Securing Processors from Time Side Channels

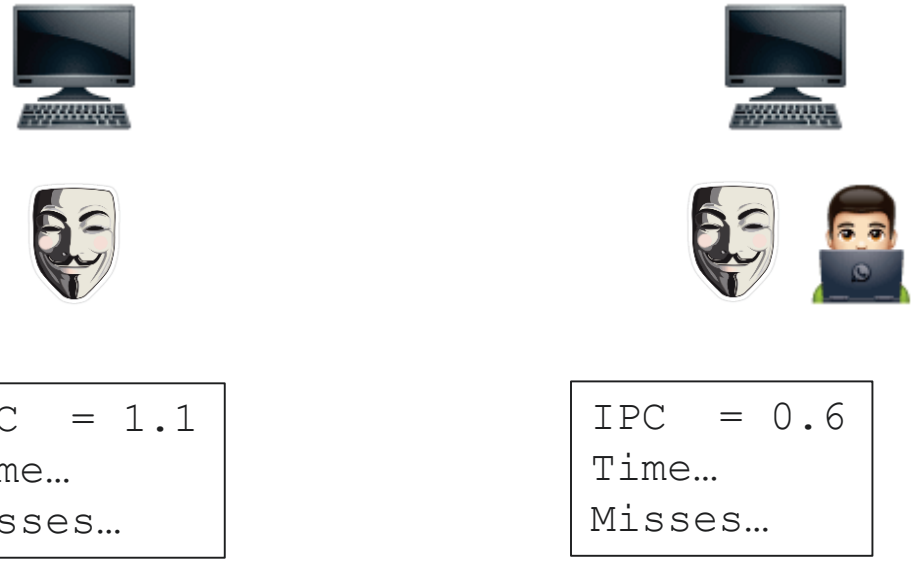
Jose Renau

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

An Attacker can use Time Leaks to Infer Usage

```
IPC  = 1.1
Time…
Misses…
```

An Attacker can use Time Leaks to Infer Usage



An Attacker can use Time Leaks to Infer Usage

IPC = 0.9
Time…
Misses…

IPC = 1.1
Time…
Misses…

IPC = 0.6
Time…
Misses…

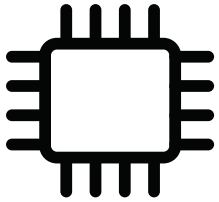
Time Domain

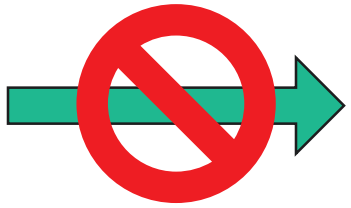
One Time Domain should not affect the performance of another Time Domain

Time Domain

One Time Domain should not affect the performance of another Time Domain

Time Domain 1
IPC1





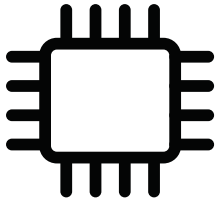
Time Impact

```
add   s0,sp,48
sw    a0,-36(s0)
sd    a1,-48(s0)
lw    a5,-36(s0)
sw    a5,-20(s0)
sw    zero,-24(s0)
j     .L2
```

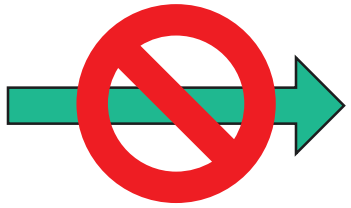
Time Domain

One Time Domain should not affect the performance of another Time Domain

Time Domain 1
IPC1

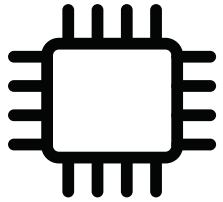


```
add   s0,sp,48
sw    a0,-36(s0)
sd    a1,-48(s0)
lw    a5,-36(s0)
sw    a5,-20(s0)
sw    zero,-24(s0)
j     .L2
```



Time Impact

Time Domain 2
IPC2

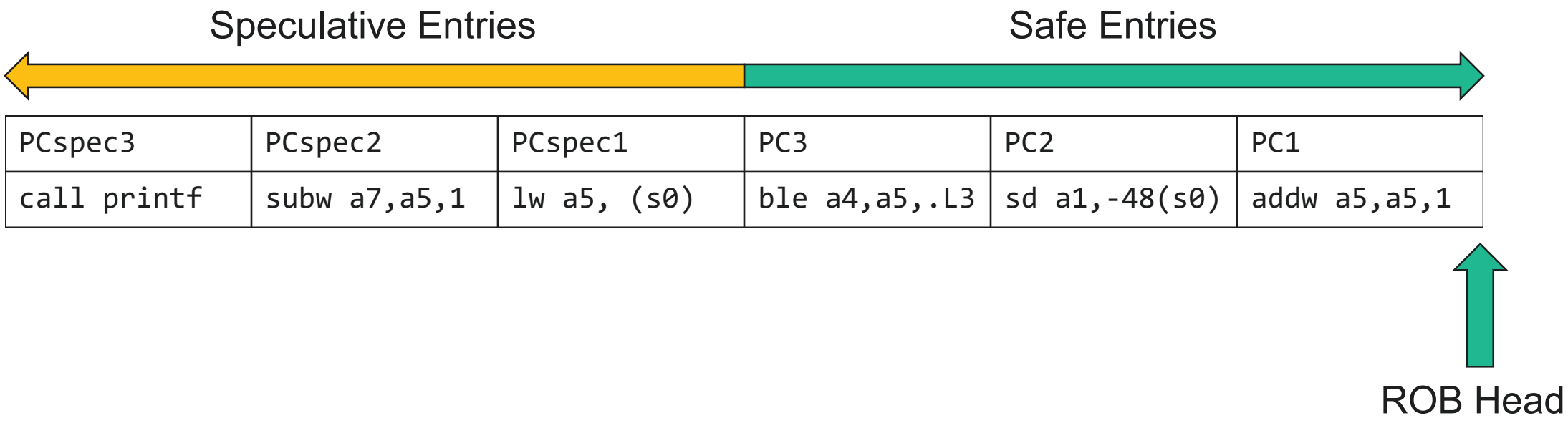


```
lw    a4,-20(s0)
lw    a5,-36(s0)
addw  a5,a4,a5
sw    a5,-20(s0)
lw    a5,-24(s0)
addw  a5,a5,1
sw    a5,-24(s0)
```

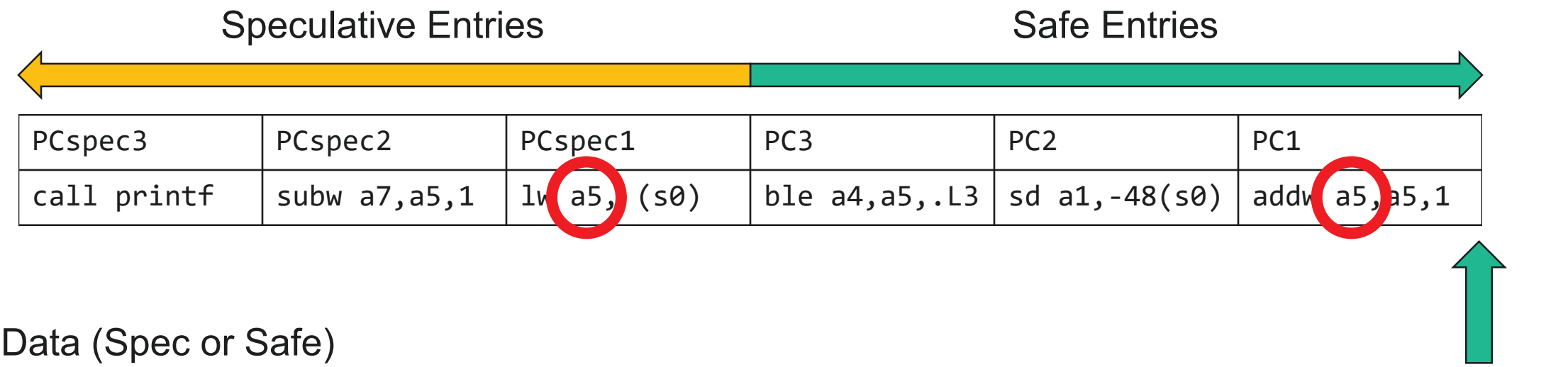
Outline

- Categorize Time Leaks
- High level techniques
- Sample cores

What can Time Leak?

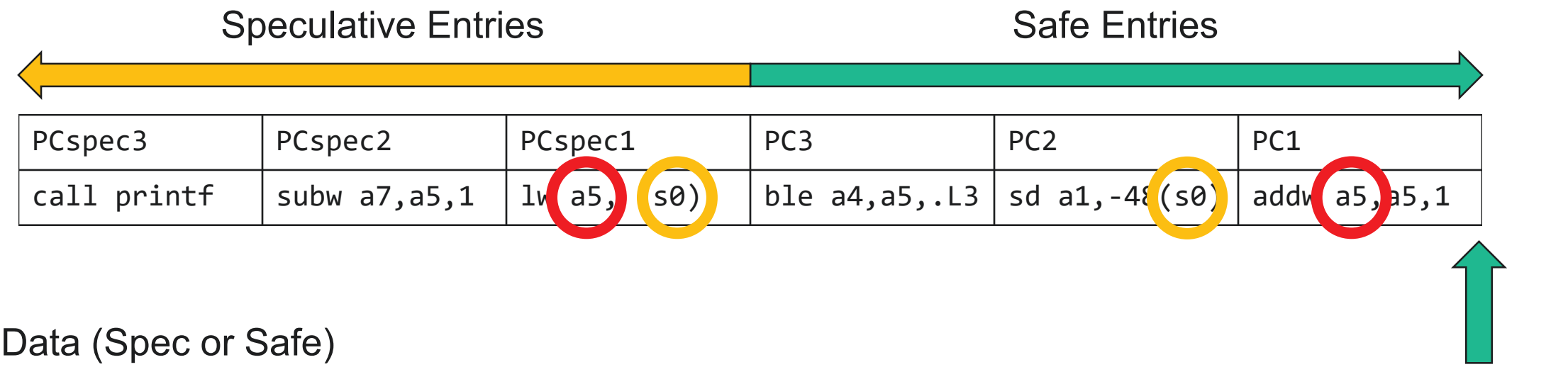


What can Time Leak?



D: Data (Spec or Safe)

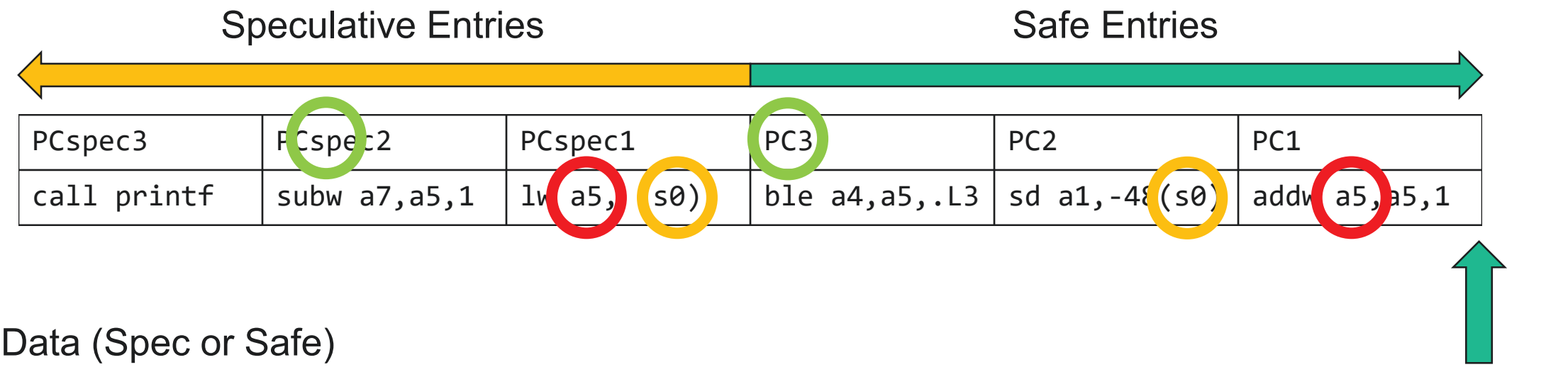
What can Time Leak?



D: Data (Spec or Safe)

A: Addresses (Spec or Safe)

What can Time Leak?



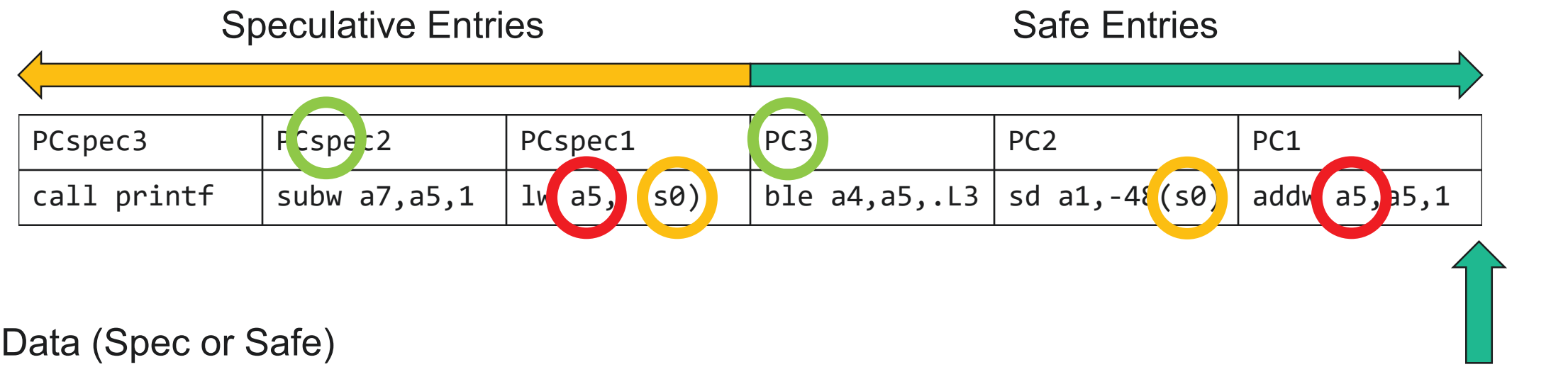
D: Data (Spec or Safe)

A: Addresses (Spec or Safe)

P: Program Counter (Spec or Safe)

E: Execution Time (Spec or Safe)

What can Time Leak?



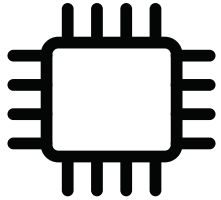
D: Data (Spec or Safe)

A: Addresses (Spec or Safe)

P: Program Counter (Spec or Safe)

E: Execution Time (Spec or Safe)

Where can Time Leaks go?

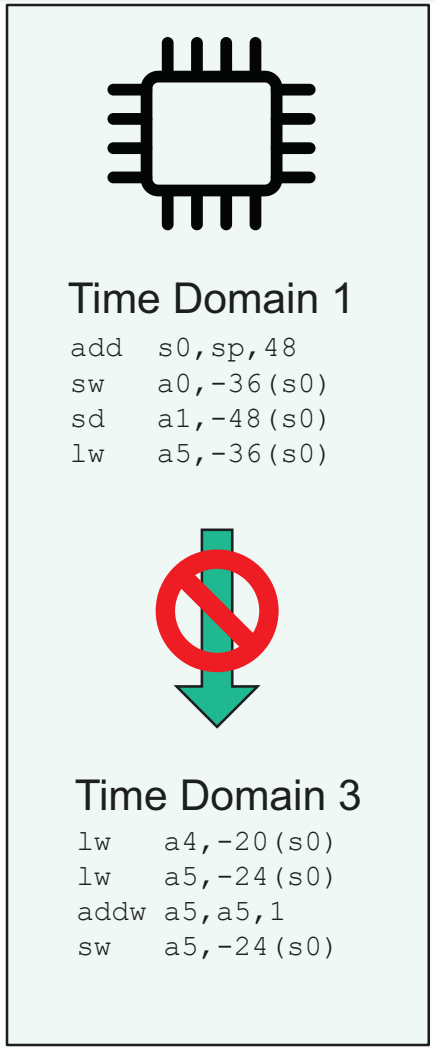


Time Domain 1

```
add   s0,sp,48
sw    a0,-36(s0)
sd    a1,-48(s0)
lw    a5,-36(s0)
```

Where can Time Leaks go?

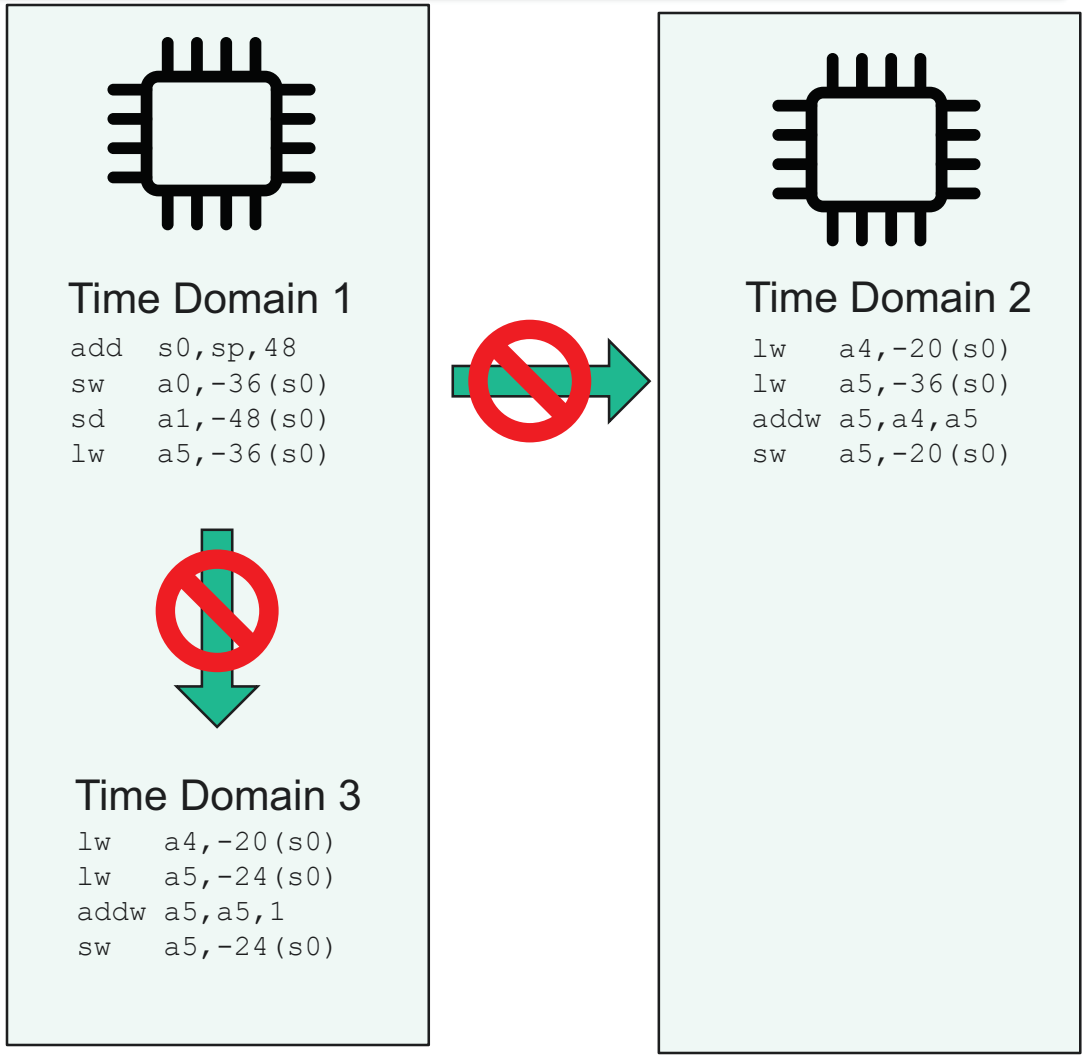
SC: no Same Core leaks



Where can Time Leaks go?

SC: no Same Core leaks

MC: no Muti-Core leaks

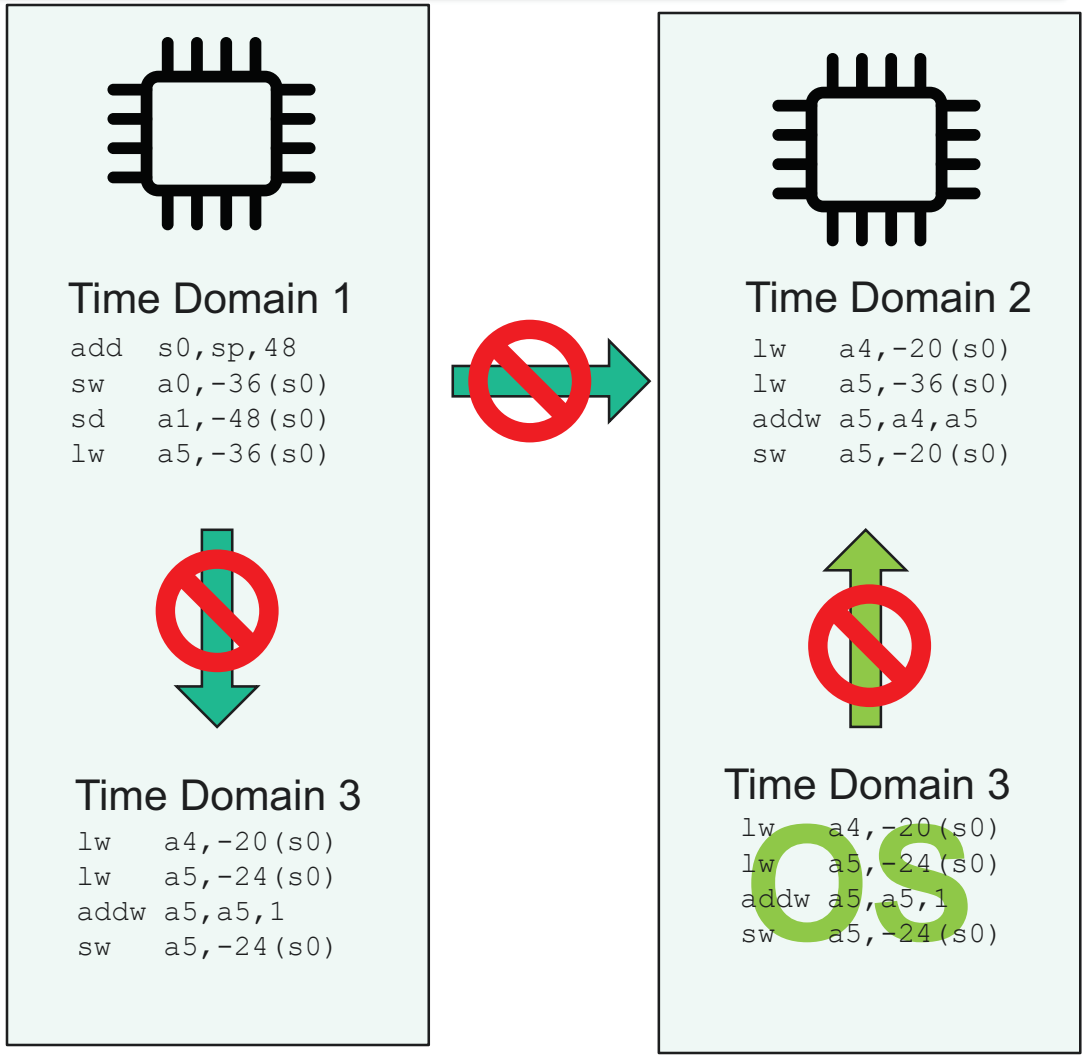


Where can Time Leaks go?

SC: no Same Core leaks

MC: no Muti-Core leaks

OS: no Operating System leaks



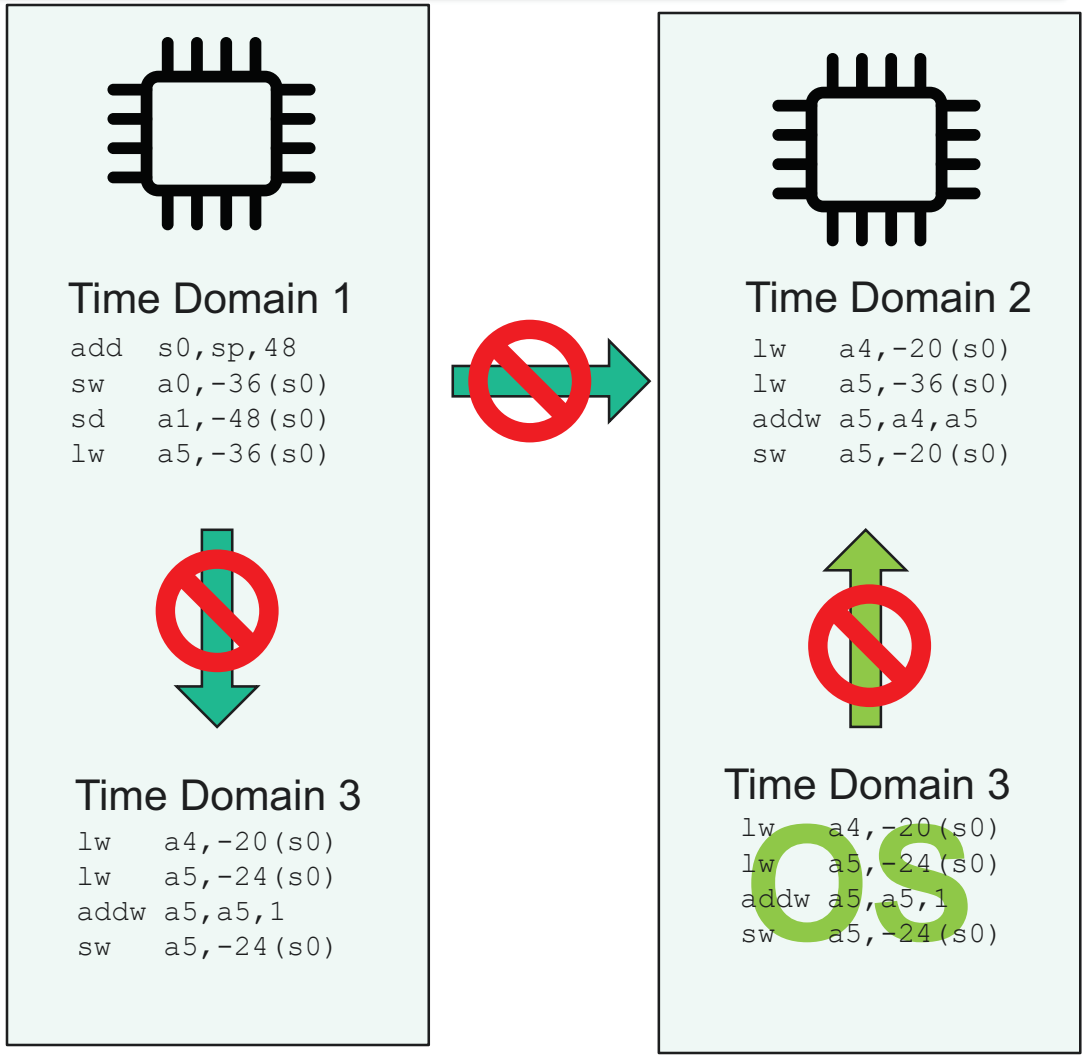
Where can Time Leaks go?

SC: no Same Core leaks

MC: no Muti-Core leaks

OS: no Operating System leaks

AI: no Application Interface leaks



What is a Safe.AI? (Application Interface)

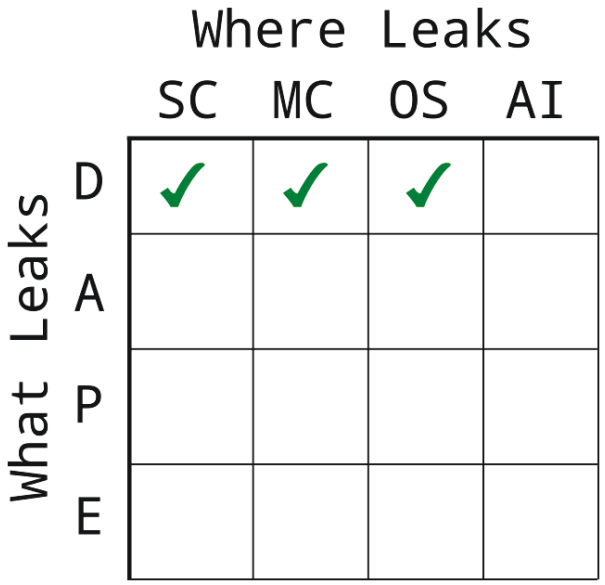
The alien SoC/App must have an AI accessible to attackers

```
x = get_time
Call Alien_API
```

```
Wait Response
y = get_time
```

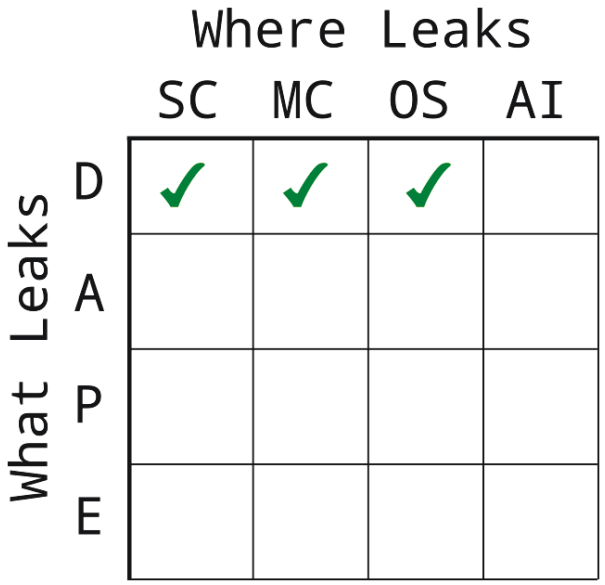
Even alien/opaque SoCs can Time Leak to an AI attack

Time Leaks Categorization

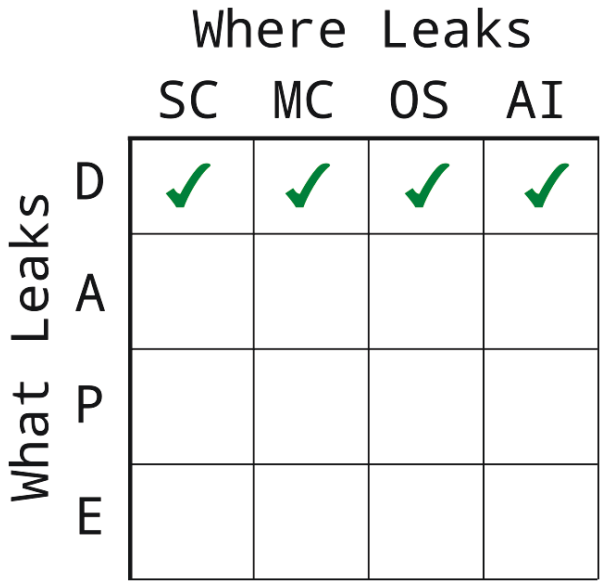


AMD Zen-like protection

Time Leaks Categorization

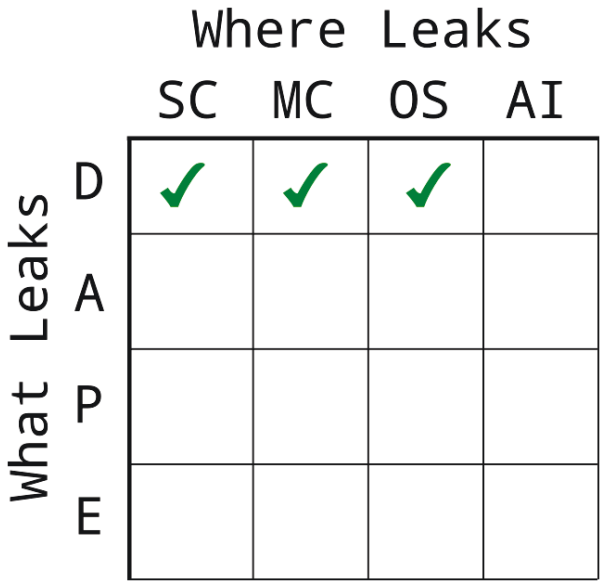


AMD Zen-like protection

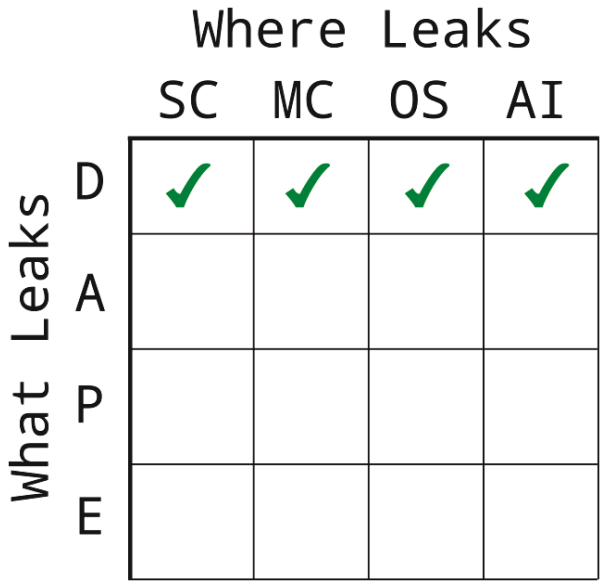


Safe[D].ALL core

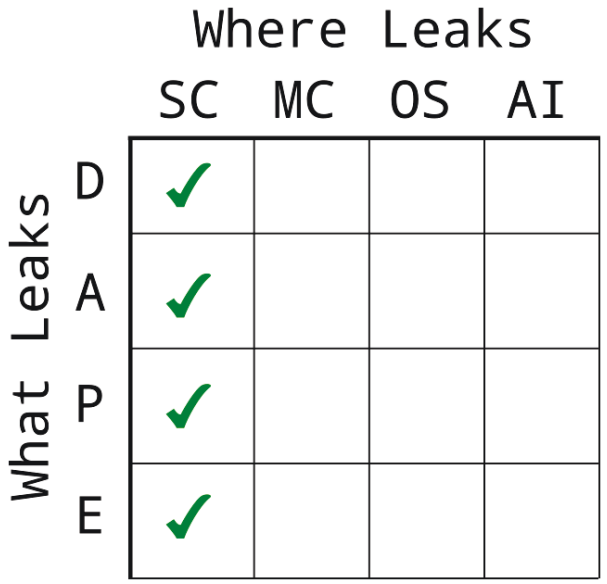
Time Leaks Categorization



AMD Zen-like protection

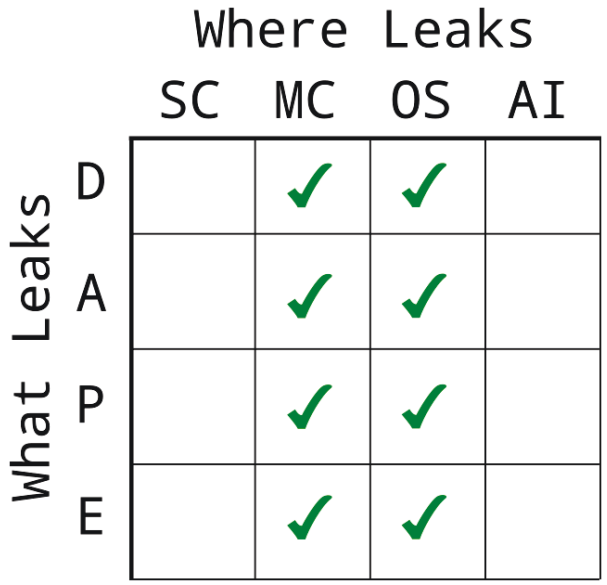


Safe[D].ALL core

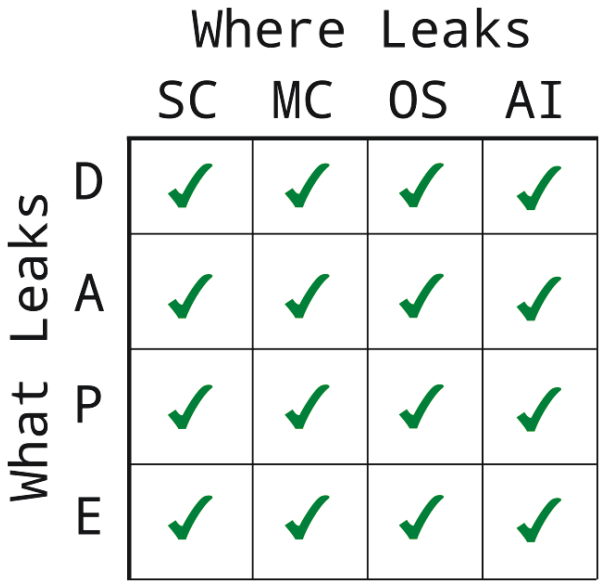


Safe.SC core

Time Leaks Categories



Safe.MC.OS



Spec.ALL

Interesting Cases

- Google/amazon/cloud provider
  - Safe.MC.OS, Spec.ALL and have different VMs in different cores
- Desktop/phone
  - Safe.SC.MC, Spec.ALL
- Autonomous cars
  - Spec.MC, Spec.ALL and use OS to separate per core
- Router/switch
  - Spec.ALL (allow BPF)
- IOT
  - Unclear, web browser IOT Spec.ALL

Outline

- Categorize Time Leaks
- High level techniques
- Sample cores

Tech 1: Point of No Return (PNR)

- Not all the instructions are speculative
- PNR in ROB indicates point were instruction is guaranteed to eventually commit

- SPEC2006 for an A72-like core ~25% of the instructions are beyond the PNR

Tech 2: No Speculative Update Predictors

- Delay and/or perfect fixup all the predictors
  - Good: Minor/no performance impact
  - Bad: More buffering
- Delay TAGE/BTB/… updates until retirement
  - No performance impact!
- Return Address Stack (RAS) and GHR
  - Have a copy at retirement, perfect fix at misspredict
- Delay AMPM/Stride/xxx prefetch updates to retirement
  - Even better performance, no pollution

Tech 3: No Speculative Cache

- No inclusive/exclusive L1/L2/L3 caches
- Allocate lines only to Store Completion Buffer (SCB) on cache miss
  - When combined with PNR, very few entries in SCB
  - For a 4-way OoO core and SPEC2006, 3 entries cover over 99% of the cases

Tech 4: Partition

- Private L1 and L2
- Many papers showing how to data partition LLC
- Bias designs towards systems with smaller amount of shared LLC

Tech 5: Partial/Full Flush

- Flush entries at Time Domain switch
- Flush can be partial like 2 LRU ways or X entries

Tech 6: Partial/Full Off-load

- Off-load/Re-load part of structures at Time Domain switch
- E.g:
  - Off-Load Data L1 MRU lines on Time Domain switch
  - Flush the other Data L1 entries

Tech 7: Dynamic every ?M cycles

- Dynamic partition/repartition/adaptation can increase throughput/utilization

- OK to dynamically adapt as long as it does not leak
  - Adapt to a slow moving average infrequently (millions of cycles)
  - Trade-off performance vs security

Other Techniques (not show due to time)

- Tech 8: Dealing with Row Hammer
- Tech 9: BW isolation per Time Domain
- Tech 10: QoS for network on chip
- Tech 11: Non-value dependent operation latency (no telescopic units)
- Tech 12: Regularization
- Tech 13: Homogenization

Outline

- Categorize Time Leaks
- High level techniques
- Sample cores

Spec.ALL Techniques

- Core
  - Private L1/L2 non-inclusive caches
  - Tech 1: PNR
  - Tech 2: No Speculative Update Predictors
    - TAGE/ITTAGE/BTB/Prefetcher
    - Stride prefetcher uses retire address + extra offset
  - Tech 3: No Speculative Cache
    - Allocate speculative loads on L1 SCB

- SoC
  - Small non-inclusive L3
  - Tech 4: Dynamic Partition
    - Fixed partition LLC, directory
    - Dynamic partition: bandwidth, Memory Controller (open pages, row hammer)
  - Tech 7: Adapt every dynamic partitioned resources every 10M cycles
  - Tech 9: Memory BW partitioning per core

Safe.MC Techniques

- Comes for free once you have the previous Spec.ALL
  - Tech 4 (Dynamic Partitioning) and Tech 9 (BW)

Mostly, Spec.ALL already provides Safe.MC

Safe.OS Techniques

- Core
  - Tech 4: Dynamic Partition if frequent OS calls
    - Dynamic partition: L2 TLB, L2, Memory Dependence Predictor
    - No L2 allocate by OS unless frequent OS
  - Tech 5: Partial Flush on OS return
    - Memory Dependence predictor, L1 TLB, Memory Controller
    - Partition of Data L1, Instruction L1, BTB, ITTAGE

Mostly, allocate a chunk of core predictors to OS or flush a fixed chunk at OS return

Safe.SC.MC.OS Techniques

- Combine Safe.MC and Safe.OS techniques, but also add
- Core with infrequent context switch
  - Tech 5: Partial Flush
    - Full Flush Memory Dependence predictor, L1 TLB
    - Partial flush if partition DL2, IL2, BTB, ITTAGE
- Core with frequent context switch
  - Tech 4: Dynamic Partition
    - Dynamic partition: L2 TLB, DL2, IL1
  - Tech 6: Partial Off-load (Frequent context switch)
    - Off-load MRU sets in data and instruction DL1 and IL1 on Time Domain switch
      - Off-load to IL1/DL2 if partition mode, flush otherwise
    - Off-load data and instruction prefetch tables for quick re-load

Mostly, flush if infrequent, partition/off-load if frequent switching

Conclusions

- Categorization of Time Leaks to classify/understand/test current SoCs
- Show several techniques that can achieve different degrees of protection
- Show techniques for Spec.ALL with small performance impact (but large redesign)
- Show techniques for Safe.OS, Safe.MC, Safe.SC
- Protection level could change at run-time

Questions?

Jose Renau

Department of Computer Engineering,
University of California, Santa Cruz
http://masc.soe.ucsc.edu

- Securing Processors from Time Side Channels, Jose Renau, April 2018.

5 Categories of "where" Time Leaks can go

- Application Interaction (*.AI)
  - No Time leak when one Time Domain uses any API in another Time Domain
- Multi Core (*.MC)
  - Time leaks would not go from one core to another
- Single Core (*.SC)
  - Time leaks would not go across Time Domains in the same core
- Operating System (*.OS)
  - OS would not time leak to application Time Domains

4 Categories of "what" can Time Leak

- Speculative or Safe Memory Data (Spec[D] or Safe[D])
- Speculative or Safe Memory Addresses (Spec[A] or Safe[A])
- Speculative or Safe Program Counters (Spec[P] or Safe[P])
- Speculative or Safe Execution Time (Spec[E] or Safe[E])