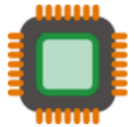


CASPER

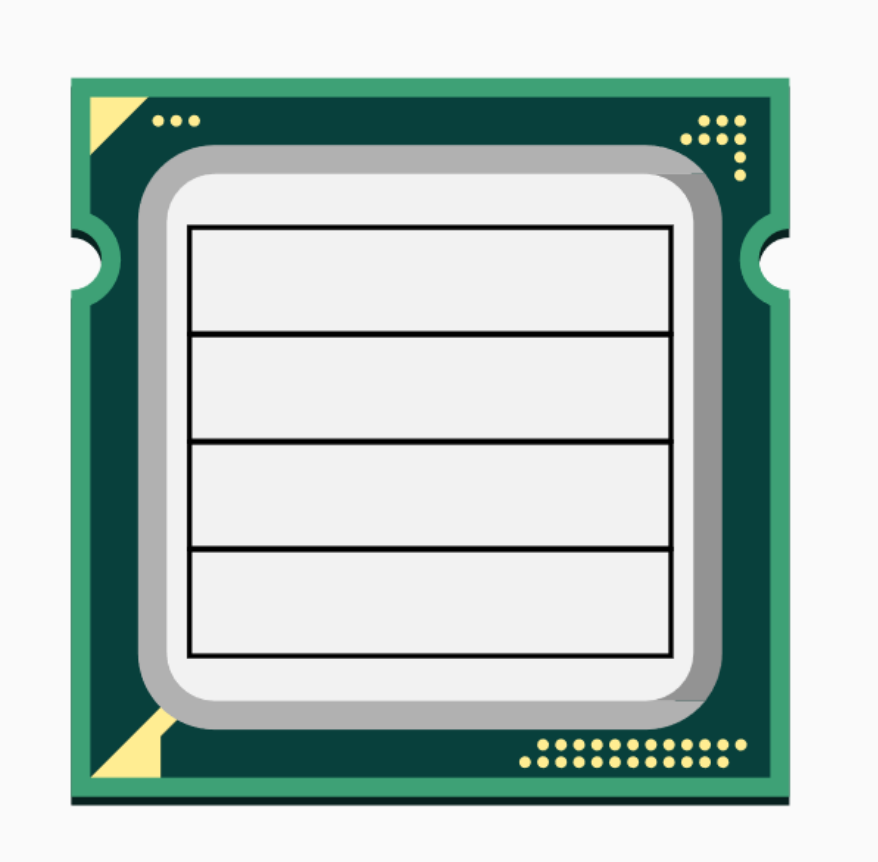
CS305: Computer Architecture

Spectre and Meltdown

https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

Microarchitecture



Programmer

ISA

Architecture

Registers

Caches, TLBs,

Branch predictors,

Prefetchers, Interconnect,

Out-of-order execution

Not exposed to programmer

Microarchitecture

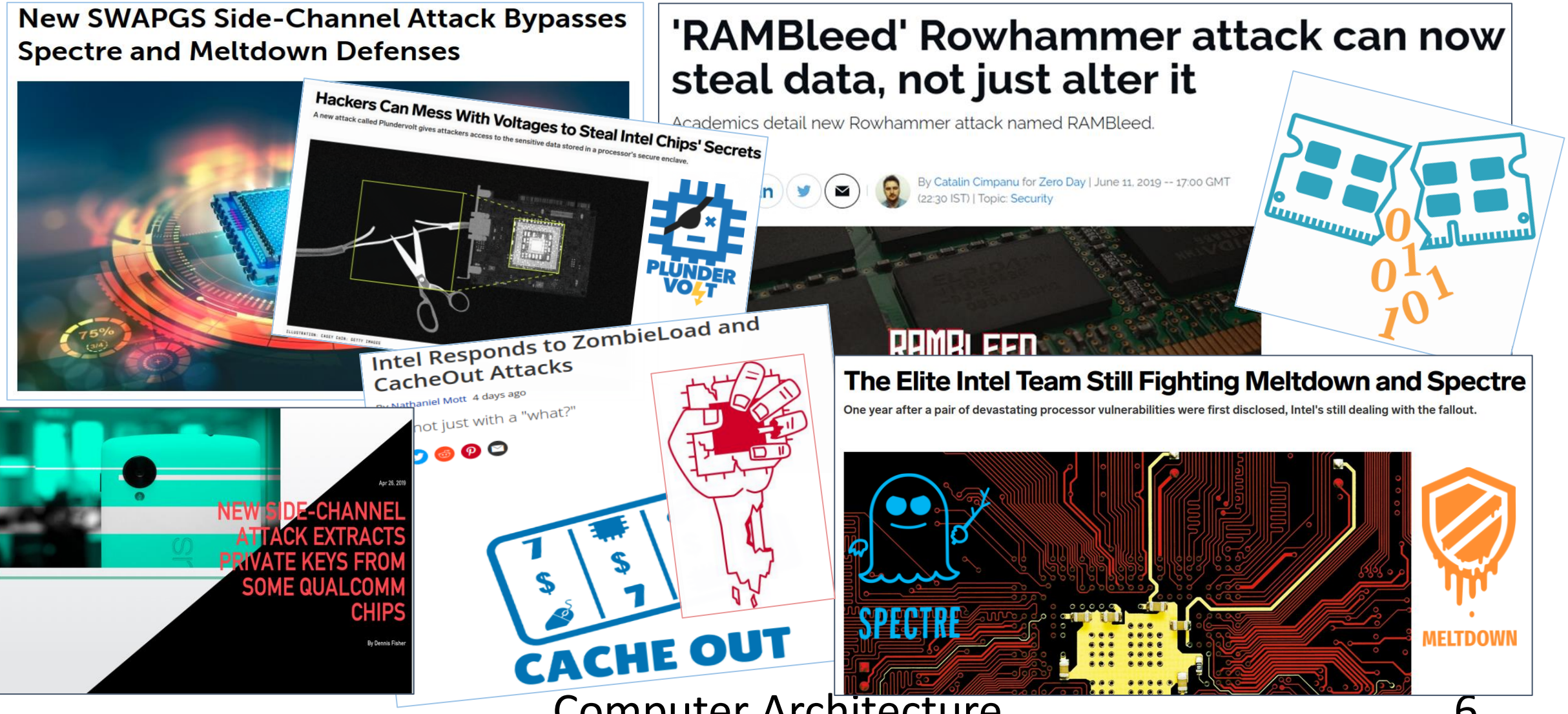
Not exposed to programmer

Security: A bit Subtle

Confidentiality	*You do not **see (READ)** what you are not supposed to see*
Integrity	*You do not **change (WRITE)** what you are not supposed to see*
Availability	*You do not **affect (DELAY)** others (un)intentionally*

Attacks Inside

In News


New SWAPGS Side-Channel Attack Bypasses Spectre and Meltdown Defenses
Hackers Can Mess With Voltages to Steal Intel Chips' Secrets
A new attack called Plundervolt gives attackers access to the sensitive data stored in a processor's secure enclave.
PLUNDER VOLT
'RAMBleed' Rowhammer attack can now steal data, not just alter it
Academics detail new Rowhammer attack named RAMBleed.
By Catalin Cimpanu for Zero Day | June 11, 2019 -- 17:00 GMT (22:30 IST) | Topic: Security
RAMBLEED
0 1 0 1 1 0
Intel Responds to ZombieLoad and CacheOut Attacks
Nathaniel Mott 4 days ago
not just with a "what?"
CACHE OUT
NEW SIDE-CHANNEL ATTACK EXTRACTS PRIVATE KEYS FROM SOME QUALCOMM CHIPS
Apr 26, 2019
By Dennis Fisher
The Elite Intel Team Still Fighting Meltdown and Spectre
One year after a pair of devastating processor vulnerabilities were first disclosed, Intel's still dealing with the fallout.
SPECTRE
MELTDOWN

Information Leakage: 101

$x \longleftarrow 1$

for $i \longleftarrow |e|\text{-}1$ **downto** 0 do

  $x \longleftarrow x^2 \bmod n$

  if $(e_i = 1)$ **then**

    $x = xb \bmod n$

  endif

done

return x

Modular exponentiation, b^e mod n

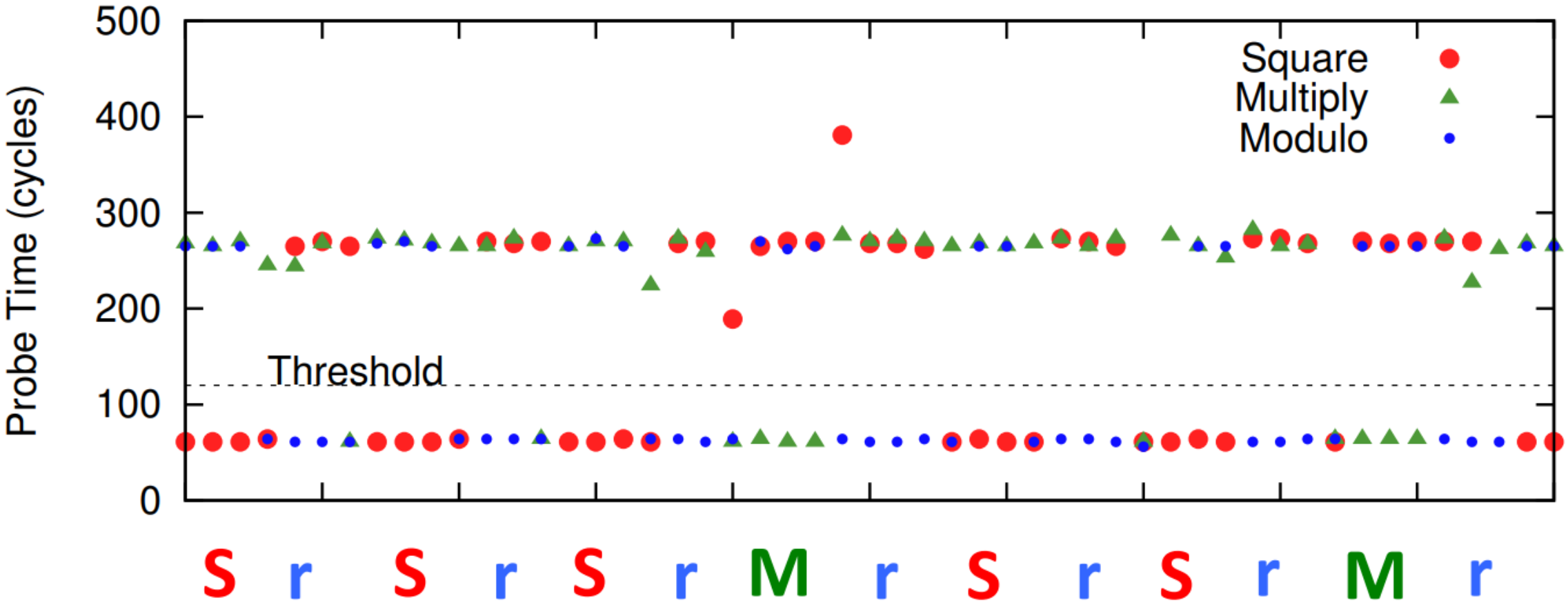
Exponent e is used for decryption

$e_i = 0$, *Square Reduce (SR)*

$e_i = 1$, *SRMR*

Attacker tries to get the e

Timing Channel



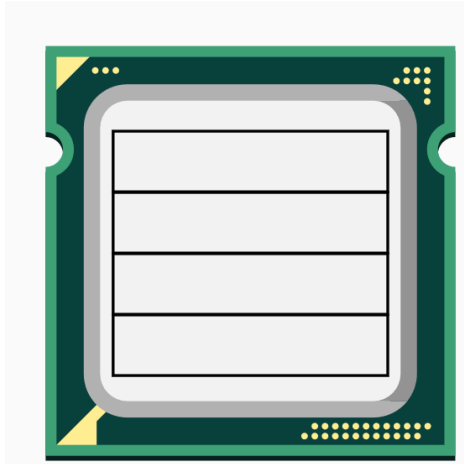
Toy Example of side-channel attacks

```
If  secret=1  do
    access(&a)
else // secret=0
    no-access
```

Victim

```
    flush(&a)
t1=start_timer
    access(&a)
t2=end_timer
```

Attacker

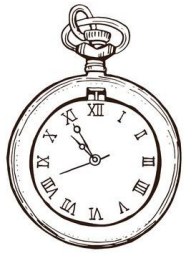


Fast – 1

Slow – 0

Cache Attacks

Attacks at the LLC exploit timing channels:
LLC miss > LLC hit



Flush + Reload

Evict + Reload

Prime + Probe

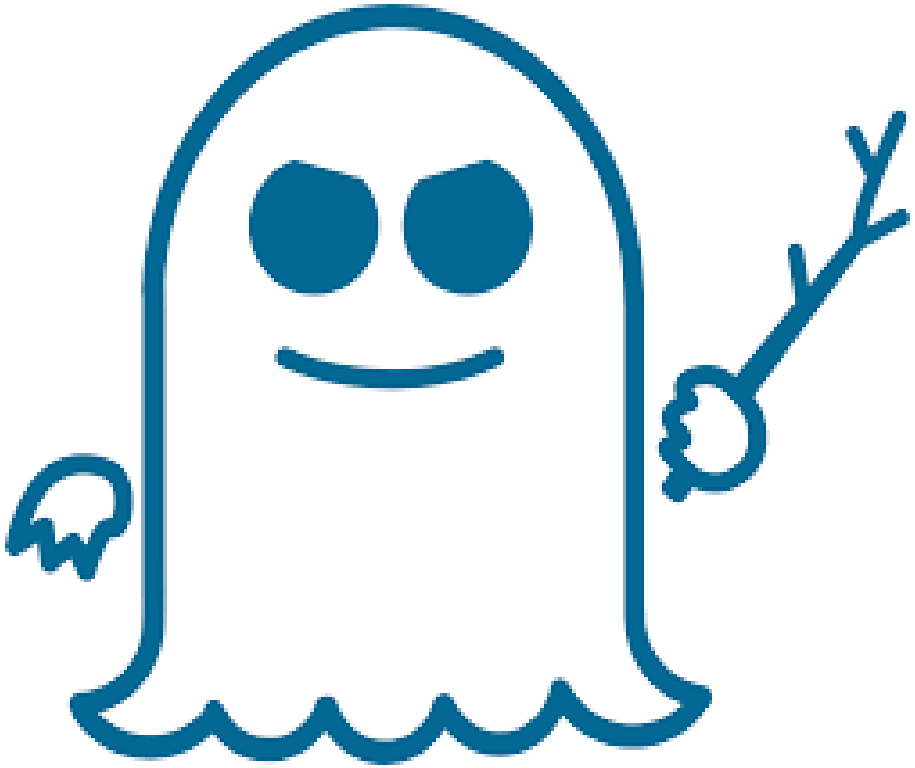
clflush

Eviction based attacks

Threat

Knowing the victim *has accessed a cache set (line)* can be considered as a *successful* attack

Spectre and Meltdown

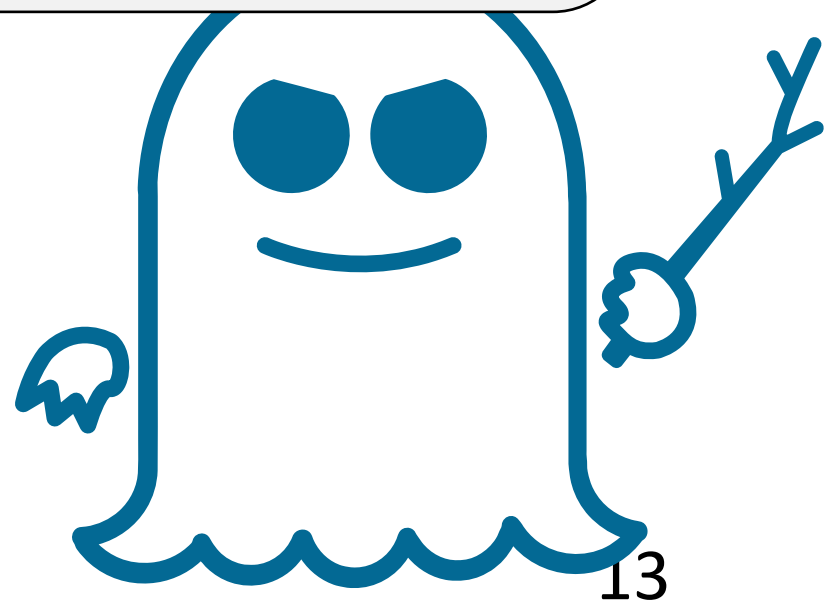


Spectre in Action

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

DRAM LOAD

DRAM LOAD



Branch Predictor and Speculative Execution

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Branch Predictor and Speculative Execution

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

Branch predictor returns TRUE ☹

T T T T T T T T T T T T

Attacker has mis-trained it ☹ ☹

How? By using values less than 3 always ☹ ☹

Branch Predictor and Speculative Execution

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch Predictor and Speculative Execution

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Branch resolution latency 200 cycles ☹ ☹ ☹ ☹

Branch Predictor and Speculative Execution

```
int CS305Array = [100, 200, 300];
int attacker = 4;
if (attacker < sizeof(CS305Array))
        y = MyArray[CS305Array[attacker]*512]
```

Branch predictor returns TRUE ☹

Attacker has mis-trained it ☹ ☹

Processor is on the wrong-path ☹ ☹ ☹

Yes, you guessed it right: cache attacks ☹ ☹ ☹

Next

Processor realized it was a mistake and *flushed* all wrong path instructions

But cache has the data ☹

y = MyArray[CS305Array[attacker]*512]

LOAD MyArray[0] 60 ns
LOAD MyArray[512] 60 ns
LOAD MyArray[1024] 5 ns Bingo !! *CS305Array[attacker] = 2*

Meltdown: The O3 curse!

1. raise_exception();
2. // line below is never reached
3. secret=KernelArray[data*4096];

Kernel Trap

Out-of-order (O3) as
it has no dependency

1. secret=KernelArray[data*4096];
2. raise_exception();

What about page-fault?

For more:

CS773: Computer Architecture for Performance and Security

January 2022

Go raibh maith agat