CPU Vulnerabilities

transient execution attacks made easy

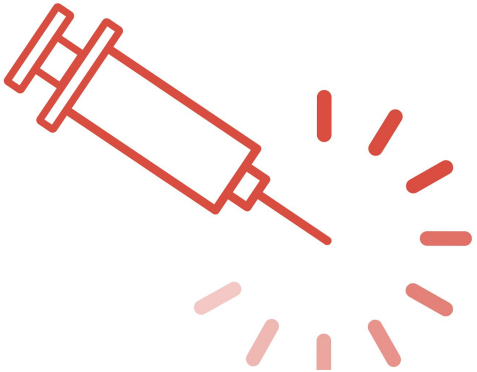
Herbert Bos

Vrije Universiteit Amsterdam

https://vusec.net

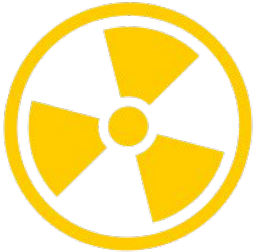
Vulnerabilities in the hardware: transient execution

There is by now a family of such bugs



RIDL

404 Logo Not Found



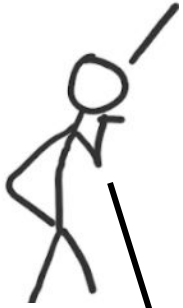
Problem

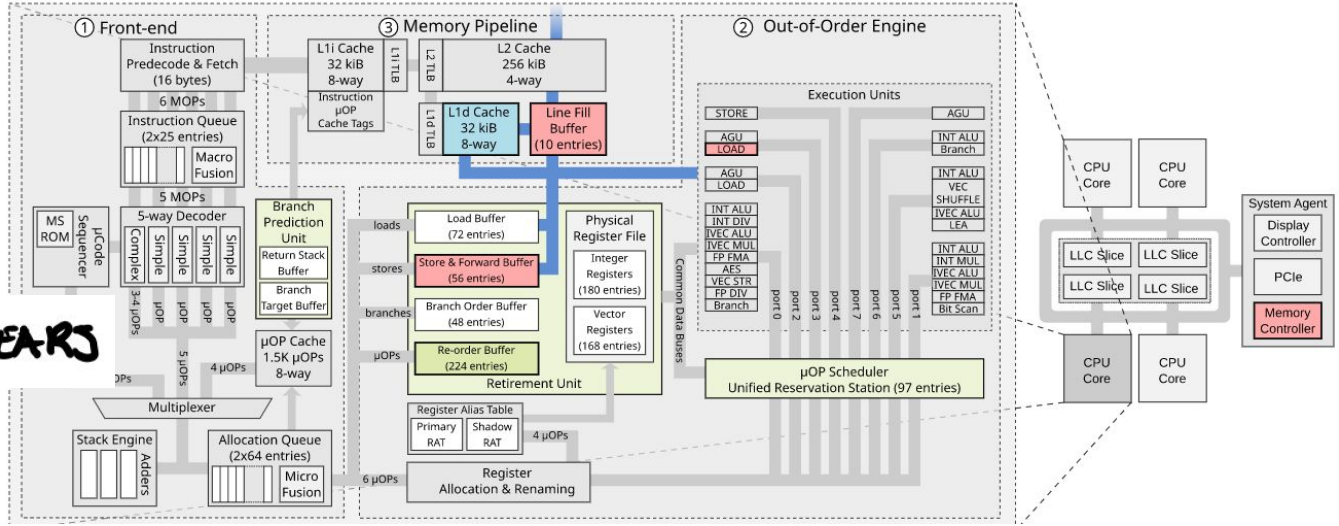
These vulnerabilities allow attackers to access data across security boundaries


SO, WHAT'S UP WITH THE HARDWARE?

GOSH...

CPUs HAVE BECOME AMAZINGLY COMPLEX





PAST 50 YEARS

EVERYTHING SACRIFICED AT ALTAR OF PERFORMANCE

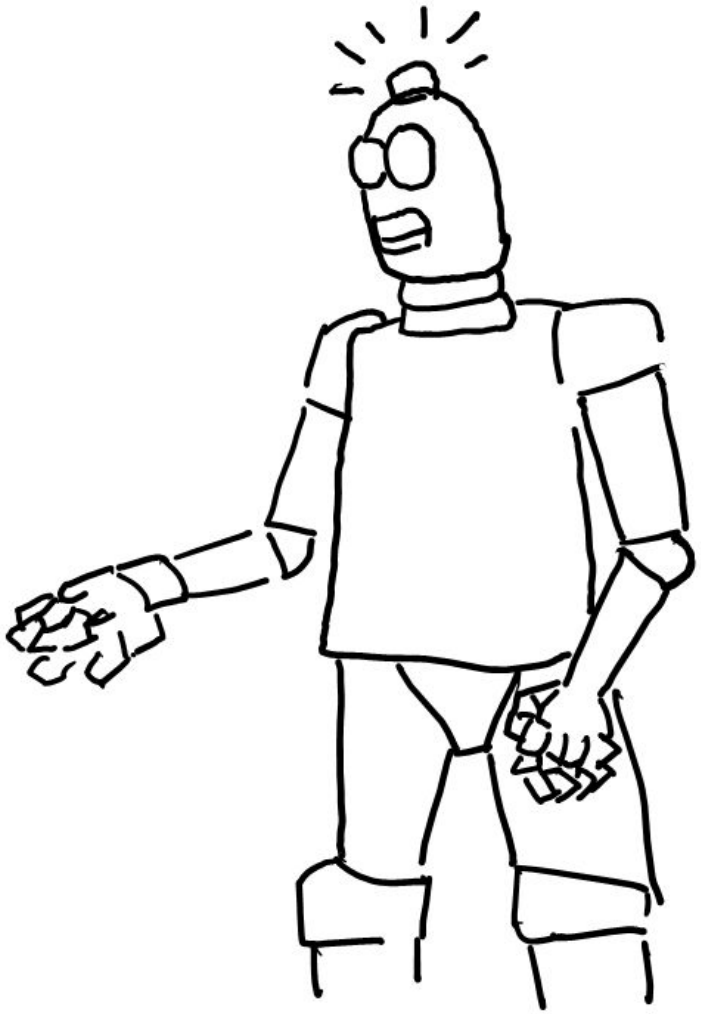
ARCHITECTURE

INSTRUCTIONS

REGISTERS

ADDRESSING MODES

⋮



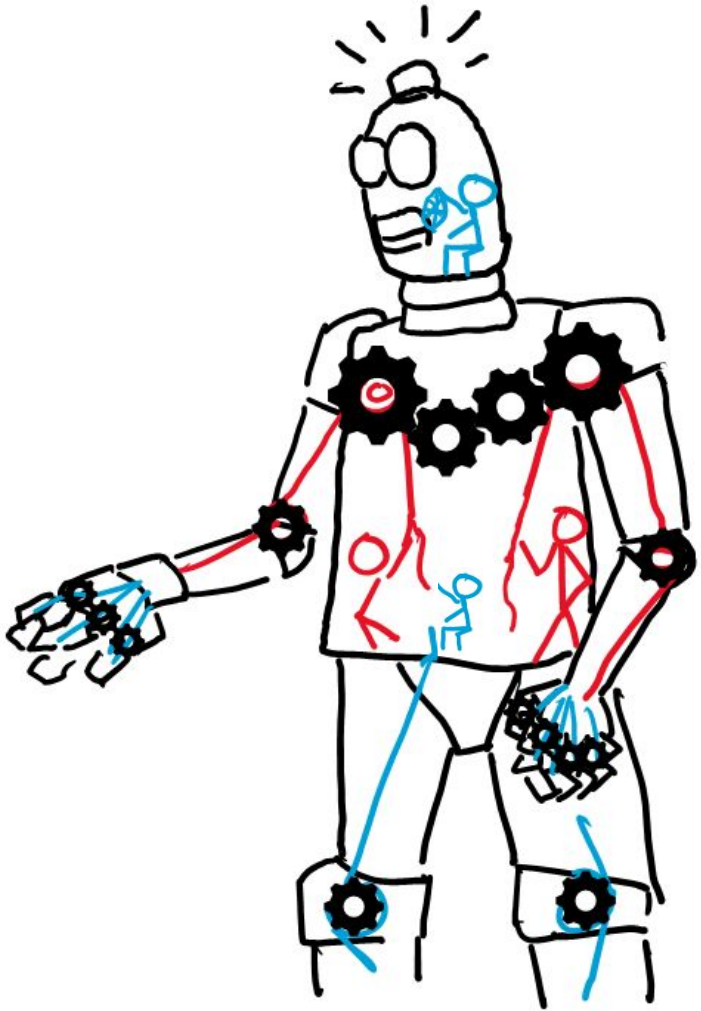
MICRO ARCHITECTURE

CACHES

HOW INSTRUCTIONS EXECUTE

OPTIMIZATIONS

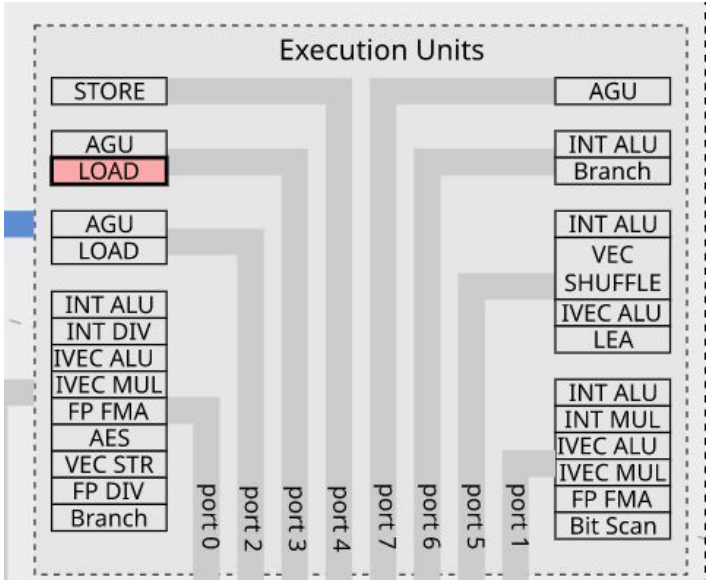
HIDDEN REGISTERS




CPUs HAVE BECOME AMAZINGLY COMPLEX
PAST 50 YEARS
EVERYTHING SACRIFICED
AT ALTAR OF PERFORMANCE

WHAT DOES THAT MEAN?

CPU DOESN'T WANT TO WAIT




```
if (a == foo) {
  x  = y + z;
  v  = b - d;
} else ...
```

SPECULATE

→ IN CASE OF ERROR → SQUASH RESULTS!

Transient Execution

SPECULATNG EXECUTION

A LOCATION IN MEMORY

```
IF ( X == 42 ) {
    Y = 3;
    Z = Y * X;
    ...
}
```

PREDICTORS

WHAT HAPPENED THE LAST 10 TIMES HERE?

→ IN CASE OF ERROR → SQUASH RESULTS!

SPECULATNG EXECUTION

JMP *RAX

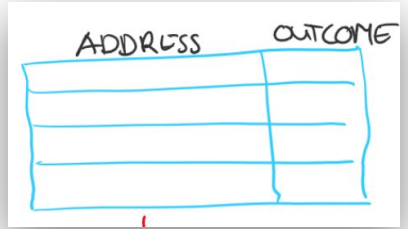
PREDICTORS

→ IN CASE OF ERROR → SQUASH RESULTS!

SPECULATNG EXECUTION

CALL *RAX

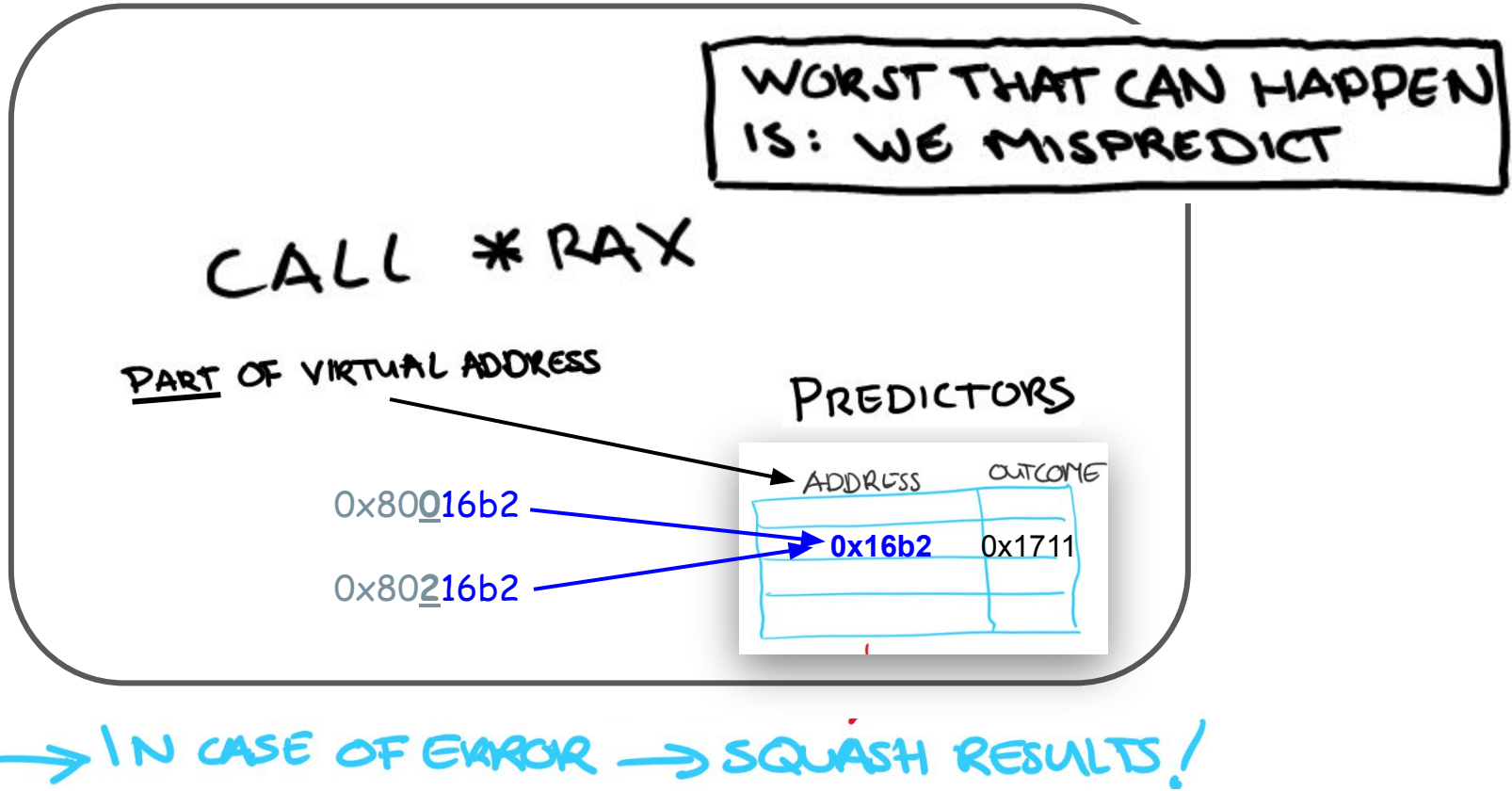
PREDICTORS



→ IN CASE OF ERROR → SQUASH RESULTS!

SPECULATNG EXECUTION

(There are yet others types of speculation also. It is speculation all the way...)



Bounds Check

```
if (index < bounds) {                    // cause mis-prediction
        data = array1[index];
        val = array2[data];
}
```

Branch to Target

```
call [reg] // cause mis-prediction to trained target
```

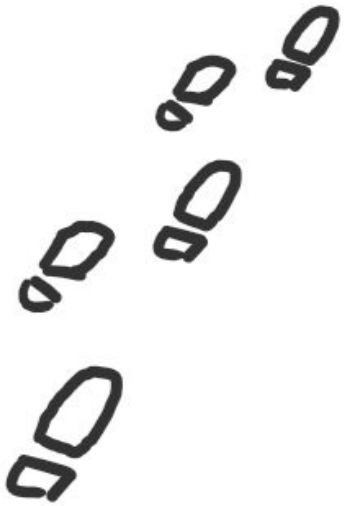
ATTACKER CAN MANIPULATE THE PREDICTORS

FOR INSTANCE

- CALL CODE WITH SMALL INDEX 100 TIMES
- INDIRECT CALL WITH CONGRUENT ADDRESS TO SOME TARGET 100 TIMES

WHAT GOOD DOES THAT DO? ALL RESULTS WILL BE SQUASHED

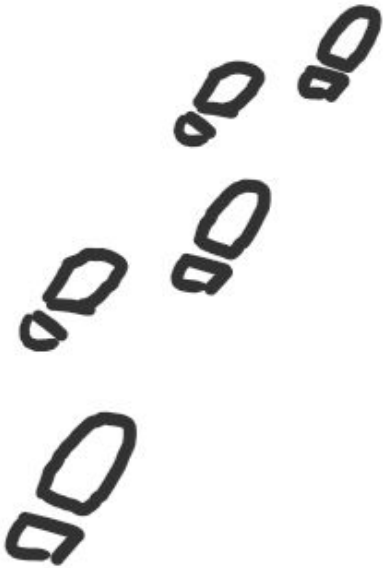
THIS IS TRUE.
BUT THERE IS STILL A TRACE!



AT THE MICROARCHITECTURAL LEVEL

Spectre V1: Bounds Check Bypass

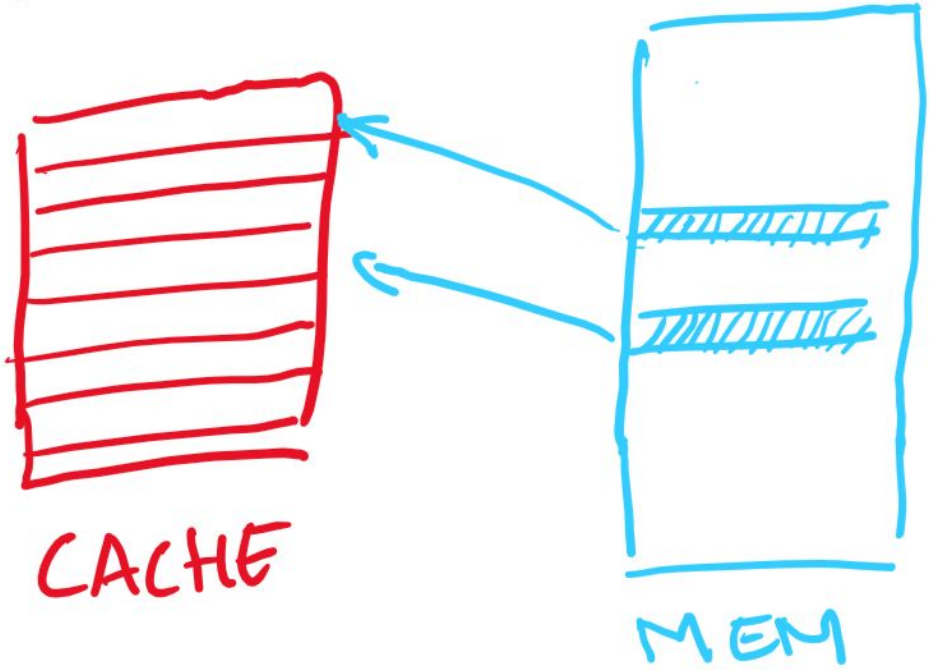
```
if (index < bounds) {                  // cause mis-prediction
        data = array1[index];          // load secret data
        val = array2[data*4096]; // leave traces in cache
}
```

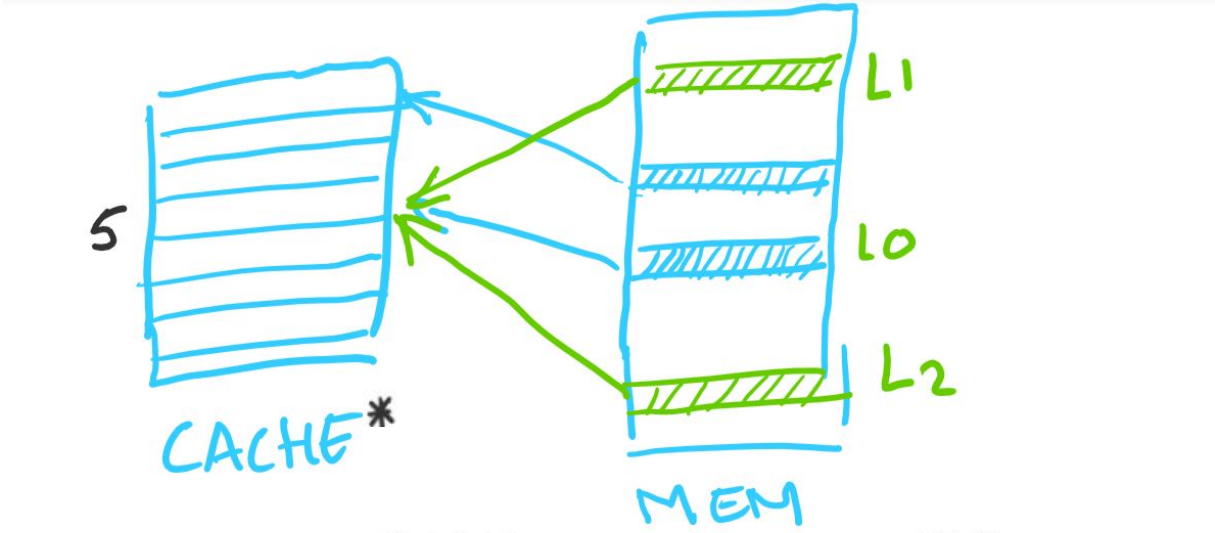


WORKS AS FOLLOWS

WHENEVER CPU ACCESSES DATA IN MEMORY, THIS DATA IS STORED IN THE CACHE FIRST → VERY FAST MEMORY

DATA STAYS THERE UNTIL THE CACHE ENTRY IS NEEDED FOR SOMETHING ELSE (THE CACHE IS VERY SMALL)





Cache is small → multiple memory locations map onto same cache entry

Consider cache line 5 above which may contain data from
L_0, L_1, L_2, L_1 (again), etc

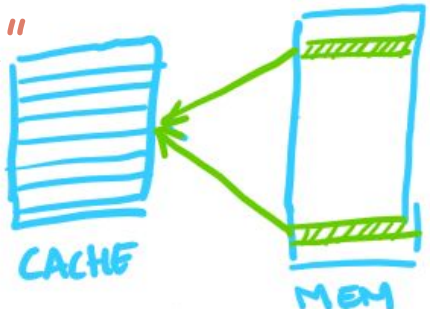
*ACTUALLY SLIGHT SIMPLIFICATION—
REAL CACHES ARE "N-WAY SET-ASSOCIATIVE"
BUT NOT IMPORTANT HERE


SO WHAT?

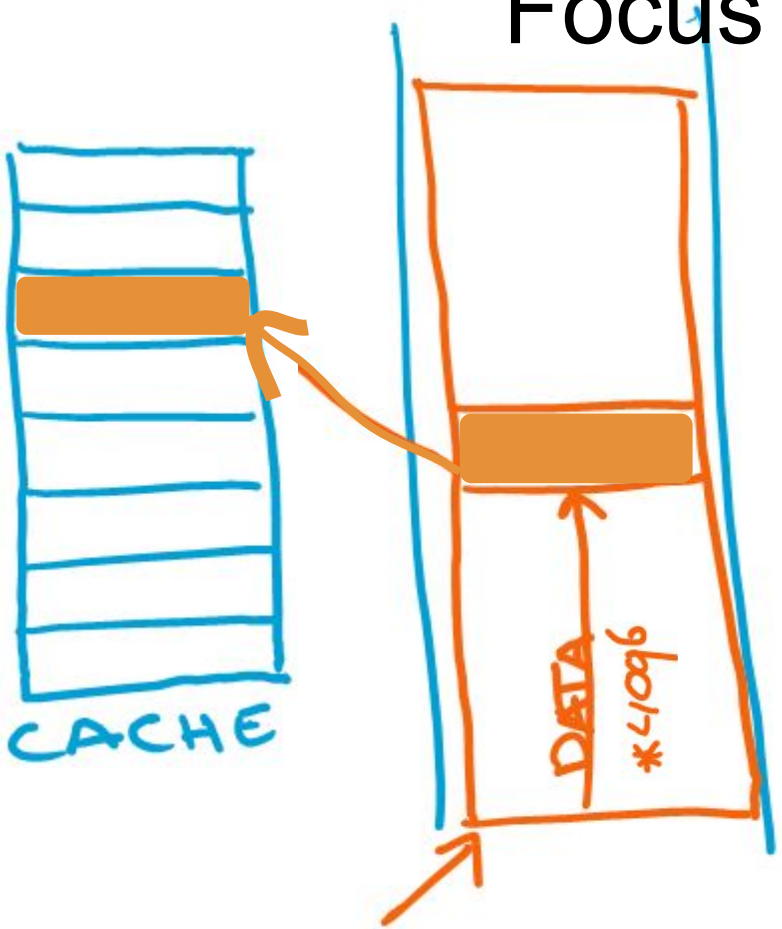
Spectre V1: Bounds Check Bypass

```
if (index < bounds) {              // cause mis-prediction
        data = array1[index];      // load secret data
        val = array2[data*4096]; // leave traces in cache
}
```

FIRST, WE MAKE SURE THAT THE "array2" TABLE IS NOT IN THE CACHE (BY ACCESSING OTHER DATA THAT MAPS ON THE SAME CACHE LINES



Focus on the cache

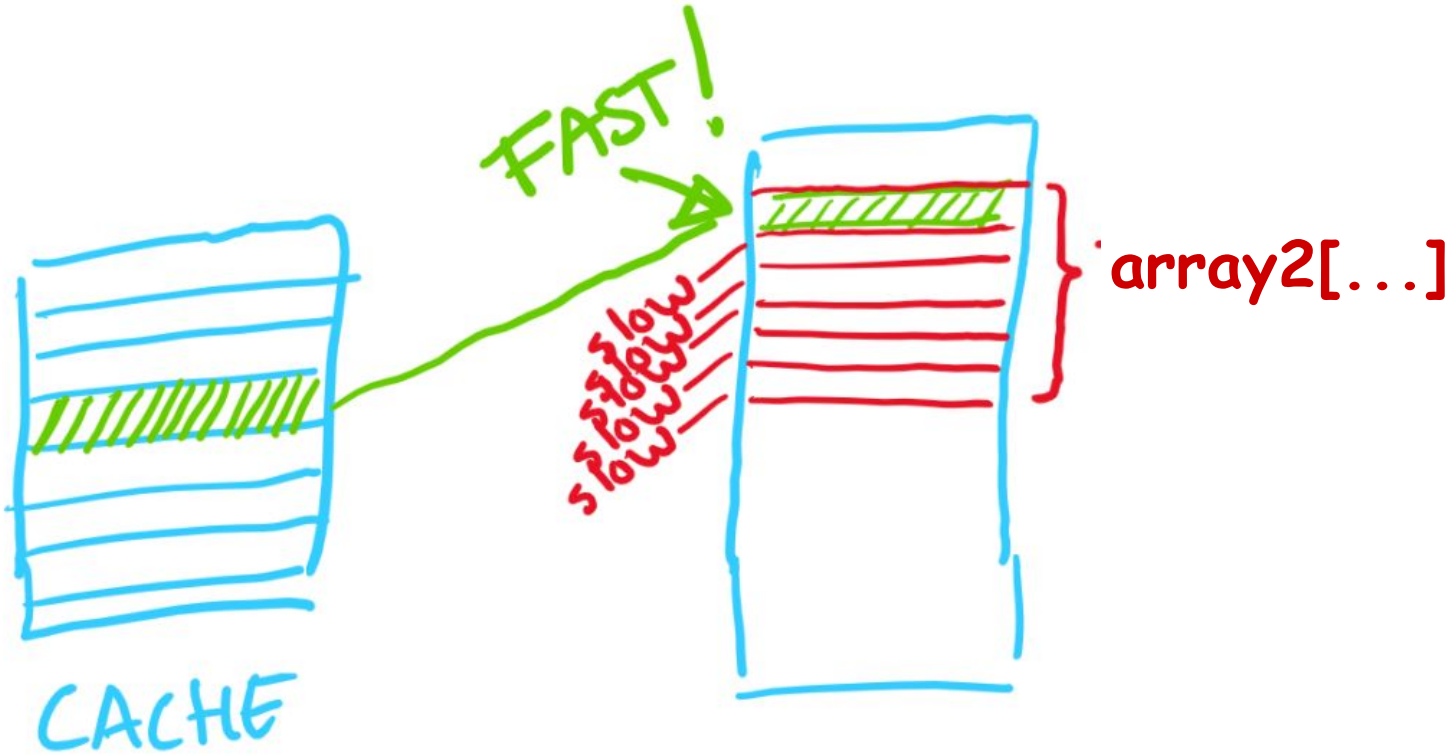


Spectre V1: Bounds Check Bypass

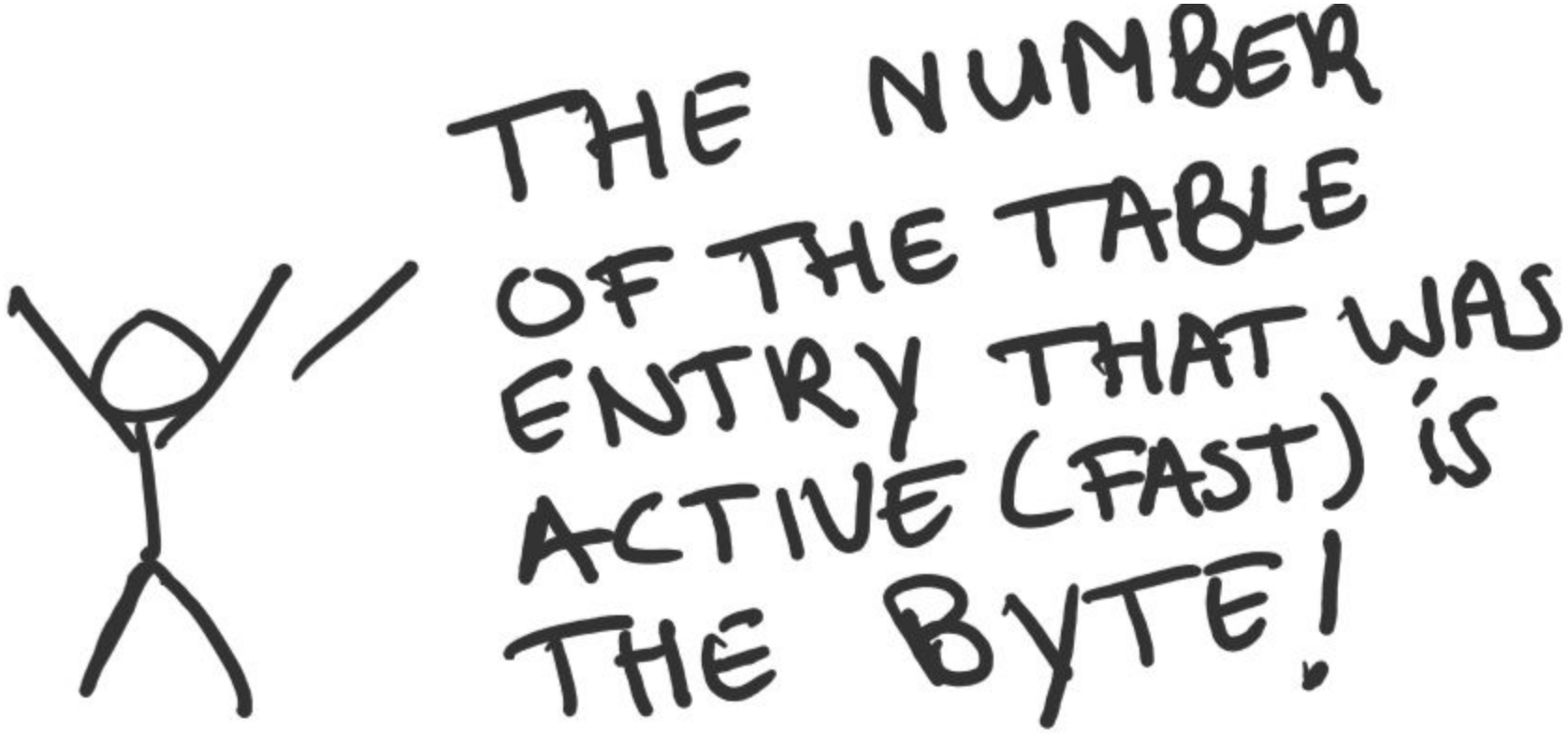
```
if (index < bounds) {                  // cause mis-predict
        data = array1[index];          // load secret data
        val = array2[data*4096]; // leave trace in cache
}
```

while assignments to data and val are squashed, data in array2[data*4096] stil cached

(the array is called a “probe array”)



SO WE TIME THE ACCESSES
IF (<FAST) → THIS DATA WAS
ACCESSED BEFORE


THE NUMBER
OF THE TABLE
ENTRY THAT WAS
ACTIVE (FAST) IS
THE BYTE!

Transient Execution

Reasons? There are many reasons

Control speculation / prediction
Data speculation / prediction
Likely invariant violations
Exceptions
…

```
1    data = *kernel_addr
2    val = array [data * 4096]
```

→ not allowed → fault

→ before fault is architecturally visible, result is used in transient execution

Meltdown/Spectre were not a one-off

New dangerous vulnerabilities have emerged

New variants of Spectre, but also:

L1TF / Foreshadow

MDS / RIDL

CrossTalk

The problem appears to be systemic

```
[sebastian@sarek ridl ]$
```

Conclusions

Implications are still not clear

Limited knowledge of issues (→ 

)

Mitigations are limited (→ performance+mistakes → see mdsattacks.com, S&P'19)

Security by obscurity and exploit Whac-A-Mole (→ see mdsattacks.com)

Often not considering combined threat models (→ BlindSide [CCS'20])