Everywhere All at Once: Co-Location Attacks on Public Cloud Faas

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

ASPLOS ’24 – Session 1D: Attacks and Mitigations
Many Microarchitectural Side-Channel Attacks

These side channels exploit **shared resources** between the attacker and victim.

**Lora of the King(s)**

**Interconnect**

**LLC Prime+Probe**  
**Last-Level Cache (LLC)**  
**Flush+Reload**

**DRAMA**  
Main Memory
Steps of Side-Channel Attacks in Public Cloud

**This Work**

**Step 1: Co-Location**

![Diagram of Co-Location: Victim and Attacker located together with Google Cloud (Cloud Run) and a fully-managed containerized environment (E.g., Function-as-a-Service (FaaS)).]

**Step 2: Extraction**

![Diagram of Extraction: Victim and Attacker separated with a 0-1 transition and the same fully-managed containerized environment.]

*Characters are based on https://xkcd.com/2176 and https://xkcd.com/1808 (under a CC Attribution-NonCommercial 2.5 License).*
Background: Fully-Managed Containerized Environment

Developers → Libraries → Program → Container Image → Configurations → Web Service

Region: us-east
vCPU: 2
Memory: 1GiB
Cloud vendor automatically launches a container instance
(The instance placement is managed by the vendor)
Cloud vendor launches more instances to handle traffic increases
Background: Fully-Managed Containerized Environment

Takeaways:
- Container instance placement is fully managed by the cloud vendor
- Container creation and destruction are automatically adapted to service's demand
Challenge of Co-Locating with the Target Victim

**Attacker’s goal:** Spread attacker containers across many hosts
⇒ Increase the chance of co-location

**Main challenge:**
Attacker has no control nor knowledge of instance placement
⇒ Naively launching containers has a low chance of co-location
Idea: Fingerprint Host → Reverse Engineer Placement Behavior

Accurate host fingerprinting

Understand container placement

Strategy 1
Idea: Fingerprint Host → Reverse Engineer Placement Behavior

Accurate host fingerprinting

Understand container placement

Strategy 1

Strategy 2

...
Main Contributions & Results

1. Accurate Host Fingerprinting
2. Inexpensive Co-Location Test
3. Exploitable Placement Behavior

Exploitable behavior of Google Cloud ⇒ High chance of co-location
Insight 1: Physical Host’s Boot Time as Fingerprint

Boot times:

10:00:00

11:24:91

13:45:28

Likely unique within a data center
Challenge: Host Information is Hidden Due to Sandboxing

Boot time?

Host Kernel

Hardware
Insight 2: Bypassing Software Protection by Asking the Hardware

Useful properties:
1. Reset to 0 when the physical host boots
2. Increments at a known fixed rate $f$

Not affected by CPU frequency scaling and power state

x86 timestamp counter (TSC)

rdtsc (unprivileged)

Boot time?

Host Kernel

Hardware
Derive Boot Time From Timestamp Counter

unknown $T_{boot}$

Host boots up

uptime = timestamp/f

$\Rightarrow T_{boot} = T_q - \text{uptime}$

Fingerprint

Real-world time

`rdtsc` (unprivileged)
Verifying Co-Location

**Instance Pair**

```plaintext
E.g., caches, random number generator*
```

**Shared Resource**

- **Contention!** ⇒ Co-Located
- **No Contention** ⇒ Not Co-Located

**Scalability issue:** it requires $O(N^2)$ pairwise tests to verify $N$ containers

The paper discusses a scalable, fingerprint-assisted method for verifying co-location

---

* Evtyushkin et al., Covert Channels through Random Number Generator: Mechanisms, Capacity Estimation and Mitigations (CCS ‘16)
Host Fingerprints are Highly Accurate

For each pair of container instances
  • **False positive (FP):** same fingerprints but not co-located
  • **False negative (FN):** different fingerprints but co-located

  • Measure accuracy in **three** data center regions (us-central1/east1/west1)
  • Repeat measurements **five** times in each data center region

  Average FN rate: 0.00%
  Average FP rate: 0.02%

😊 14 out of 15 measurements generate perfect fingerprints (no FP nor FN)
Understanding Instance Placement Policy

Method:  

Launch $\times N$  
Fingerprint ...

Google Cloud  
$\bullet =$ Physical Host
Observation 1: An Account Has a Preferred Set of Hosts

**Why:** Affinity scheduling to reduce communication overhead
Observation 2: Different Accounts Have Different Preferred Hosts

**Implication:** Low chance of co-location with a target user
Observation 3: Repeated Launches Spread Instances

Repeated launches:  \( \text{Launch} \times N \text{ Terminate} \)

\( \text{Google Cloud} \)

\( \text{Physical Host} \)

Victim 1

Attacker

Victim 2
Observation 3: Repeated Launches Spread Instances

Repeated launches: $\mathbf{\text{Launch}} \times N \text{ Terminate}$

Repeater container launches

$\mathbf{\text{Google Cloud}}$

$\mathbf{\text{Physical Host}}$

Victim 1

Victim 2

Attacker
Observation 3: Repeated Launches Spread Instances

**Why:** Repeated launches ⇒ User has high demand ⇒ Load balance
Evaluation: Co-Location with Victims

Victim coverage: Percentage of victim instances that are co-located with the attacker
High Victim Coverage and Low Attack Cost

Average Victim Instance Coverage (3 repetitions in each region)

<table>
<thead>
<tr>
<th>Region</th>
<th>Account 2 Coverage</th>
<th>Account 3 Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>us-east1</td>
<td>97.7%</td>
<td>99.7%</td>
</tr>
<tr>
<td>us-central1</td>
<td>90.0%</td>
<td></td>
</tr>
<tr>
<td>us-west1</td>
<td></td>
<td>100.0%</td>
</tr>
</tbody>
</table>

Avg. Attack Cost:
- 24 USD
- 23 USD
- 27 USD

**Takeaway:** High victim coverage and low attack cost
Conclusions

**Insight:** Fingerprint Host → Reverse Engineer Placement Behavior

1. Accurate Host Fingerprinting
2. Inexpensive Co-Location Test
3. Exploitable Placement Behavior

⇒ High victim coverage and low attack cost

**GitHub Repo:** https://github.com/zzrcxb/EAAO
Steps of Side-Channel Attacks in Public Cloud

**Step 1: Co-Location**
- Victim
- Attacker

**Step 2: Extraction**
- Victim
- Attacker

*Characters are based on [https://xkcd.com/2176](https://xkcd.com/2176) and [https://xkcd.com/1808](https://xkcd.com/1808) (under a CC Attribution-NonCommercial 2.5 License)

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