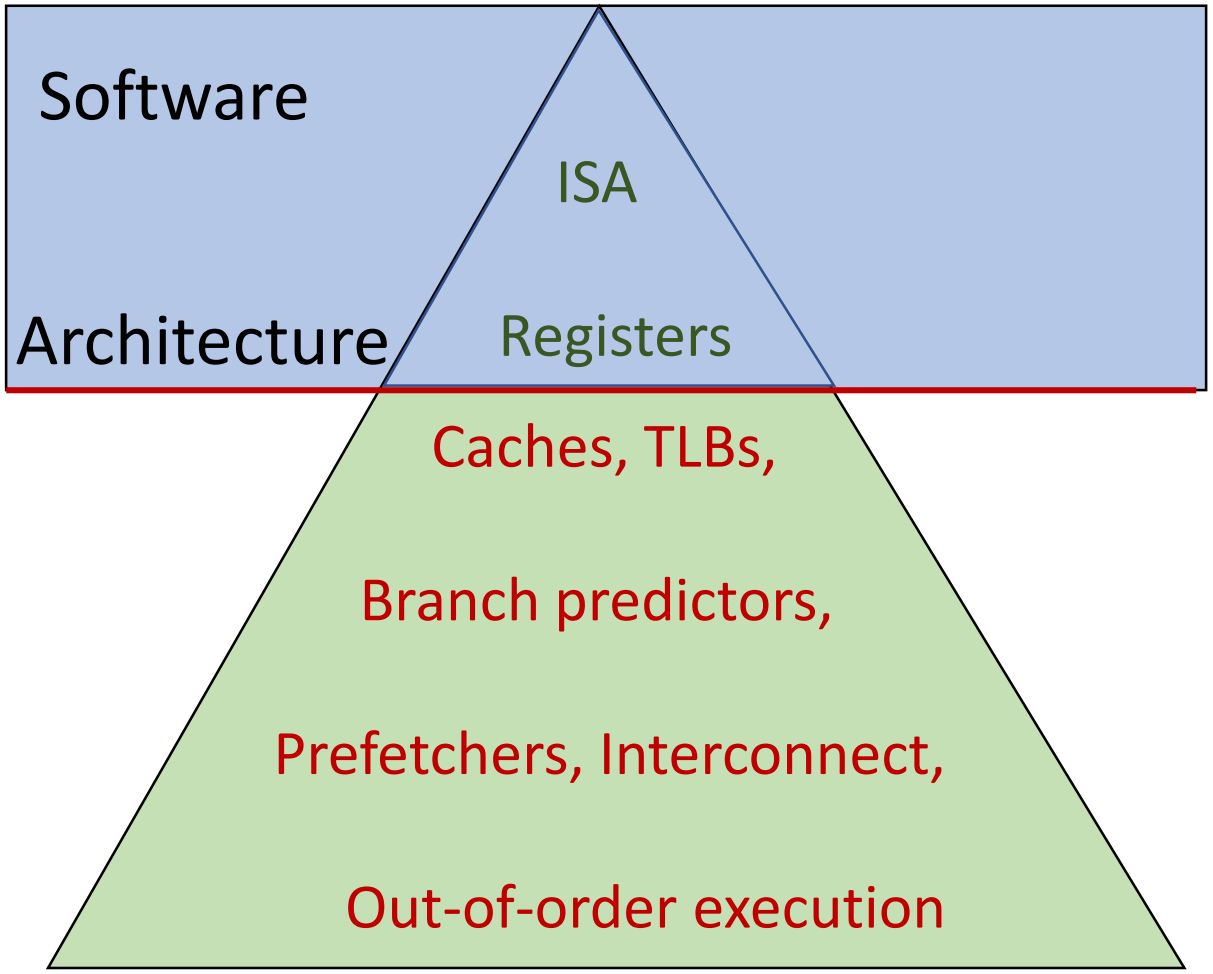
# CS230-Wrapup

# From Performance to Security: 10K Feet View



# Attacker

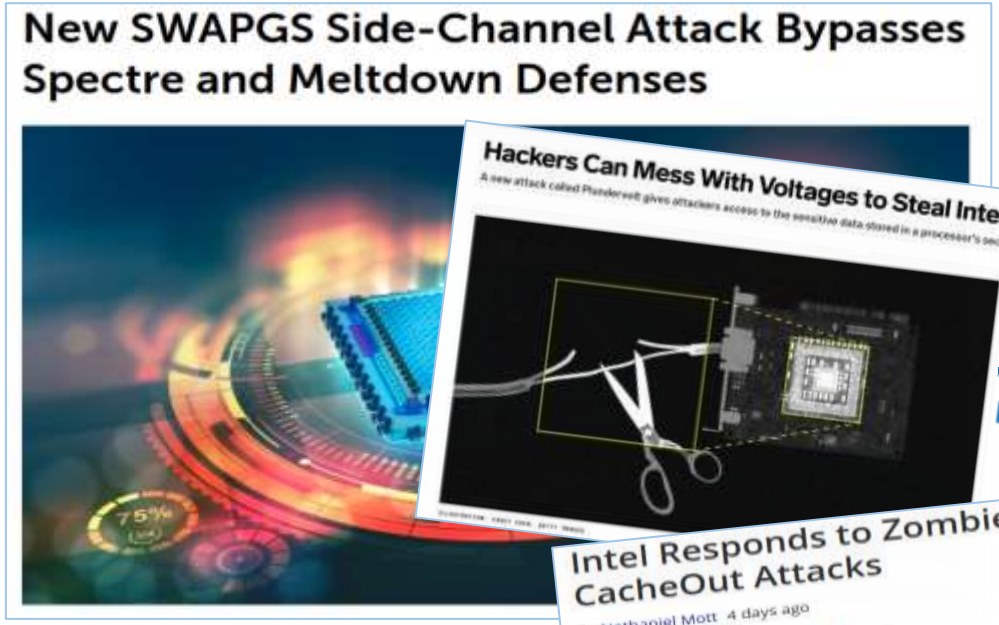

O3 processor ka, cache ka, prefetcher ka, coherence protocol ka, DRAM ka, sabka khabar lega ☹

# 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

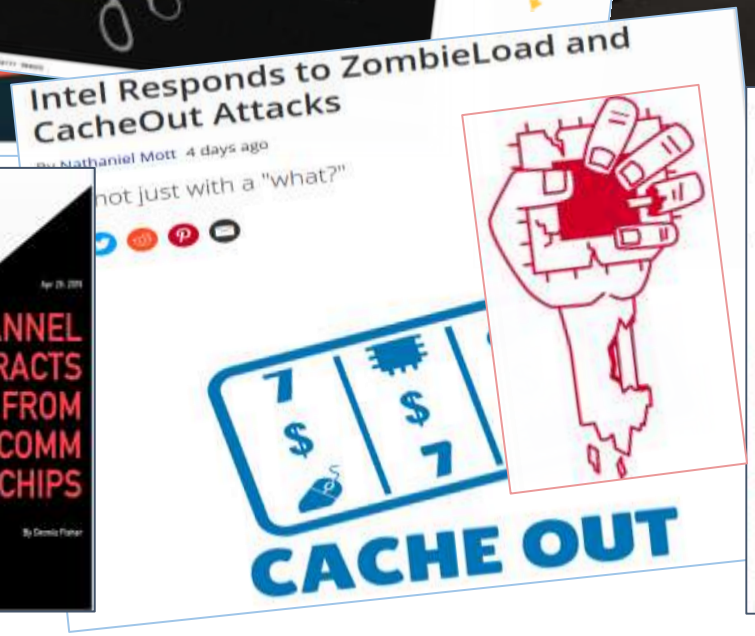
# Media Articles ☹



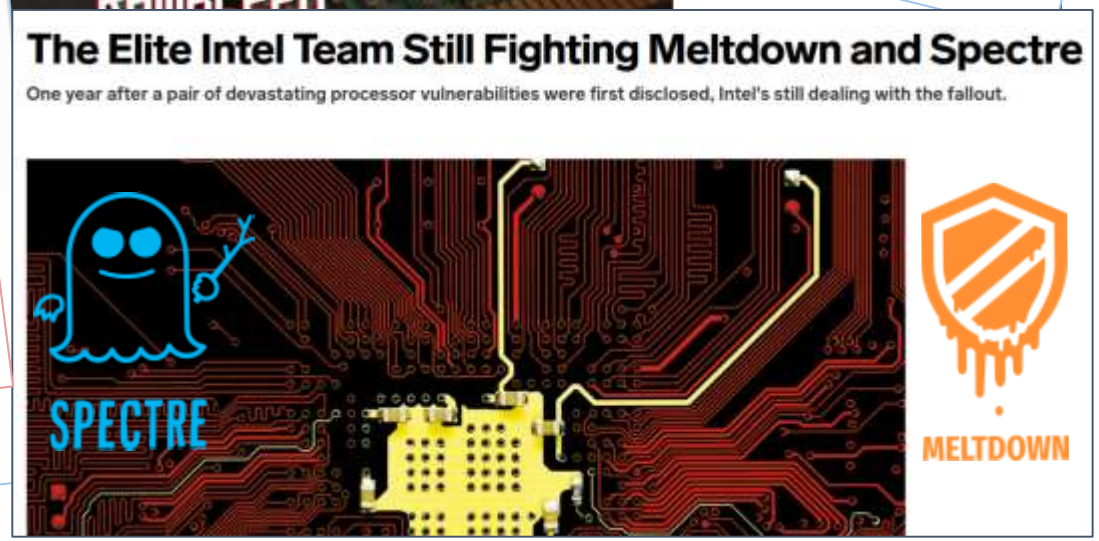










# Brushing-up: Information Leakage

$x \leftarrow 1$

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

$x \leftarrow x^2 \bmod n$

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

$x = xb \bmod n$

**endif**

**done**

**return** $x$

*square*

*reduce*

*multiply*

Modular exponentiation, $b^e$ mod n

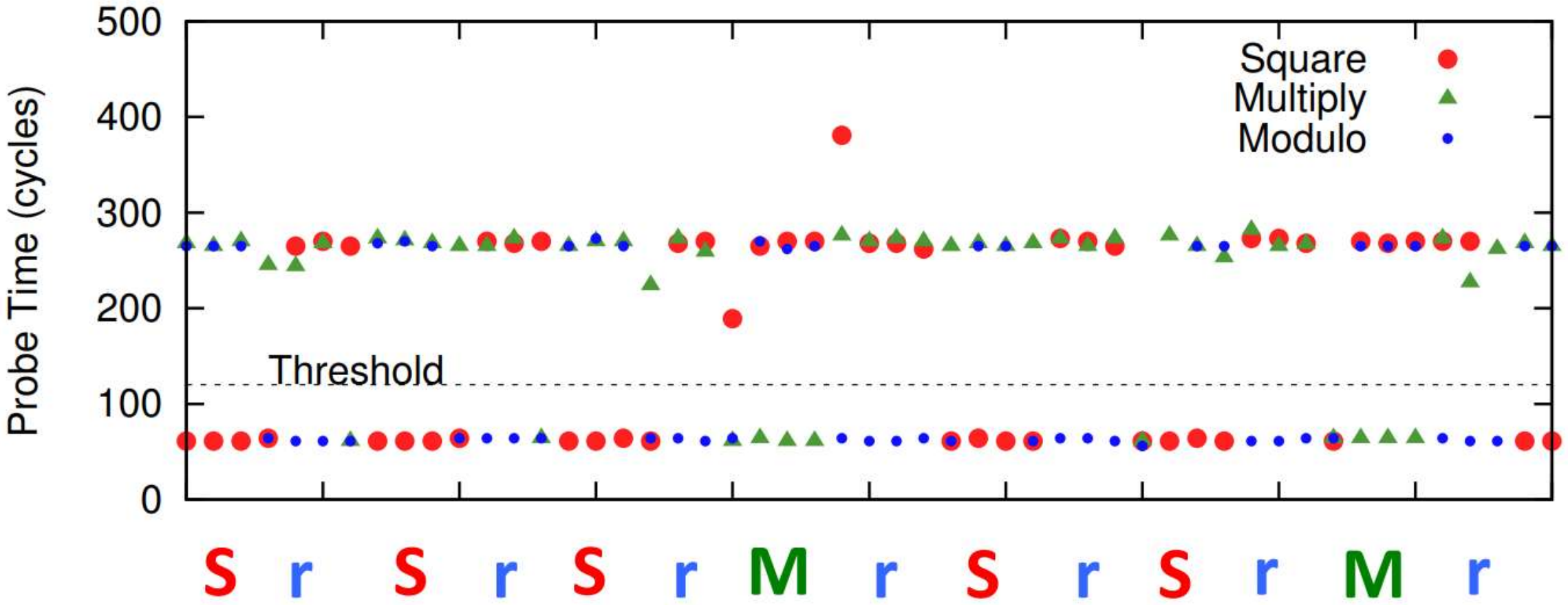
Exponent $e$ is used for decryption

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

$e_i = 1$, *SRMR*

*Attacker tries to get the e*

# Timing Channel




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

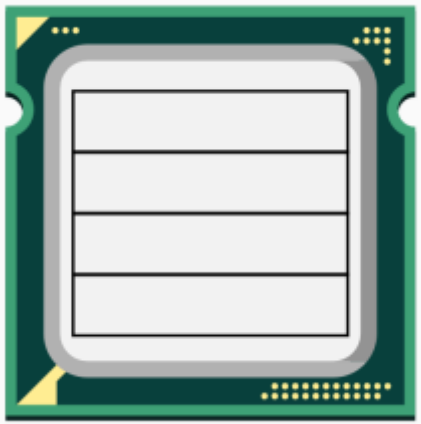
# Toy Example: Flush Based Attacks

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

Victim

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

Attacker

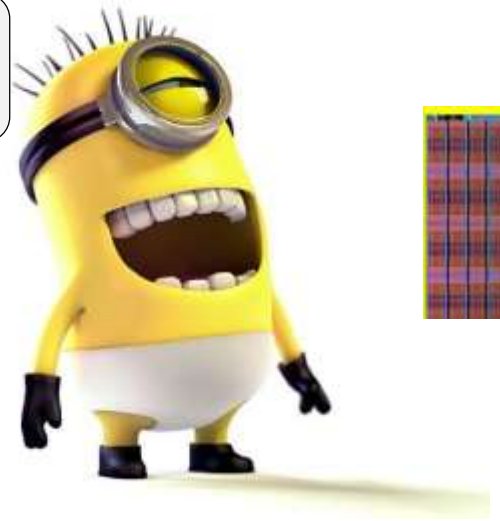


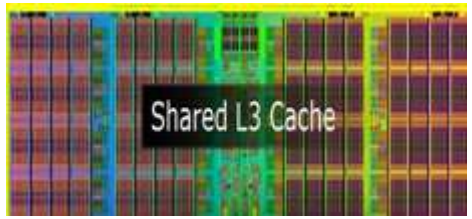
Fast – 1

Slow – 0

# Side and Covert Channels

Spy

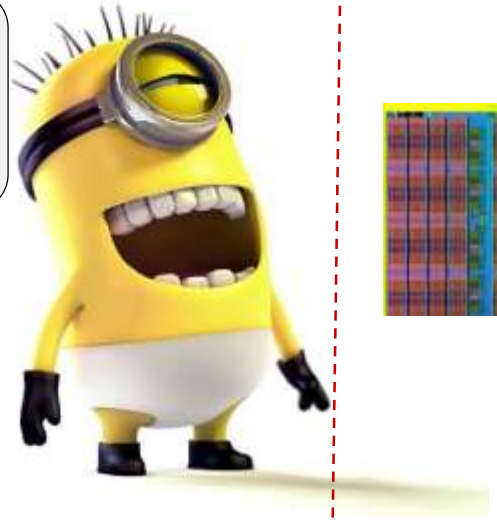


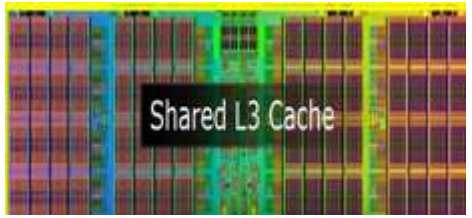


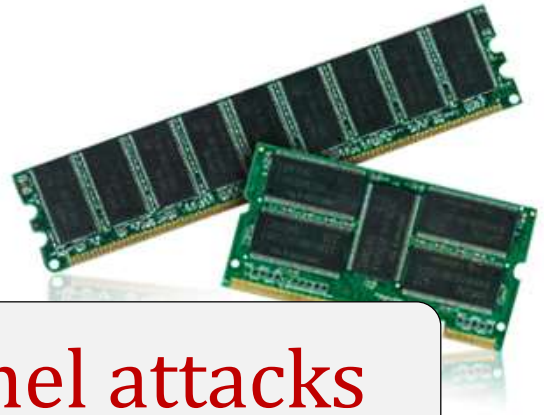

Victim

Side-channel attacks

Let's play

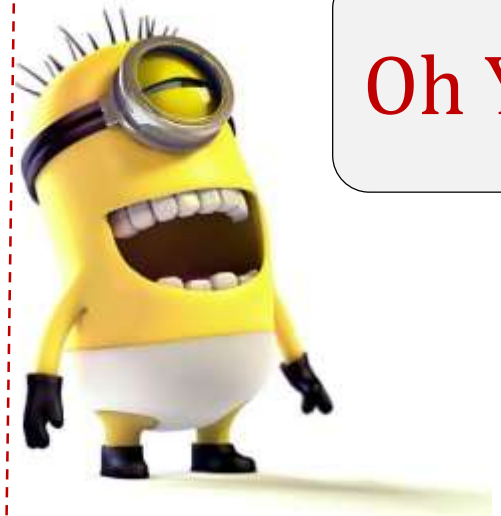




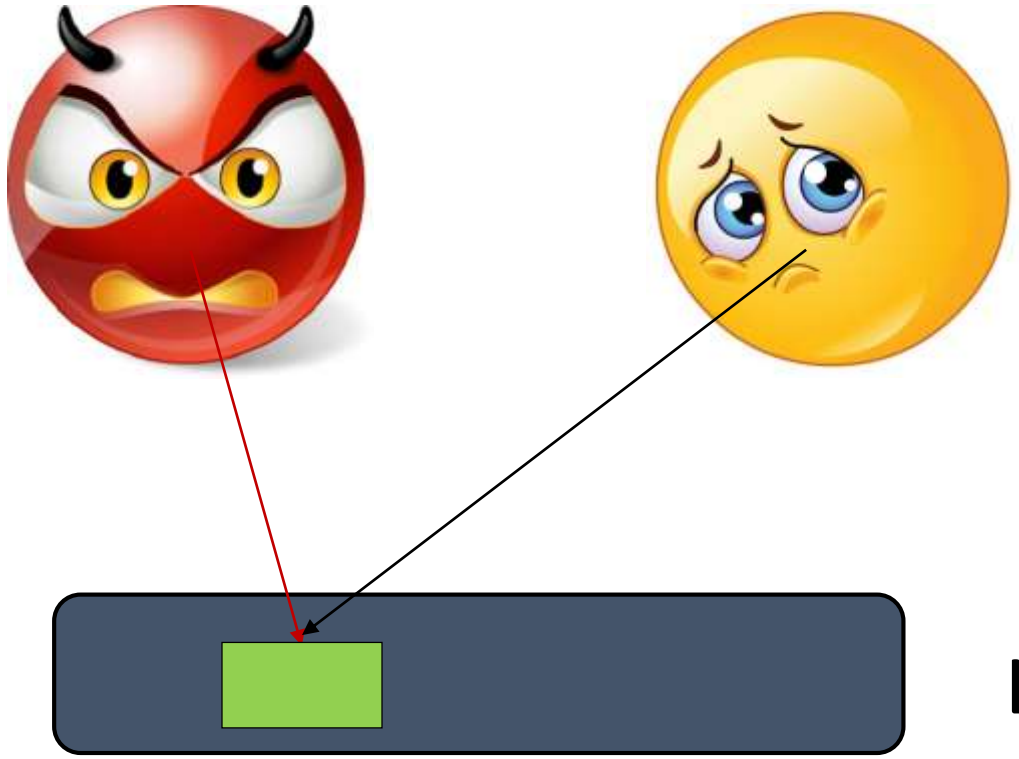
Covert-channel attacks

Oh Yes!!



# Pause

# Flush+Reload Attack




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

# Flush+Reload Attack

*Clflush*

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 ☹*

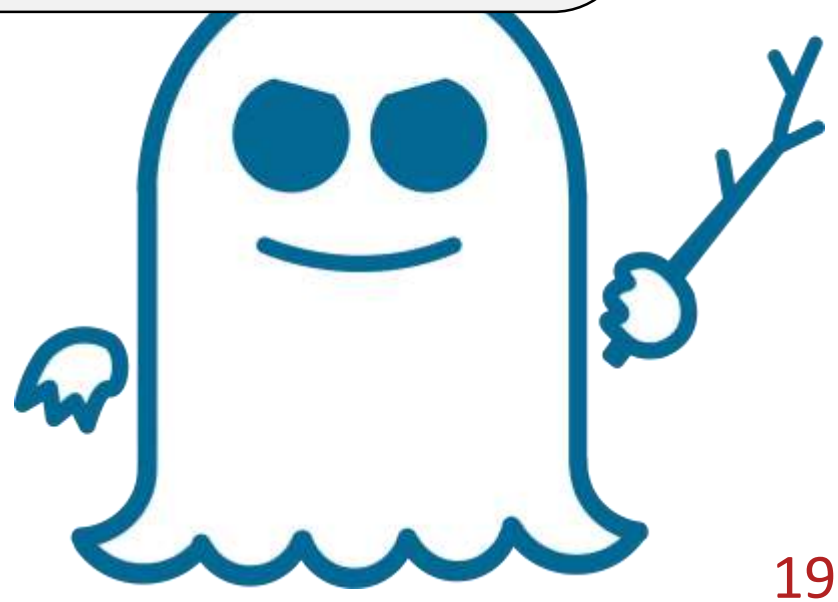
# Spectre and Meltdown

# 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 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

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

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

What about page-fault?

# Summary: CS230+CS231

(Connecting the dots on the board)