SC-K9: A Self-synchronizing Detection Framework to Counter Microarchitecture Side Channels

Hongyu Fang, Milos Doroslovacki, Guru Venkataramani

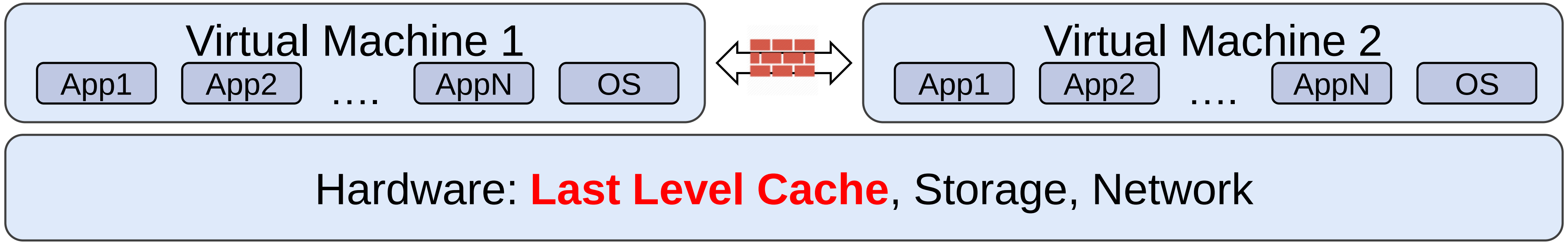
The George Washington University

Washington, DC, USA

Acknowledgments: US National Science Foundation, Office of Naval Research

Security Challenge on Shared Platforms

- Applications from mutually **untrusted** sources share one physical machine.
- Shared hardware (last level cache, random number generator, and GPU) can be media of information leakage.



Challenges to Defend against Cache Timing Channel

- The only thing adversaries do is to modulate their accesses to microarchitecture.
- Shared microarchitecture cannot be disabled without performance degradation.
- Microarchitecture side channel can be implemented with various protocols.

Pros and Cons of Prior Defense Mechanisms

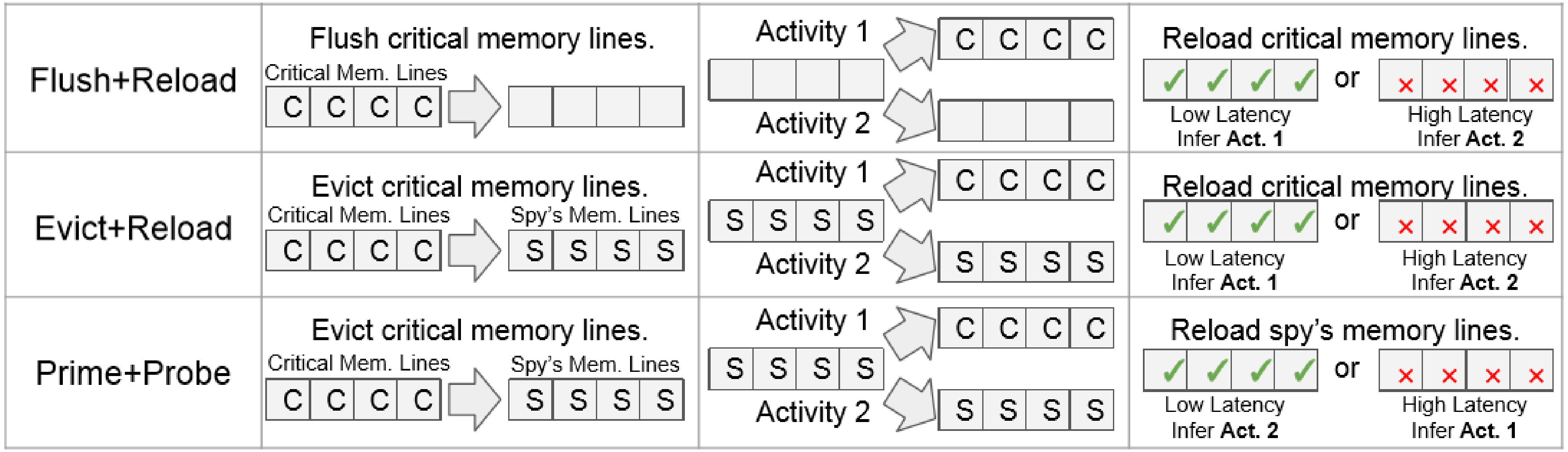
- Microarchitecture Partitioning:
  - Pros: Straightforward mitigation with existing hardware.
  - Cons: Either requires SW-HW co-design or impact performance of benign workloads.
- Secured Hardware Design:
  - Pros: Defense without limiting available hard resource of each process.
  - Cons: Complicated to implement; Annul the optimizations.
- Detection:
  - Pros: On-demand protection without influence on benign workloads.
  - Cons: High false positive penalty; May be evaded by advanced spy.

Typical Iteration of Information Leakage

- **Spy's Setup**: Setup hardware status to make future activities of victim observable.
- **Victim's Leakage**: Victim's secret-dependent activities change hardware status.
- **Spy's Observation**: Spy observes status changed by victim and decode the secret.

Example Iteration of Cache Side Channel

- **All cache timing channel attacks involved three phases:**
  - **Spy's Setup:** Spy removes critical memory lines from cache.
  - **Victim's Leakage:** Victim accesses critical memory lines.
  - **Spy's Observation:** Spy reloads memory lines and measures latency.

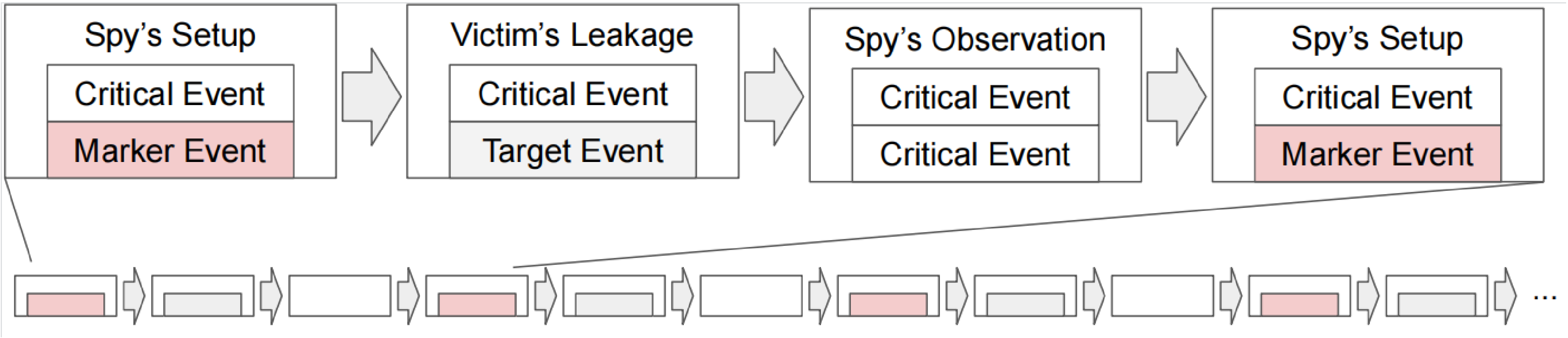


Iterations of Various Side Channels

Attack Variant	Spy's Setup	Victim's Leakage	Spy's Observation
BranchScope[4]	Spy manipulates predictor status	Victim executes branch	Spy executes primed branches
TLBleed[10]	Spy occupies TLB set with its addrs.	Victim accesses memory lines	Spy accesses occupied TLB
Cache Prime+Probe[15]	Spy occupies a cache set	Victim accesses occupied cache set	Spy probes the cache set
Cache Flush+Reload[22]	Spy flushes victim mem. lines	Victim accesses victim mem. lines	Spy reloads flushed mem. lines
Speculation-based side channel[13, 14]	Spy flushes exploited array	Victim transiently loads secret-dependent addr.	Spy reloads exploited array

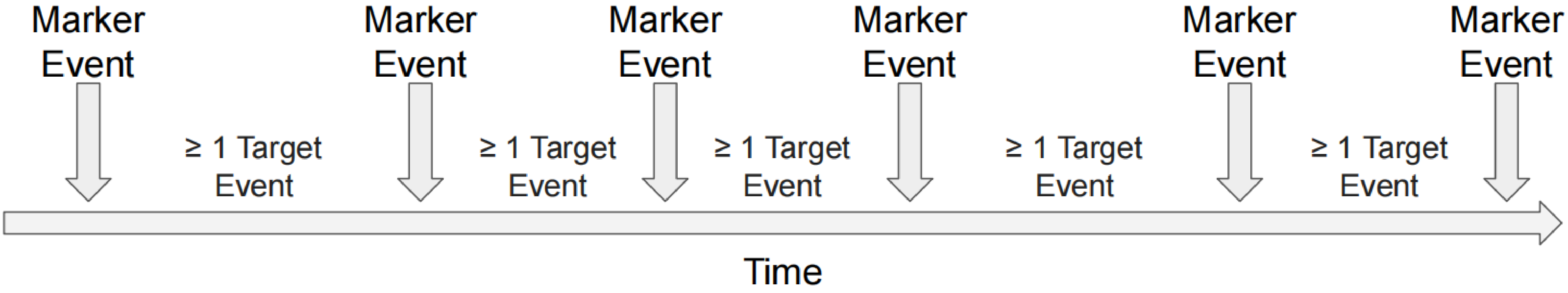
Capturing Iterations of Information Leakage

- **Marker Event:** a critical event which appears in no less than half of iterations of a side channel attack
- **Target Event:** a series of events that occur inbetween marker events.



Capturing Iterations of Information Leakage

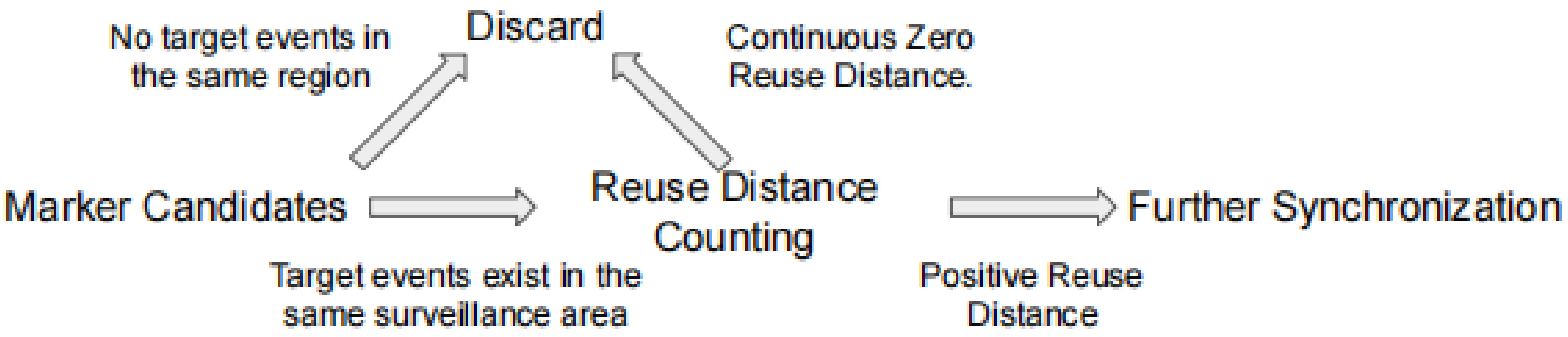
- **Reuse Distance:** The number of target events between a pair of repetitive marker events.
- Multiple positive reuse distance value would be observed in side channels.



Event Pattern of Typical Side Channel

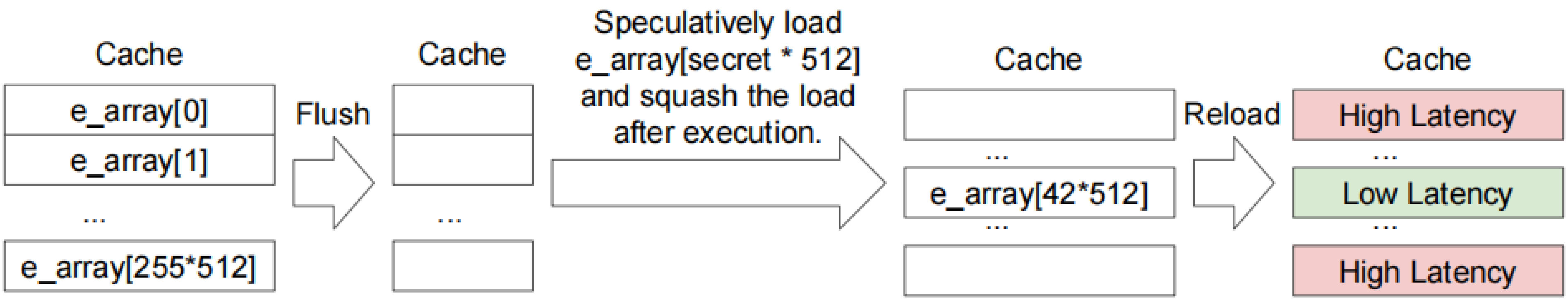
Filtering the Events

- For some types of microarchitecture side channels, marker events of a side channel could happen within specific regions.
- We define these hardware events that has the same event type with the marker events as marker candidate.
- Aggressive filtering methods are needed before detection in order to reduce the number of marker candidates.



Case Study: Detecting Speculation-based Attack

- Typical Implementation of Speculation-based Side Channel.



Case Study: Detecting Speculation-based Attack

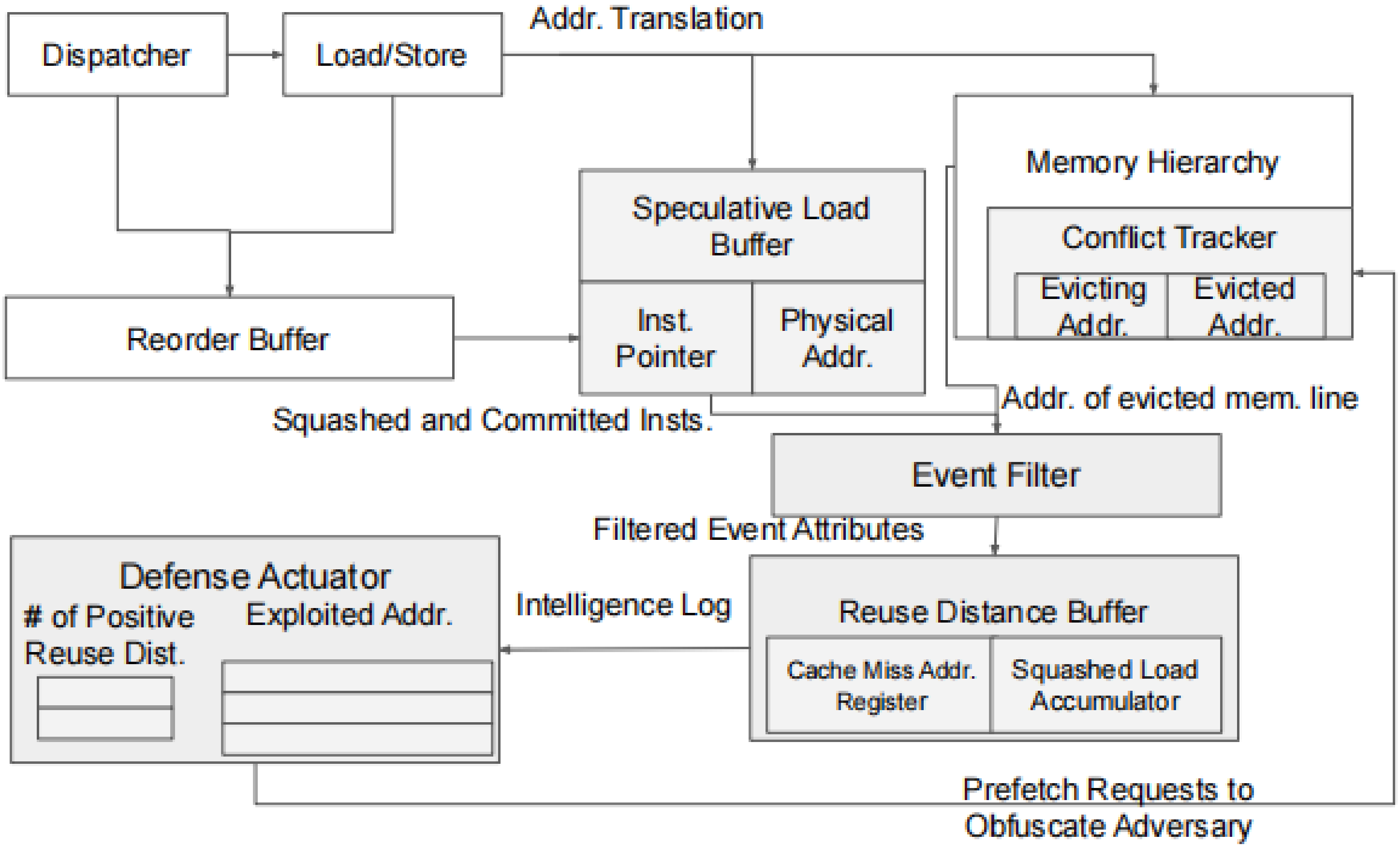
- Repetitive Activities of Speculation-based Side Channel

Attack Variant	Byte/Iteration	Byte Accuracy
Spectre v1	1	99%
Spectre v2	4	98%
Meltdown	1	94%
Foreshadow	1	70 - 99%

- Event Definition
    - Marker Event: Conflict Misses
    - Target Event: Mis-speculated Load Instructions

Case Study: Detecting Speculation-based Attack

- Overall Design



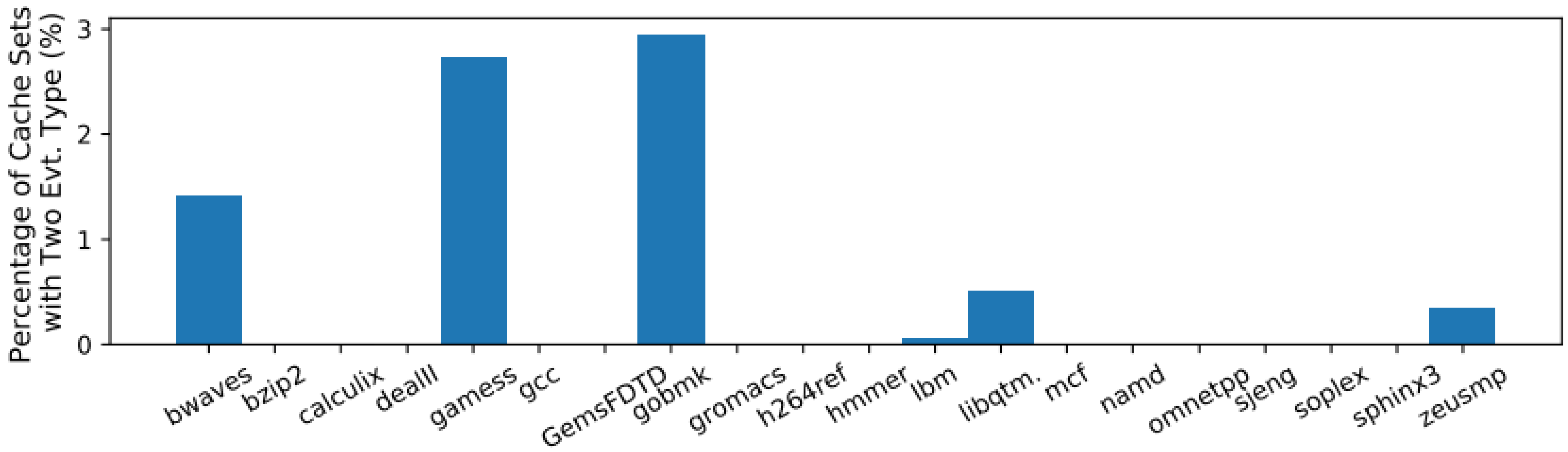
Case Study: Detecting Speculation-based Attack

- Experimental Setups
  - Gem5 with four x86 cores, 32 KB private L1 and 4 MB, 8-way shared L2 caches.
  - Implemnt Spectre v1 and v2 to evaluate our design.
  - Both adversaries repeat attack iteration 100 times for single byte
  - Both adversaries manage to steal 40 bytes of the secret.
  - We implement adversary with different transmission rates.

Attack Variant	Iteration/Second
Spectre-vx-1	0.5k
Spectre-vx-2	1.5k
Spectre-vx-3	3k
Spectre-vx-4	5k
Spectre-vx-5	10k

Case Study: Detecting Speculation-based Attack

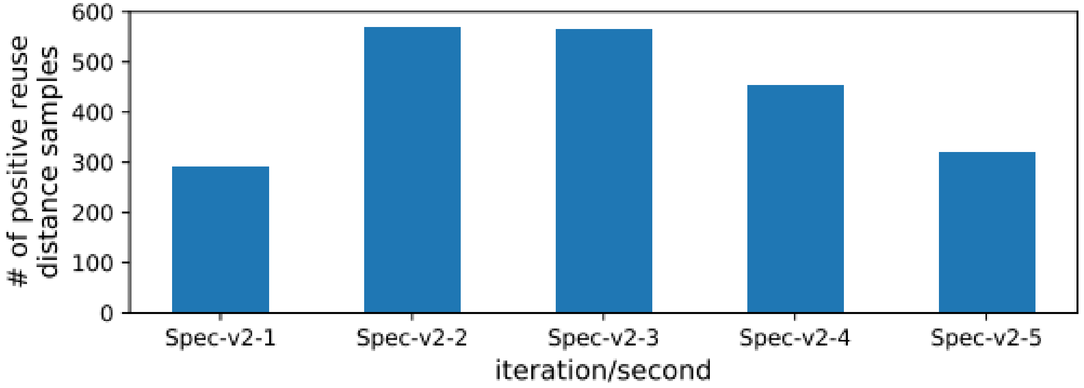
- Efficiency of Event Filtering



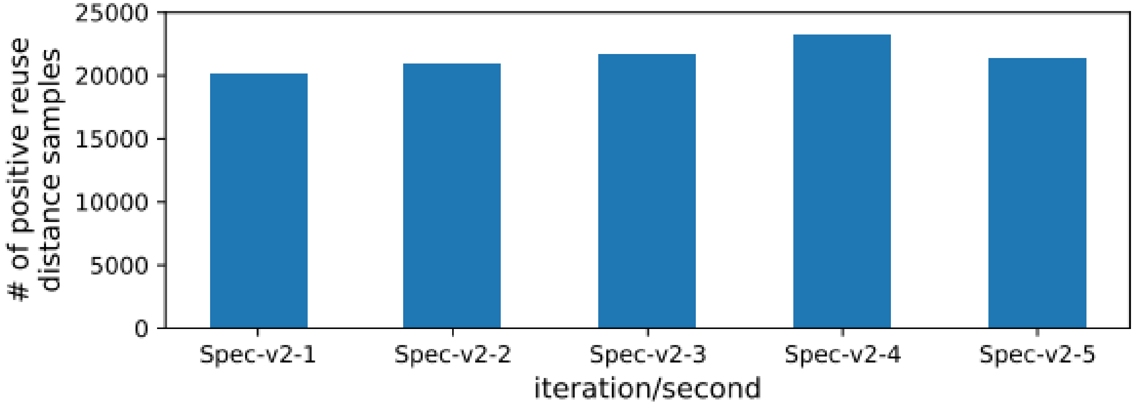
The first level event filtering remove **97%** of cache sets potentially with one of the events

Case Study: Detecting Speculation-based Attack

- Number of Positive Reuse Distance Observed in Speculation-based Side Channels



(a) Number of positive reuse distance samples of Spectre v1 in different frequencies



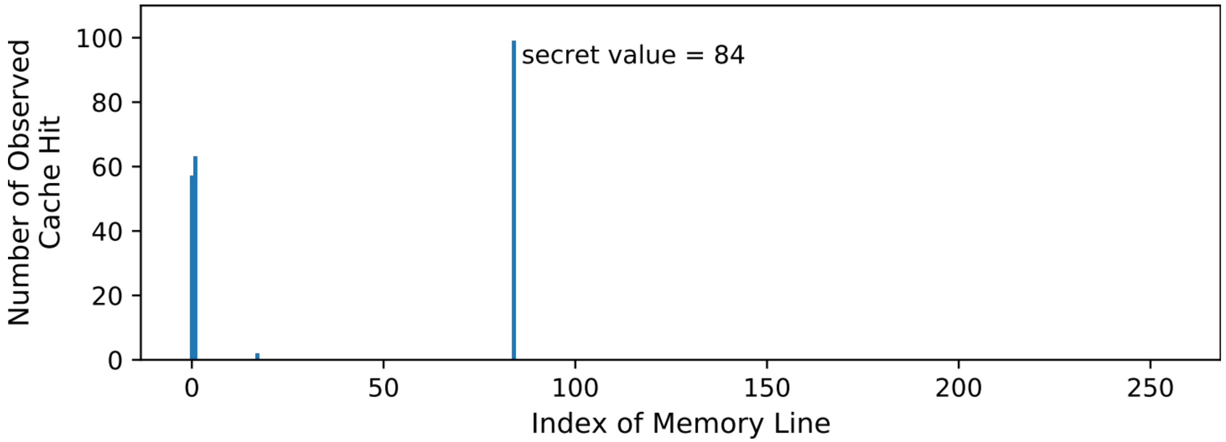
(b) Number of positive reuse distance samples of Spectre v2 in different frequencies

- Number of Positive Reuse Distance Observed in Benign Workloads

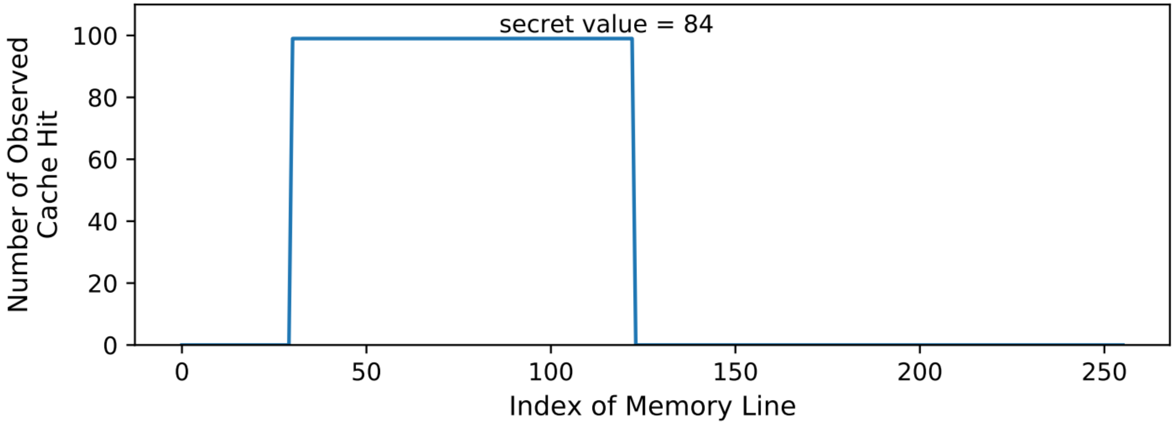
Process Name	Count of Positive Reuse Distances
benignSpec v1	0
benignSpec v2	0
hmmer	3
Other SPEC2006	0

Case Study: Detecting Speculation-based Attack

- Obfuscating Side Channel using Prefetcher
  - SC-K9 provides rich information for further defense.
  - In this case study, we leverage prefetcher to obfuscate victim's leakage phase.



Before Obfuscation



After Obfuscation

Conclusion

- We leverage the fundamental behavior of side channels and develop a generic framework to capture the repetitive interference observed in these attacks.
- We evaluate SC-K9 using recently notorious case study: speculation-based cache.
- Our experimental results show that SC-K9 can effectively distinguish adversaries from various types of benign workloads with high accuracy.
- Our evaluation shows that the information provided by SC-K9 can be used in efficient defense mechanism, which can make it difficult or impossible for the spy to recover any leaked secrets.