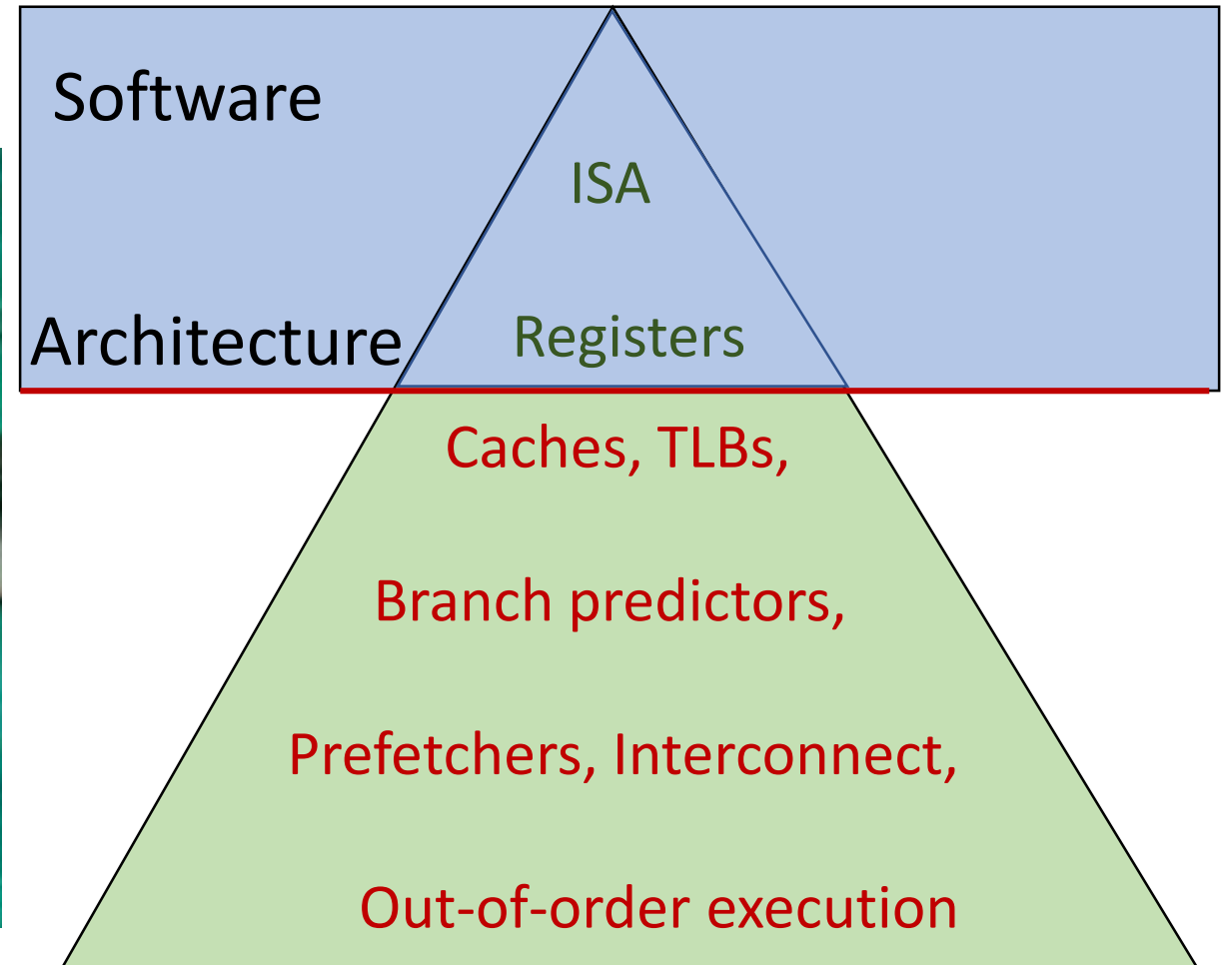




CS230-Wrapup

From Performance to Security: 10K Feet View



Attacker



O3 processor ka, cache ka, prefetcher ka, coherence protocol ka, DRAM ka, sabka khabar lega ☹️

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



inside™



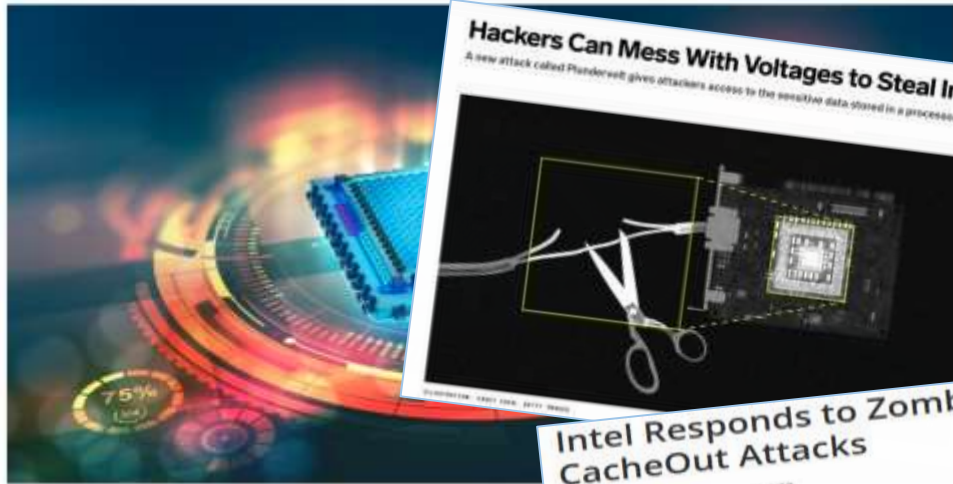
inside™



inside™

Media Articles ☹️

New SWAPGS Side-Channel Attack Bypasses Spectre and Meltdown Defenses



Hackers Can Mess With Voltages to Steal Intel Chips' Secrets

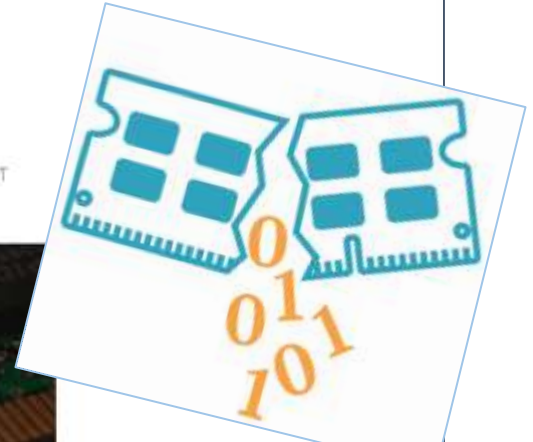
A new attack called PlunderVolt gives attackers access to the sensitive data stored in a processor's secure enclave.



'RAMBleed' Rowhammer attack can now steal data, not just alter it

Academics detail new Rowhammer attack named RAMBleed.

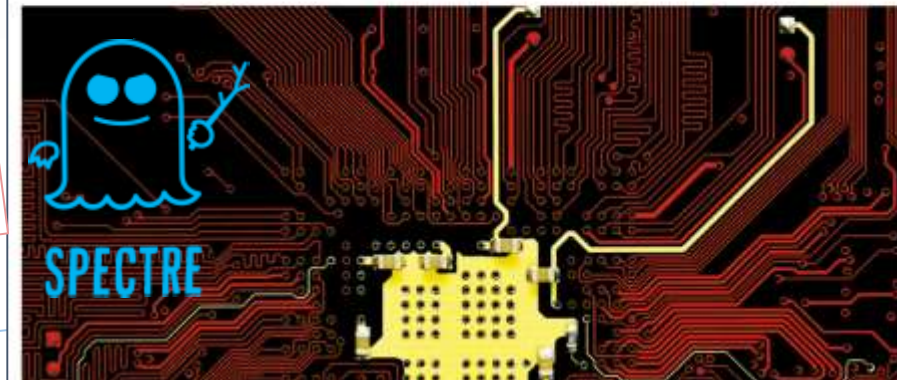
By Catalin Cimpanu for Zero Day | June 11, 2019 -- 17:00 GMT (12:30 IST) | Topic: Security



RAMBLEED

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.



NEW SIDE-CHANNEL
ATTACK EXTRACTS
PRIVATE KEYS FROM
SOME QUALCOMM
CHIPS

By Dennis Fisher

Intel Responds to ZombieLoad and CacheOut Attacks

by Nathaniel Mott 4 days ago

not just with a "what?"



Brushing-up: Information Leakage

$x \leftarrow 1$

Modular exponentiation, $b^e \bmod n$

for $i \leftarrow |e|-1$ **downto** 0 **do**

Exponent e is used for decryption

$x \leftarrow x^2 \bmod n$

square

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

reduce

$x = xb \bmod n$

endif

multiply

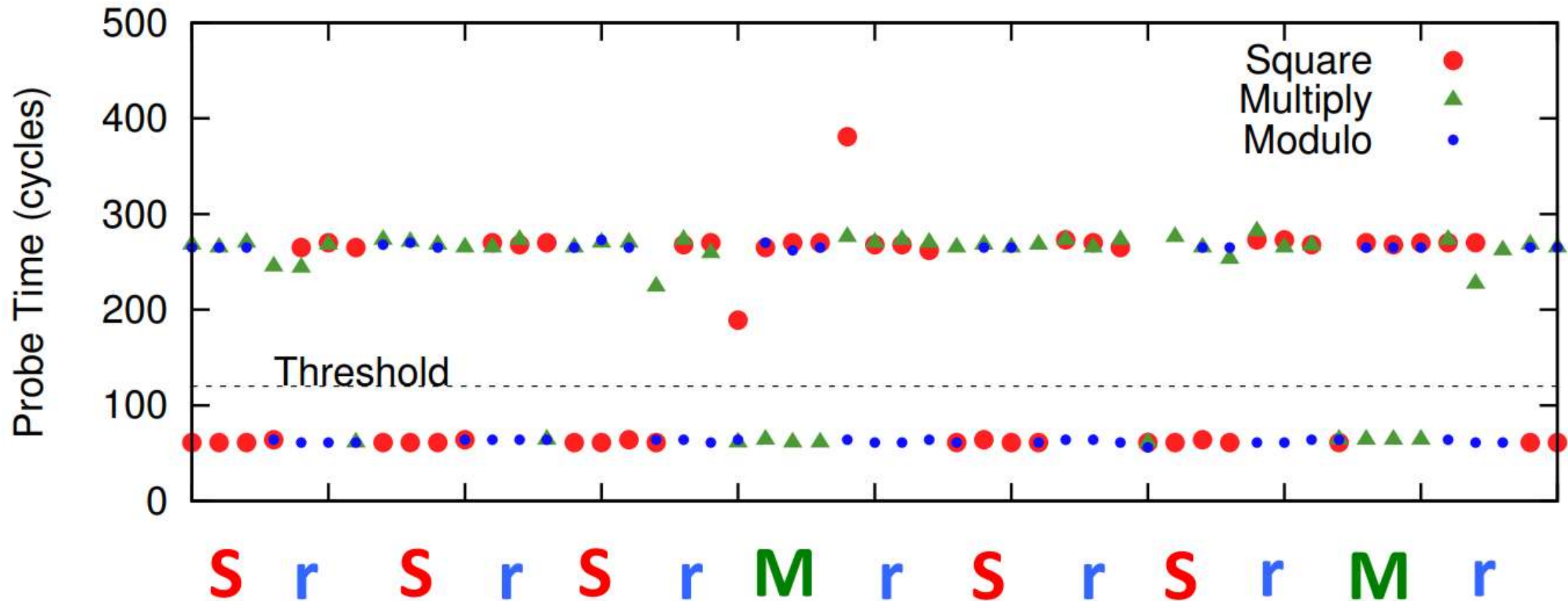
done

return x

$e_i = 0$, Square Reduce (SR)
 $e_i = 1$, SRMR

Attacker tries to get the e

Timing Channel



Adpated from flush+reload attack [Usenix Security '14]

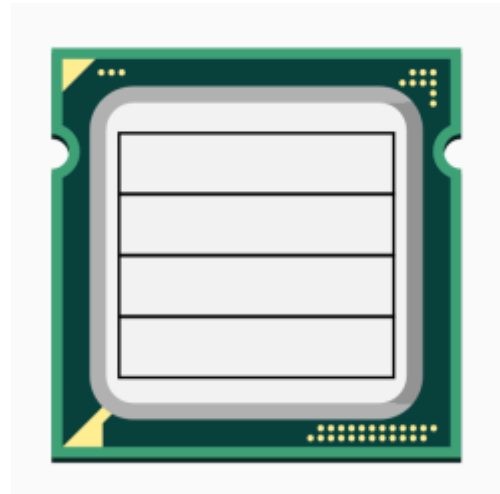
Toy Example: Flush Based Attacks

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

Victim

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

Attacker

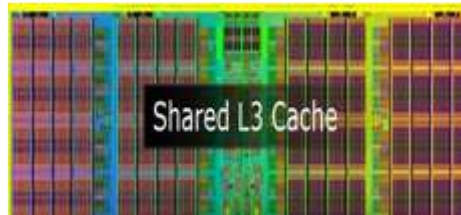


Fast – 1

Slow – 0

Side and Covert Channels

Spy



Side-channel attacks

Victim



Let's
play



Covert-channel attacks

Oh Yes!!

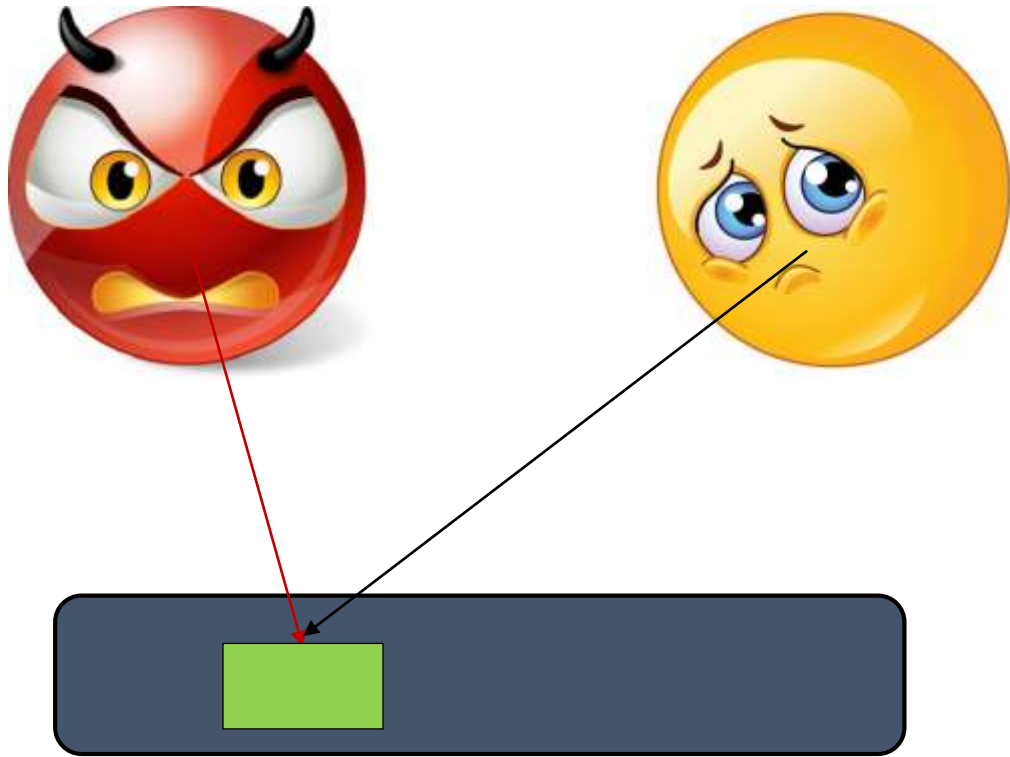




Pause



Flush+Reload Attack



Step 0: Spy *maps* the shared library, shared in the cache



Flush+Reload Attack



Cflush



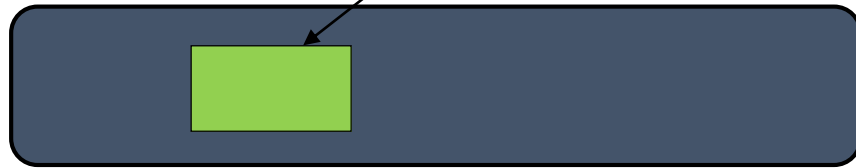
LLC

Step 0: Spy *maps* the shared library, shared in the cache

Step 1: Spy *flushes* the cache block



Flush+Reload Attack



LLC

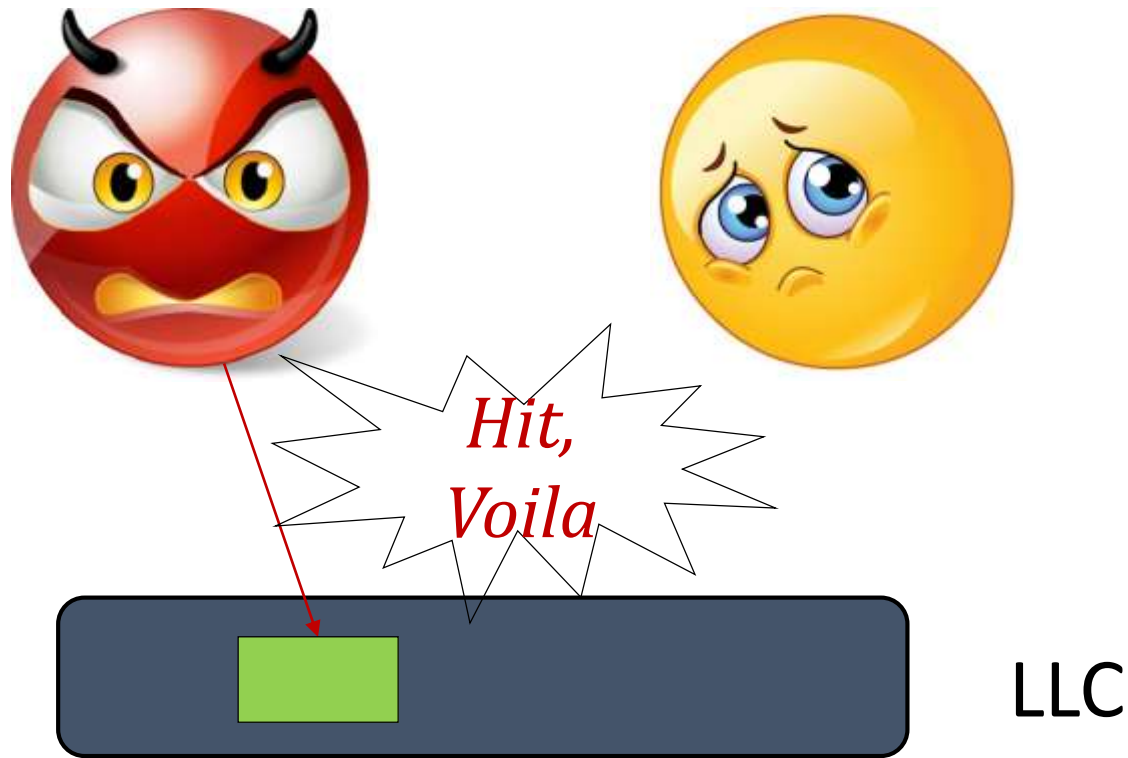
Step 0: Spy *maps* the shared library, shared in the cache

Step 1: Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block



Flush+Reload Attack



Step 0: Spy *maps* the shared library, shared in the cache

Step 1: Spy *flushes* the cache block

Step 2: Victim *reloads* the cache block

Step 3: Spy *reloads* the cache block (hit/miss)





Pause

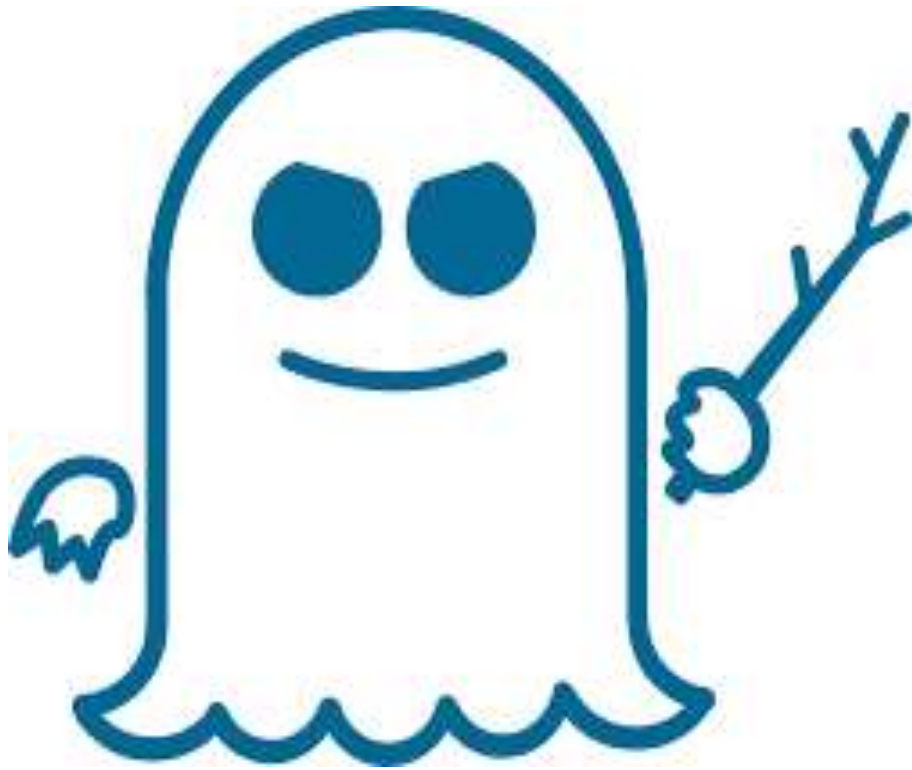


How Practical?



Future is uncertain, if we do not take care of present attacks, future may be worse ☹️

Spectre and Meltdown

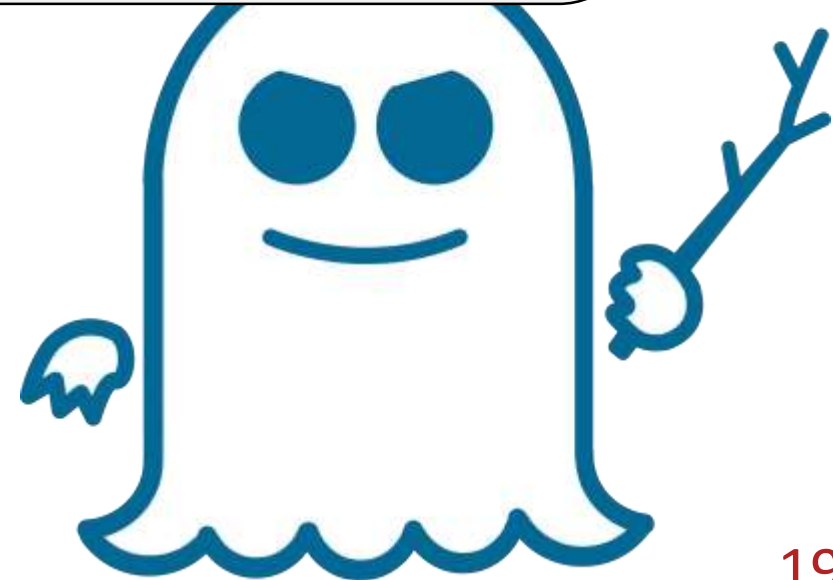


Spectre in Action: Fasten Your Seat Belts

```
int CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```

DRAM LOAD

DRAM LOAD



Branch Predictor and Speculative Execution

```
int CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```



Branch predictor returns TRUE ☹️

Branch Predictor and Speculative Execution

```
int CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```



Branch predictor returns TRUE ☹️

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 CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```

Branch predictor returns TRUE ☹️

Attacker has mis-trained it ☹️ ☹️

Processor is on the wrong-path ☹️ ☹️ ☹️

Branch Predictor and Speculative Execution

```
int CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```

Branch predictor returns TRUE ☹️

Attacker has mis-trained it ☹️ ☹️

Processor is on the wrong-path ☹️ ☹️ ☹️

Branch resolution latency 200 cycles ☹️ ☹️ ☹️ ☹️

Within these 200 cycles 😊

```
int CS230Array = [100, 200, 300];  
int attacker = 4;  
if (attacker < sizeof(CS230Array))  
    y = MyArray[CS230Array[attacker]*512]
```

CS230Array[4] is in L1/L2/L3 ☹️

The address is in the cache ☹️ ☹️

Yes, you guessed it right: F+R, P+P cache attacks ☹️ ☹️ ☹️

Picture Abhi Baki Hai 😊 After 200 cycles

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

But cache has the data 😞

*y = MyArray[CS230Array[attacker]*512]*

LOAD MyArray[0] 60 ns

LOAD MyArray[512] 60 ns

LOAD MyArray[1024] 5 ns Bingo !! CS230Array[attacker] = 2



Meltdown: The O3 Curse!!

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

Kernel Trap

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

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

What about page-fault?

Summary:
CS230+CS231

(Connecting the dots
on the board)

