33RD USENIX
SECURITY SYMPOSIUM

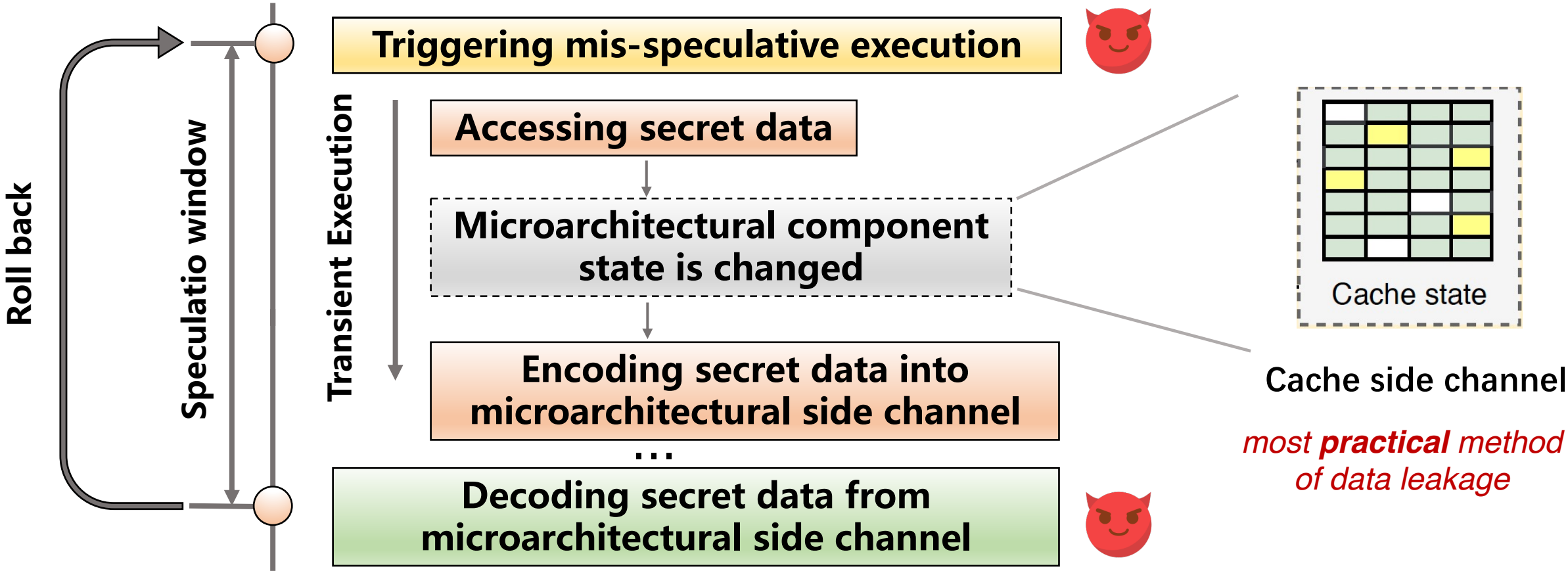
SpecLFB: Eliminating Cache Side Channels in Speculative Executions

Xiaoyu Cheng, Fei Tong, Hongyu Wang, Zhe Zhou,
Fang Jiang, and Yuxing Mao

Presented at USENIX Security 2024

Contents

Spectre Attack

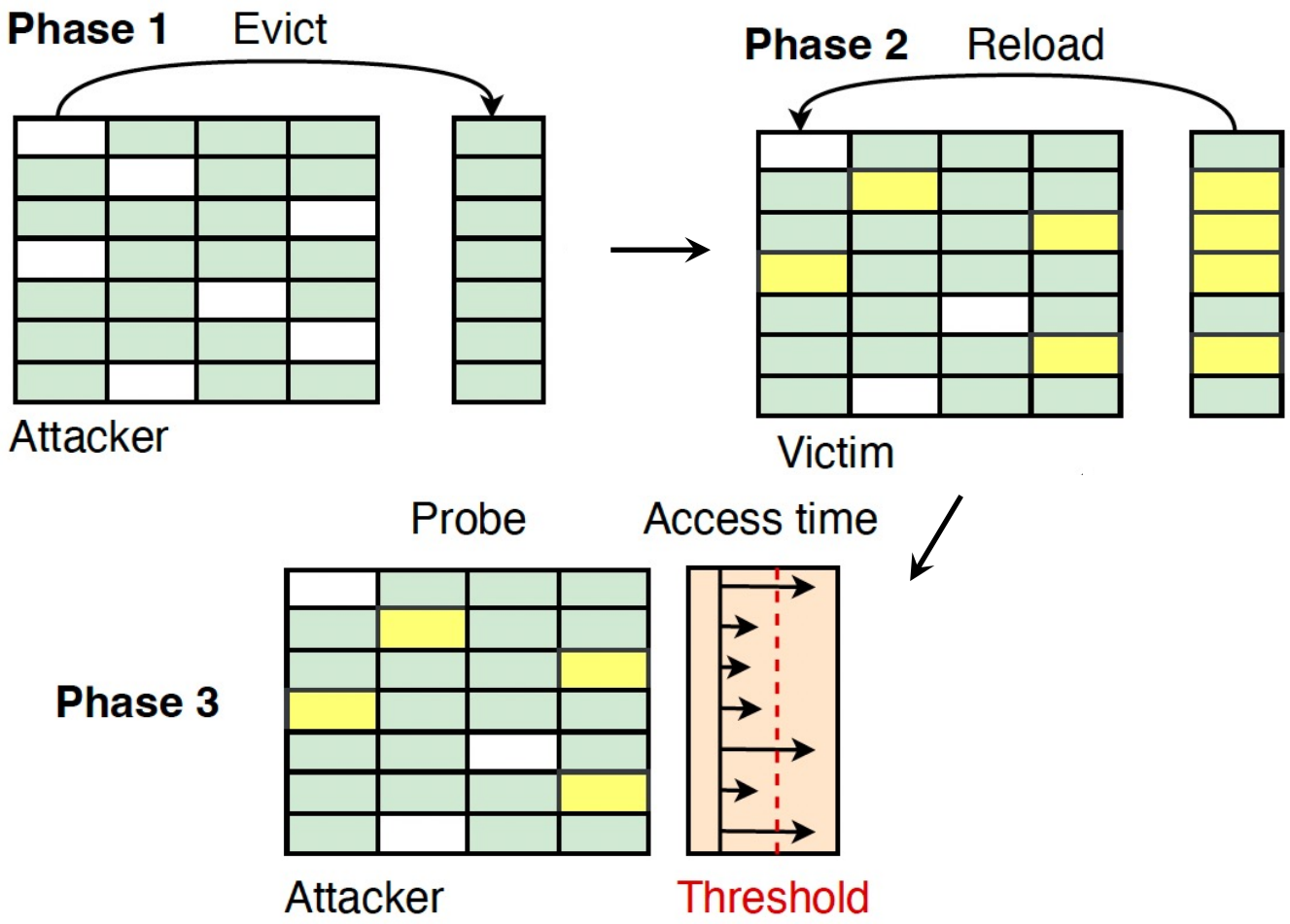


Spectre v1 [S&P' 19]

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Secret byte

Cache Side-channel Attack



Flush-Based Attack:

An attacker uses specialized machine instructions, such as the x86 CLFLUSH instruction used in Flush+Reload [USENIX Security'14], to flush the target cache line.

Flush+Flush [DIMVA' 16]

Flush+Coherence [HPCA'18]

Reload+Refresh [USENIX Security'20]

Conflict-Based Attack:

An attacker constructs a conflict set to evict the target cache line.

Evict+Reload [USENIX Security'16]

Prime+Probe [S&P'19]

Prime+Abort [USENIX Security'19]

Contents

Existing defense solutions

- Using serialization instructions within branches.
  - *Equivalent to completely disabling branch prediction*
- Invisible speculation

  InvisiSpec [MICRO'18] *SafeSpec* [DAC'19] *MuonTrap* [ISCA'20]
  - *Additional data structures to hide the cache lines accessed by the speculative loads*
  - *The data movement caused by the re-installing operations*
- Selective speculation

  STT [MICRO'19] *NDA* [ISCA'20] *SDO* [ISM'19] *SSE-RV* [CARRV'21]
  - *Complex tracking analysis logic*
  - *Untained technique cannot deprotect in time*
  - *A large taint file as a shadow structure for the PRF*

Problems

The combination of cache side-channel and speculative execution allows attackers to leak arbitrary data from the victim’s memory space

↓

Solutions

Sometimes block instructions without security threats

Require additional data structures and data movement operations, or complex logical implementation

→ **Significant overhead**

Most of the hardware defense solutions lack real hardware prototypes

Contents

Threat Model

Focus on eliminating cache side channels in speculative execution established by **flush- and conflict-based cache side-channel** attacks, which **require pre-evicting the cache lines to be leaked.**

- Attackers are allowed to run arbitrary code before and during the victim's executions to affect the victim's speculation.
- Attackers are aware of the cache index method and re-placement strategy, enabling them to evict target cache lines from the cache.
- Attackers can locate gadgets within the victim's executable memory space.

Design concept

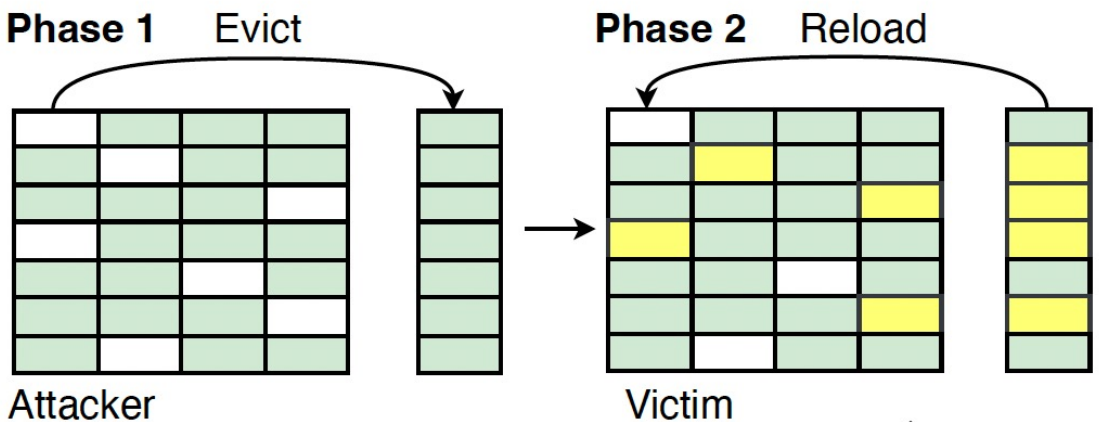
The larger the scope of protected instructions and the longer they are protected, the more dependent instructions will be blocked

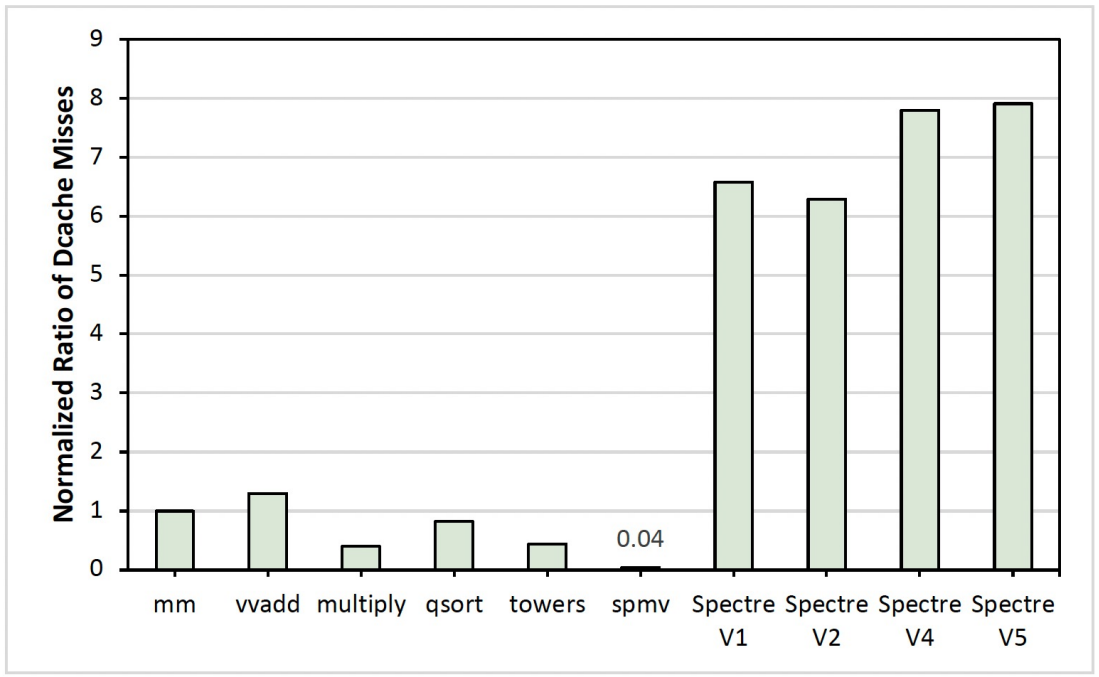
Protection scope **Which speculative loads should be protected?**

↓

All speculative loads could be exploited by attacker ?

Attack Characteristic Analysis





- *The cache lines that are exploited to leak data by the attacker must be under cache misses states in Phase 1.*
- *The victim's speculative loads which reload these cache lines thereby change the cache's occupancy state in Phase 2.*

The speculative loads causing cache misses are considered unsafe and should be protected.

Unsafe **S**peculative **l**oads caused cache **m**isses —— **MUSL**

Design concept

The larger the scope of protected instructions and the longer they are protected, the more dependent instructions will be blocked

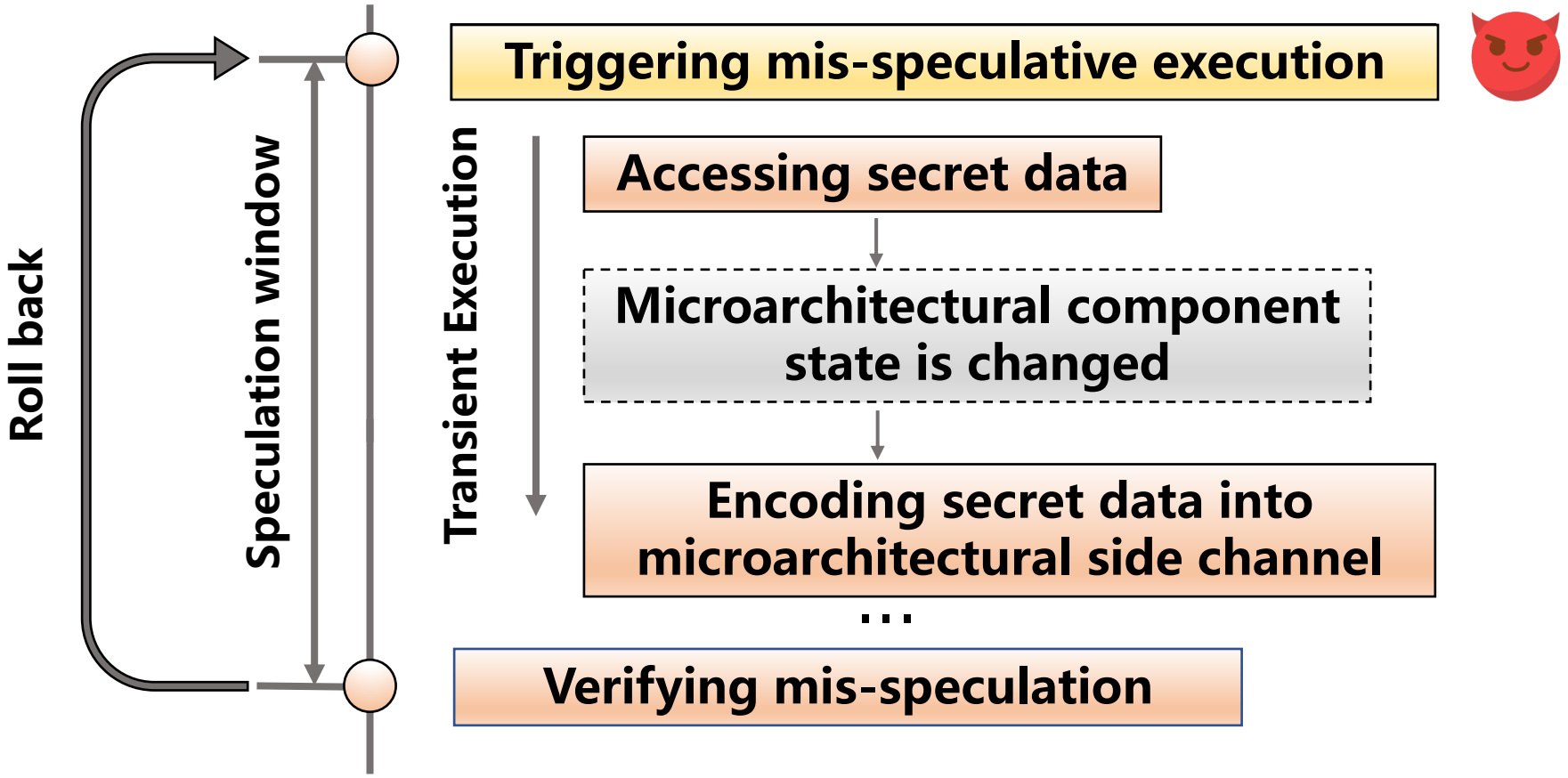
Protection scope — **Which speculative loads should be protected?**

↓

The speculative loads causing cache misses.

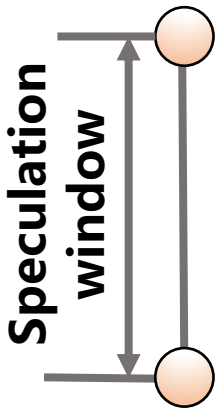
Protection duration — **When to protect and deprotect?**

Spectre Attack



Time to protect and deprotect

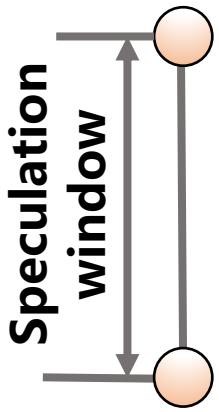
Unresolved Control Flow:



When control instructions (e.g. jumps, branches) enter the ROB

When the conditions and addresses can be verified by the processor

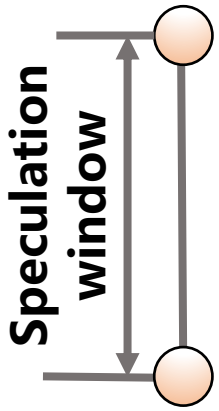
Unresolved Memory Access Order:



When stores with unresolved addresses enter the ROB

When the processor obtains the address of the store

Unresolved Value:



When the dependent instruction is executed using the predicted source operand value

When the predicted value can be verified by the processor

The loads within speculation window need to be protected

↓

The speculative loads under correct speculation need to be deprotected once the speculation window ends

Design concept

The larger the scope of protected instructions and the longer they are deprotected, the more dependent instructions will be blocked

Protection scope — **Which speculative loads should be protected?**

↓

The speculative loads causing cache misses.

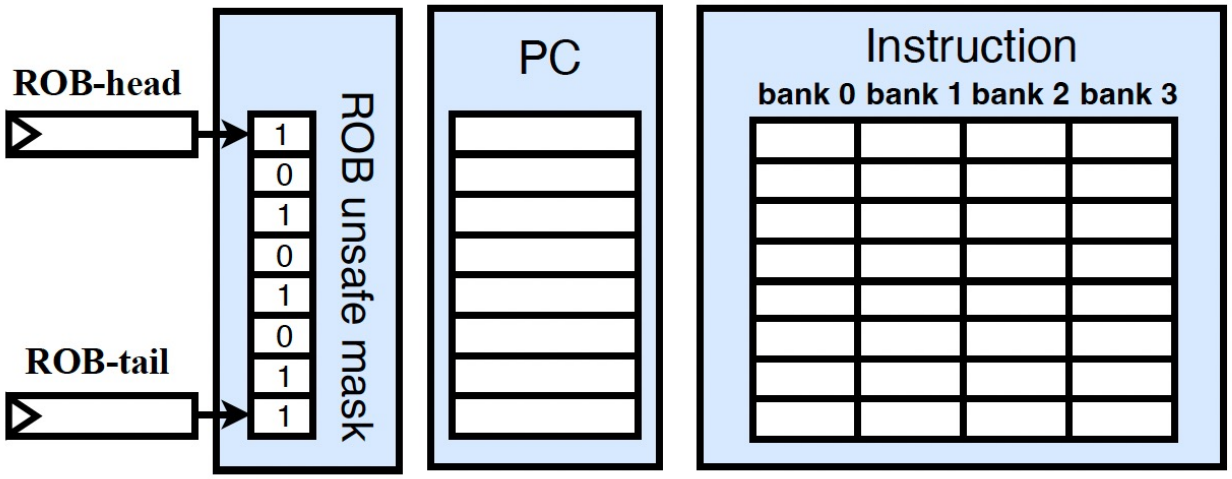
Protection duration — **When to protect and deprotect ?**

↓

Speculation windows caused by different speculation sources

Contents

ROB unsafe mask

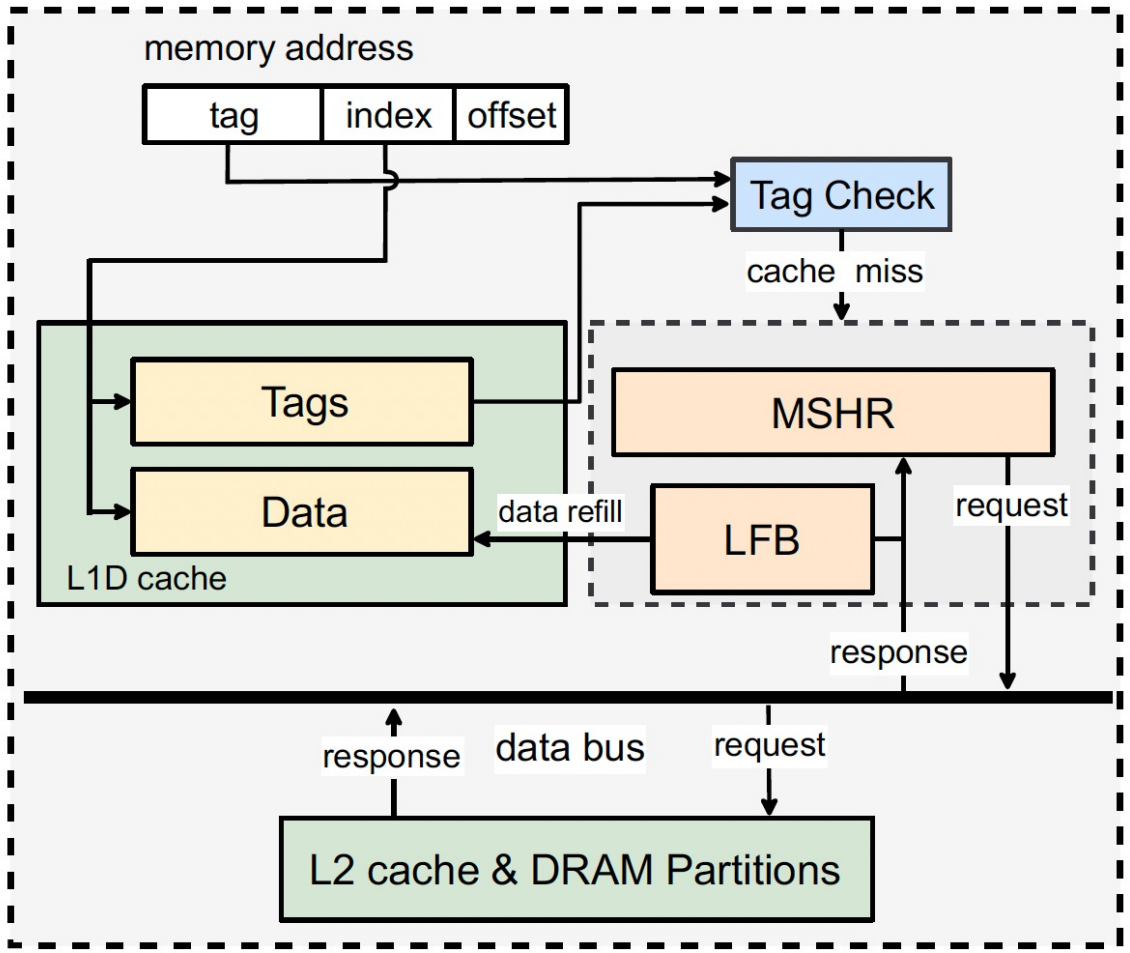


- *Each ROB rows are divided into multiple banks in multi-bank ROB architecture.*
- *Each entry of a bank stores the information of one µop.*
- *A one-to-one mapping exists between the ROB rows and the bits of ROB unsafe mask.*
- *The value of each bit in the ROB unsafe mask is jointly determined by the instructions in all banks of that ROB row.*

Number of ROB entries	M	
Number of banks	N	
ith bit of the ROB unsafe mask	r_i	$(i \in [0, M-1])$
jth bank of the ith entry of the ROB	b_{ij}	$(j \in [0, N-1])$

$$r_i = b_{i0} || b_{i1} || b_{i2} || \dots || b_{ij} || b_{i(N-1)}$$

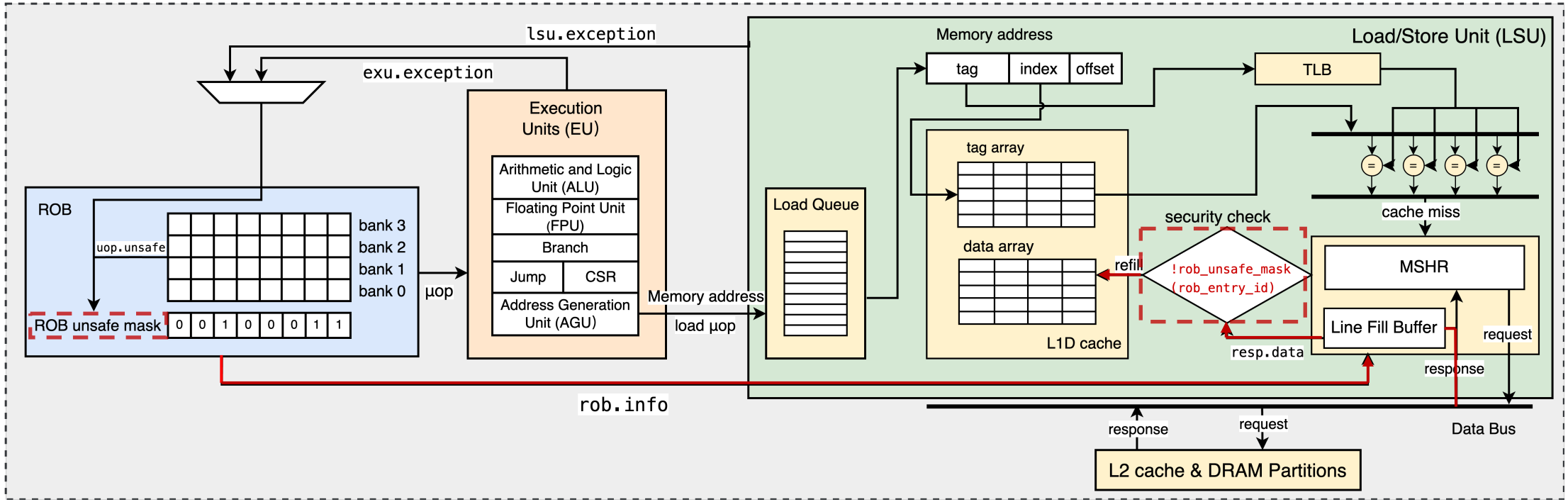
➢Line-fill-buffer (LFB)



Missing Status Holding Register (MSHR): Manages and sends data requests generated by cache misses to retrieve the missing cache line from lower-level caches or memory.

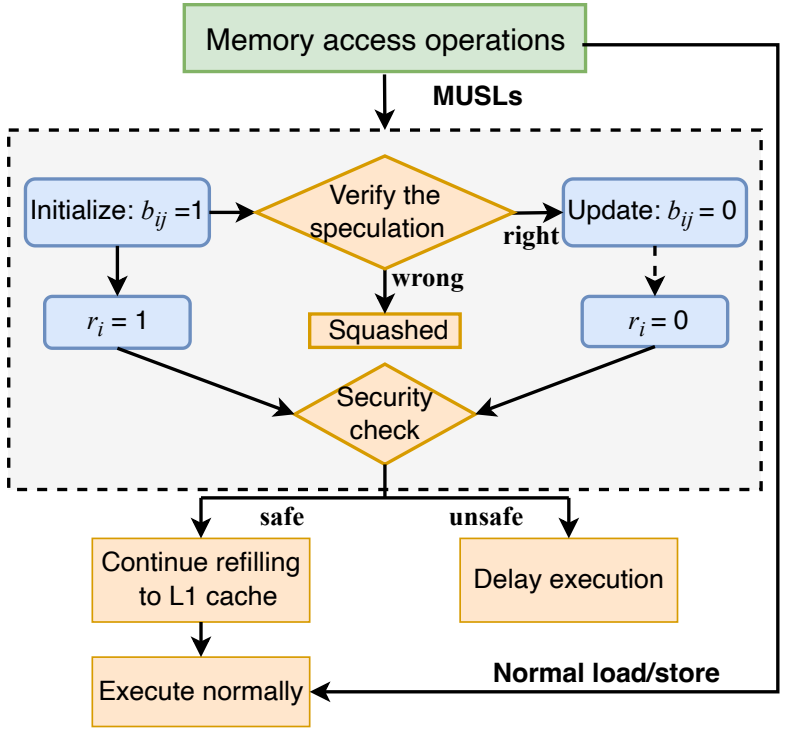
Line Fill Buffer (LFB): Temporarily stores retrieved cache lines by loads under cache misses, allowing cache eviction and refilling to occur in parallel, further reducing memory access latency caused by cache misses.

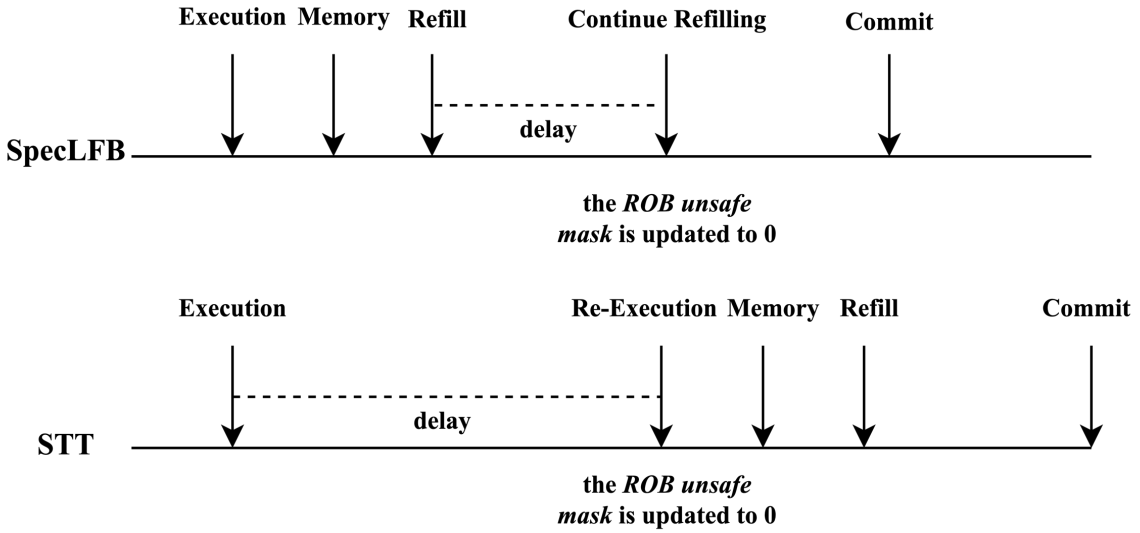
Security check mechanism



- The lower-level cache is requested by MSHRs to retrieve the missing data of loads under cache misses.
- The response data pass through the **security check mechanism** introduced to **LFB**.
- Only when the security check mechanism determines that the requested load is not a **MUSL**, can the data be **refilled into** the cache.

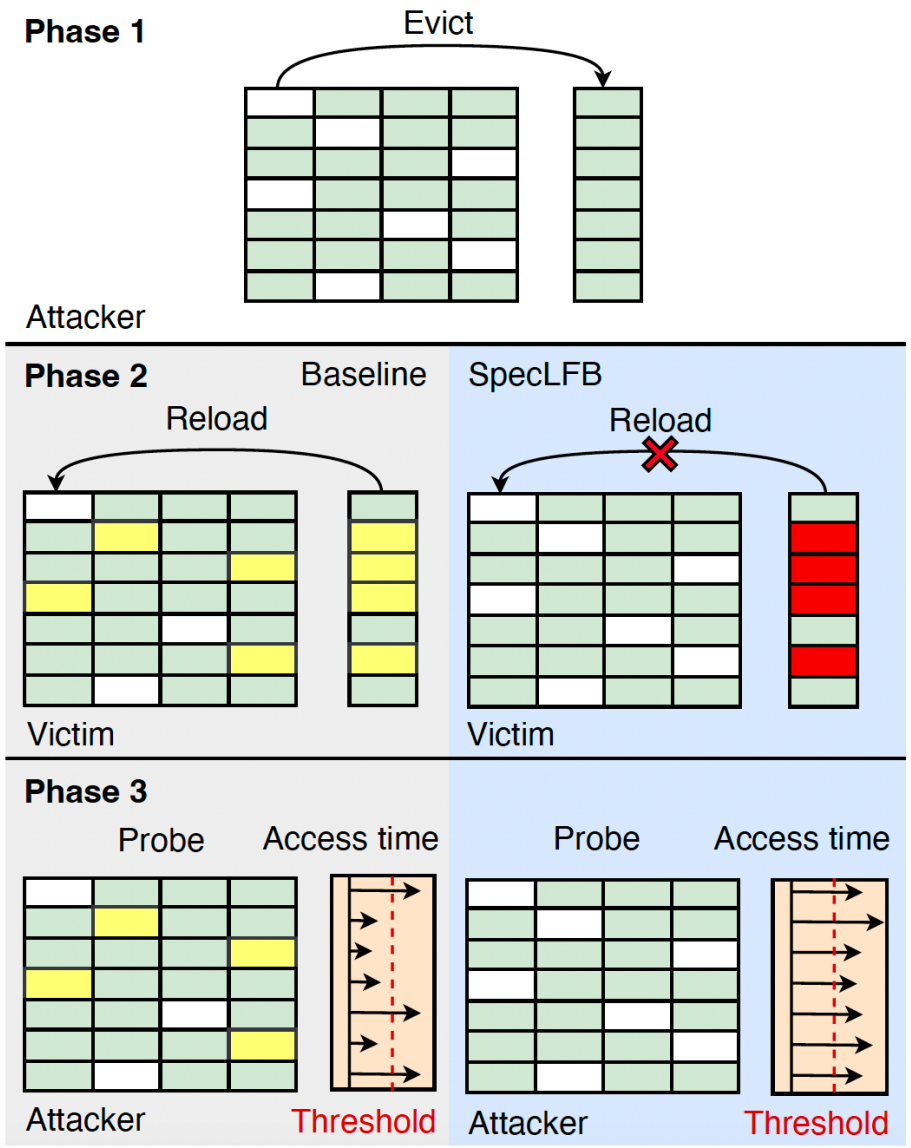
Security check mechanism





- During the waiting period for SpecLFB to update the ROB unsafe mask, the MSHR can simultaneously request data of MUSLs from the lower-level caches or memory
- As soon as the protection is disabled, the requested data can be refilled directly from the LFB into the cache instead of the lower-level caches or memory

SpecLFB Security Analysis



Phase 2:

Victim's load

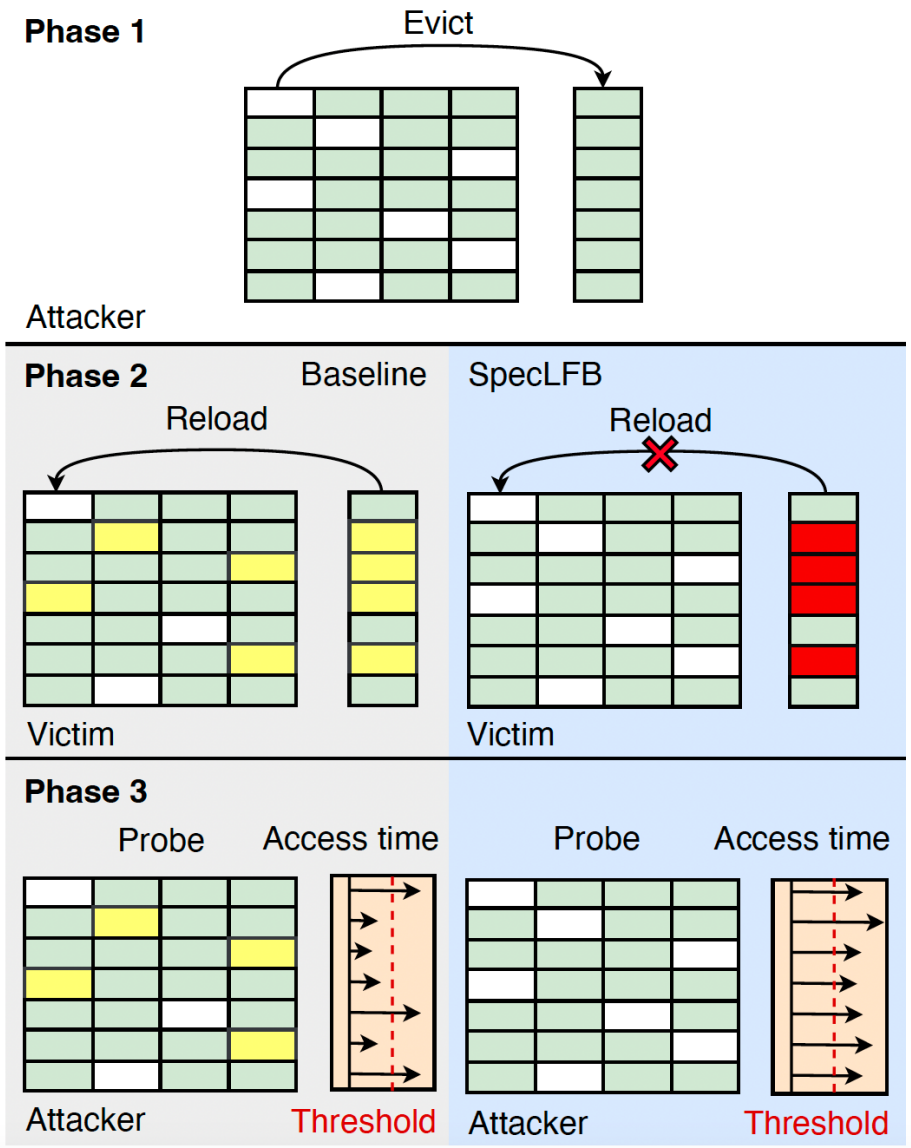
Is a cache miss generated? → **A cache refill occurs**

Is it within the speculation window? → **ROB unsafe mask bit is set to 1**

MUSL

*The cache lines requested by **MUSLs** cannot pass through the **security check in LFB** to be refilled into the cache*

SpecLFB Security Analysis



Phase 2:

Victim's load

Is a cache miss generated? → **A cache refill occurs**

Is it within the speculative window? → **ROB unsafe mask bit is set to 1**

MUSL

*The cache lines requested by **MUSLs** and cannot pass through the **security check in LFB** to be refilled into the cache*

Phase 3:

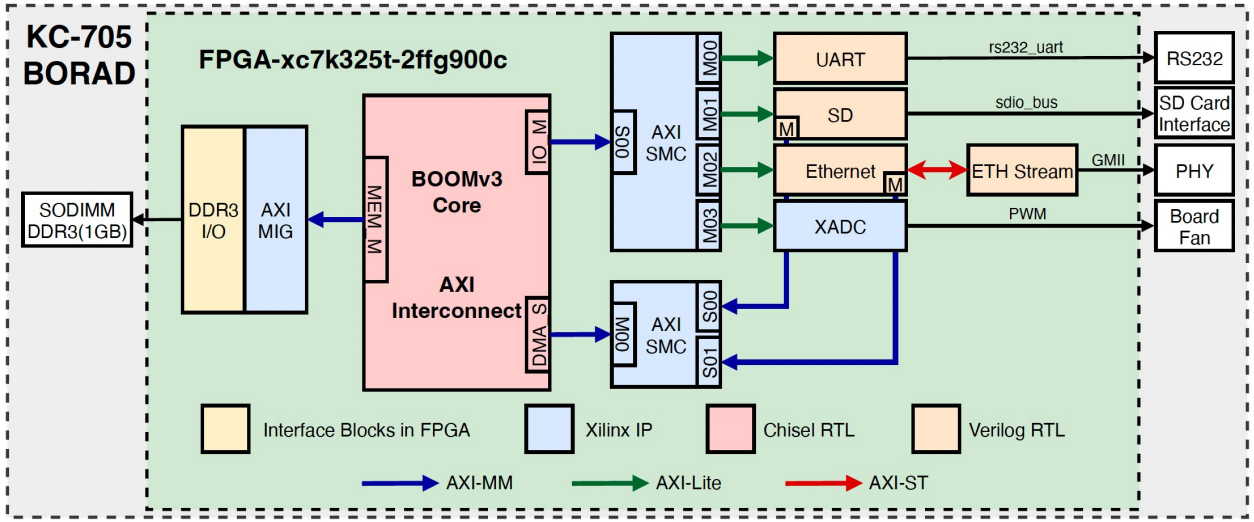
Attackers cannot probe access time differences

Contents

Experimental Setup

- *SpecLFB is implemented in the L1D cache of open-source RISC-V core, SonicBOOM and both cache levels of X86 O3 CPU model in Gem5 simulator.*
- *Hardware prototypes based on **Xilinx EK-KC-705 FPGA** platform burned with three **SonicBOOM cores** and running a Linux-kernel-based operating system **are develeped**.*

Processor	SonicBOOM Configurations	Gem5 Configurations
Baseline	Original SonicBOOM	Original O3 CPU
SSE-RV	SonicBOOM enhanced with SSE-RV	\
STT	\	O3 CPU enhanced with STT
SpecLFB	SonicBOOM enhanced with SpecLFB	O3 CPU enhanced with SpecLFB

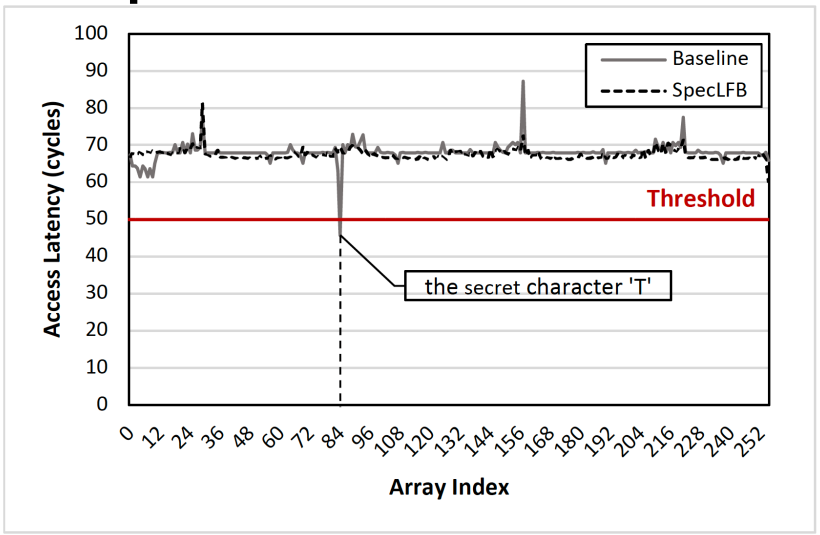


Parameter	SonicBOOM-FPGA	Gem5
ISA	RV64	X86-64
Frequency	FPGA@50MHz	simulate@2GHz
Processor type	2-decode 4-issue MediumBoom O3CPU	8-decode 8-issue DerivO3CPU
ROB/LDQ/STQ	64/16/16 entries	192/32/32 entries
L1I Cache	16KB, 4-way, 64B line	32KB, 8-way, 64B line, 4 MSHRs
L1D Cache	16KB, 4-way, 64B line, 2 MSHRs	32KB, 8-way, 64B line, 4 MSHRs
L2 Cache	512KB, 16-way, 64B line	2MB, 16-way, 64B line, 20 MSHRs

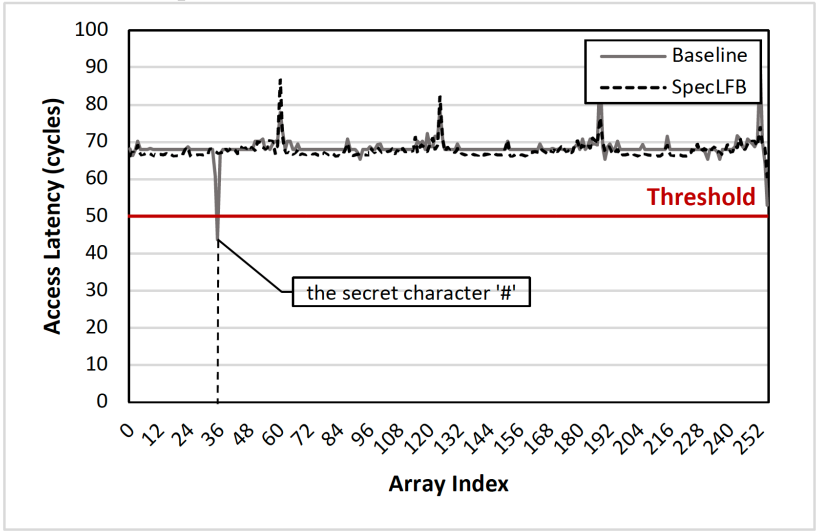
Security Evaluation

- Chipyard-SonicBOOM
- Gem5-X86 O3

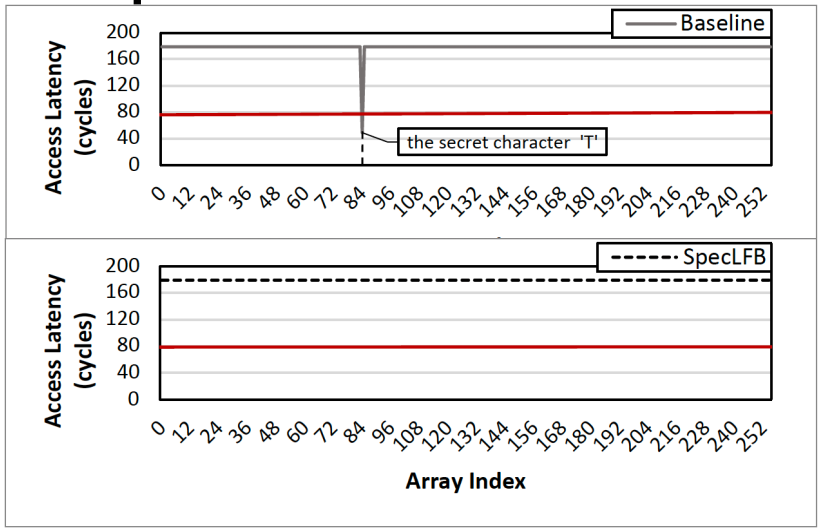
Spectre v1-Evict+Reload



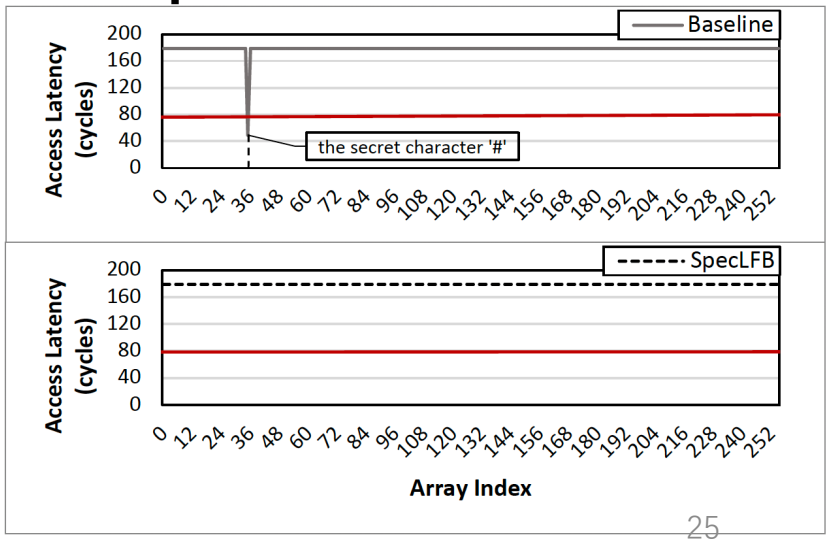
Spectre v4-Evict+reload



Spectre v1-Flush+Reload



Spectre v4-Flush+Reload



Performance Evaluation

Scheme	Performance overhead	Evaluation Method	Speculation			
			Value Prediction	Control Flow	Memory Access	Exception
STT-*Sp*	8.5%	Gem5	✗	✓	✗	✗
STT-*Fu*	14.5%		✓	✓	✓	✓
SpecLFB	3.20%		✗	✓	✓	✓
	1.85%	RTL-FPGA				
SSE-RV	4.8%		✗	✓	✓	✗

- ***Shorter protection time***
- ***Smaller protection scope***
- ***Shorter memory access path after deprotection***

FPGA resources utilization

Scheme	Core	Device	LUTs	FFs
Baseline	SonicBOOM	Xc7a325T	169,463	93,994
SSE-RV	SonicBOOM	Xc7a325T	172,538 (+1.81%)	94,567 (+0.61%)
SpecLFB	SonicBOOM	Xc7a325T	170,765 (+0.77%)	94,283 (+0.31%)

Tips:

LUTs (Look-Up-Tables) *are primarily used to implement logic circuit functionality.*

FFs (Flip-Flops) *are sequential logic elements used for storing and transferring binary data indigital circuits on FPGAs.*

- ***Lower LUT utilization.*** *The additional logic in the taint initialization and propagation stages in SSE-RV is more complex than the security mechanism of SpecLFB.*
- ***Lower FF utilization.*** *SSE-RV adds the taint file, propagation queue, and other intermediate registers related to taint tracking technique, while SpecLFB only adds intermediate registers related to the ROB unsafe mask.*

Contents

Conclusion

- Limit the scope of unsafe speculative loads that need to be delayed to **MUSLs**, which denotes unsafe speculative loads that cause cache misses.
- Introduce a simple yet effective **security check mechanism** that works inconjuction with the **ROB unsafe mask** to the LFB, preventing the cache refill of MUSLs.
- Implemented both in **SonicBOOM** and **Gem5 O3 processor**.
- average hardware resource overhead: **0.6%**

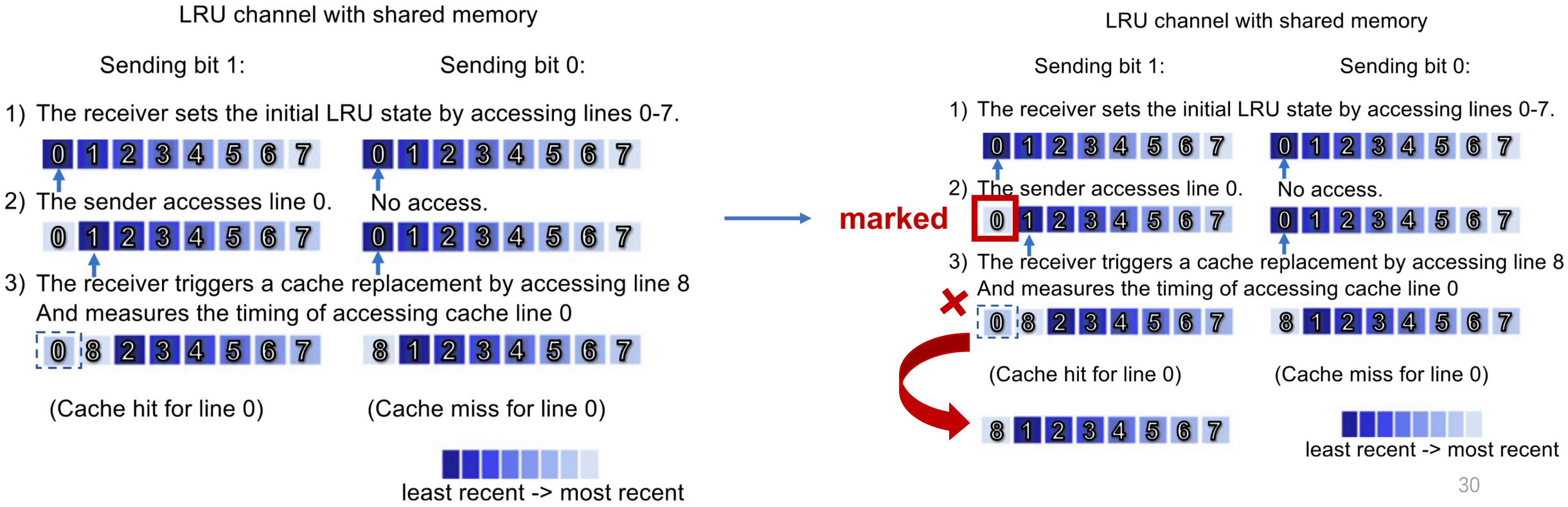
  performance overhead: **1.85% in the FPGA hardware prototype experiment**

  3.20% in the Gem5 simulation

Discussion

Speculative accesses resulting in cache hits. Where a cache replacement (i.e., a cache miss) by the sender is not required to change the cache state.

Example **Speculative Cache side-channel attack based on Least-Recently-Used (LRU) replacement policy [TC' 21]**



SpecLFB: Eliminating Cache Side Channels in Speculative Executions

Xiaoyu Cheng, Fei Tong, Hongyu Wang, Zhe Zhou,
Fang Jiang, and Yuxing Mao

Welcome for questions !

Email: xiaoyu_cheng@seu.edu.cn

References

[1] YAROM, Y., ANDFALKNER, K. FLUSH+RELOAD: A high resolution, low noise, l3 cache side-channel attack. In 2014 USENIX Security Symposium (USENIX Security), pp. 719–732.
[2] GRUSS, D., MAURICE, C., WAGNER, K., ANDMANGARD, S. Flush+Flush: a fast and stealthy cache attack. In Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7-8, 2016, Proceedings13, Springer, pp. 279–299.
[3] YAO, F., DOROSLOVACKI, M.,ANDVENKATARAMANI, G. Are coherence protocol states vulnerable to information leakage? In 2018 IEEE International Symposium on High Performance Computer Architecture(HPCA), IEEE, pp. 168–179.
[4] BRIONGOS, S., MALAGÓN, P., MOYA, J. M., ANDEISENBARTH,T. Reload+Refresh: Abusing cache replacement policies to perform stealthy cache attacks. In2020 USENIX Security Symposium (USENIX Security), pp. 1967–1984.
[5] LIPP, M., GRUSS, D., SPREITZER, R., MAURICE, C.,ANDMAN-GARD, S. Armageddon: Cache attacks on mobile devices. In2016USENIX Security Symposium (USENIX Security), pp. 549–564.
[6] IRAZOQUI, G., INCI, M. S., EISENBARTH, T., ANDSUNAR, B. Wait a minute! a fast, cross-vm attack on aes. In Research in Attacks, Intrusions and Defenses: 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings 17, Springer, pp. 299–319.
[7] YAN, M., SPRABERY, R., GOPIREDDY, B., FLETCHER, C., CAMP-BELL, R., ANDTORRELLAS, J. Attack directories, not caches: Side channel attacks in a non-inclusive world. In 2019 IEEE Symposium on Security and Privacy (S&P), IEEE, pp. 888–904.
[8]CANELLA, C., VANBULCK, J., SCHWARZ, M., LIPP, M., VONBERG,B., ORTNER, P., PIESSENS, F., EVTYUSHKIN, D., ANDGRUSS, D. A systematic evaluation of transient execution attacks and defenses. In 2019 USENIX Security Symposium (USENIX Security), pp. 249–266.
[9] KOCHER, P., HORN, J., FOGH, A., GENKIN, D., GRUSS, D., HAAS,W., HAMBURG, M., LIPP, M., MANGARD, S., PRESCHER, T., ET AL. Spectre attacks: Exploiting speculative execution. Communications of the ACM 63, 7 (2020), 93–101.
[10] HORN, ANDJANN. Speculative Execution, variant 4: Speculative store bypass, 2018. https://bugs.chromium.org/p/project-zero/issues/detail?id=1528.
[11] KORUYEH, E. M., KHASAWNEH, K. N., SONG, C., ANDABU-GHAZALEH, N. B. Spectre Returns! Speculation Attacks using the Return Stack Buffer. In 2018 USENIX Security Symposium (USENIX Security)
[12] YAN, M., CHOI, J., SKARLATOS, D., MORRISON, A., FLETCHER,C.,ANDTORRELLAS, J. InvisiSpec: Making speculative execution invisible in the cache hierarchy. In 2018 ACM/IEEE 51th International Symposium on Microarchitecture (MICRO), pp. 428–441.
[13] KHASAWNEH, K. N., KORUYEH, E. M., SONG, C., EVTYUSHKIN,D., PONOMAREV, D., ANDABU-GHAZALEH, N. SafeSpec: Banishing the spectre of a meltdown with leakage-free speculation. In 2019 ACM/IEEE 56th Design Automation Conference (DAC), IEEE, pp. 1–6.
[14] AINSWORTH, S.,ANDJONES, T. M. MuonTrap: Preventing cross-domain spectre-like attacks by capturing speculative state. In 2020 ACM/IEEE 47th International Symposium on Computer Architecture (ISCA), IEEE, pp. 132–144.
[15] YU, J., YAN, M., KHYZHA, A., MORRISON, A., TORRELLAS, J.,ANDFLETCHER, C. W. Speculative taint tracking (STT) a comprehensive protection for speculatively accessed data. In2019 ACM/IEEE 52thInternational Symposium on Microarchitecture (MICRO), pp. 954–968.
[16] WEISSE, O., NEAL, I., LOUGHLIN, K., WENISCH, T. F., ANDKASIKCI, B. NDA: Preventing speculative execution attacks at their source. In 2019 ACM/IEEE 52th International Symposium on Microarchitecture, pp. 572–586.
[17] YU, J., MANTRI, N., TORRELLAS, J., MORRISON, A.,ANDFLETCHER, C. W. Speculative data-oblivious execution: Mobilizing safe prediction for safe and efficient speculative execution. In 2020 ACM/IEEE 47th International Symposium on Computer Architecture (ISCA), IEEE, pp. 707–720.
[18] SABBAGH, M., ANDFEI, Y. Secure speculative execution via riscv open hardware design. In 2021 Fifth Workshop on Computer Architecture Research with RISC-V (CARRV).
[19] XIONG, W., KATZENBEISSER, S., ANDSZEFER, J. Leaking information through cache lru states in commercial processors and secure caches. IEEE Transactions on Computers 70, 4 (2021), 511–523.