ReViCe: Reusing Victim Cache to Prevent Speculative Cache Leakage

Sungkeun Kim, Farabi Mahmud, Jiayi Huang, Pritam Majumder, Neophytos Christou*, Abdullah Muzahid, Chia-Che Tsai, Eun Jung Kim

Texas A&M University   *University Of Cyprus

#IEEESecDev   https://secdev.ieee.org/2020
Vulnerable Performance Optimization

- Attackers can access the secret through speculative execution.
- Attackers transmit the secret through cache side channel.
Problem: Speculation Based Attacks (Spectre V1)

- Mis-speculative Execution Due to Mistraining
- Leak Secret
- Non-speculative Execution

Victim Function

```
Array[ secret ];
```

Install

Cache Blocks

```
For i = 0 to n
    T1 = time();
    Array[i];
    T2 = time() - T1;
```

Attacker
Solution: ReViCe - An Undo-Based Mitigation

Mis-speculative Execution Due to Mistrain

Array[ secret ];

Install

Leak Secret

Cache Side Channel

Non-speculative Execution

For i = 0 to n
T1 = time();
Array[i];
T2 = time() – T1;

Victim Function

Allow Early Update
Hide Using Jitter

Cache Blocks

Restore Using Victim Cache

Attacker
ReViCe – Motivations
Prior work – Redo VS. Undo

- Delay update until Branch resolution
- Penalized by correctly speculated load.

Redo - InvisiSpec [Yan et al. MICRO `18]

- Early Update on Response
- Penalized by incorrectly speculated load.

Undo - [Saileshwar et al. MICRO `19]

Prediction is very accurate
Prior work – CleanupSpec [Saileshwar et al. MICRO `19]

Speculative Executions

Core

Access cache block A

Cleanup on Mis-Speculation

Eviction

Install

L1D$

A

Core

A

Invalidation

L1D$

B

Restore

L2$

A

No Speculation

During Speculation
Threat model

- Mis-Speculative load can access the secret.
- Cache side channel transmits the secret.
- Attacker has access to the source code of the victim program
- OS is correct and trusted by the victim.

- Out of Scope
  - Other side channels: TLB, Branch Prediction History
  - Foreshadow
ReViCe - Design
Jitter – Mimics cache miss to hide speculation

1. Speculative Load [A]
2. Load [A]
3. Response after Jitter length

.cache

Speculative Line [A]

Mimics Cache miss

Core

Cache

Allow Early Update
Hide Using Jitter

Restore
Using Victim Cache
Victim Cache – Confirm Correct Speculative Changes

1. Speculative Load [B]
2. Victimize
3. Conform [B]
4. Clear STag
5. Update Replacement state

Consistent access latency is guaranteed.

[Image: Diagram showing the process of Victim Cache with Tag A and Tag B, and the steps for confirming correct speculative changes.]
Victim Cache – Restore Speculative Changes

1. Speculative Load [B]
2. Victimize
3. Restore [B]
4. Restore
Delayed Downgrade Coherence State [Yao et al. HPCA `18]

Core 0 (Requester)  Core 1 (Owner)  Core 0 (Requester)  Core 1 (Sharer)

L1  E/M  L1  S

On-chip Network  On-chip Network  Owner

Shared Cache

Downgrade E/M → S
ReViCe – Evaluation
ReViCe – Evaluation Methodology

- Simulation based
  - gem5 full system simulator
  - Out of order processor (Single, Octa cores)
- Proof-of-concept (4 x 3 x 2 = 24 attack programs)
  - Four Spectre Variants
  - Three Cache Side Channels
  - Same Core and Cross Cores
- Performance evaluation
  - SPEC2017, PARSEC
  - Compared against InvisiSpec, Selective Delay, CleanupSpec
ReViCe – Performance Overhead (SPEC2017)

Normalized Performance Overhead (%)

- 12.6
- 8.6
- 8.8
- 5.8

InvisiSpec  Selective Delay  CleanupSpec  ReViCe

Redo-Based  Undo-Based

Details in the paper

Dummy Request and Fixed window Size
ReViCe – Conclusion

- Problem: Mitigating Speculation based attack leveraging cache side channel.
- Prior works: Either high overhead or incomplete
- Key insights: Hide speculation using Jitter and Restore from Victim Cache.
- ReViCe is secure with better performance.
Thank you

Sungkeun Kim
ksungkeun84@tamu.edu