AN OVERVIEW ON HARDWARE ATTACKS

Dr. Tara Ghasempouri
Computer Systems
Tallinn University of Technology

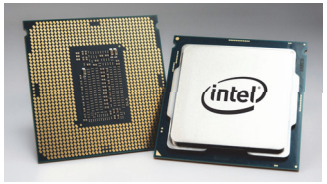
12.11.2021

MY BACKGROUND

- 2019-present, Senior researcher, Computer Systems, Taltech.
  - Topic of interest
    - Hardware security
    - Cyber security
- 2017-2019, Postdoc, Computer Systems, Taltech.
  - Topic of interest
    - Assertion based verification
    - Hardware testing
- 2013-2016, PhD, University of Verona, Italy
  - Topic of interest
    - Assertion based verification
      - Automatic assertion mining

WHY HARDWARE SECURITY IS NEEDED?

- Real-world hardware is vulnerable



- Early in 2020, security researchers warned of a security flaw found within certain Intel processors that allowed hackers to install malware at the hardware level, thus rendering OS-based malware protection ineffective.

- A hardware vulnerability that was recently identified in Comcast's intelligent, XR11 voice-controlled remote control. If a user updated the remote with a compromised version of firmware, it could effectively be turned into a listening device.

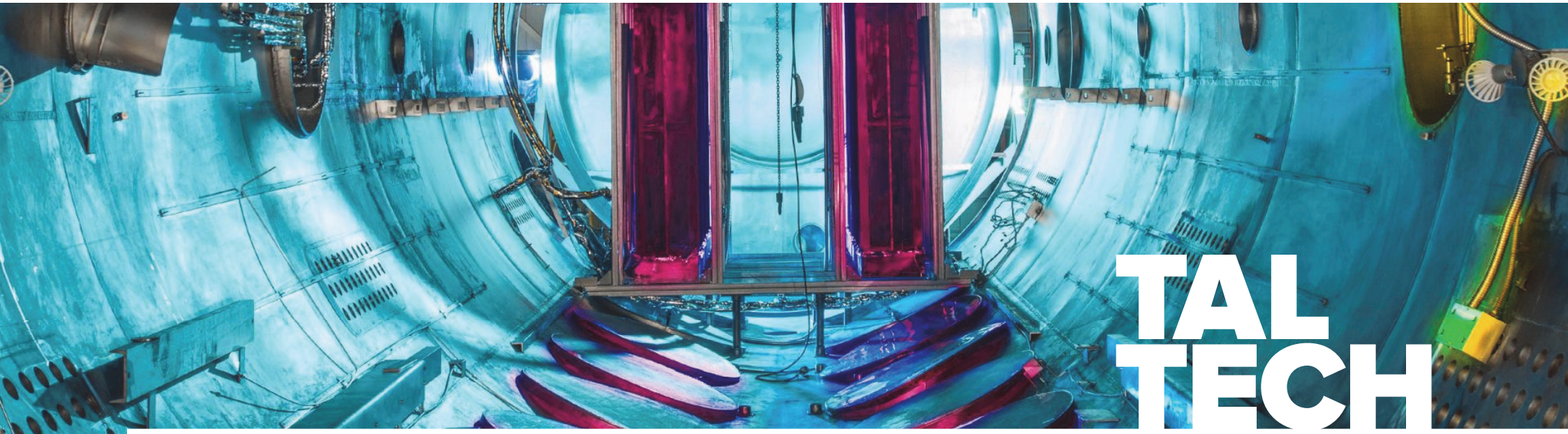
- Nvidia released a patch to plug a vulnerability that could have allowed hackers to remotely take control of the company's high-end line of DGX servers

OUTLINE

- Vulnerable hardware components
- Introduction to hardware attacks
- Memory attacks
- Mitigation strategy

WHAT HARDWARE COMPONENTS CAN BE ATTACKED?

- Node:
  - Central Processing Unit (CPU)
  - Graphic Processing Unit (GPU)
  - General-purpose computing on graphics processing units (GPGPU)
  - …
- Memory:
  - Main memory
  - Cache memory
  - Flash
  - …
- Communication channels:
  - Data bus
  - Address bus
  - …



NODE, MEMORY AND COMMUNICATION ATTACKS

EXAMPLE OF ATTACKS IN LITERATURE


Spectre


Meltdown


Firecracker


Evict+Time


Flush+Reload


Raw Hammer


Sink Hole


Buffer Overflow


x86 Hidden Instruction


Hourglass


Laser injection


Cache Trace-driven


Prime+Probe


Flush+Flush

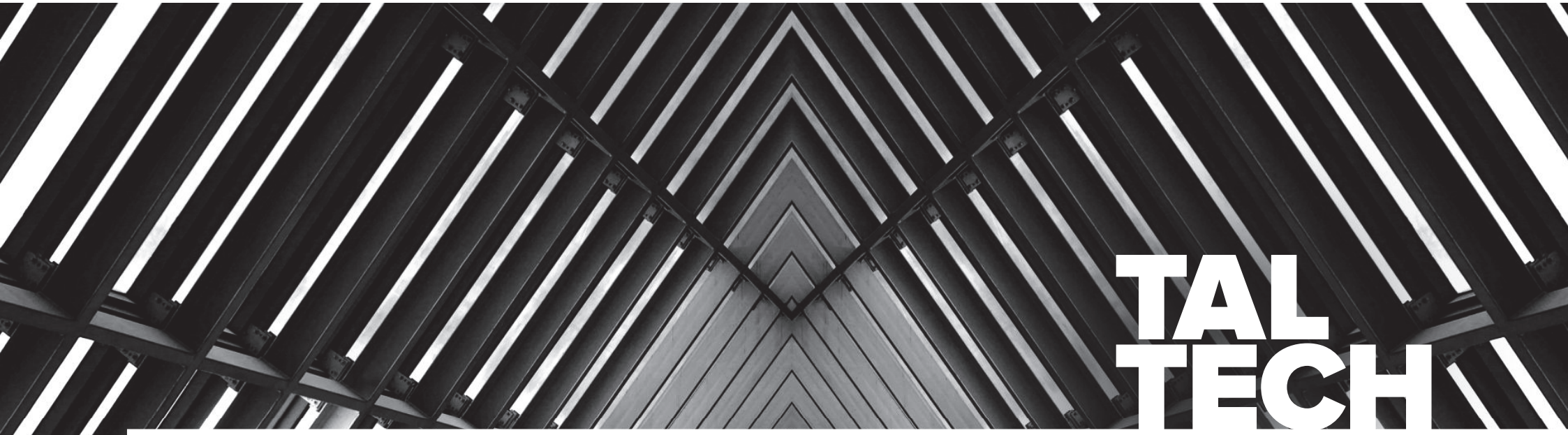
TAL TECH

WHERE EACH ATTACK TARGETS?

- Spectre
- Meltdown
- Laser injection
- Evict+Time
- Flush+Reload
- Raw Hammer
- Flush+Flush

- Sink Hole
- Buffer Overflow
- x86 Hidden Instruction
- Hourglass
- Firecracker
- Cache Trace-driven
- Prime+Probe

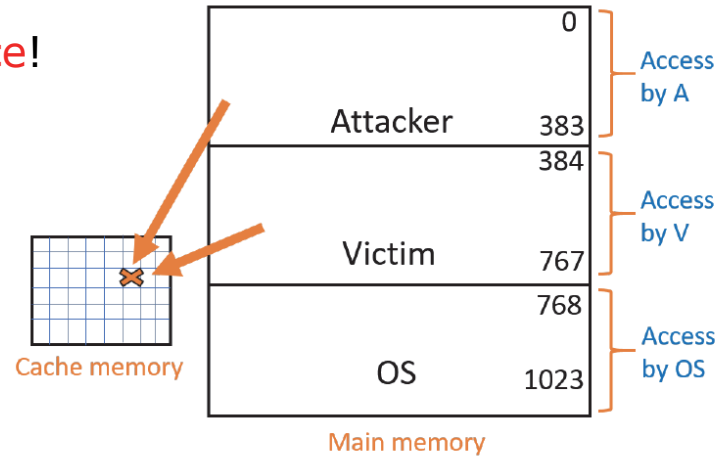
TAL TECH



MEMORY ATTACK

What to know to undrestand ?

Attacks can happen when there is a shared resource!

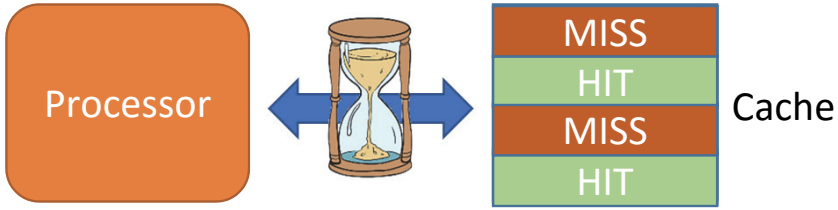


TAL TECH

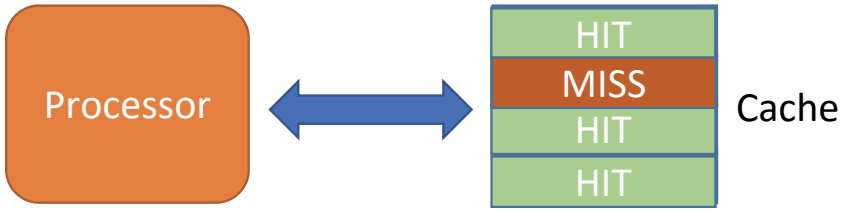
 Flush+Flush  Flush+Flush  Flush+Reload  Prime+Probe

What to know to undrestand  ?

- Timing (Bernstein, Tech Report 2005)
  - Exploit execution time differences due to cache usage
  - More cache misses means higher latency
- Access (Osvik, Springer 2006)
  - Use cache behaviour to understand the used addresses by the victim
  - Attacker changes the cache and observes



Time_a = Operation (Input_a)
Time_b = Operation (Input_b)



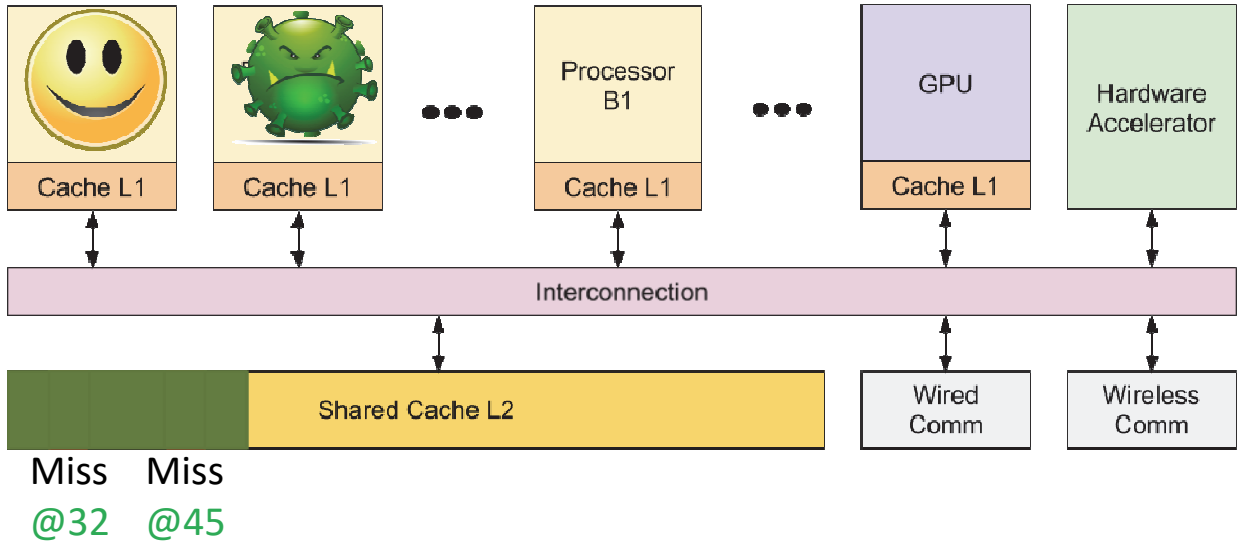
1) Attacker prepare the Cache
2) Victim uses the Cache
3) Attacker access the Cache

TAL TECH

 Evict+Time  Flush+Flush  Flush+Reload  Prime+Probe

Flush+Reload Flush+Flush Prime+Probe

- Cache Attack on AES128:
  1. Attacker fills Cache
  2. Attacker asks Encryption
  3. Encryption takes place
  4. Attacker fills Cache
  5. Attacker calculates the Key

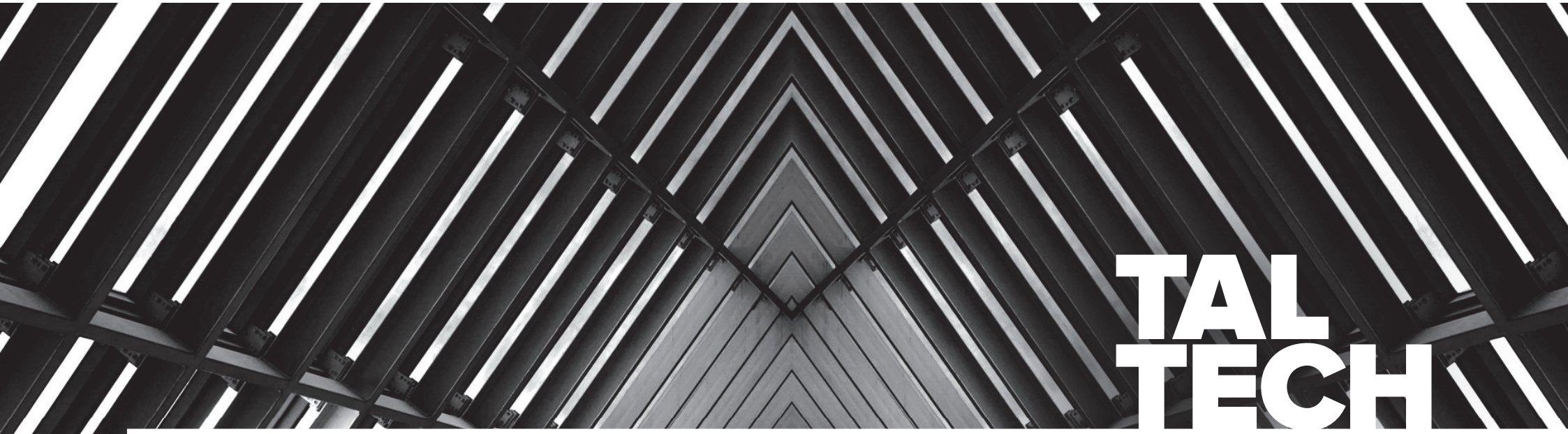


AES Round 1 $\leftarrow T_0\left[x_0^{(r)}\right] \oplus T_1\left[x_5^{(r)}\right] \oplus T_2\left[x_{10}^{(r)}\right] \oplus T_3\left[x_{15}^{(r)}\right] \oplus K_0^{(r)}$

X0 = Plaintext[0] XOR Key[0]
X5 = Plaintext[5] XOR Key[5]

32 = 25 XOR Key[0]
45 = 14 XOR Key[5]

Key[0] = 57
Key[5] = 35



OTHER KIND OF MEMORY ATTACKS

Meltdown

- Meltdown manipulates out-of-order execution to access cache memory.

Any modern processor rearranges the order of instructions to be executed to add a gap between instructions with data dependency. Those filling instructions are executed out-of-order, and in the last part of the pipeline, a module reorganizes the outputs (i.e., put again in order). However, a common vulnerability inside most processors is that illegal operations are just checked in the last part when the reordering happens. As a result, this causes a transient execution.

Meltdown organizes a program where the independent instruction performs an illegal operation with sensitive data. Such operation typically accesses the cache, which modifies its state. Hence, even with the undo performed by the processor, the cache state is already changed. Finally, the attacker needs to reread the cache and verify what address the state has changed.

TAL TECH

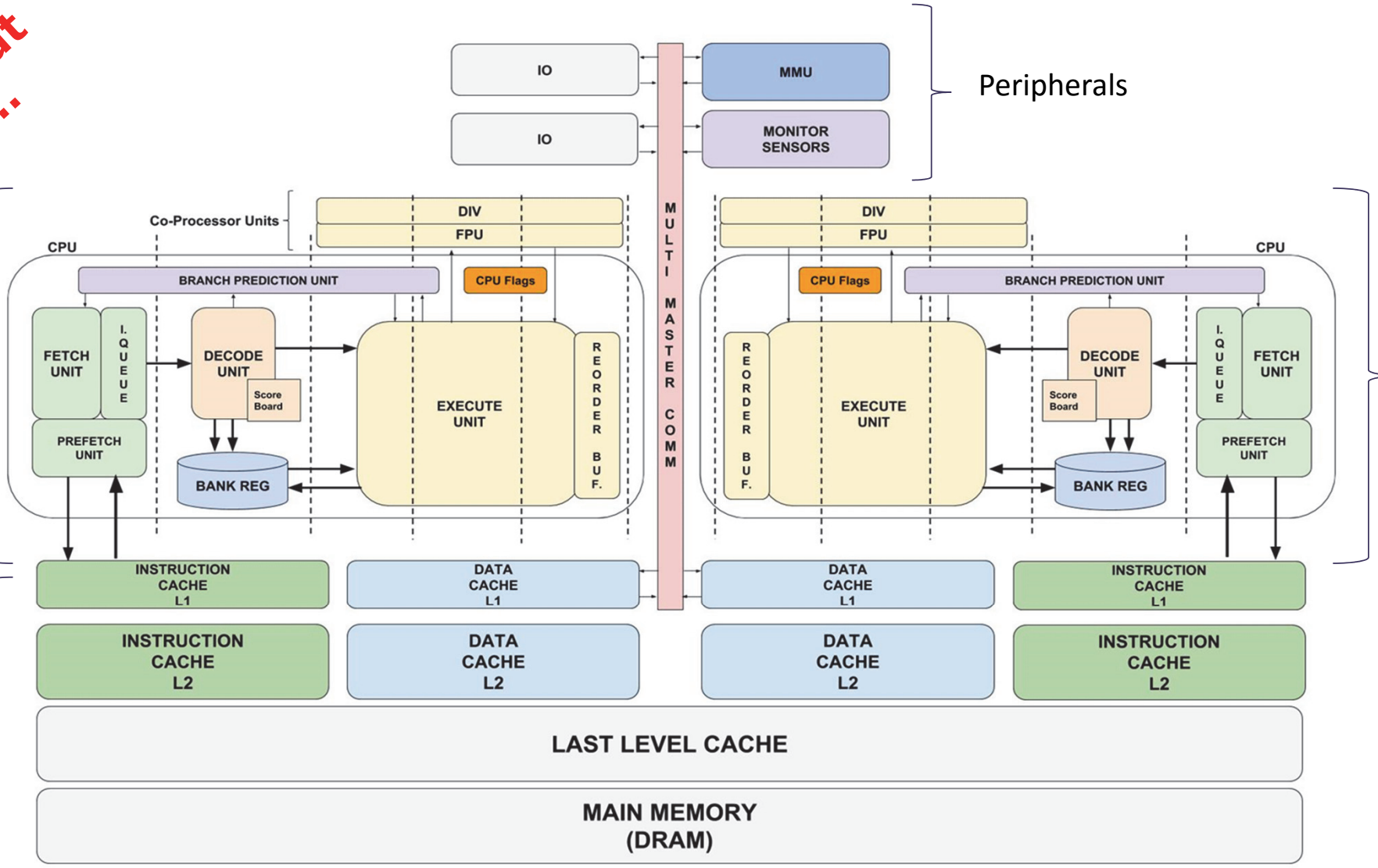
Spectre

- Spectre manipulates branch prediction unit to stole data from cache memory.

Spectre refer to attacks on the branch prediction unit (BPU) of modern processors. They exploit the speculative behaviour of such component to force illegal instructions to execute. Although the processor can detect such mistake caused by speculation and undo the illegal operation, the processor or the system might have changed. If carefully designed, the attacker can hide the stolen information in such different states of the system. The most typical example is to use the cache memory, where the access to an address will permanently change the cache state.

TAL TECH

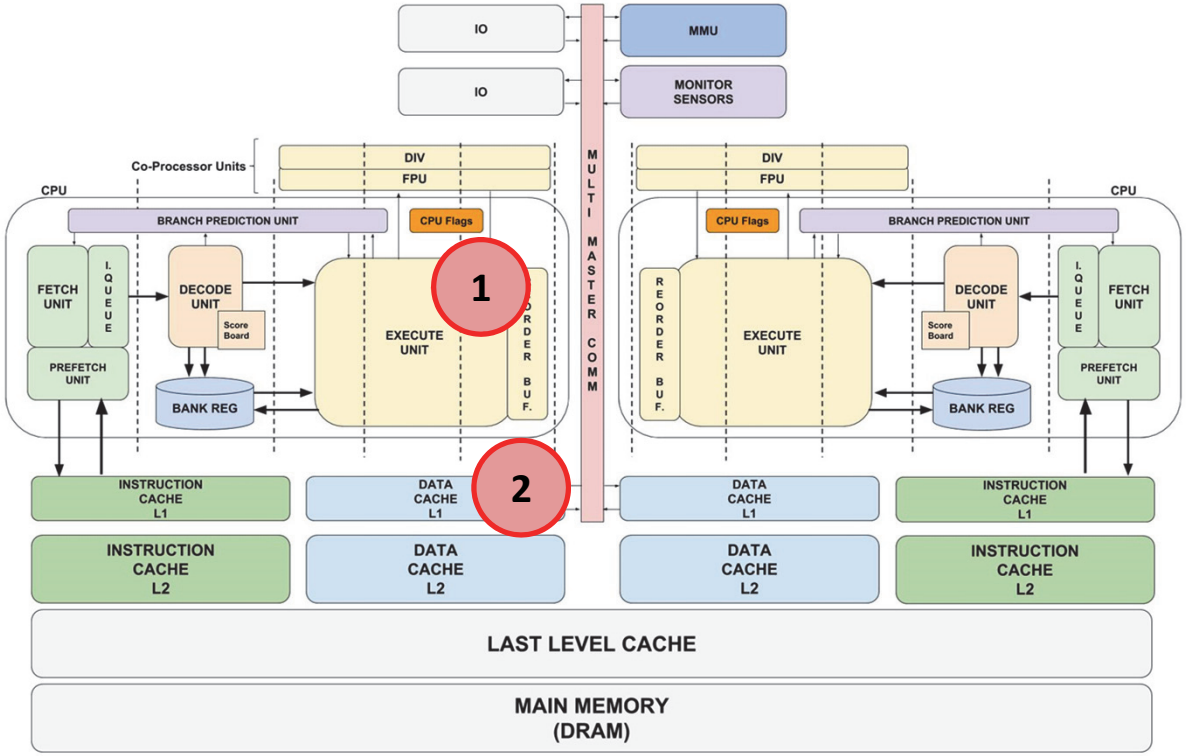
Let's check out some more...



TAL TECH

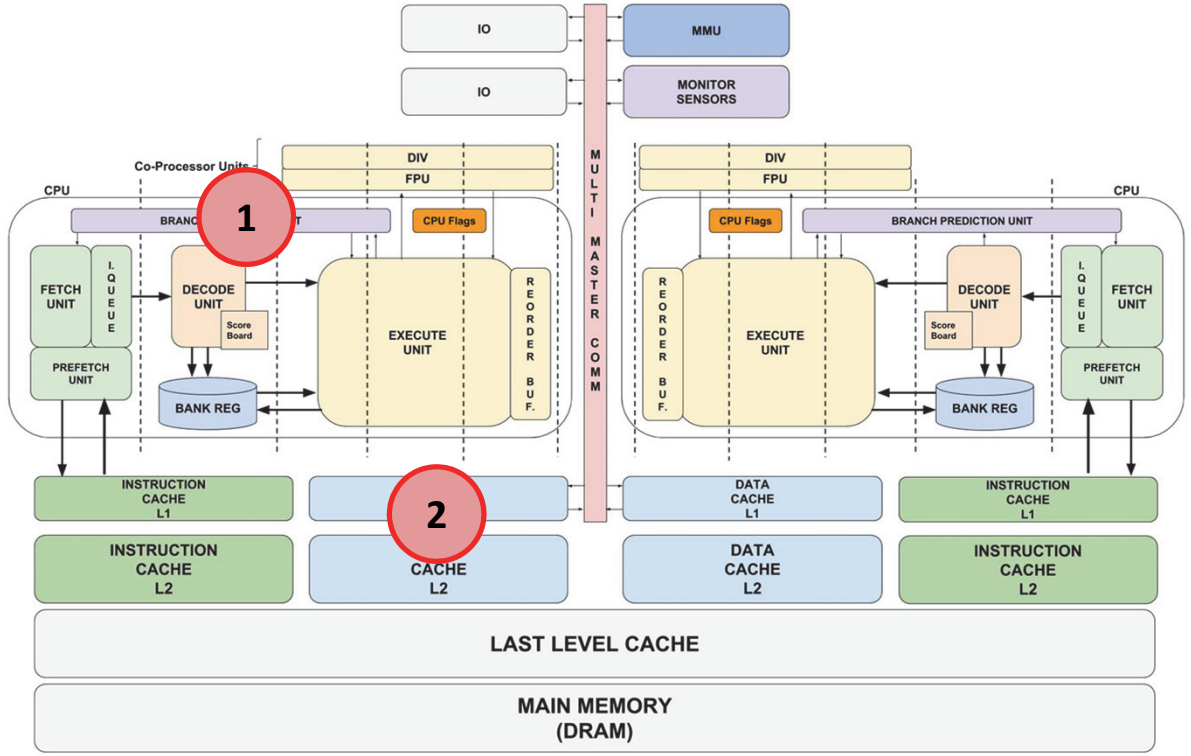
Meltdown

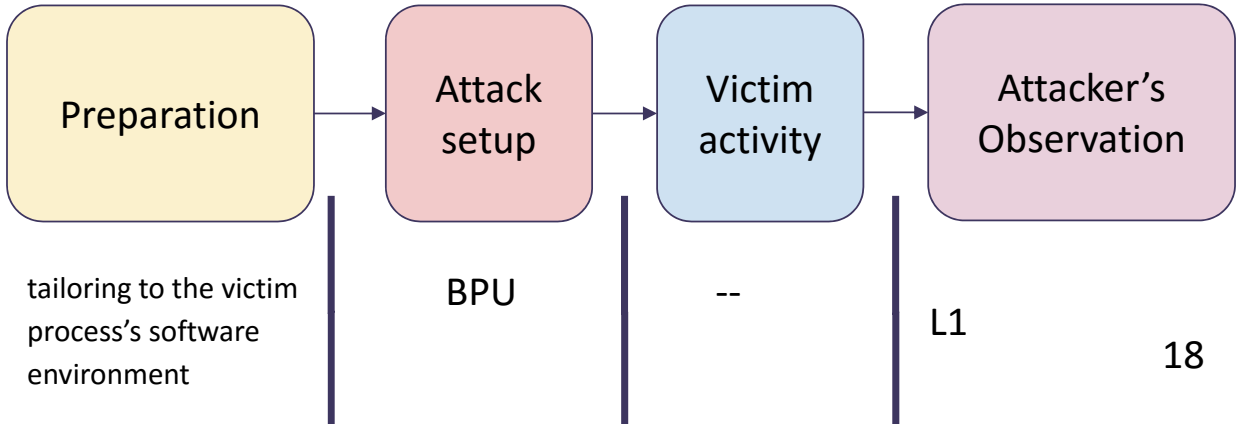
A: Runs Out of Order instructions
A: Observes L1 Cache



Spectre

A: Runs illegal instructions after wrong branch!
A: Observes L1 Cache





Rowhammer

- Rowhammer performs repeatedly access to certain memory rows with a high frequency in order to cause a bitflip, 0 to 1 or viceversa.

Rowhammer is a security exploit that takes advantage of an unintended and undesirable side effect in dynamic random-access memory (DRAM) in which memory cells interact electrically between themselves by leaking their charges possibly changing the contents of nearby memory rows that were not addressed in the original memory access. These attacks perform repeatedly access to certain memory rows with a high frequency in order to degrade the internal charging capacitance. Such operation is defined as “hammering” and that is why such threat is called as Rowhammer.

TAL TECH

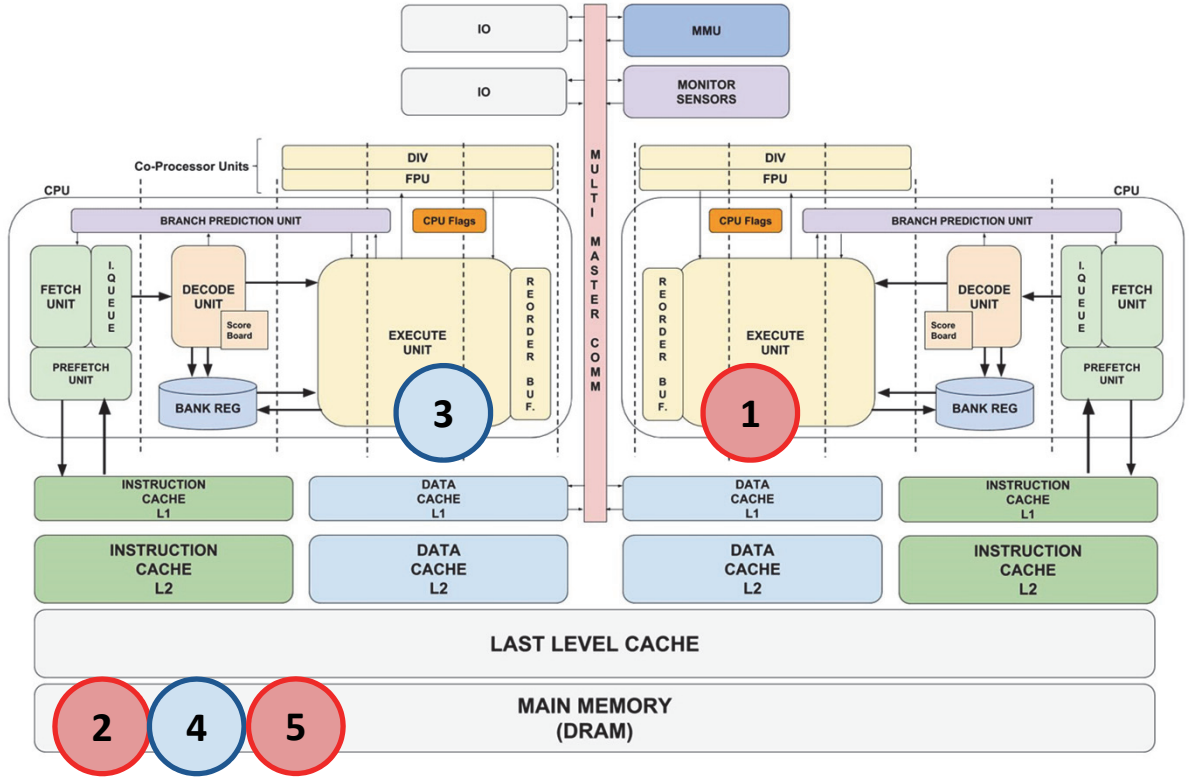
Rowhammer

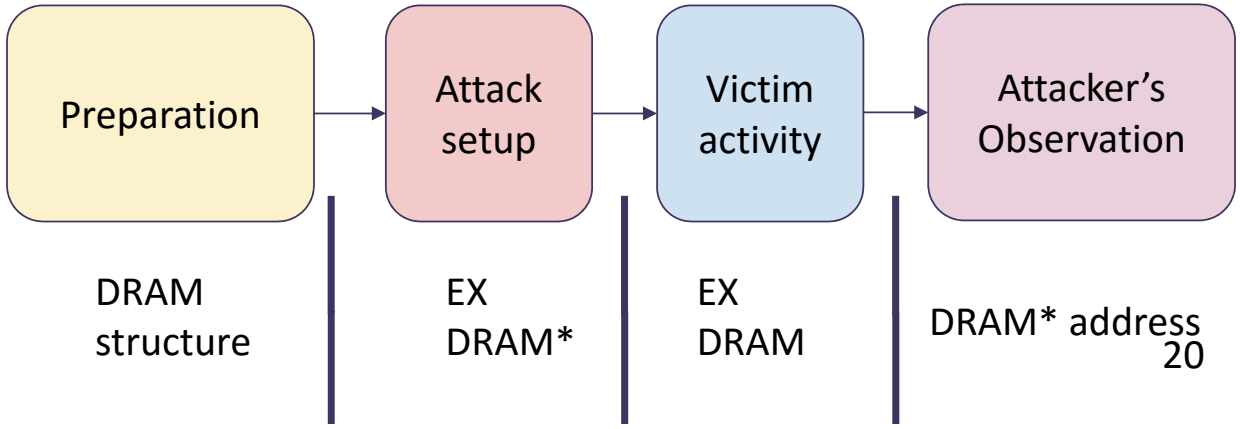
A: Flushes row buffer in DRAM

V: Executes the instructions

V: Makes changes in DRAM

A: Observes row buffer in DRAM





Flush+Reload

Prime+Probe

Flush+Flush

- Flush-and-Reload family, which relies on sharing pages between the attacker and the victim processes. With shared pages, the attacker can ensure that a specific memory line is evicted from the whole cache hierarchy. The spy uses this to monitor access to the memory line.
- PRIME+PROBE is another attack which extracts the time measurement information by manipulating the state of the cache before each encryption, and observes the execution time of the subsequent encryption.
- Flush+Flush attack relies only on the difference in timing of the flush instruction between cached and non cached memory accesses. In contrast to other cache attacks, it does not perform any memory accesses. Indeed, it builds upon the observation that the flush instruction leaks information on the state of the cache.

TAL TECH

Flush+Reload

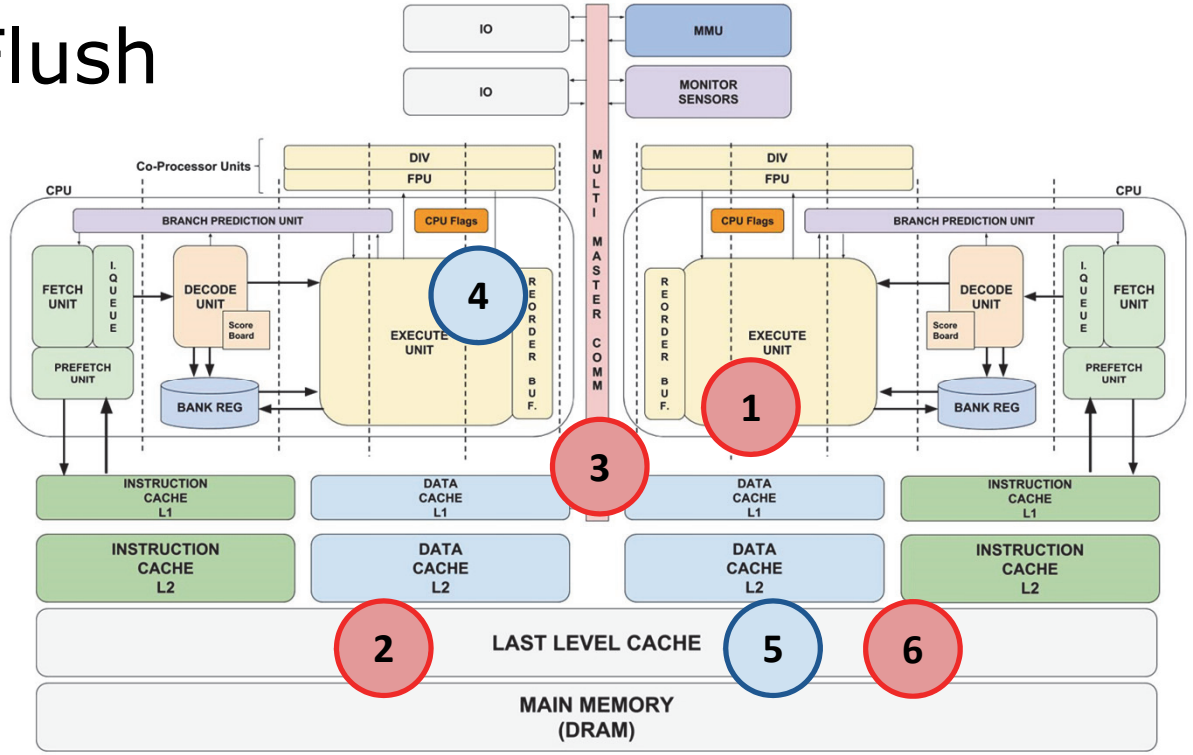
Flush+Flush

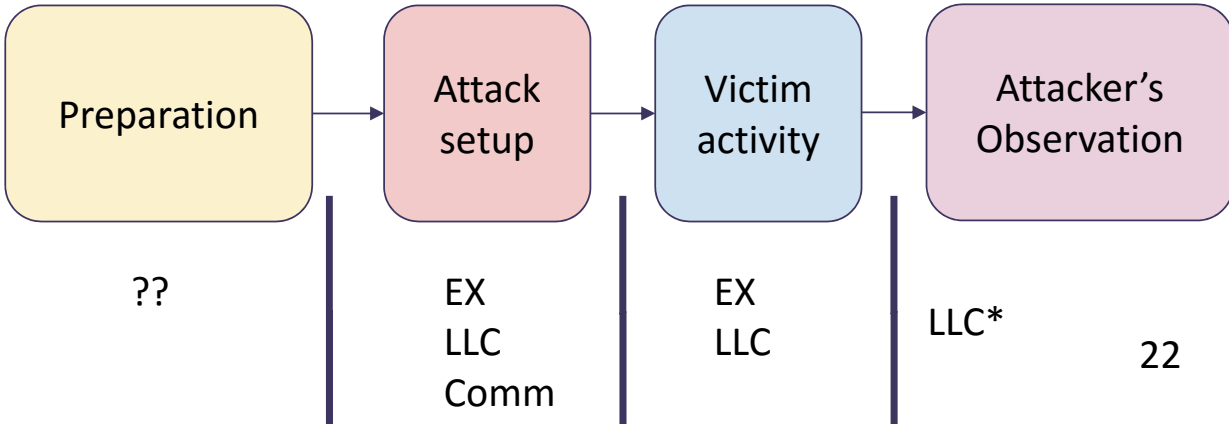
Prime+Probe

A: Takes over the core
A: Flushes the cache
A: Asks the victim to perform encryption

V: Executes the instructions
V: Accesses memory, resulting in a change in cache

A: Observes the changed addresses





Evict+Time

- EVICT+TIME is an attack which extracts the time measurement information by manipulating the state of the cache before each encryption, and observes the execution time of the subsequent encryption.

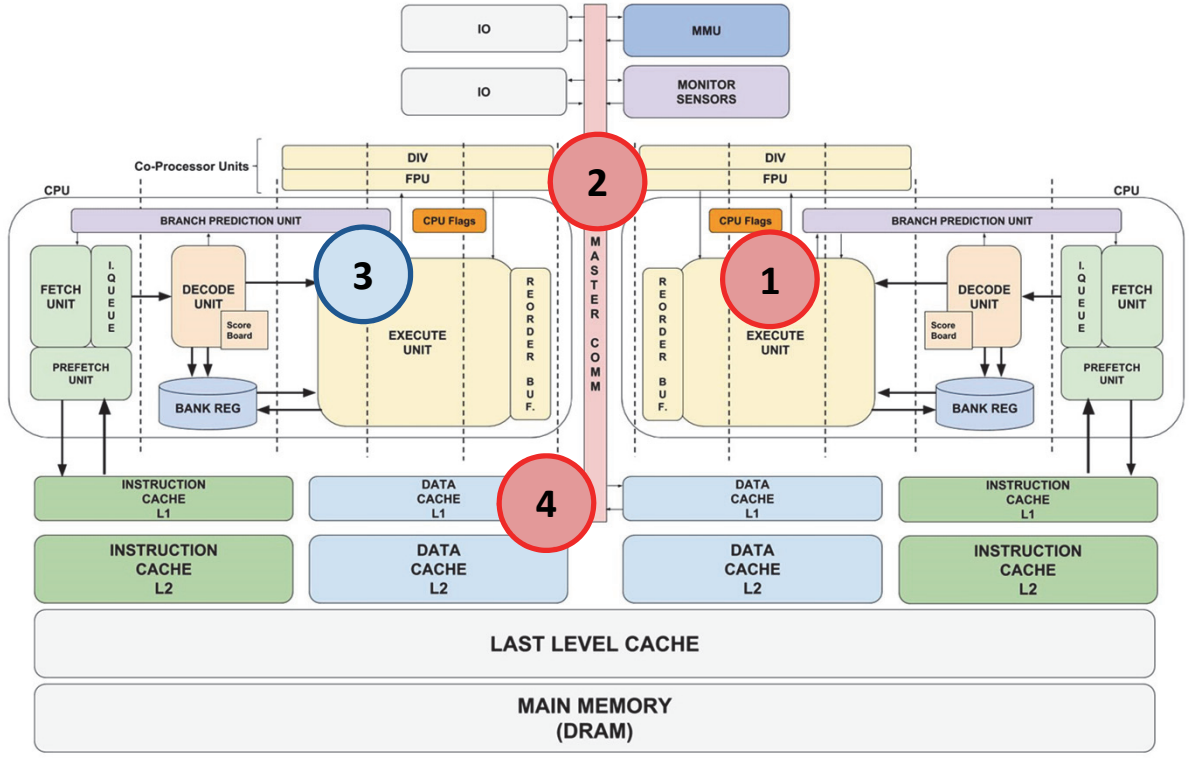
TAL TECH

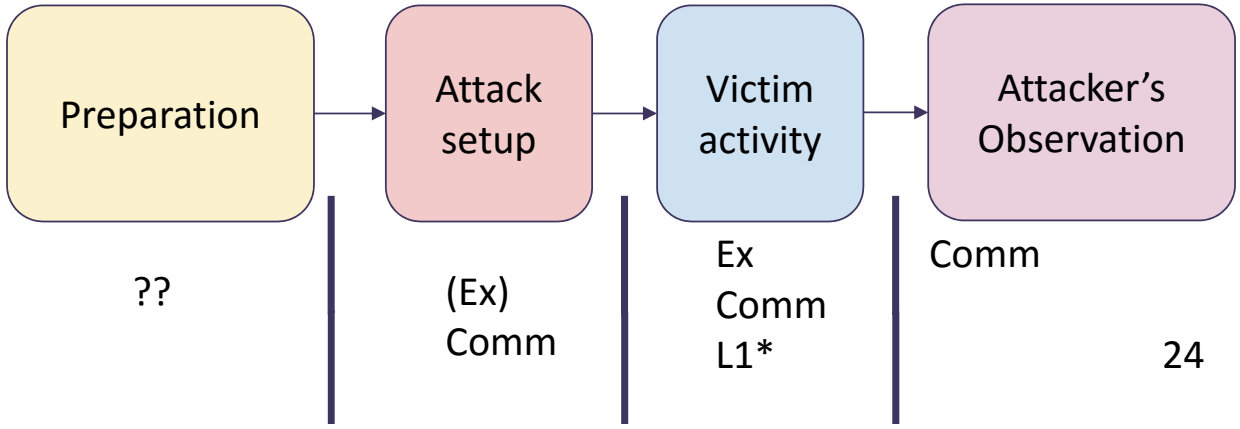
Evict+Time

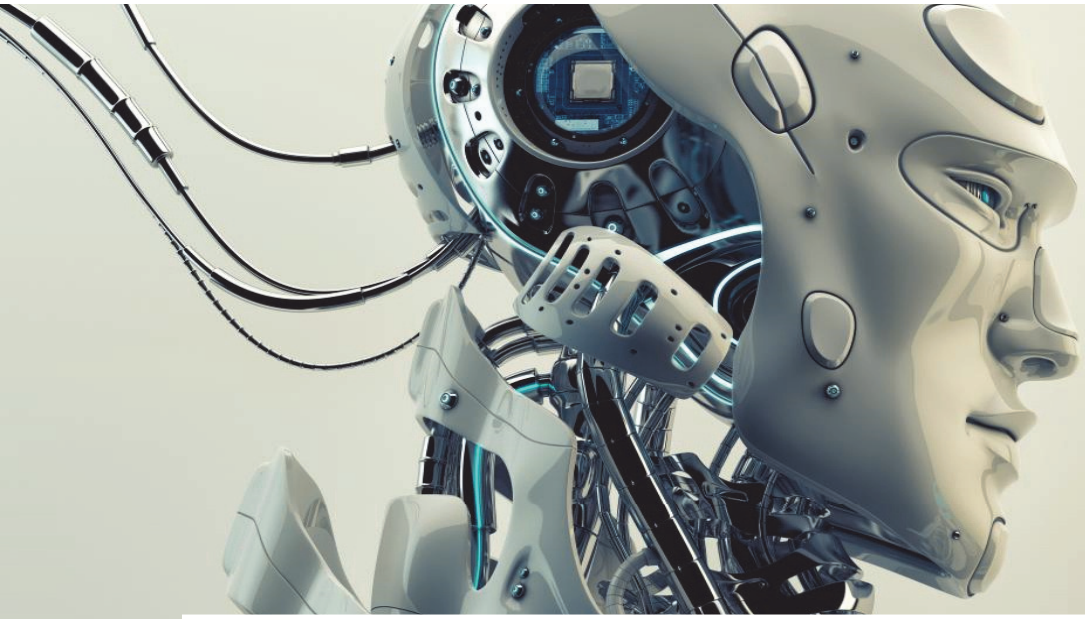
A: Asks the victim to perform encryption

V: Executes the instructions

A: Observes the timing







MITIGATION STRATEGY FOR CACHES

DIFFERENT TYPE OF CACHES

- Conventional or non-partitioned Cache
- Statically-Partitioned Cache
- Dynamically-Partitioned Cache

CONVENTIONAL CACHE

- The conventional non-partitioned cache is a commonly used cache type where all applications share the complete cache address space. Due to this, the memory space of both the common and sensitive may (partially) overlap.

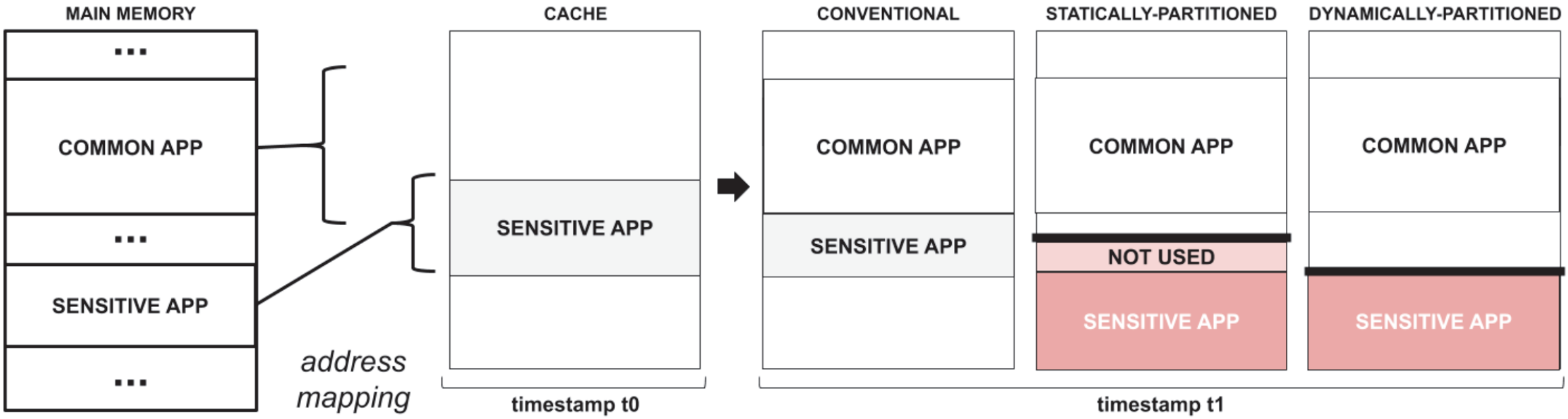
STATICALLY-PARTITIONED CACHE

- In statically-partitioned caches, the cache is physically divided into different partitions. Each secure partition can only be used by one application, thus it effectively eliminates the cache interference between different applications. The drawback of statically-partitioned caches is their performance degradation. The fixed size of the partition effectively reduces the cache size as parts of the cache might not be used.

DYNAMICALLY-PARTITIONED CACHE

- Dynamically-partitioned caches are similar to static partitioned cache but determine the partition size dynamically at run-time based on the application needs.

CACHES: CONVENTIONAL-STATIC PARTITIONED–DYNAMIC PARTITIONED



WHICH CACHE IS THE MOST SECURE ONE?

- Conventional or non-partitioned cache
- Statically-Partitioned Cache
- Dynamically-Partitioned Cache

WHICH CACHE IS THE MOST EXPENSIVE ONE?

- Conventional or non-partitioned cache
- Statically-Partitioned Cache
- Dynamically-Partitioned Cache

IF YOU HAVE ANY QUESTION YOU CAN SEND ME AN EMAIL:

TARA.GHASEMPOURI@TALTECH.EE

OR FIND ME AT:

AKADEEMIA TEE 15A, ICT BUILDING, ROOM 504.