + Meltdown
blackhat
ACM CCS 2016
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
Meltdown
Rowhammer
KOCHER
GENKIN
YAROM
Spectre
HAAS
CYBERUS
PRESCHER
intel


FOGH
Academic?
Hacker?
Project
Zero
HORN
Spectre +
Meltdown
black hat
ACM
CCS
2016
MANGARD
SCHWARZ
LIPP
GRUSS
TU
Graz
Meltdown
Ph.D. proposal
Rowhammer
KOCHER
GENKIN
YAROM
Spectre
CYBERUS
HAAS
PRESCHER
intel



Academic?
Hacker?
FOGH
CYBERWTF
Project Zero
HORN
Spectre + Meltdown
blackhat
ACM CCS 2016
Meeting (2016)?
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
Meltdown
Ph.D. proposal
Rowhammer
KOCHER
GENKIN
YAROM
Spectre
HAAS
CYBERUS
PRESCHER
intel


Academic?
Hacker?
FOGH
Project Zero
HORN
Spectre + Meltdown
blackhat
ACM CCS 2016
Meeting (2016)?
Ph.D. proposal
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
KOCHER
GENKIN
YAROM
Rowhammer
Meltdown
Spectre
HAAS
CYBERUS
PRESCHER
CHES 2017
intel


FOGH
CyberWTF
Academic?
Hacker?
Project
Zero
HORN
Online discussions (2015)
Meeting (2016)?
Spectre +
Meltdown
blackhat
ACM
CCS
2016
MANGARD
SCHWARZ
LIPP
GRUSS
TU Graz
Meltdown
Ph.D. proposal
Rowhammer
KOCHER
GENKIN
YAROM
Spectre
HAAS
CYBERUS
PRESCHER
CHES
2017
intel

Timeline of a Vulnerability

Not a conspiracy

- Tools to detect the bug only invented in 2014

Timeline of a Vulnerability

Not a conspiracy

- Tools to detect the bug only invented in 2014
- No broad interest in microarchitectural attacks before

Timeline of a Vulnerability

Not a conspiracy

- Tools to detect the bug only invented in 2014
- No broad interest in microarchitectural attacks before
- Discovering teams quite knowledgeable in this area

Timeline of a Vulnerability

Not a conspiracy

- Tools to detect the bug only invented in 2014
- No broad interest in microarchitectural attacks before
- Discovering teams quite knowledgeable in this area
- The bug was “ripe” $\Rightarrow$ a consequence of research in this area

Timeline of a Vulnerability

Not a conspiracy

- Tools to detect the bug only invented in 2014
- No broad interest in microarchitectural attacks before
- Discovering teams quite knowledgeable in this area
- The bug was "ripe" $\Rightarrow$ a consequence of research in this area

$\rightarrow$ bug collision nearly inevitable

The Fallout

You realize it is something big when...

The Fallout

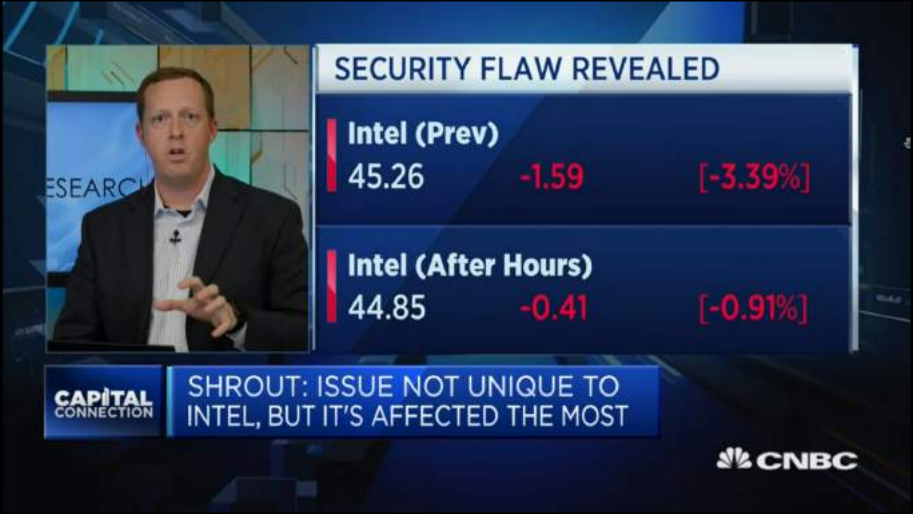
You realize it is something big when...

- it is in the news, all over the world

FOX BUSINESS WASHINGTON, D.C.
WASHINGTON, D.C.
WINTER STORM
NEWS ALERT
INTEL REVEALS DESIGN FLAW THAT COULD ALLOW HACKERS TO ACCESS DATA
FOX BUSINESS NETWORK
@FOXBUSINESS

FIREEYE CYBER THREAT MAP
DEVELOPING STORY
COMPUTER CHIP FLAWS IMPACT BILLIONS OF DEVICES
LIVE
CNN
DAX ▲164.69
NEWS STREAM

GLOBAL
COMPUTER CHIP SCARE
The bugs are known as 'Spectre' and 'Meltdown'
BBC WORLD NEWS
£:HK$ 10.58
EURO:£ 0.891



SECURITY FLAW REVEALED

Intel (Prev)
45.26 -1.59 [-3.39%]

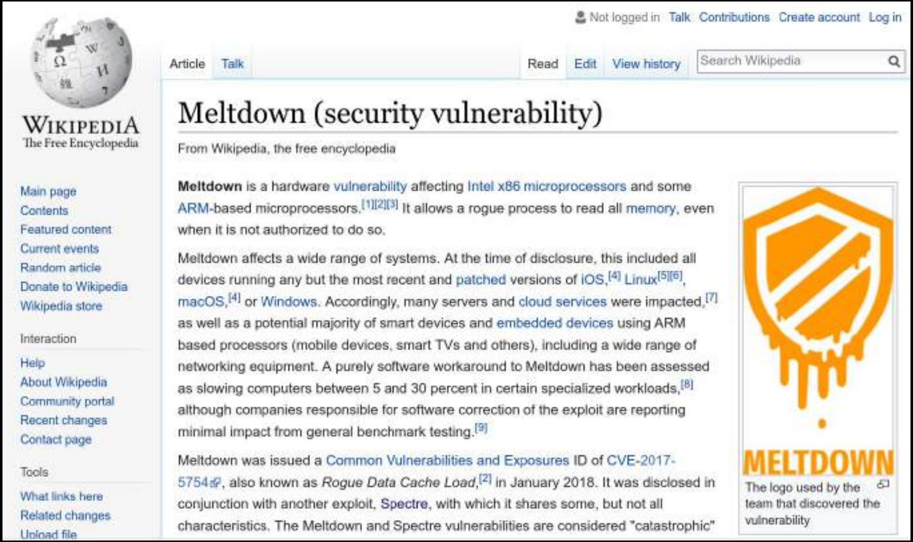
Intel (After Hours)
44.85 -0.41 [-0.91%]

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

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



Meltdown (security vulnerability)

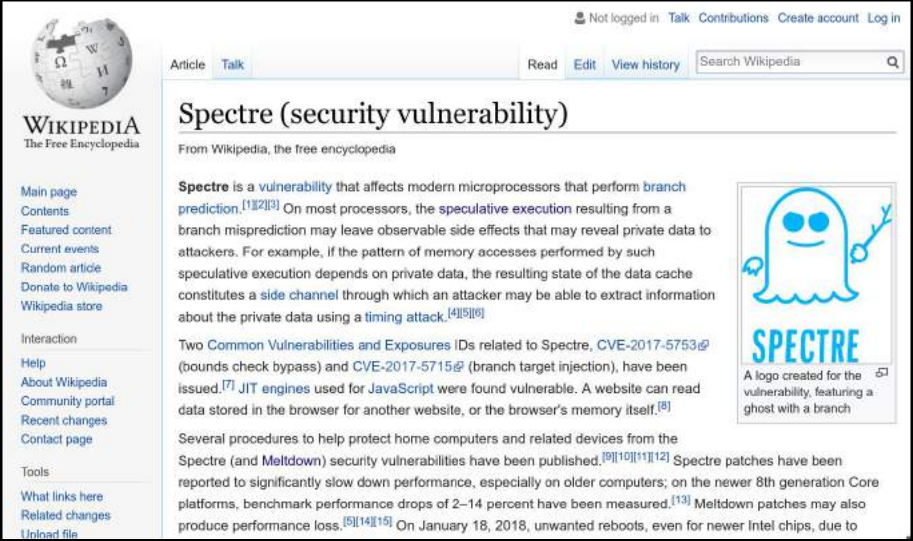
From Wikipedia, the free encyclopedia

Meltdown is a hardware vulnerability affecting Intel x86 microprocessors and some ARM-based microprocessors.[1][2][3] 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,[4] Linux[5][6], macOS,[4] or Windows. Accordingly, many servers and cloud services were impacted,[7] 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,[8] although companies responsible for software correction of the exploit are reporting minimal impact from general benchmark testing.[9]

Meltdown was issued a Common Vulnerabilities and Exposures ID of CVE-2017-5754, also known as *Rogue Data Cache Load*,[2] 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"

The logo used by the team that discovered the vulnerability



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file

Spectre (security vulnerability)

From Wikipedia, the free encyclopedia

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 (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]

SPECTRE

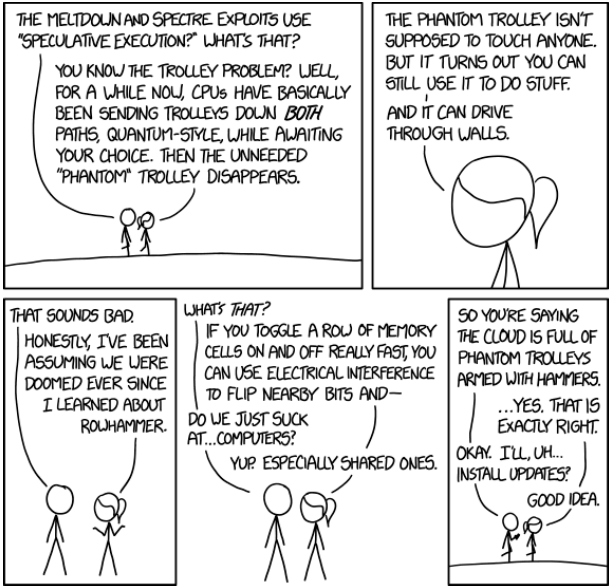
A logo created for the vulnerability, featuring a ghost with a branch

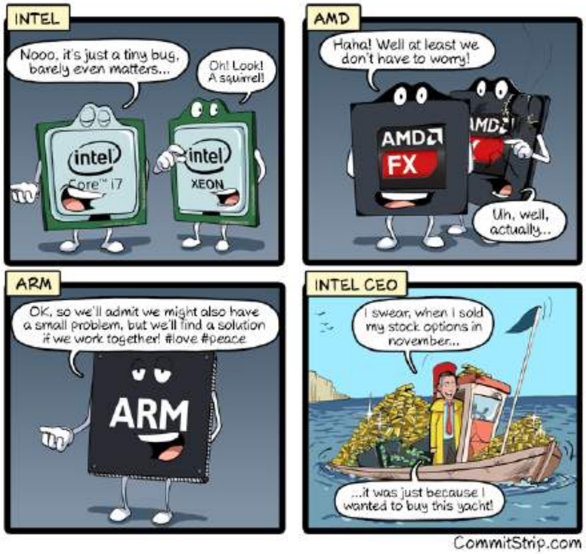
Several procedures to help protect home computers and related devices from the Spectre (and Meltdown) security vulnerabilities have been published.[9][10][11][12] Spectre patches have been reported to significantly slow down performance, especially on older computers; on the newer 8th generation Core 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

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

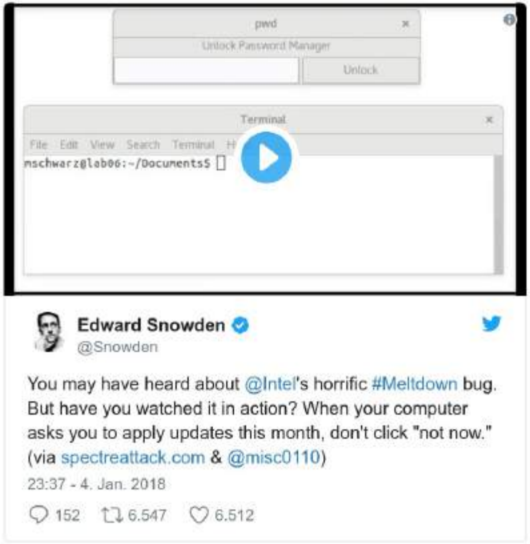

THE MELTDOWN AND SPECTRE EXPLOITS USE "SPECULATIVE EXECUTION?" WHAT'S THAT?
YOU KNOW THE TROLLEY PROBLEM? WELL, FOR A WHILE NOW, CPUs HAVE BASICALLY BEEN SENDING TROLLEYS DOWN *BOTH* PATHS, QUANTUM-STYLE, WHILE AWAITING YOUR CHOICE. THEN THE UNNEEDED "PHANTOM" TROLLEY DISAPPEARS.
THE PHANTOM TROLLEY ISN'T SUPPOSED TO TOUCH ANYONE. BUT IT TURNS OUT YOU CAN STILL USE IT TO DO STUFF.
AND IT CAN DRIVE THROUGH WALLS.
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 WE 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. I'LL, UH... INSTALL UPDATES?
GOOD IDEA.


INTEL
Nooo, it's just a tiny bug, barely even matters...
Oh! Look! A squirrel!
AMD
Haha! Well at least we don't have to worry!
Uh, well, actually...
ARM
OK, so we'll admit we might also have a small problem, but we'll find a solution if we work together! #love #peace
INTEL CEO
I swear, when I sold my stock options in november...
...it was just because I wanted to buy this yacht!
CommitStrip.com

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 follower after Snowden mentioned you



```
mschwarz@lab06:~/Documents$
```

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)

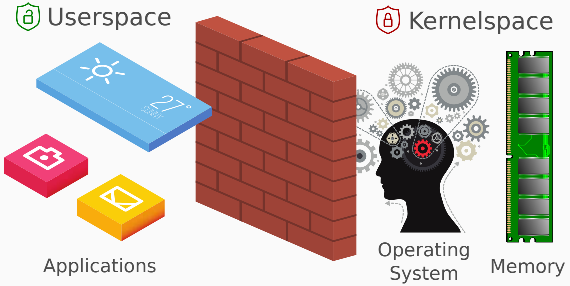
23:37 - 4. Jan. 2018

152 6.547 6.512

The Wall

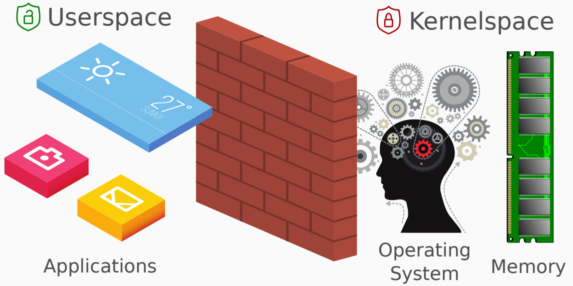
The Core of Meltdown/Spectre

- Kernel is isolated from user space



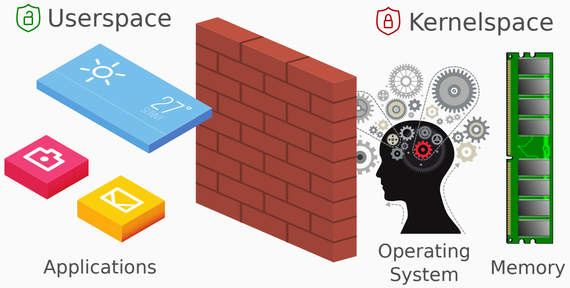
The Core of Meltdown/Spectre

- Kernel is isolated from user space
- This isolation is a combination of hardware and software



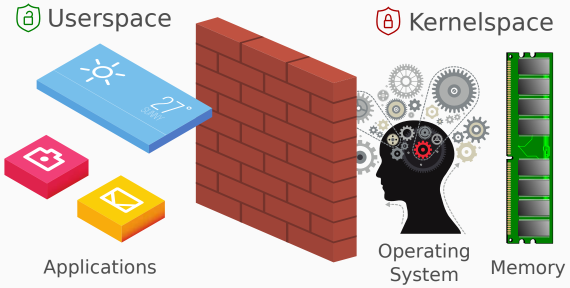
The Core of Meltdown/Spectre

- Kernel is isolated from user space
- This isolation is a combination of hardware and software
- User applications cannot access anything from the kernel



The Core of Meltdown/Spectre

- Kernel is isolated from user space
- This isolation is a combination of hardware and software
- User applications cannot access anything from the kernel
- There is only a well-defined interface → syscalls



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.

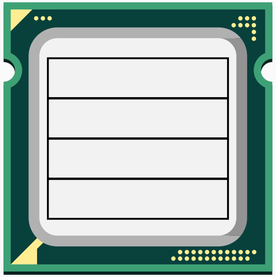
ONLY TODAY INSTEAD OF ~~$1,300~~

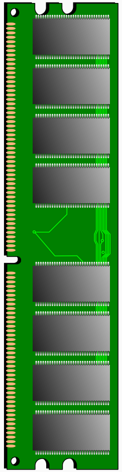
$1,299

ORDER VIA PHONE: +555 12345

CPU Cache

```
printf("%d", i);
printf("%d", i);
```

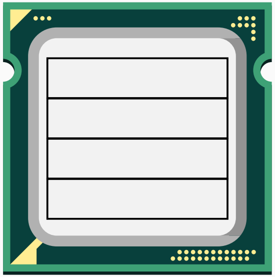


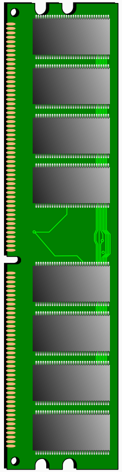


CPU Cache

```
printf("%d", i);
printf("%d", i);
```

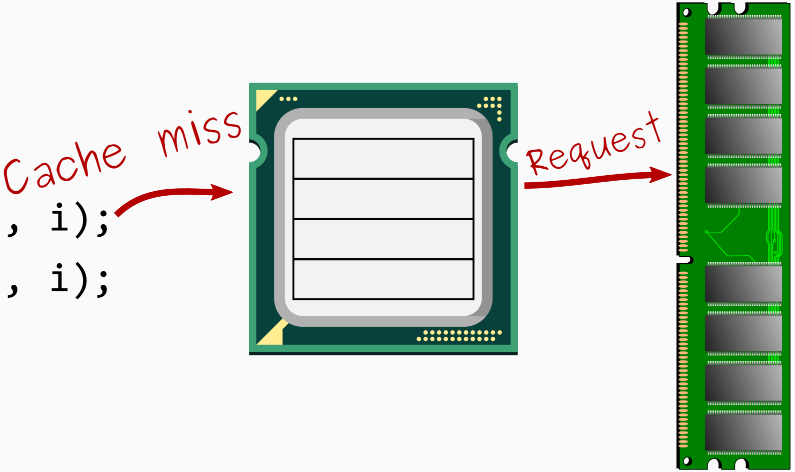
Cache miss





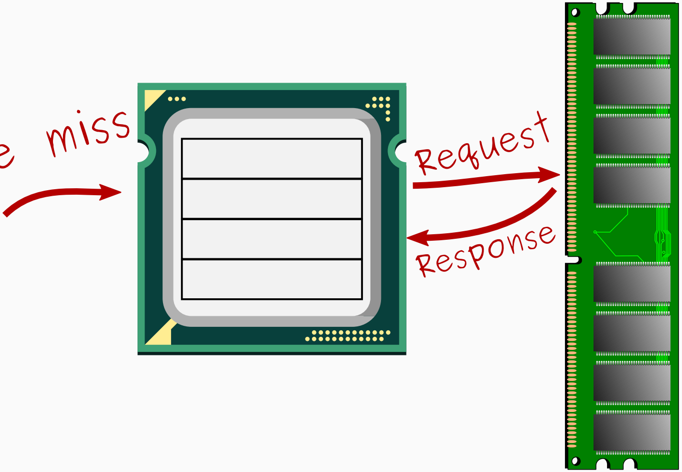
CPU Cache

```
printf("%d", i);
printf("%d", i);
```



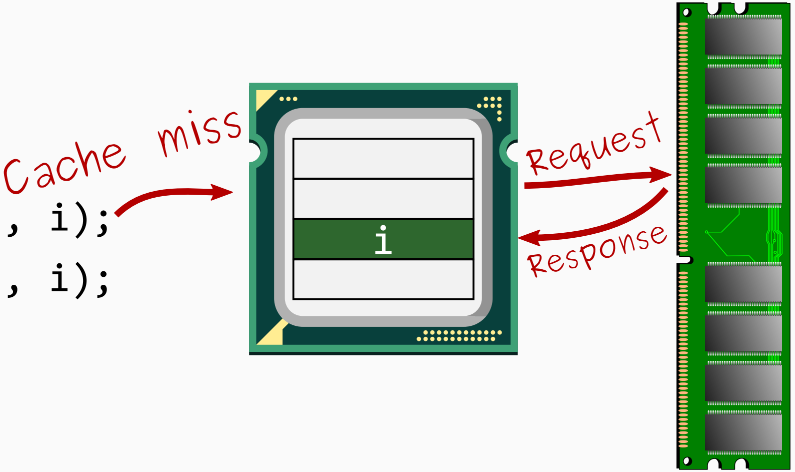
CPU Cache

```
printf("%d", i);
printf("%d", i);
```



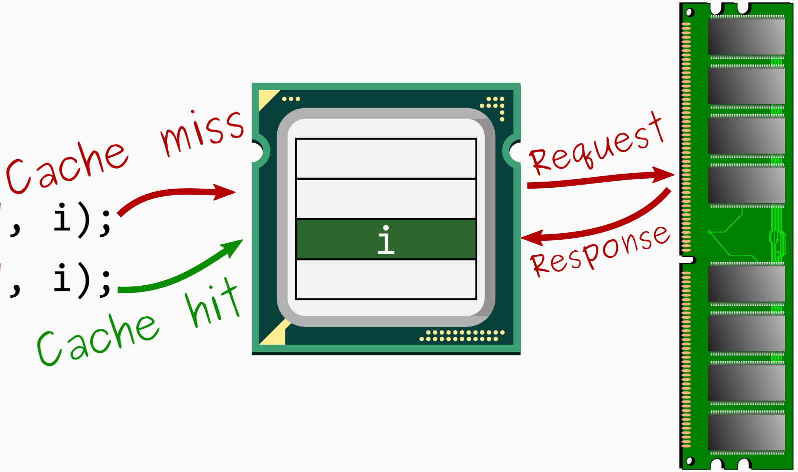
CPU Cache

```
printf("%d", i);
printf("%d", i);
```

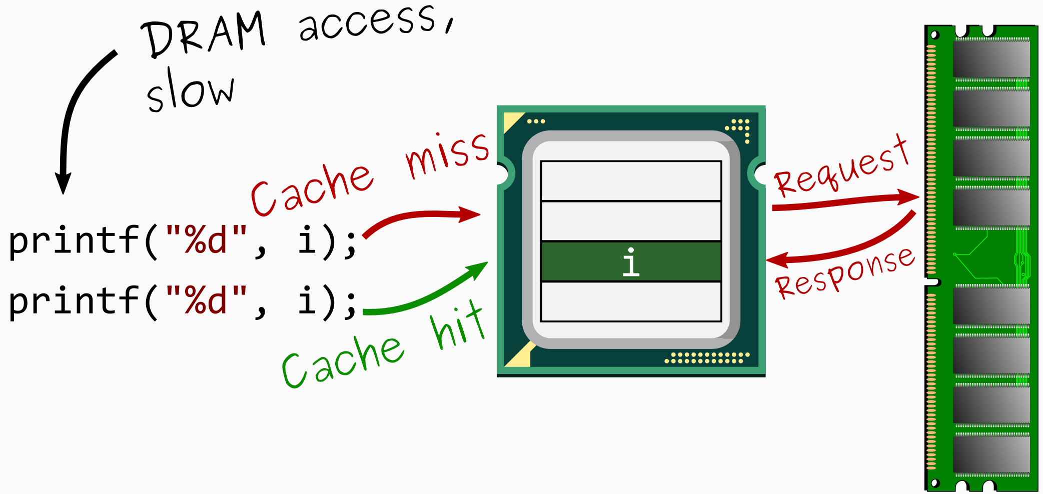


CPU Cache

```
printf("%d", i);
printf("%d", i);
```

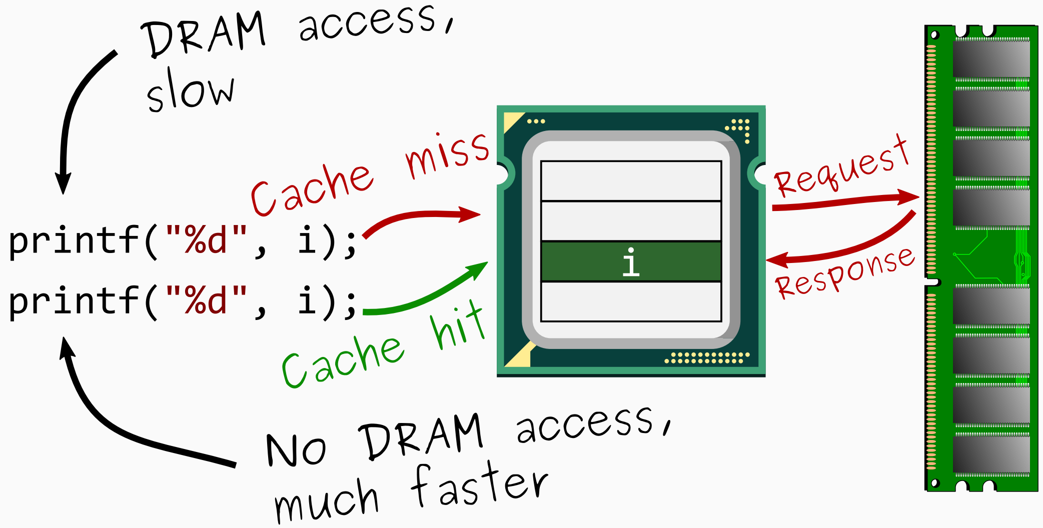


CPU Cache



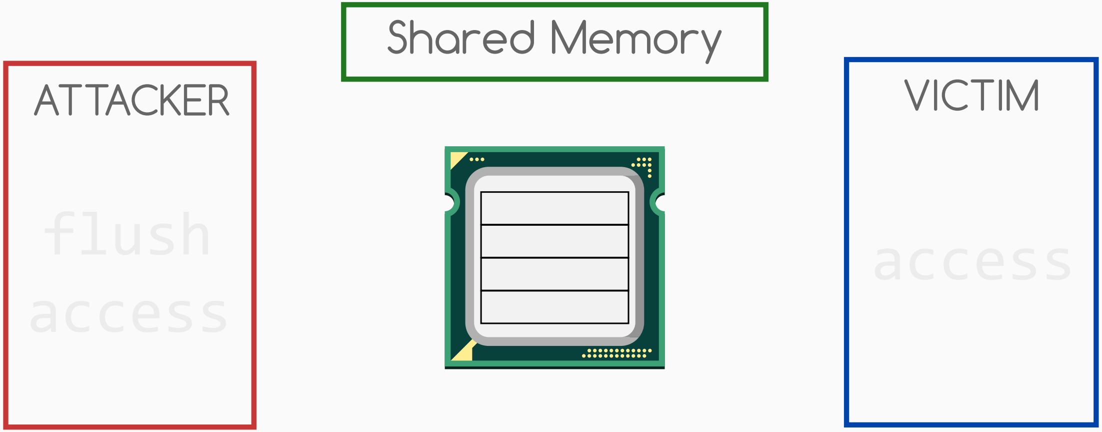
```
printf("%d", i);
printf("%d", i);
```

CPU Cache

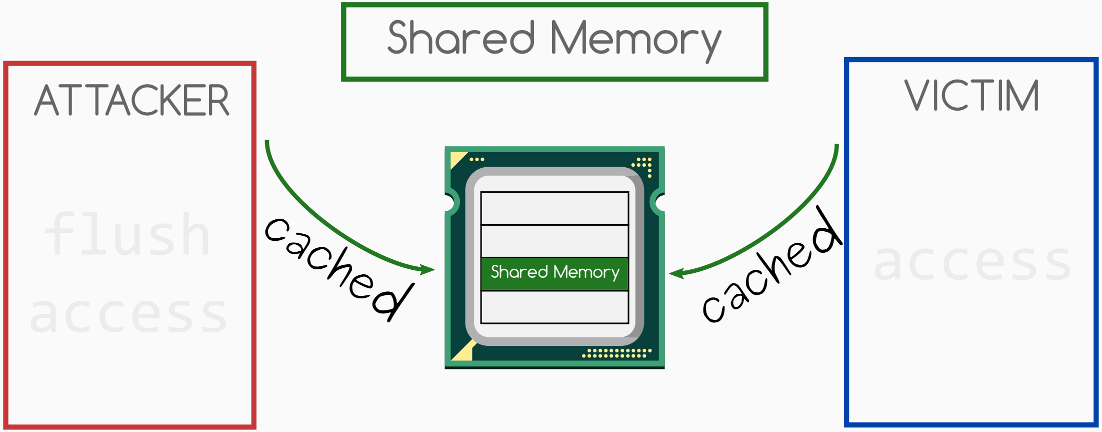


```
printf("%d", i);
printf("%d", i);
```

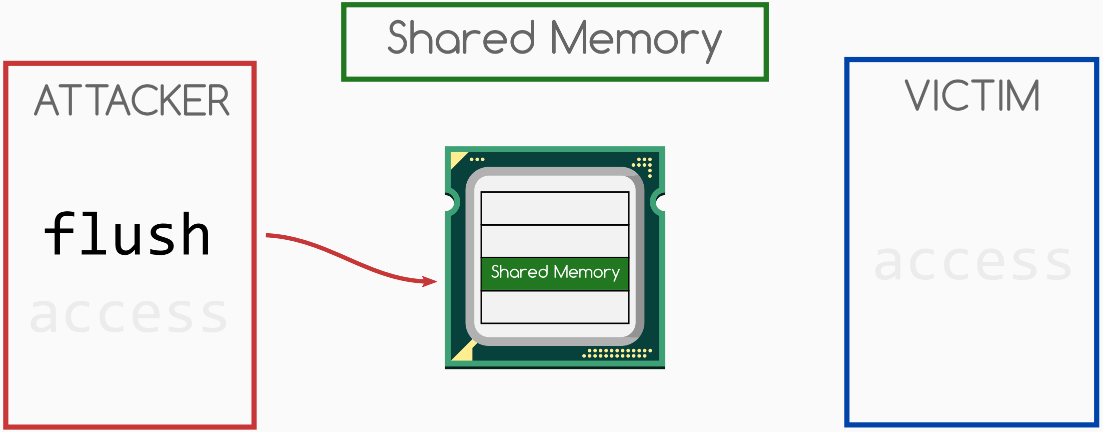
Flush+Reload



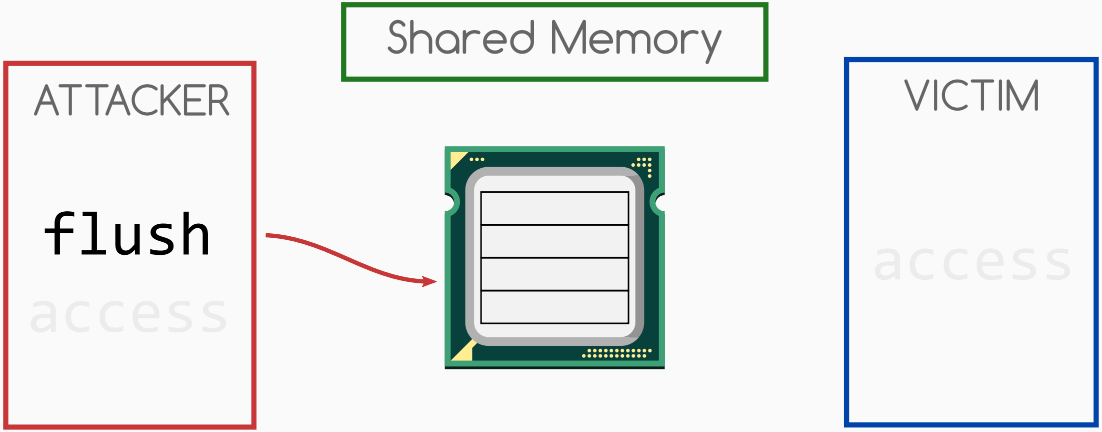
Flush+Reload



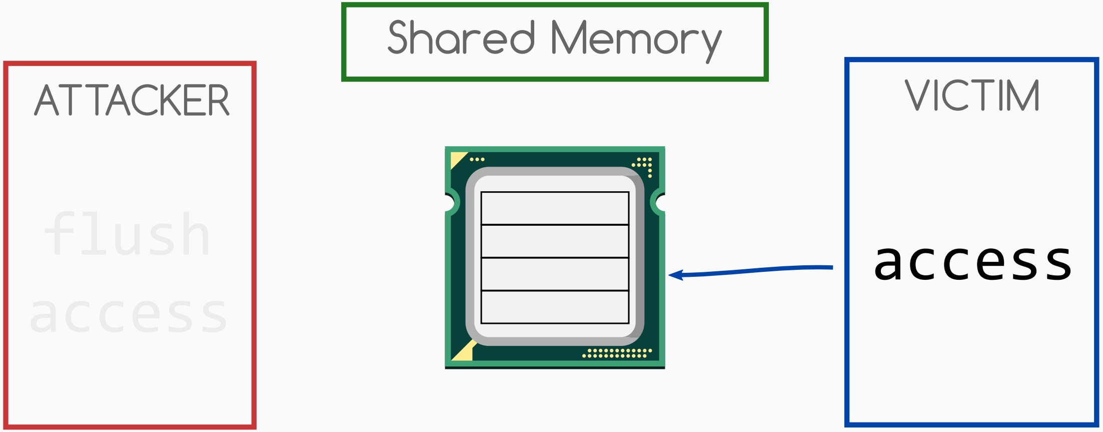
Flush+Reload



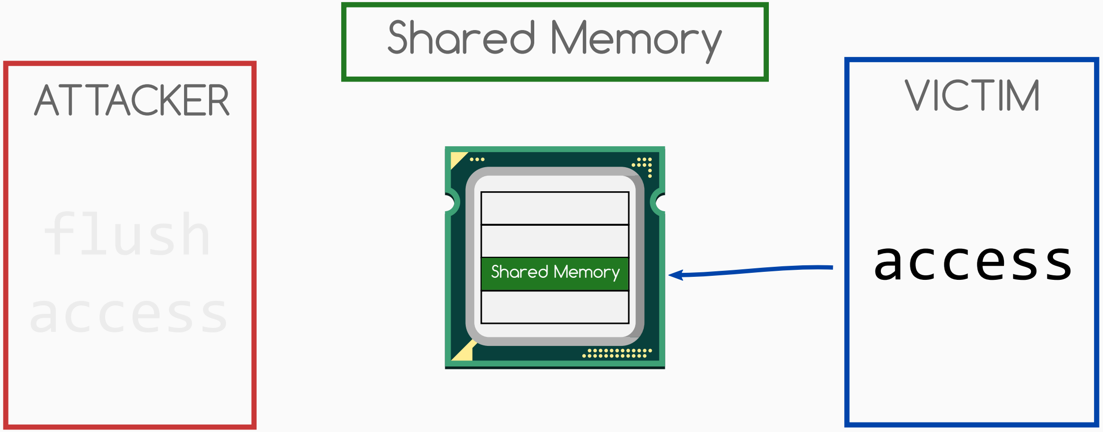
Flush+Reload



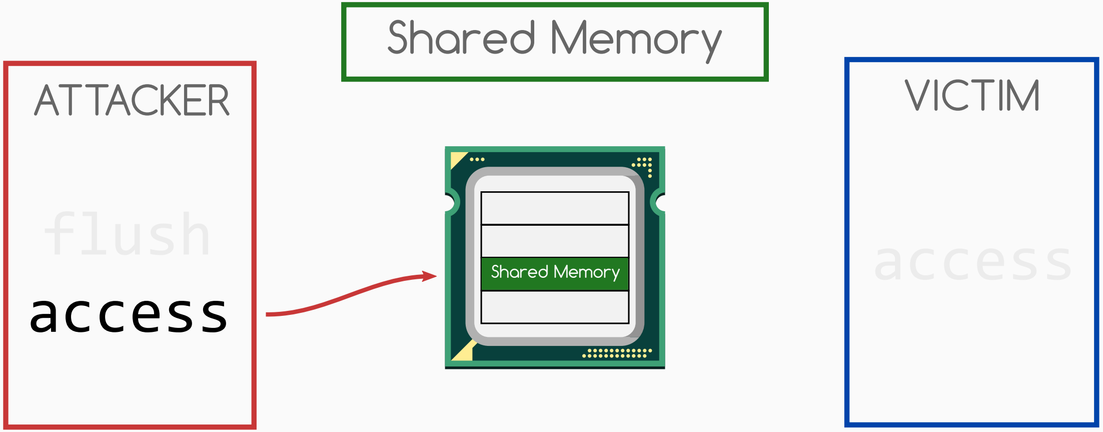
Flush+Reload



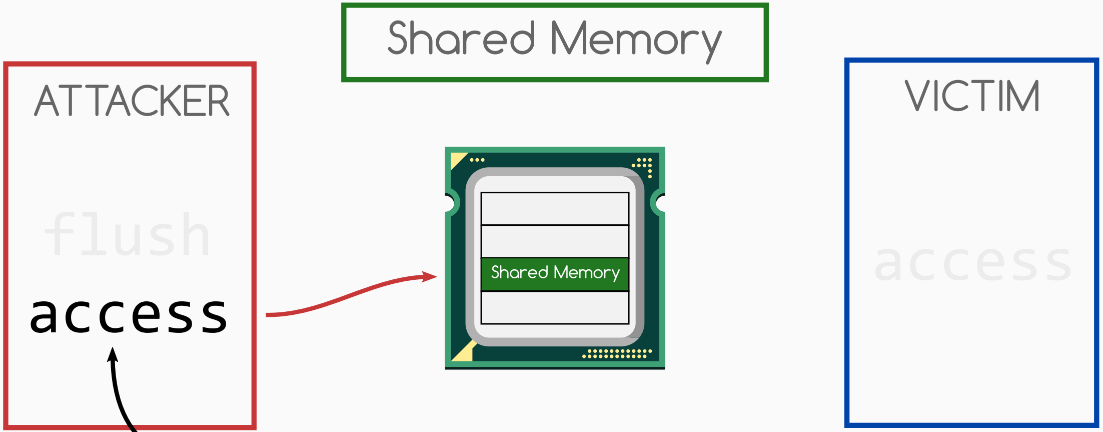
Flush+Reload



Flush+Reload



Flush+Reload



fast if victim accessed data,
slow otherwise

Back to Work

5. Cook everything until vegetables are soft

6. Add spices to taste [illegible] for 10 minutes

7. Serve with cooked and peeled potatoes

Wait for an hour

Wait for an hour

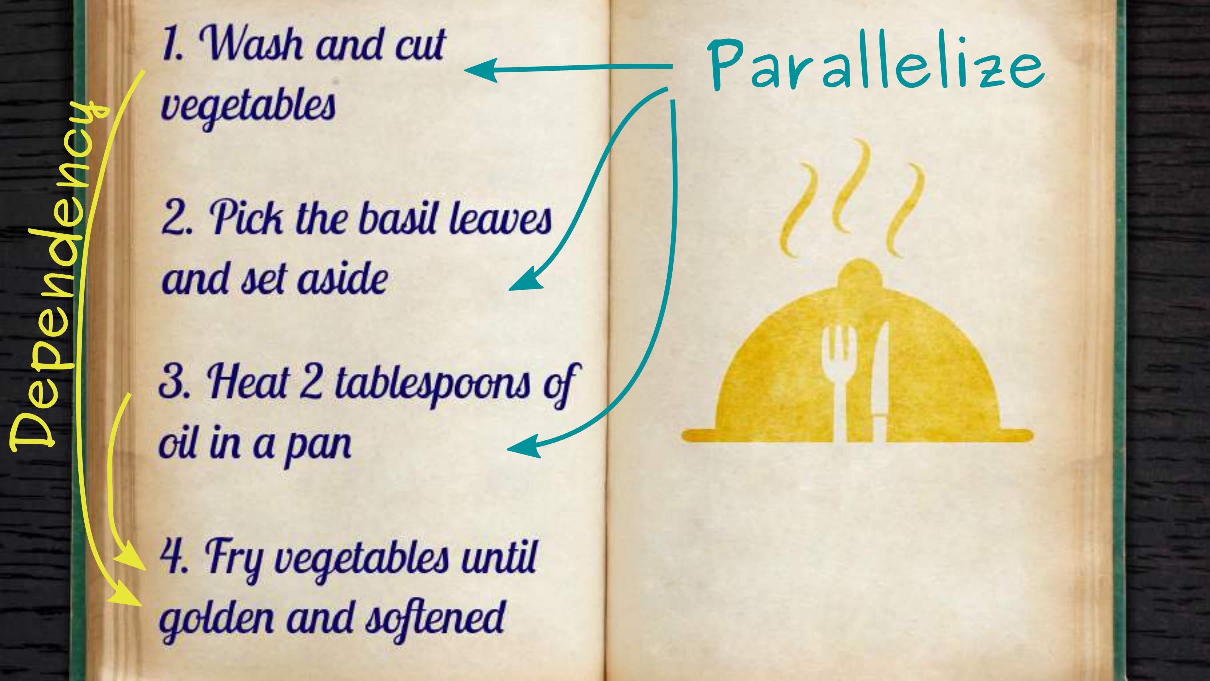
LATENCY

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

Out-of-order Execution

```c
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);
```

Out-of-order Execution

Parallelize

Dependency

```
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);
```

Building the Code

- Find something human readable, e.g., the Linux version

```
# sudo grep linux_banner /proc/kallsyms
ffffffff81a000e0 R linux_banner
```

Building the Code

```
char data = *(char*)0xffffffff81a000e0;
printf("%c\n", data);
```

Building the Code

- Compile and run

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

Building the Code

- Compile and run

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

- Kernel addresses are of course not accessible

Building the Code

- Compile and run

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

- Kernel addresses are of course not accessible
- Any invalid access throws an exception → segmentation fault

Building the Code

- Just catch the segmentation fault!

Building the Code

- Just catch the segmentation fault!
- We can simply install a signal handler

Building the Code

- Just catch the segmentation fault!
- We can simply install a signal handler
- And if an exception occurs, just jump back and continue

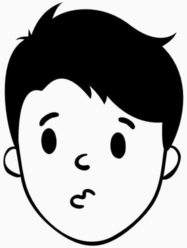
Building the Code

- Just catch the segmentation fault!
- We can simply install a signal handler
- And if an exception occurs, just jump back and continue
- Then we can read the value

Building the Code

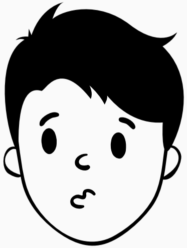
- Just catch the segmentation fault!
- We can simply install a signal handler
- And if an exception occurs, just jump back and continue
- Then we can read the value
- Sounds like a good idea

Building the Code



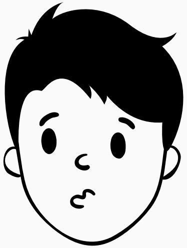
- Still no kernel memory

Building the Code



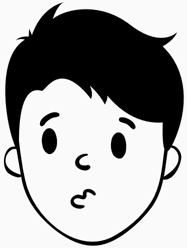
- Still no kernel memory
- Maybe it is not that straight forward

Building the Code



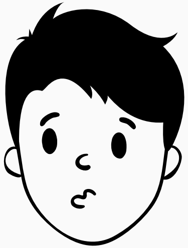
- Still no kernel memory
- Maybe it is not that straight forward
- Privilege checks seem to work

Building the Code



- Still no kernel memory
- Maybe it is not that straight forward
- Privilege checks seem to work
- Are privilege checks also done when executing instructions out of order?

Building the Code

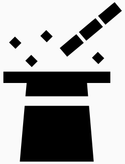


- Still no kernel memory
- Maybe it is not that straight forward
- Privilege checks seem to work
- Are privilege checks also done when executing instructions out of order?
- Problem: out-of-order instructions are not visible

Building the Code

- Adapted code

```
*(volatile char*)0;
array[0] = 0;
```



Building the Code

- Adapted code

```
*(volatile char*)0;
array[0] = 0;
```

- `volatile` because compiler was not happy

```
warning: statement with no effect [-Wunused-value]
         *(char*)0;
```

Building the Code

- Adapted code

```
*(volatile char*)0;
array[0] = 0;
```

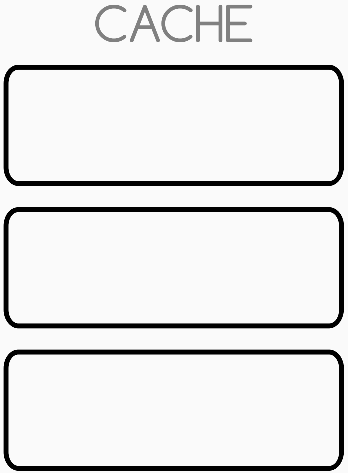
- `volatile` because compiler was not happy

```
warning: statement with no effect [-Wunused-value]
          *(char*)0;
```

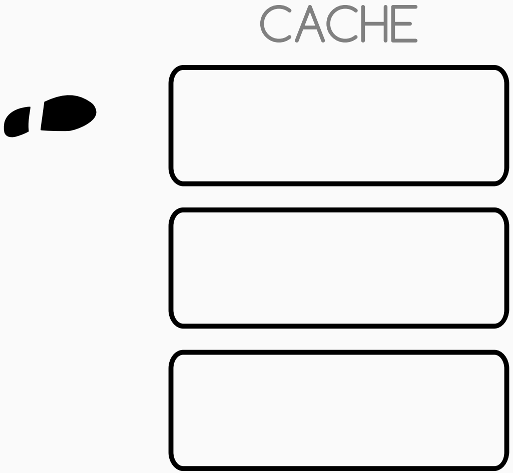
- Static code analyzer is still not happy

```
warning: Dereference of null pointer
         *(volatile char*)0;
```

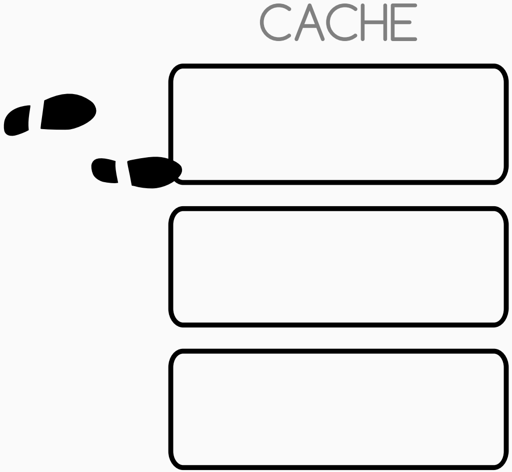
Traces in the Cache



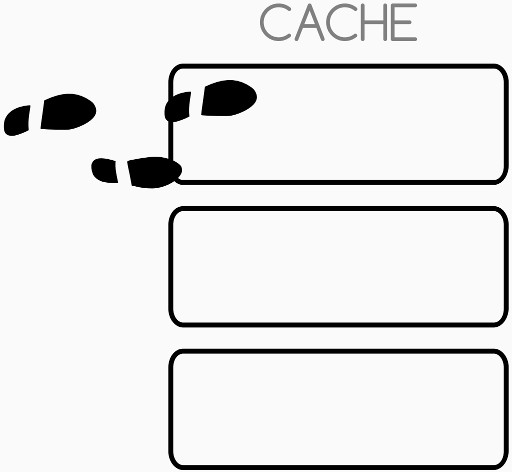
Traces in the Cache



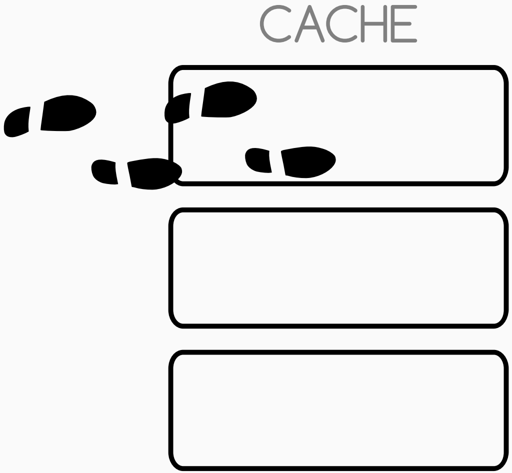
Traces in the Cache



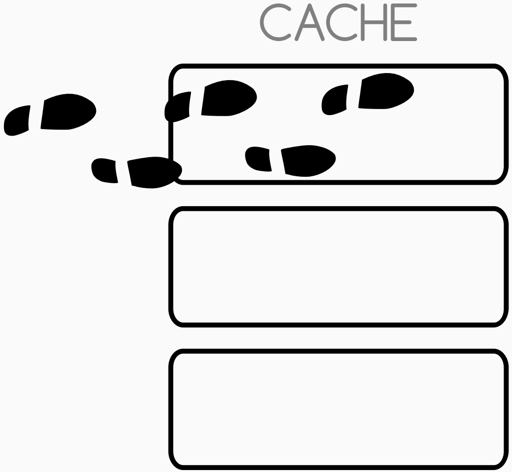
Traces in the Cache



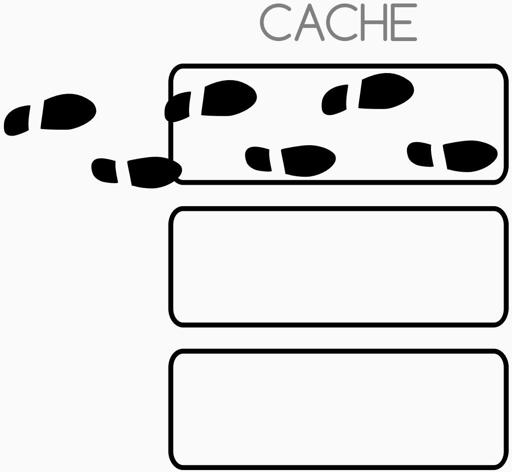
Traces in the Cache



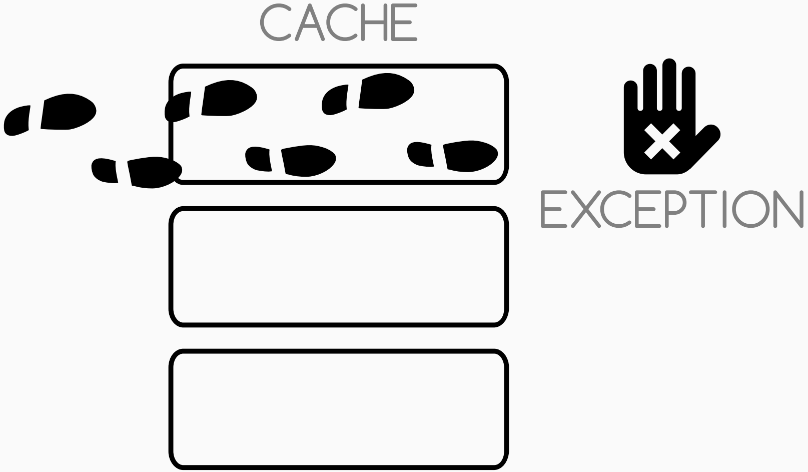
Traces in the Cache



Traces in the Cache



Traces in the Cache



Building the Code

- Out-of-order instructions leave microarchitectural traces

Building the Code

- Out-of-order instructions leave microarchitectural traces
- We can see them for example in the cache

Building the Code

- Out-of-order instructions leave microarchitectural traces
- We can see them for example in the cache
- Give such instructions a name: transient instructions

Building the Code

- Out-of-order instructions leave microarchitectural traces
- We can see them for example in the cache
- Give such instructions a name: transient instructions
- We can indirectly observe the execution of transient instructions

Building the Code

- Maybe there is no permission check in transient instructions...

Building the Code

- Maybe there is no permission check in transient instructions...
- ...or it is only done when commiting them

Building the Code

- Maybe there is no permission check in transient instructions...
- ...or it is only done when commiting them
- Add another layer of indirection to test

```
char data = *(char*)0xffffffff81a000e0;
array[data * 4096] = 0;
```

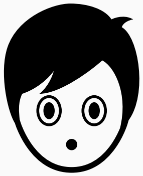
Building the Code

- Maybe there is no permission check in transient instructions...
- ...or it is only done when commiting them
- Add another layer of indirection to test

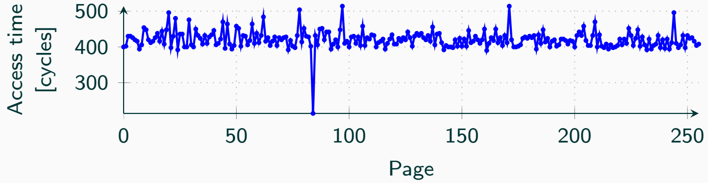
```
char data = *(char*)0xffffffff81a000e0;
array[data * 4096] = 0;
```

- Then check whether any part of `array` is cached

Building the Code

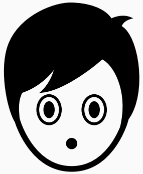


- Flush+Reload over all pages of the array

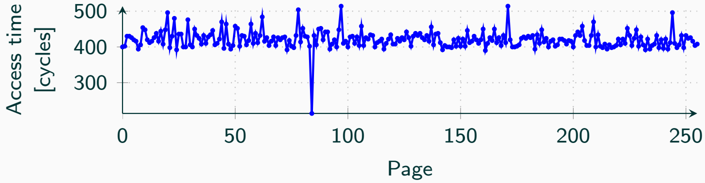


- Index of cache hit reveals data

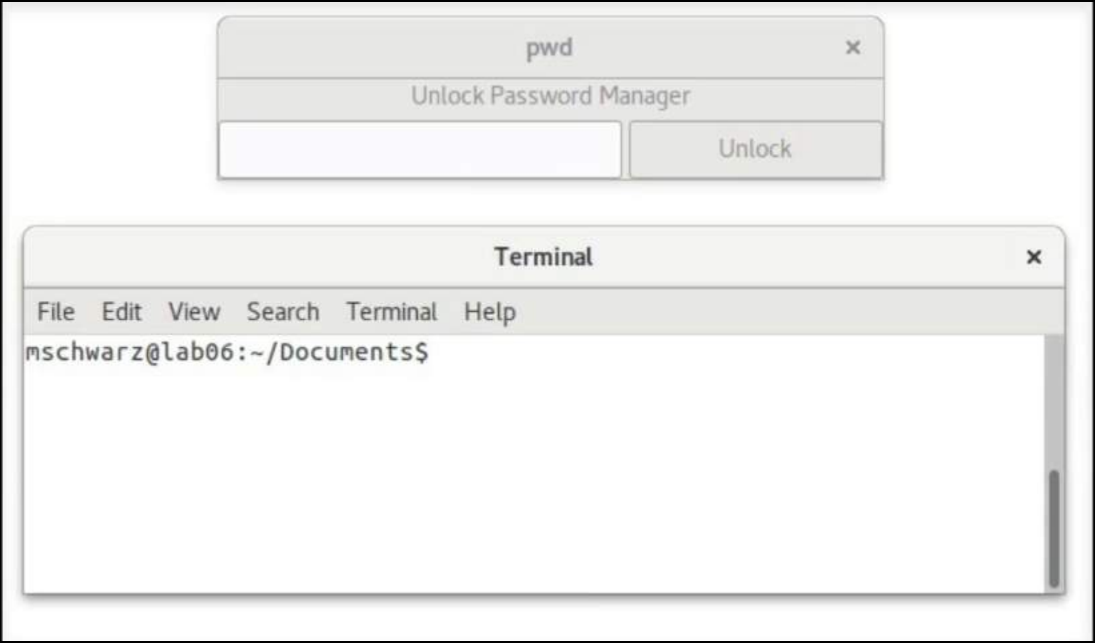
Building the Code



- Flush+Reload over all pages of the array

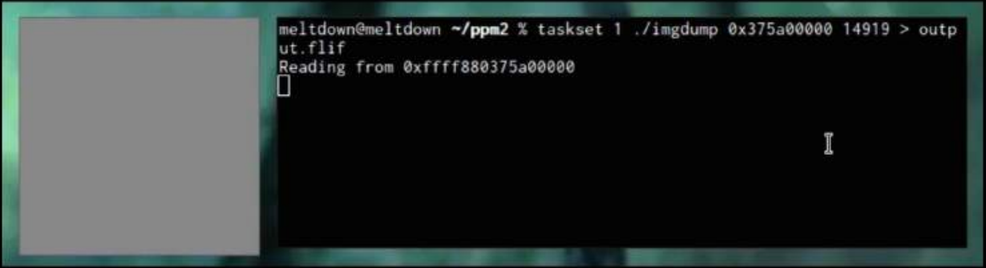


- Index of cache hit reveals data
- Permission check is in some cases not fast enough



```
mschwarz@lab06:~/Documents$
```

CAN YOU
ENHANCE THAT



```
meltdown@meltdown ~/ppm2 % taskset 1 ./imgdump 0x375a00000 14919 > outp
ut.flif
Reading from 0xffff880375a00000
```

AND IN OTHER NEWS...
MELTDOWN
WE'RE ALL DOOMED, SANDRA.
HOW ABOUT THE WEATHER?
COMEDY CENTRAL

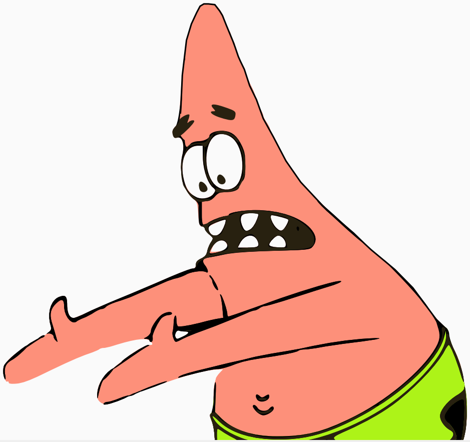
Not so fast. . .

Take the kernel addresses...

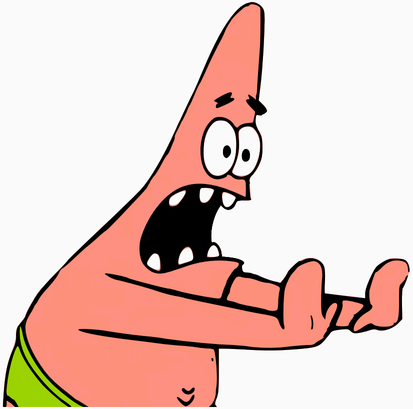
- Kernel addresses in user space are a problem

Take the kernel addresses...

- Kernel addresses in user space are a problem
- Why don't we take the kernel addresses...

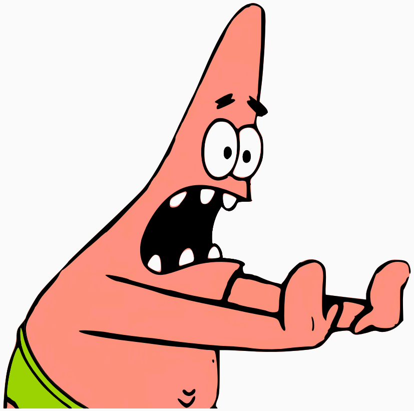


...and remove them



- ...and remove them if not needed?

...and remove them



- ...and remove them if not needed?
- User accessible check in hardware is not reliable

Idea

- Let's just unmap the kernel in user space

Idea

- Let's just unmap the kernel in user space
- Kernel addresses are then no longer present

Idea

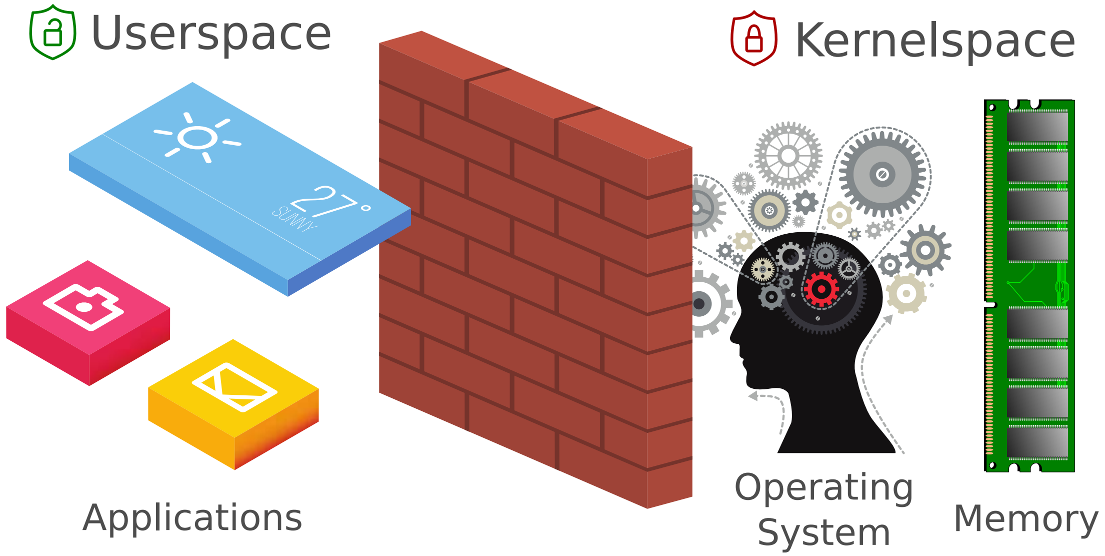
- Let's just unmap the kernel in user space
- Kernel addresses are then no longer present
- Memory which is not mapped cannot be accessed at all

Kernel **A**ddress **I**solation to have **S**ide channels **E**fficiently **R**emoved

KAISER /ˈkʌɪzə/

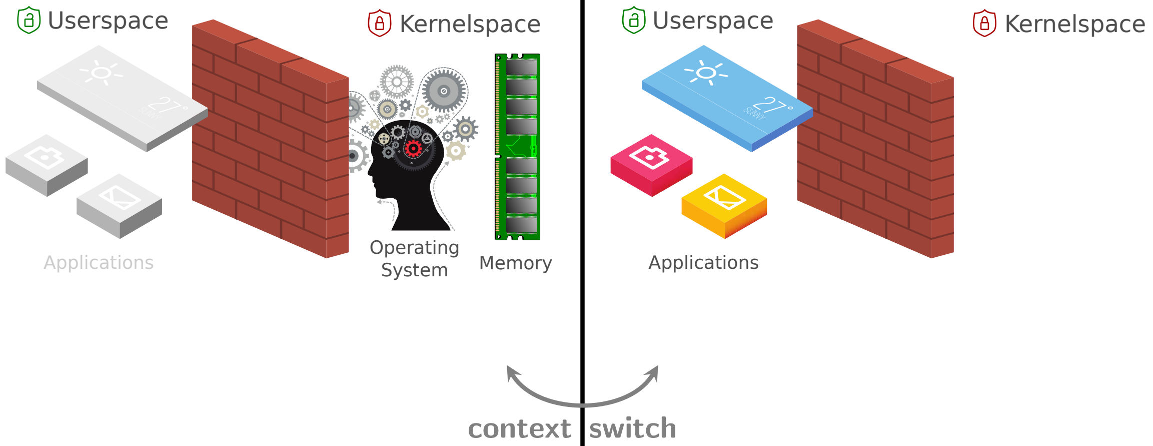
1. [german] Emperor, ruler of an empire
2. largest penguin, emperor penguin

Kernel **A**ddress **I**solation to have **S**ide channels **E**fficiently **R**emoved

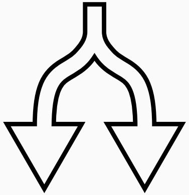

Userspace
Kernelspace
27°
SUNNY
Applications
Operating System
Memory

Kernel View

User View

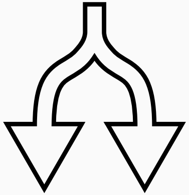


Kernel Address Space Isolation



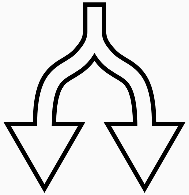
- We published KAISER in July 2017

Kernel Address Space Isolation



- We published KAISER in July 2017
- Intel and others improved and merged it into Linux as KPTI (Kernel Page Table Isolation)

Kernel Address Space Isolation

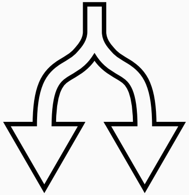


- We published KAISER in July 2017
- Intel and others improved and merged it into Linux as KPTI (Kernel Page Table Isolation)
- Microsoft implemented similar concept in Windows 10

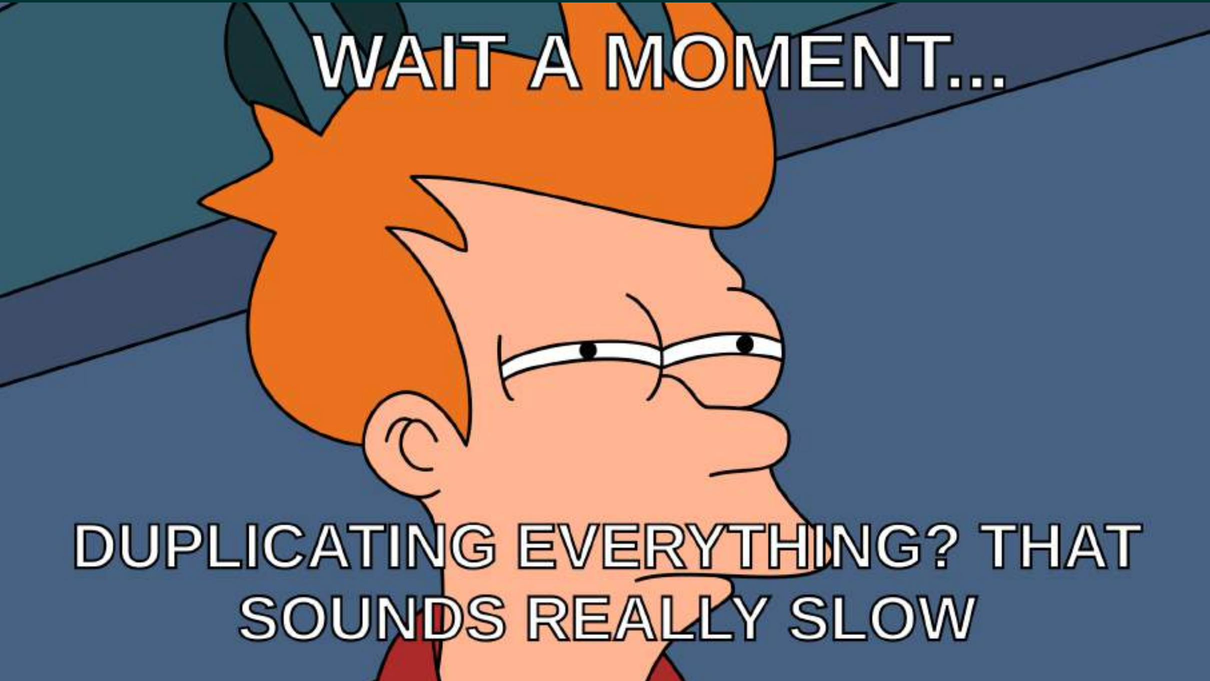
Kernel Address Space Isolation

- We published KAISER in July 2017
- Intel and others improved and merged it into Linux as KPTI (Kernel Page Table Isolation)
- Microsoft implemented similar concept in Windows 10
- Apple implemented it in macOS 10.13.2 and called it “Double Map”

Kernel Address Space Isolation



- We published KAISER in July 2017
- Intel and others improved and merged it into Linux as KPTI (Kernel Page Table Isolation)
- Microsoft implemented similar concept in Windows 10
- Apple implemented it in macOS 10.13.2 and called it “Double Map”
- All share the same idea: switching address spaces on context switch


WAIT A MOMENT...
DUPLICATING EVERYTHING? THAT SOUNDS REALLY SLOW

Performance

- Depends on how often you need to switch between kernel and user space

Performance

- Depends on how often you need to switch between kernel and user space
- Can be slow, 40% or more on old hardware

Performance

- Depends on how often you need to switch between kernel and user space
- Can be slow, 40% or more on old hardware
- But modern CPUs have additional features

Performance

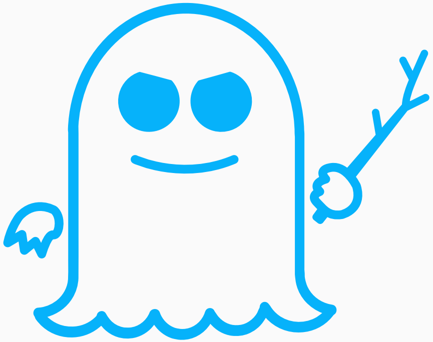
- Depends on how often you need to switch between kernel and user space
- Can be slow, 40% or more on old hardware
- But modern CPUs have additional features
- ⇒ Performance overhead on average below 2%

Meltdown and Spectre

MELTDOWN

SPECTRE

~~Meltdown and~~ Spectre



MELTDOWN

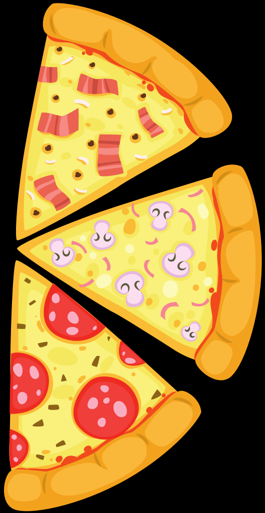
SPECTRE

PIZZA

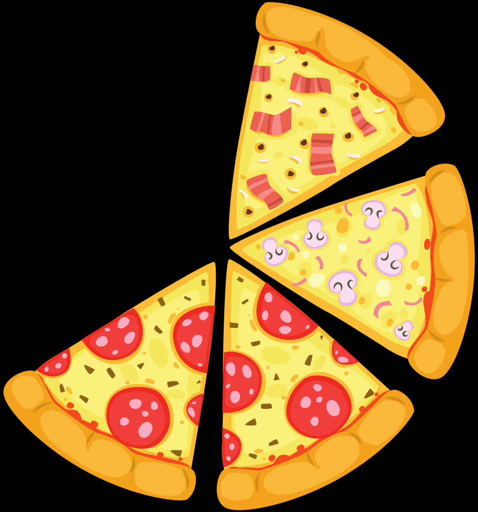
SPECIAL RECIPES

Prosciutto

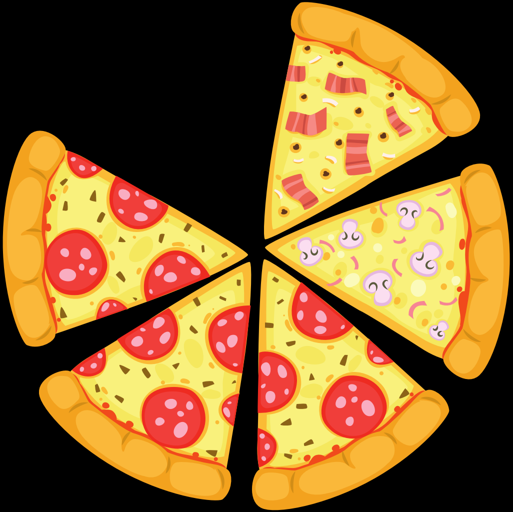
Funghi



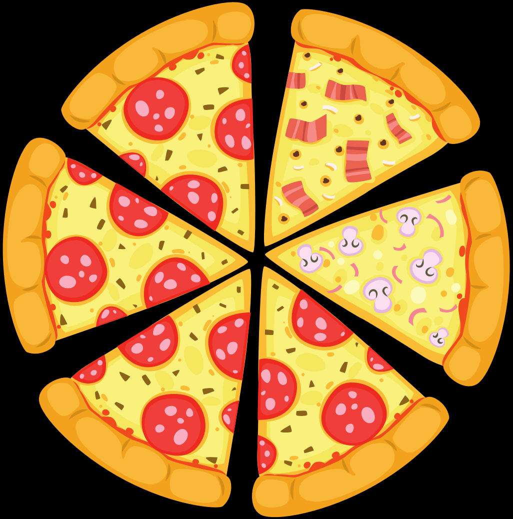
Diavolo



Diavolo



Diavolo



Diavolo

»A table for 6 please«

Speculative Cooking

»A table for 6 please«

PIZZA
SPECIAL RECIPES

PIZZA
SPECIAL RECIPES

Pizza

PIZZA
SPECIAL RECIPES

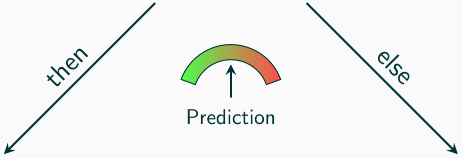
Pizza

PIZZA
SPECIAL RECIPES

Pizza

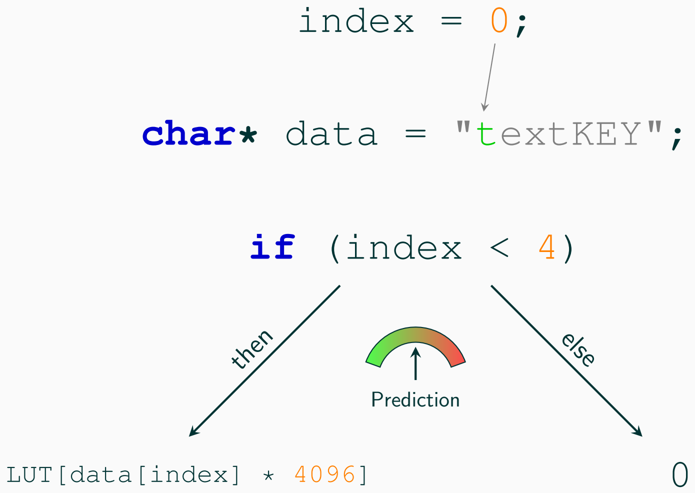
Spectre (variant 1)

```
index = 0;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                    0
```

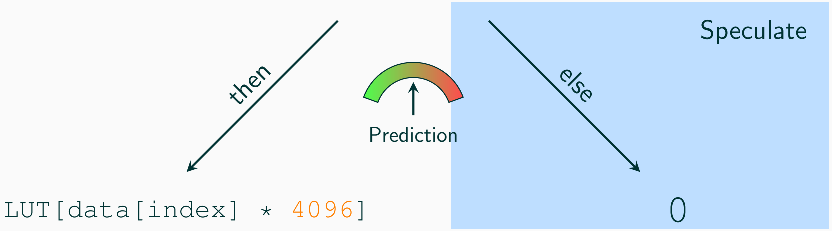
Spectre (variant 1)



```
index = 0;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]                0
```

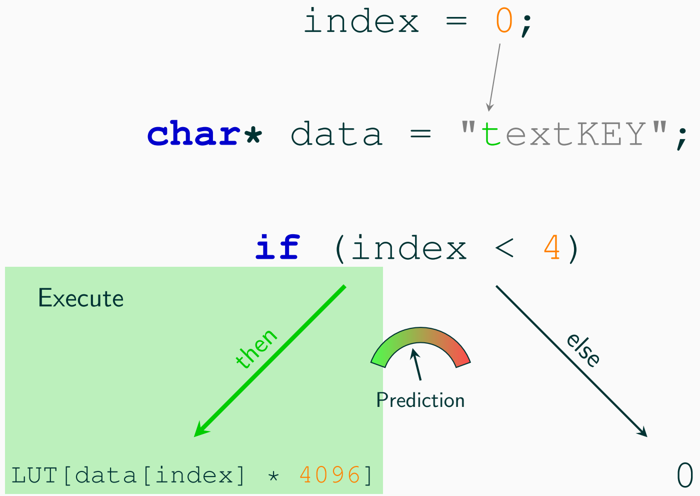
Spectre (variant 1)

```
index = 0;

char* data = "textKEY";

if (index < 4)
```



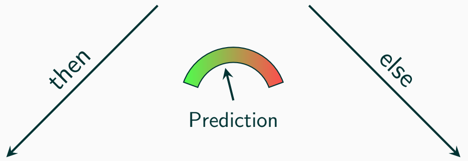
```
LUT[data[index] * 4096]
```

Spectre (variant 1)


```
index = 0;

char* data = "textKEY";

if (index < 4)
```

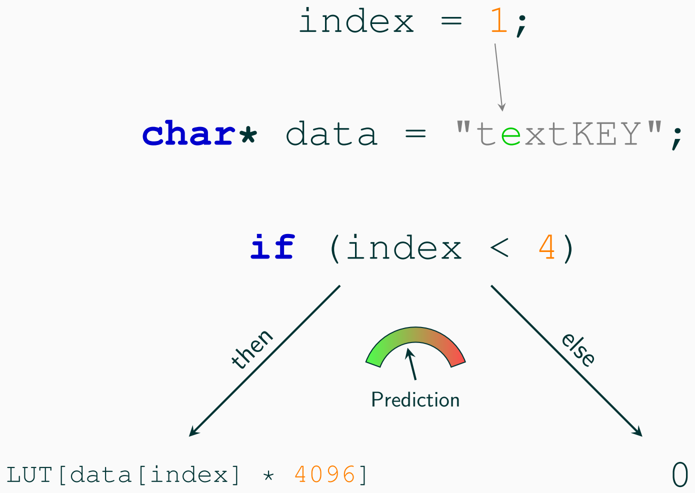
Spectre (variant 1)

```
index = 1;

char* data = "textKEY";

if (index < 4)
```



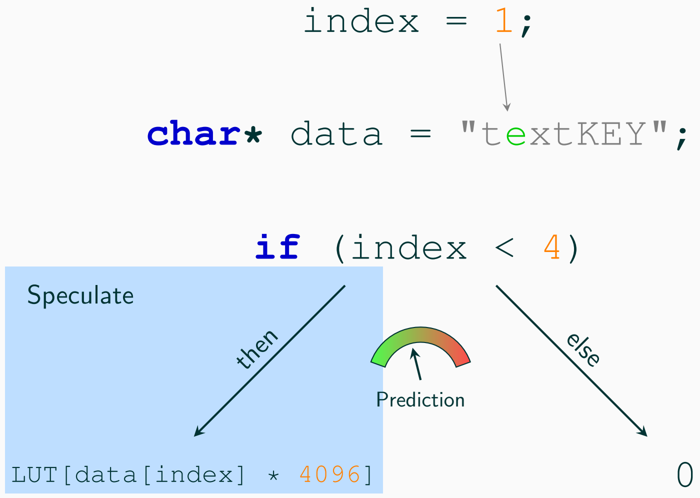
```
LUT[data[index] * 4096]                    0
```

Spectre (variant 1)



```
index = 1;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]                     0
```

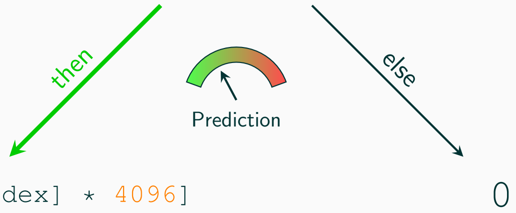
Spectre (variant 1)



```
index = 1;

char* data = "textKEY";

if (index < 4)
```

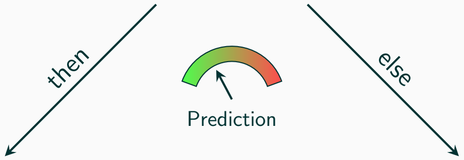
Spectre (variant 1)

```
index = 1;

char* data = "textKEY";

if (index < 4)
```



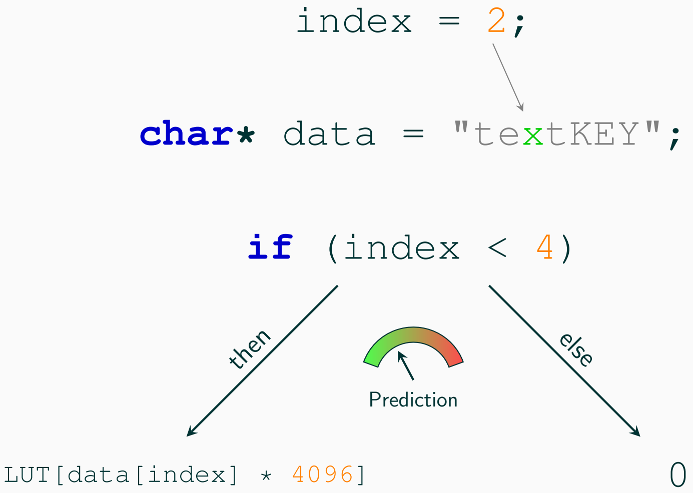
Spectre (variant 1)

```
index = 2;

char* data = "textKEY";

if (index < 4)
```



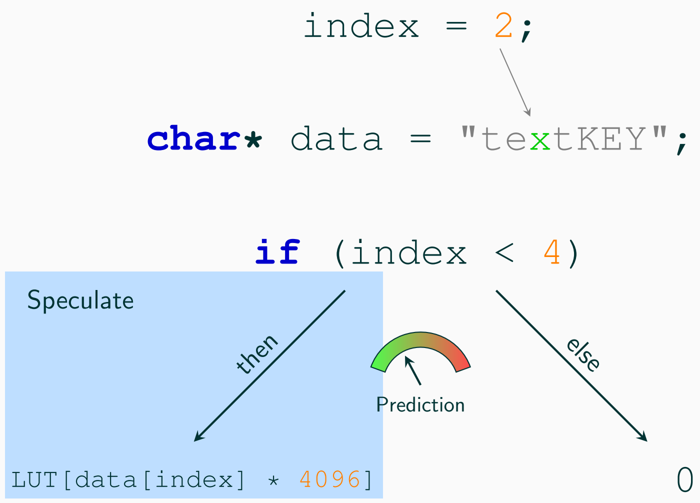
```
LUT[data[index] * 4096]                0
```

Spectre (variant 1)

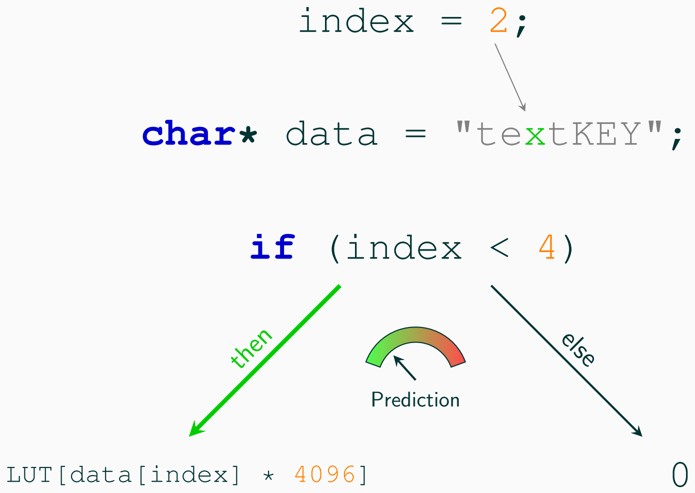


```
index = 2;

char* data = "textKEY";

if (index < 4)
```

Spectre (variant 1)



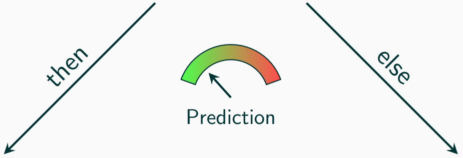
Spectre (variant 1)



```
index = 2;

char* data = "textKEY";

if (index < 4)

LUT[data[index] * 4096]                    0
```

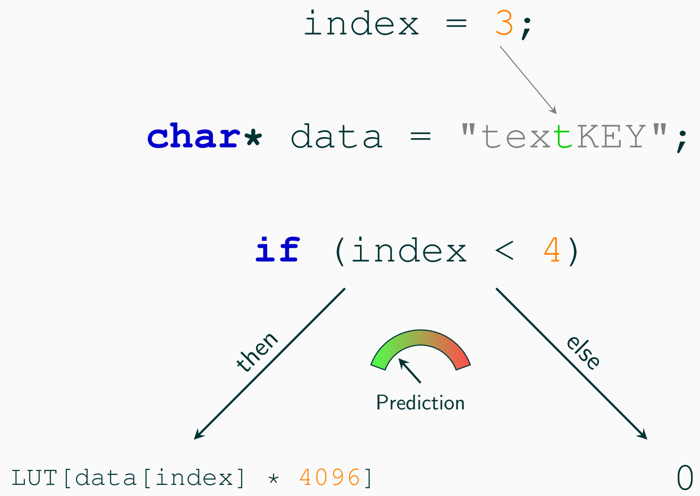
Spectre (variant 1)

```
index = 3;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                0
```

Spectre (variant 1)

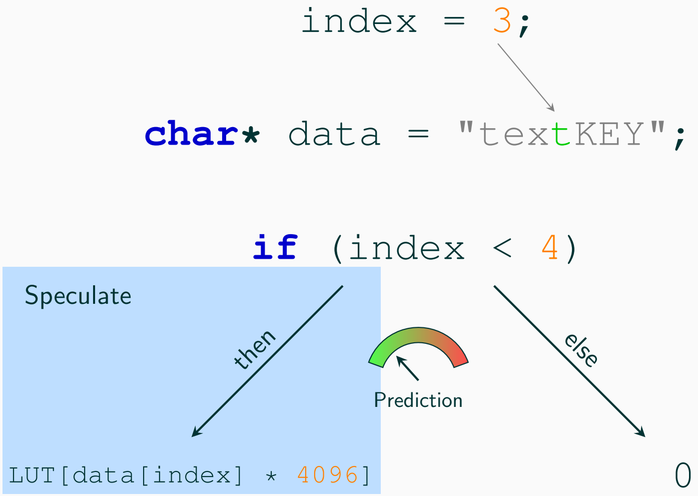


```
index = 3;

char* data = "textKEY";

if (index < 4)
```

```
LUT[data[index] * 4096]
```

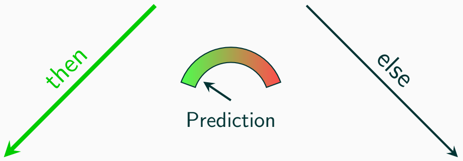
```
0
```

Spectre (variant 1)



Spectre (variant 1)

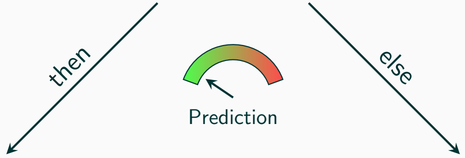
```
index = 3;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                0
```

Spectre (variant 1)

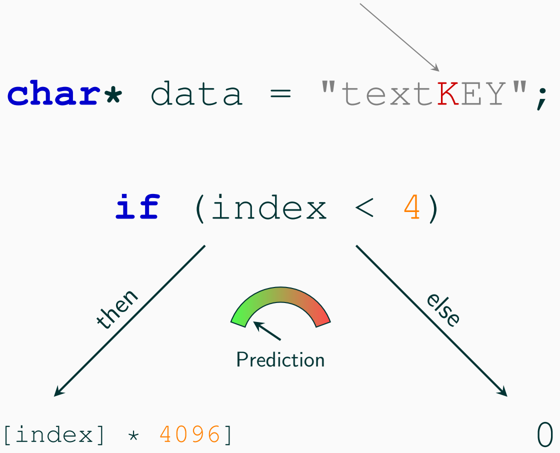
```
index = 4;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                    0
```

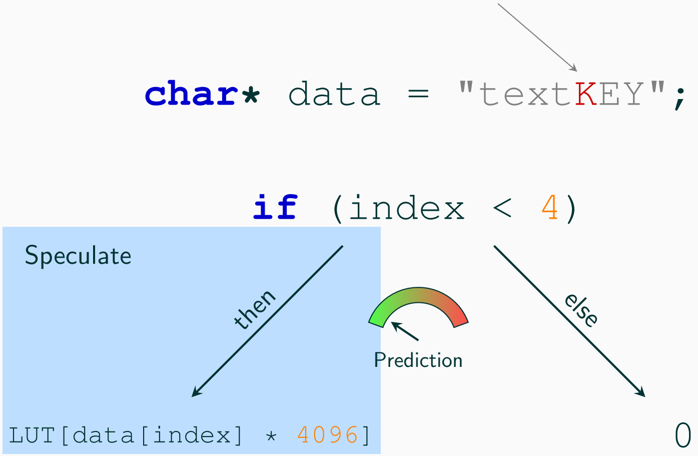
Spectre (variant 1)

```
index = 4;

char* data = "textKEY";

if (index < 4)
```



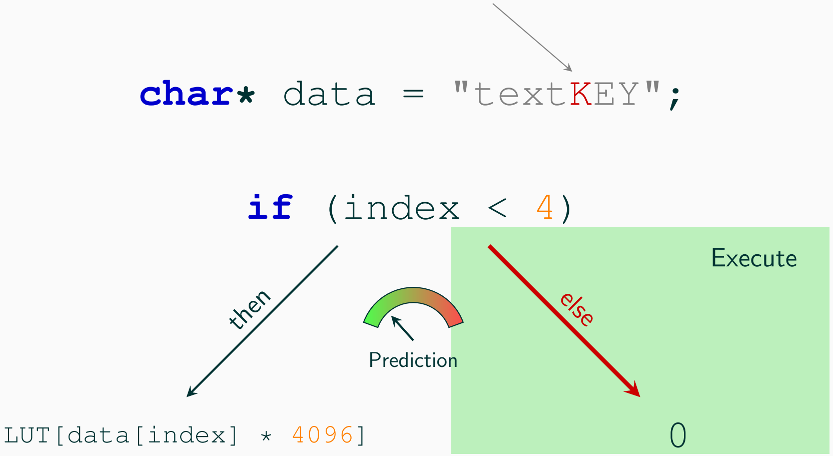
Spectre (variant 1)

```
index = 4;

char* data = "textKEY";

if (index < 4)
```



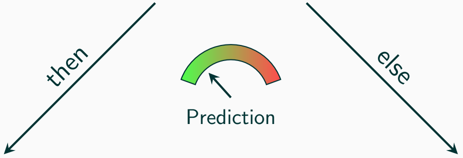
Spectre (variant 1)

```
index = 4;

char* data = "textKEY";

if (index < 4)
```



Spectre (variant 1)

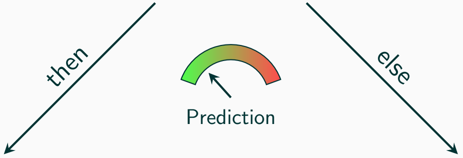
```
index = 5;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                    0
```

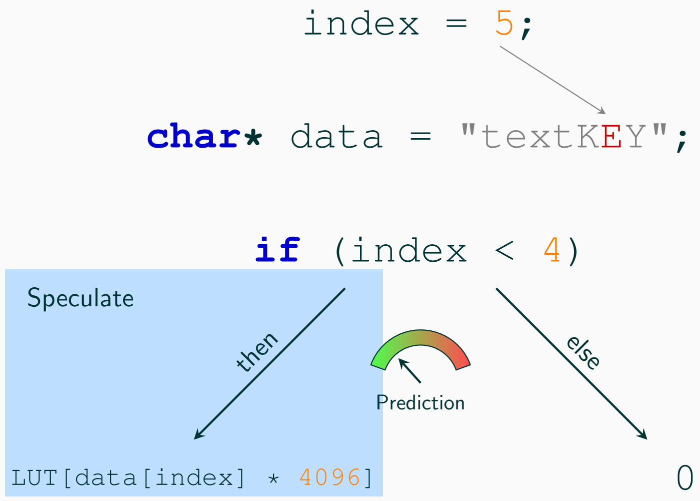
Spectre (variant 1)

```
index = 5;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                    0
```

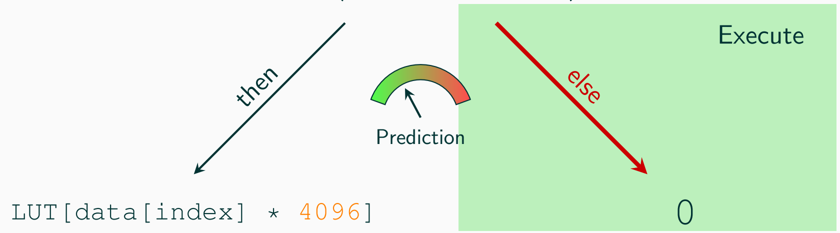
Spectre (variant 1)



```
index = 5;

char* data = "textKEY";

if (index < 4)
```

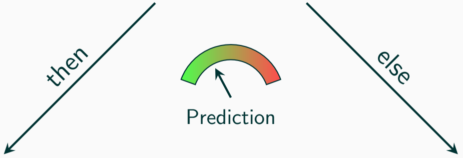
Spectre (variant 1)

```
index = 5;

char* data = "textKEY";

if (index < 4)
```



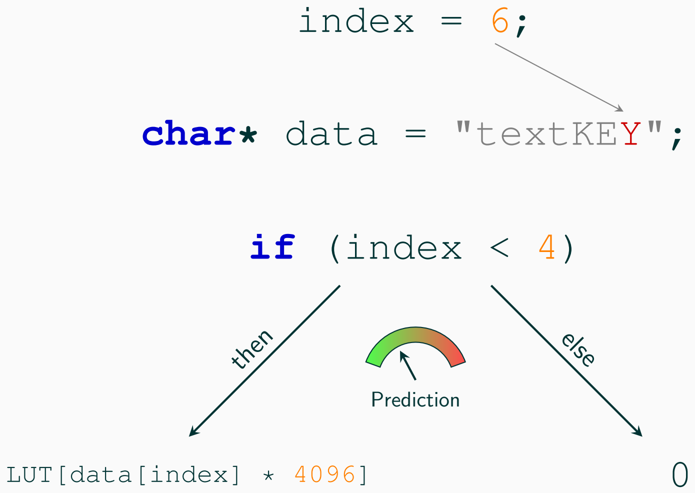
Spectre (variant 1)

```
index = 6;

char* data = "textKEY";

if (index < 4)
```



```
LUT[data[index] * 4096]                    0
```

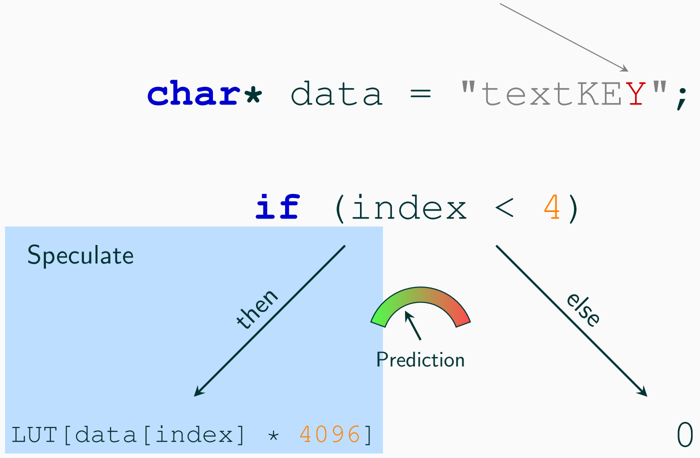
Spectre (variant 1)



```
index = 6;

char* data = "textKEY";

if (index < 4)
```

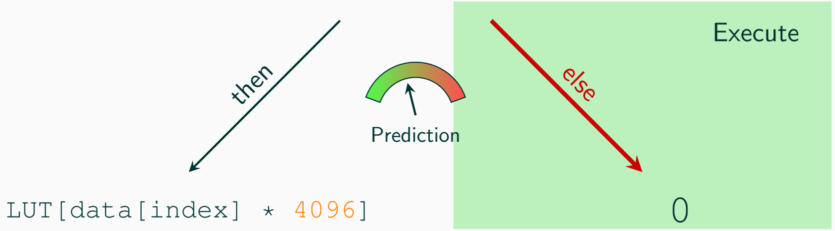
Spectre (variant 1)

```
index = 6;

char* data = "textKEY";

if (index < 4)
```



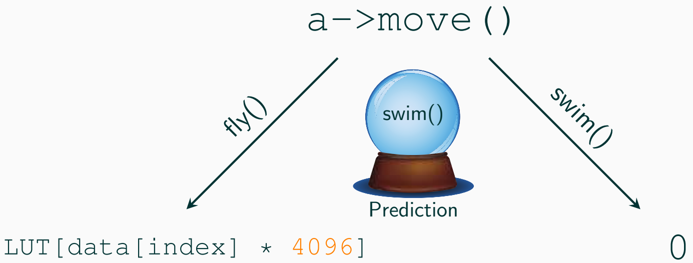
Spectre (variant 1)

```
index = 6;

char* data = "textKEY";

if (index < 4)
```



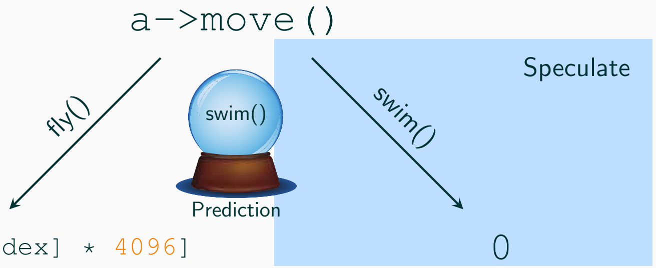
Spectre (variant 2)

```
Animal* a = bird;
```



Spectre (variant 2)

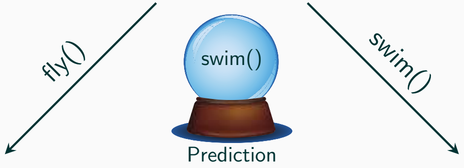
```
Animal* a = bird;
```



Spectre (variant 2)

```
Animal* a = bird;
```

```
a->move()
```

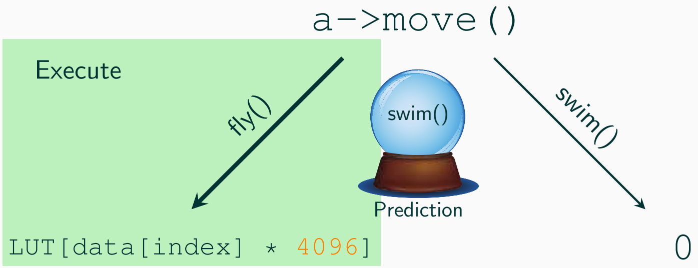


```
LUT[data[index] * 4096]
```

0

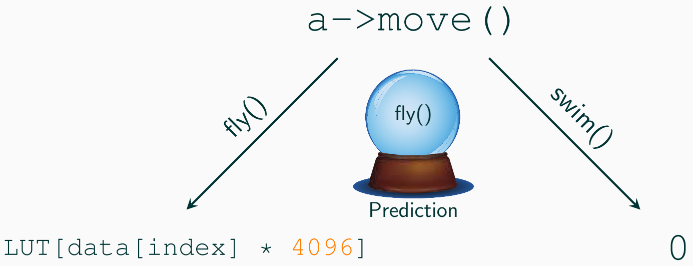
Spectre (variant 2)

```
Animal* a = bird;
```



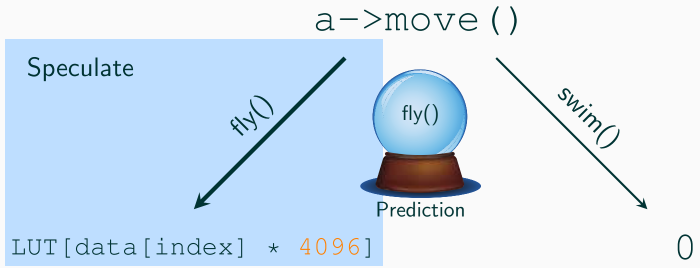
Spectre (variant 2)

```
Animal* a = bird;
```



Spectre (variant 2)

```
Animal* a = bird;
```



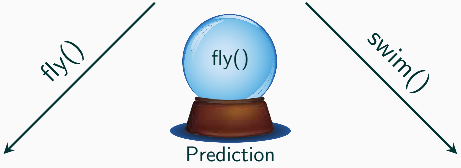
```
a->move()
```

```
LUT[data[index] * 4096]
```

Spectre (variant 2)

```
Animal* a = bird;
```

```
a->move()
```



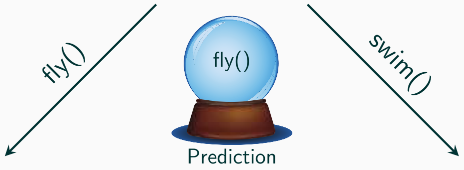
```
LUT[data[index] * 4096]
```

0

Spectre (variant 2)

```
Animal* a = fish;
```

```
a->move()
```

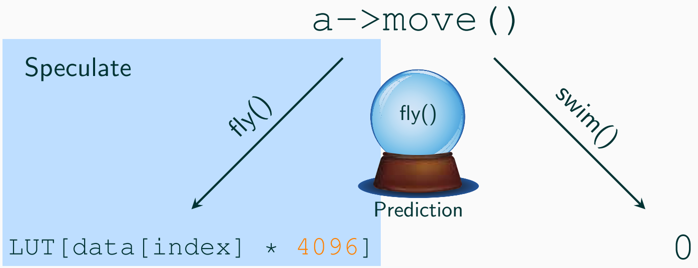


```
LUT[data[index] * 4096]
```

0

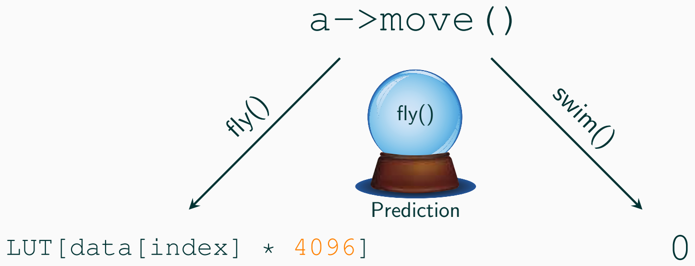
Spectre (variant 2)

```
Animal* a = fish;
```



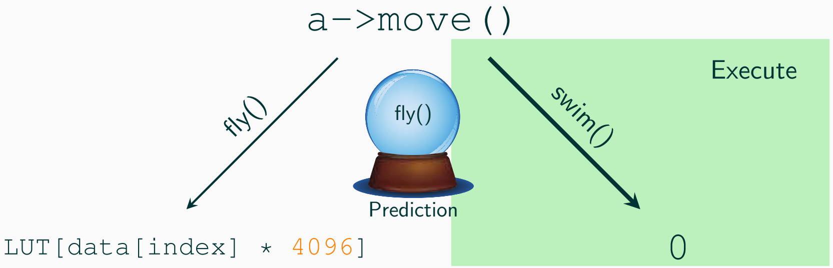
Spectre (variant 2)

```
Animal* a = fish;
```



Spectre (variant 2)

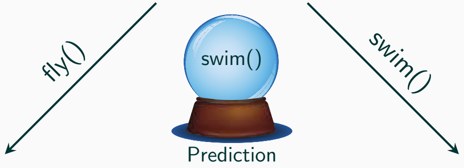
```
Animal* a = fish;
```



Spectre (variant 2)

```
Animal* a = fish;
```

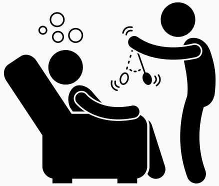
```
a->move()
```



```
LUT[data[index] * 4096]
```

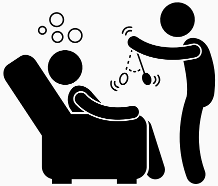
0

Spectre



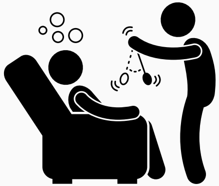
- Read own memory (e.g., sandbox escape)

Spectre



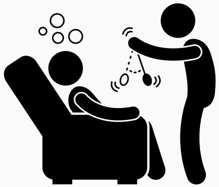
- Read own memory (e.g., sandbox escape)
- “Convince” other programs to reveal their secrets

Spectre



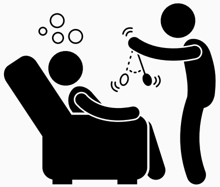
- Read own memory (e.g., sandbox escape)
- “Convince” other programs to reveal their secrets
- Again, a cache attack (Flush+Reload) is used to read the secret

Spectre



- Read own memory (e.g., sandbox escape)
- “Convince” other programs to reveal their secrets
- Again, a cache attack (Flush+Reload) is used to read the secret
- Much harder to fix, KAISER does not help

Spectre



- Read own memory (e.g., sandbox escape)
- "Convince" other programs to reveal their secrets
- Again, a cache attack (Flush+Reload) is used to read the secret
- Much harder to fix, KAISER does not help
- Ongoing effort to patch via microcode update and compiler extensions

Spectre Variant 1 Mitigations

Spectre Variant 1 Mitigations

- LFENCE

Spectre Variant 1 Mitigations

- LFENCE

→ speculation barrier to insert after every bounds check

Spectre Variant 1 Mitigations

- LFENCE

→ speculation barrier to insert after every bounds check

- implemented as a compiler extension

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):
    - do not speculate based on anything before entering IBRS mode

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):
  - do not speculate based on anything before entering IBRS mode
  - hyperthreading?

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):
  - do not speculate based on anything before entering IBRS mode
  - hyperthreading?
- Indirect Branch Predictor Barrier (IBPB):

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):
  - do not speculate based on anything before entering IBRS mode
  - hyperthreading?
- Indirect Branch Predictor Barrier (IBPB):
  - flush branch-target buffer

Spectre Variant 2 Mitigations (Microcode/MSRs)

- Indirect Branch Restricted Speculation (IBRS):
  - do not speculate based on anything before entering IBRS mode
  - hyperthreading?
- Indirect Branch Predictor Barrier (IBPB):
  - flush branch-target buffer
  - hyperthreading?

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP)

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
  push <call_target>
  call 1f
2:                        ; speculation will continue here
  lfence                  ; speculation barrier
  jmp 2b                  ; endless loop
1:
  lea 8(%rsp), %rsp       ; restore stack pointer
  ret                     ; the actual call to <call_target>
```

→ always predict to enter an endless loop

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
  push <call_target>
  call 1f
2:                          ; speculation will continue here
  lfence                    ; speculation barrier
  jmp 2b                    ; endless loop
1:
  lea 8(%rsp), %rsp         ; restore stack pointer
  ret                       ; the actual call to <call_target>
```

→ always predict to enter an endless loop

- instead of the correct (or wrong) target function

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
  push <call_target>
  call 1f
2:                      ; speculation will continue here
  lfence                ; speculation barrier
  jmp 2b                ; endless loop
1:
  lea 8(%rsp), %rsp     ; restore stack pointer
  ret                   ; the actual call to <call_target>
```

→ always predict to enter an endless loop

- instead of the correct (or wrong) target function → performance?

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
  push <call_target>
  call 1f
2:                          ; speculation will continue here
  lfence                    ; speculation barrier
  jmp 2b                    ; endless loop
1:
  lea 8(%rsp), %rsp         ; restore stack pointer
  ret                       ; the actual call to <call_target>
```

→ always predict to enter an endless loop

- instead of the correct (or wrong) target function → performance?
- On Broadwell or newer:

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
    push <call_target>
    call 1f
  2:                      ; speculation will continue here
    lfence                ; speculation barrier
    jmp 2b                ; endless loop
  1:
    lea 8(%rsp), %rsp     ; restore stack pointer
    ret                   ; the actual call to <call_target>
```

→ always predict to enter an endless loop

- instead of the correct (or wrong) target function → performance?
- On Broadwell or newer:
  - **ret** may fall-back to the BTB for prediction

Spectre Variant 2 Mitigations (Software)

Single Thread Indirect Branch Predictors (STIBP) = **retpoline**

```
  push <call_target>
  call 1f
2:                      ; speculation will continue here
  lfence                ; speculation barrier
  jmp 2b                ; endless loop
1:
  lea 8(%rsp), %rsp     ; restore stack pointer
  ret                   ; the actual call to <call_target>
```

→ always predict to enter an endless loop

- instead of the correct (or wrong) target function → performance?
- On Broadwell or newer:
  - **ret** may fall-back to the BTB for prediction

  → microcode patches to prevent that

What do we learn from it?

We have ignored software side-channels for many many years:

What do we learn from it?

We have ignored software side-channels for many many years:

- attacks on crypto

What do we learn from it?

We have ignored software side-channels for many many years:

- attacks on crypto → “software should be fixed”

What do we learn from it?

We have ignored software side-channels for many many years:

- attacks on crypto → "software should be fixed"
- attacks on ASLR

What do we learn from it?

We have ignored software side-channels for many many years:

- attacks on crypto → “software should be fixed”
- attacks on ASLR → “ASLR is broken anyway”

What do we learn from it?

We have ignored software side-channels for many many years:

- attacks on crypto $\rightarrow$ "software should be fixed"
- attacks on ASLR $\rightarrow$ "ASLR is broken anyway"
- attacks on SGX and TrustZone

What do we learn from it?

We have ignored software side-channels 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”

What do we learn from it?

We have ignored software side-channels 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”

→ for years we solely optimized for performance

When you read the Intel manuals...

After learning about a side channel you realize:

When you read the Intel manuals...

After learning about a side channel you realize:

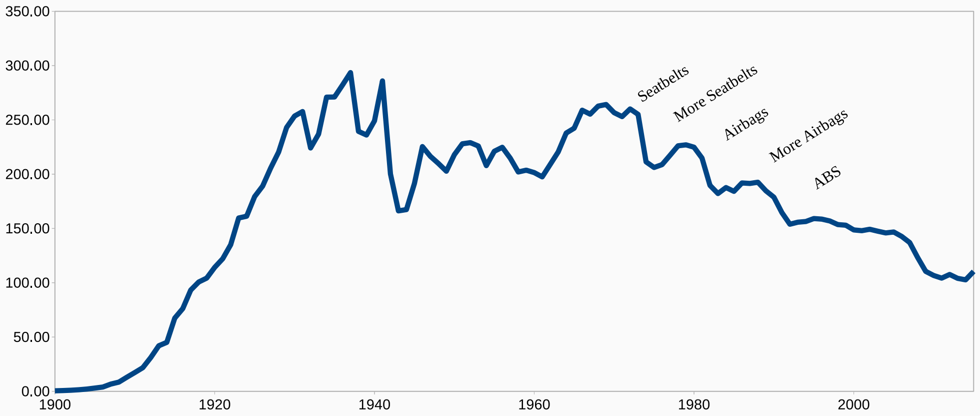
- the side channels were documented in the Intel manual

When you read the Intel manuals...

After learning about a side channel you realize:

- the side channels were documented in the Intel manual
- only now we understand the implications

What do we learn from it?



Motor Vehicle Deaths in U.S. by Year

A unique chance

A unique chance to

- rethink processor design
- grow up, like other fields (car industry, construction industry)
- find good trade-offs between security and performance

Conclusion

- Underestimated microarchitectural attacks for a long time
    - Basic techniques were there for years
- Industry and customers must embrace security mechanisms
    - Run through the same development (for security) as the automobile industry (for safety)
    - It should not be “performance first”, but “security first”

MELTDOWN

SPECTRE

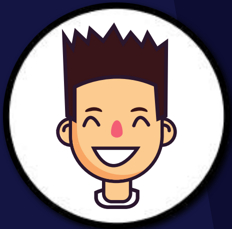
FACt

MELTDOWN

SPECTRE

Any Questions?

Beyond Belief: Spectre and Meltdown



Daniel Gruss
@lavados

Moritz Lipp
@mlqxyz

Michael Schwarz
@misc0110