Last-Level Cache Side-Channel Attacks Are Feasible in the Modern Public Cloud

Zirui Neil Zhao, Adam Morrison, Christopher W. Fletcher, Josep Torrellas
University of Illinois       Tel Aviv University

ASPLOS ’24 – Session 2B: Side Channels
Shared Last-Level Cache (LLC) Enables Information Leakage

Liu et al., Last-Level Cache Side-Channel Attacks are Practical (S&P 2015)
Cloud Vendors Claim LLC Prime+Probe is Impractical

**Example:** AWS Whitepaper – The Security Design of the AWS Nitro System (Version November 18, 2022)

*Paraphrased:*

Last-level cache (LLC) Prime+Probe is impractical due to the noise; therefore, our side-channel mitigations are **very strong** even if we do not protect VMs against LLC Prime+Probe
Attacker’s Challenge 1: Production Cloud is Noisy

Victim

Non-victim

Non-victim

Victim

Attacker

*Characters are based on https://xkcd.com/2176 and https://xkcd.com/1808 (under a CC Attribution-NonCommercial 2.5 License)
Attacker’s Challenge 2: Modern Clouds (e.g., FaaS) are Dynamic

*Characters are based on https://xkcd.com/2176 and https://xkcd.com/1808 (under a CC Attribution-NonCommercial 2.5 License)
Contributions of This Work (at 31,000 Feet)

Cross-tenant information leakage with LLC Prime+Probe

1. Fast LLC Channel Setup
   > 10 hours → 2.4 minutes

2. Noise-Resilient Victim Monitoring

3. Information Extraction

+ Google filed a critical-level bug to their product team
+ AWS revised their whitepaper on February 15, 2024
**Threat Model & Assumptions**

**✓ Step 1: Co-Location**

- Victim
- Attacker

**✓ Everywhere All at Once: Co-Location**

Attacks on Public Cloud FaaS

*(ASPLOS ’24 – Session 1D)*

**Step 2: LLC Prime+Probe**

- Victim
- Attacker

- Prime+Probe
Threat Model & Assumptions

Step 2: LLC Prime+Probe

Assumption 1: Co-location

Assumption 2: Victim is running during the attack

Assumption 3: Prior knowledge of victim’s cache access behavior
Background: LLC Prime+Probe Attack

**Eviction set:** A set of cache lines that fully occupy a cache set

4-way set-associative last-level cache (LLC)

Victim

Attacker
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**

Victim

Attacker

Eviction Set

4-way set-associative last-level cache (LLC)
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**

4-way set-associative last-level cache (LLC)
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**

4-way set-associative last-level cache (LLC)

Cache hit!  Low latency
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**

4-way set-associative last-level cache (LLC)
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with Prime+Probe
Background: LLC Prime+Probe Attack

Eviction set ⇒ Monitor memory accesses to an LLC set with **Prime+Probe**

**Takeaway**

4-way set-associative last-level cache (LLC)
An Unprivileged Attacker Does Not Know the Target Set

**Target set:** An LLC set accessed by the victim in a secret-dependent manner

Victim

Attacker

Which set to monitor?

4-way set-associative last-level cache (LLC)
Step 2.1: Build Many Eviction Sets

Attacker needs an eviction set for every LLC set in the system

~57k eviction sets
(Intel CPU in Google Cloud)

4-way set-associative last-level cache (LLC)
Step 2.2: Identify Target LLC Set to Monitor

Attacker collects an access trace from each LLC set
⇒ Checks whether the access trace matches victim’s access behavior

4-way set-associative last-level cache (LLC)
Step 2.2: Identify Target LLC Set to Monitor

Attacker collects an access trace from each LLC set
⇒ Checks whether the access trace matches victim’s access behavior
Step 2.3: Extract Information from the Victim

Attacker monitors the target set and extracts the sensitive information

4-way set-associative last-level cache (LLC)
Attack Roadmap

Step 1: Co-Locate

Step 2.1: Build Many Eviction Sets
Step 2.2: Identify Target Set
Step 2.3: Extract Information

End-to-end, cross-tenant information leakage in production Google Cloud

Challenges: Noise and dynamism

This Work

Zhao et al., Everywhere All at Once... (ASPLOS '24)
Step 1: Co-Locate

Step 2.1: Build Many Eviction Sets
  Step 2.2: Identify Target Set
  Step 2.3: Extract Information

End-to-end, cross-tenant information leakage in production Google Cloud

Challenges: Noise and dynamism

Zhao et al., Everywhere All at Once... (ASPLOS ’24)
Our Contribution: **Fast Eviction Set Construction**

**Contribution:** Fast eviction set construction in \(~2.4\) minutes

**Existing approaches:** would take \(>10\) hours to complete due to noise

Exceeds execution time limits!

4-way set-associative last-level cache (LLC)
Background: Constructing an Eviction Set

A set of virtual addresses (VAs) might be mapped to Set 1

* Congruent address

Candidate Set

Set 1

Set n

4-way set-associative last-level cache (LLC)
Background: Pruning the Candidate Set (Group Testing\textsuperscript{1,2})

Vila et al., Theory and Practice of Finding Eviction Sets (S&P 2019)
Qureshi et al., New Attacks and Defense for Encrypted-Address Cache (ISCA 2019)
Background: Pruning the Candidate Set (Group Testing\textsuperscript{1,2})

Withheld

| VA1 | VA2* |

Candidate Set

| VA3 | VA4* | VA5 | VA6* | VA7 | VA8* | VA9 | VA10 |

Access addresses

Test eviction?

Not evicted

| Taddr |

... Set 1

... Set n

Contains congruent addresses ⇒ preserved

4-way set-associative last-level cache (LLC)

\textsuperscript{1}Vila et al., Theory and Practice of Finding Eviction Sets (S&P 2019)
\textsuperscript{2}Qureshi et al., New Attacks and Defense for Encrypted-Address Cache (ISCA 2019)
Background: Pruning the Candidate Set (Group Testing\textsuperscript{1,2})

Candidate Set

\begin{itemize}
\item VA1
\item VA2\textsuperscript{*}
\item VA3
\item VA4\textsuperscript{*}
\item VA5
\item VA6\textsuperscript{*}
\item VA7
\item VA8\textsuperscript{*}
\item VA9
\item VA10
\end{itemize}

4-way set-associative
last-level cache (LLC)

\textsuperscript{1}Vila et al., Theory and Practice of Finding Eviction Sets (S&P 2019)
\textsuperscript{2}Qureshi et al., New Attacks and Defense for Encrypted-Address Cache (ISCA 2019)
Background: Pruning the Candidate Set (Group Testing\textsuperscript{1,2})

Vila et al., Theory and Practice of Finding Eviction Sets (S&P 2019)
Qureshi et al., New Attacks and Defense for Encrypted-Address Cache (ISCA 2019)

Withheld

Not congruent $\Rightarrow$ discarded

Candidate Set

VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*

Access addresses

Requires $O(W^2 N)$ memory accesses\textsuperscript{1}, $N$ is the candidate set size, $W$ is the associativity

4-way set-associative last-level cache (LLC)

1\textsuperscript{Vila et al., Theory and Practice of Finding Eviction Sets (S&P 2019)}
2\textsuperscript{Qureshi et al., New Attacks and Defense for Encrypted-Address Cache (ISCA 2019)}
This Work: Test Eviction is Susceptible to Noise

Withheld

Removed by mistake, fail to construction an eviction set

Candidate Set

Access addresses

Test eviction?

Evicted (False positive)

Noisy access

Prime+Scope $^1$ is similarly susceptible to noise

Our paper provides more detailed quantitative analyses of both algorithms

$^1$Purnal et al., Prime+Scope: Overcoming the Observer Effect for High-Precision Cache Contention Attacks (CCS 2021)
Intuition of the New Algorithm

* Congruent address

Access \( n = 6 \) addresses

Candidate Set

Test eviction?

Not evicted

4-way set-associative last-level cache (LLC)
Intuition of the New Algorithm

* Congruent address

Access $n = 7$ addresses

Candidate Set

Test eviction?

Not evicted

4-way set-associative last-level cache (LLC)
Intuition of the New Algorithm

* Congruent address

Access $n = 8$ addresses

Candidate Set

Test eviction?

Evicted!

4-way set-associative last-level cache (LLC)

Must be a congruent address

Tipping point = 8
Intuition of the New Algorithm

* Congruent address

Candidate Set

Access $n = 8$ addresses

Test eviction?

Evicted! Taddr

Tipping point = 8

Must be a congruent address

Idea: Identifying congruent addresses by finding tipping points

Insight: Can evict $\Rightarrow n \geq$ tipping point; Cannot evict $\Rightarrow n <$ tipping point

$\Rightarrow$ Speed up the process with **binary search**
New Technique 1: A Binary Search-Based Algorithm

Access $n = 5$ addresses

Test eviction?

Not evicted!

4-way set-associative last-level cache (LLC)

LB = 0

VA1    VA2*    VA3    VA4*    VA5

VA6*    VA7    VA8*    VA9    VA10

UB = 10

$n = 5$

$\text{not evicted!}$

$\text{Taddr Set 1}$

...
New Technique 1: A Binary Search-Based Algorithm

Access $n = 7$ addresses

Test eviction?

Not evicted!

4-way set-associative last-level cache (LLC)

VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*  VA9  VA10

$LB = 5$  $n = 7$  $UB = 10$

VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*  VA9  VA10

$VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*  VA9  VA10$
New Technique 1: A Binary Search-Based Algorithm

Access $n = 8$ addresses

Test eviction?

Evicted!

4-way set-associative last-level cache (LLC)

VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*  VA9  VA10

LB = 7  $n = 8$  UB = 10

VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA8*  VA9  VA10

Test eviction?

Evicted!

4-way set-associative last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

Set 1

4-way set-associative last-level cache (LLC)

LB = 7  UB = 8

Identified a congruent address
New Technique 1: A Binary Search-Based Algorithm

Access $n = 4$ addresses

Test eviction?

Not evicted

4-way set-associative last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

Access $n = 6$ addresses

Test eviction?

Not evicted

4-way set-associative last-level cache (LLC)

VA8*  VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA9  VA10

$LB = 4$  $n = 6$  $UB = 8$

VA8* VA1 VA2* VA3 VA4* VA5 VA6* VA7 VA9 VA10
New Technique 1: A Binary Search-Based Algorithm

Access $n = 7$ addresses

Test eviction?

Evicted!

4-way set-associative last-level cache (LLC)

$LB = 6 \quad n = 7 \quad UB = 8$
New Technique 1: A Binary Search-Based Algorithm

Set 1...

VA8*  VA1  VA2*  VA3  VA4*  VA5  VA6*  VA7  VA9  VA10

LB = 6  UB = 7

Identified a congruent address

Taddr

4-way set-associative last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

4-way set-associative last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

4-way set-associative
last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

Eviction set

4-way set-associative last-level cache (LLC)
New Technique 1: A Binary Search-Based Algorithm

Eviction set

Intuition: Fewer accesses on caches with higher associativity

Our algorithm requires $O(WN \log N)$ memory accesses.

Group Testing requires $O(W^2N)$ memory accesses.

Intuition: Check out our paper for another optimization that reduces $N$
Attack Roadmap

- **Step 1**: Co-Locate
- **Step 2.1**: Build Many Eviction Sets
- **Step 2.2**: Identify Target Set

**Insight**: Victim accesses are periodic ⇒ Spectral analysis

- Challenges: Noise and dynamism

Zhao et al., *Everywhere All at Once...* (ASPLOS ’24)

End-to-end, cross-tenant information leakage in production Google Cloud
Attack Roadmap

✅ Step 1: Co-Locate

✅ Step 2.1: Build Many Eviction Sets

✅ Step 2.2: Identify Target Set

➡️ Step 2.3: Extract Information

Zhao et al., Everywhere All at Once… (ASPLOS ’24)

Challenges: Noise and dynamism

Insight: Overlap memory accesses of Prime/Probe to exploit memory-level parallelism

⇒ Low Prime/Probe latency ⇒ Good time resolution and noise resilience
Attack Roadmap

- **Step 1: Co-Locate**
- **Step 2.1: Build Many Eviction Sets**

**Challenges:** Noise and dynamism

**Result:** Can extract an average of 68% of secret nonce bits (or a median value of 81%)

**Target victim:** A vulnerable ECDSA program from OpenSSL 1.0.1e

**End-to-end, cross-tenant information leakage in production Google Cloud**
Conclusions

Cross-tenant information leakage with LLC Prime+Probe

1. Fast LLC Channel Setup
> 10 hours → 2.4 minutes

2. Noise-Resilient Victim Monitoring

3. Information Extraction

+ Google filed a critical-level bug to their product team
+ AWS revised their whitepaper on February 15, 2024

GitHub Repo: https://github.com/zzrcxb/LLCFeasible