Bit of a Close Talker: A Practical Guide to Serverless Cloud Co-Location Attacks

Wei Shao, Najmeh Nazari, Behnam Omidi, Setareh Rafatirad, Khaled N. Khasawneh, Houman Homayoun, and Chongzhou Fang

UC DAVIS
UNIVERSITY OF CALIFORNIA

GEORGE MASON UNIVERSITY

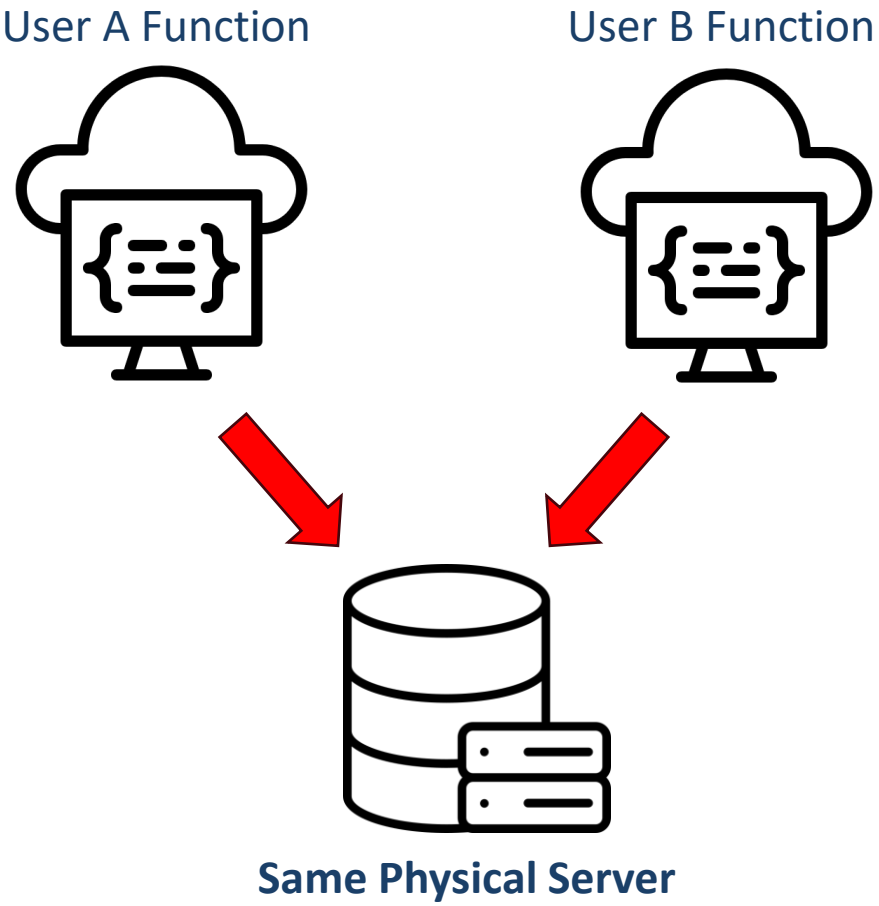
RIT | Rochester Institute of Technology

Serverless Is Everywhere — But Isolation Isn't

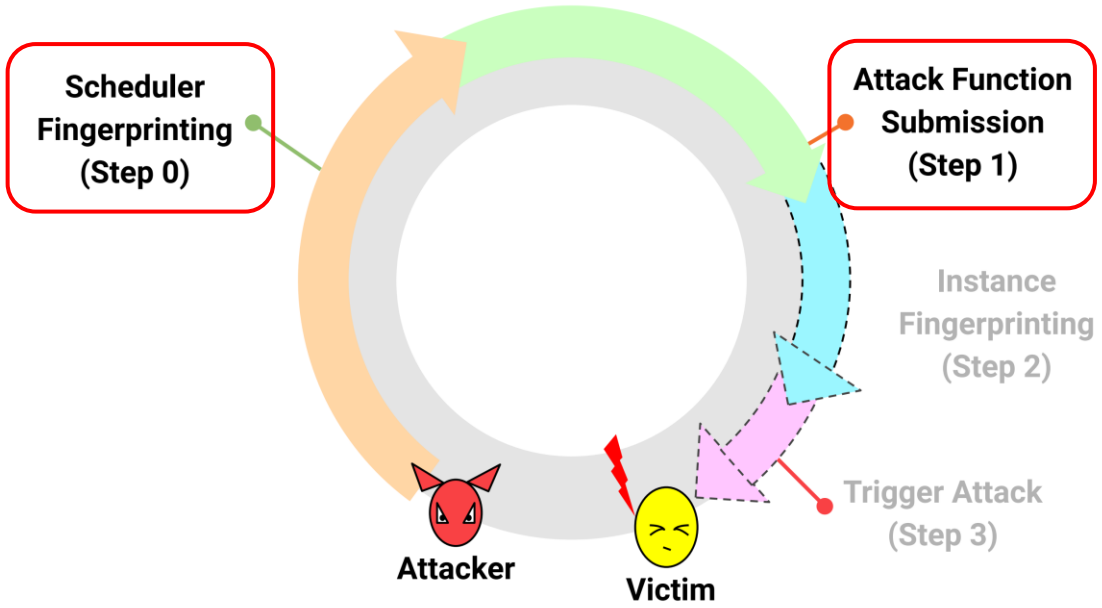
Serverless platforms are fully managed, elastic & cost-efficient, e.g.:

- AWS Lambda
- Azure Functions
- Google Cloud Run
- etc.

However, functions share physical machines — and microarchitectural attacks require **co-location**.



Where Does Co-Location Fit in the Attack Pipeline?



Most prior work studies Step 3 — the side-channel itself.

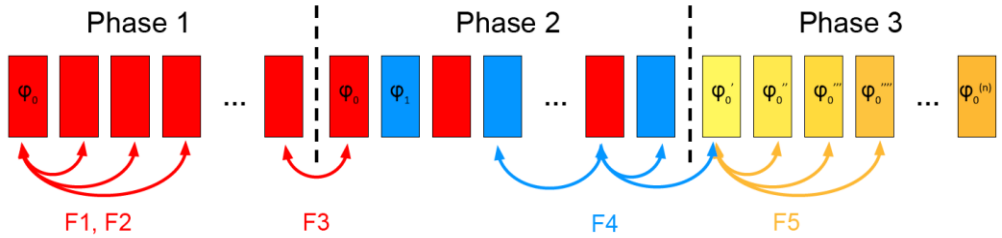
We focus on Steps 0–1: systematically achieving co-location.

Core Idea: Fingerprint the Scheduler

Scheduler behavior leaves placement patterns.

Three-phase probing strategy:

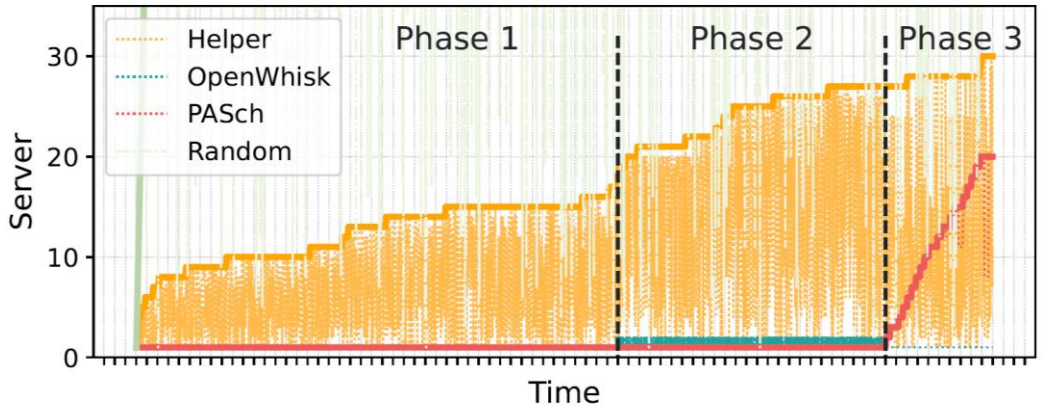
- Phase 1 — Single Function Flood
  - Detect invocation locality (F1)
  - Detect auto-scaling (F2)
- Phase 2 — Duplicate Function Comparison
  - Detect cold-start determinism (F3)
  - Detect account locality (F4)
- Phase 3 — Configuration Variation
  - Detect configuration-based locality (F5)



Placement traces reveal exploitable scheduler features — without privileged access.

Fingerprinting Reveals Scheduler Features

Scheduler	F_1	F_2	F_3	F_4	F_5
Random	×	×	×	×	×
Helper	✓	✓	✓	×	×
OpenWhisk	✓	×	✓	×	×
PASch	✓	×	✓	×	✓(package)



(b) Zoom-in view of the circled area in (a).

Different schedulers expose different exploitable features.

From Scheduler Features to Attack Construction

- **F1 — Invocation Locality** => Deploy multiple attack functions to increase server coverage
- **F2 — Auto-Scaling** => Burst invocations to force instance spreading
- **F4 — Account Locality** => Use multiple attacker accounts
- **F5 — Configuration-Based Locality** => Match victim configuration for precise targeting, or vary configurations to scatter

Once features are known, attack construction becomes systematic.

Constructed Co-Location Attack Methods

Scatter-Based Attacks:

- M1 — Multi-Function Scatter
  - Target: Invocation locality schedulers
  - Deploy multiple functions to increase server coverage
- M2 — Auto-Scaling Burst
  - Target: Auto-scaling schedulers
  - Burst invocations to force spreading
- M3-2 — Configuration Scatter
  - Target: Config-based locality schedulers
  - Vary configurations to scatter across the cluster

Targeted Attack:

- M3-1 — Configuration Matching
  - Match victim's configuration
  - Precisely target victim placement

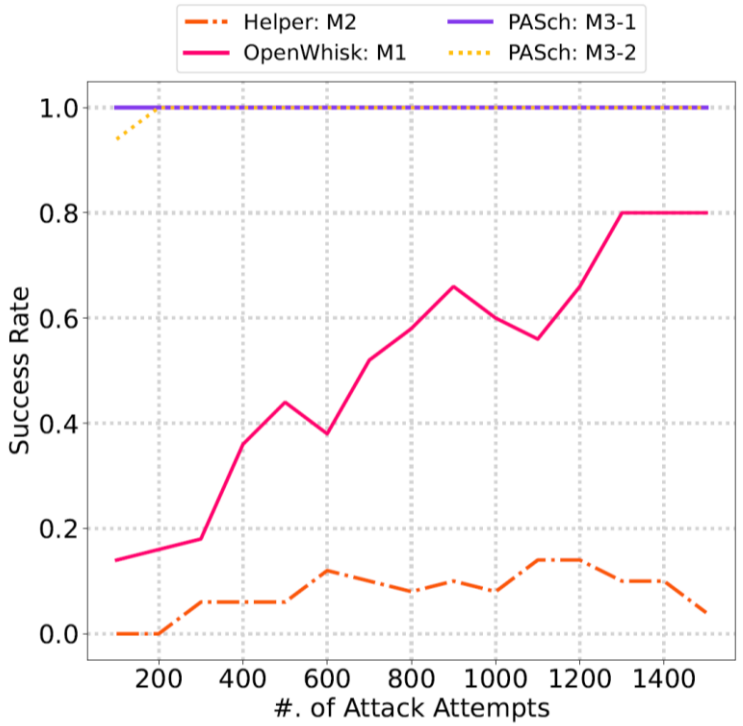
Co-location attacks must match scheduler features.

Simulation: Co-Location Success Rate

Key Findings:

- Exploiting configuration locality (F5) is extremely effective
- Auto-scaling alone provides limited advantage
- Scatter-based attacks require many attempts

Fingerprinting enables high-probability co-location.

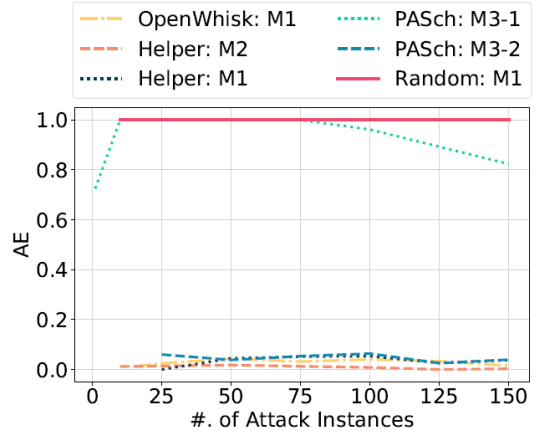


Real-World Evaluation on CloudLab

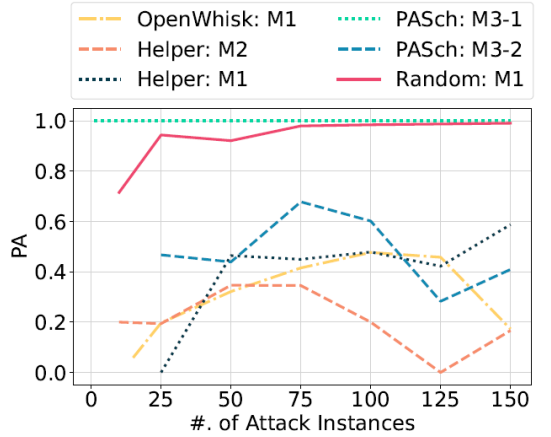
Experiment setup:

- 50-node cluster
- Continuous function invocations
- Measure AE (Attacker Efficiency) and PA (Placement Accuracy)

Attacker can co-locate with ~50% of victim hosts in practice.



(a) AE results.

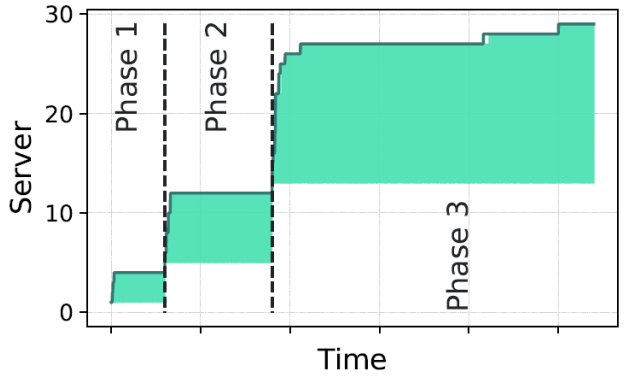


(b) PA results.

Case Study: Microsoft Azure Functions

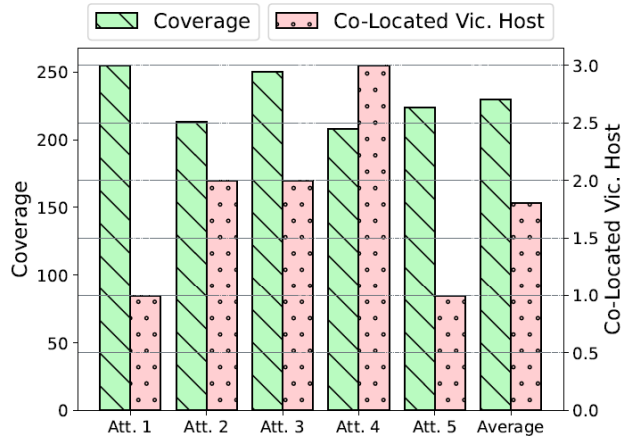
Fingerprinting Results:

- F1: Invocation locality
- F2: Auto-scaling
- Application-level locality (F5-like)
- No account-level isolation



Attack Results:

- Co-location achieved in all trials
- Avg. ~1.8 victim hosts co-located, ~230 machines covered
- **Cost: < $25**



Real-world serverless platforms are vulnerable.

Mitigation: Double-Dip Scheduler

Core Idea: Minimize cross-tenant overlap without sacrificing elasticity.

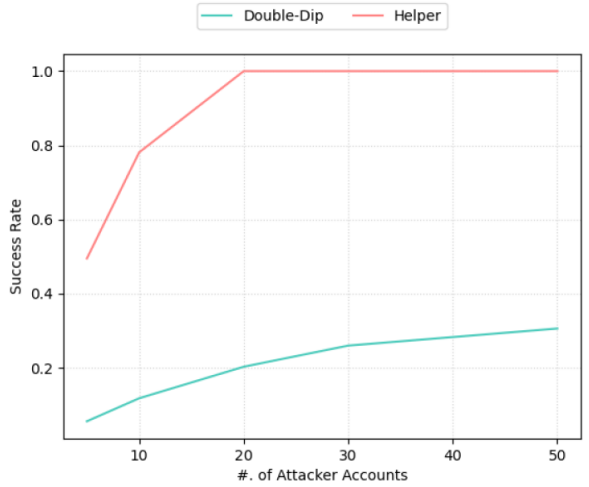
Scheduling Rule:

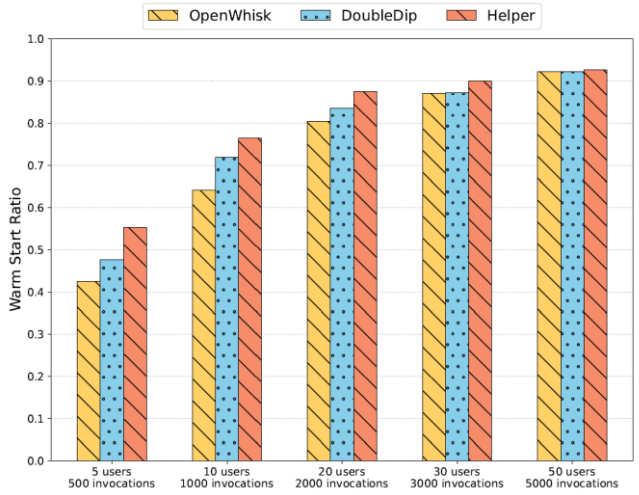
- First, reuse hosts already serving the same user
- Otherwise, choose the host with the least user diversity

Key Property:

- Reduces attacker–victim co-location probability
- Preserves warm-start performance

Soft isolation without dedicated hosts.





Key Takeaways

- Serverless schedulers leak exploitable placement patterns through locality.
- Co-location attacks can be systematic and practical.
- Performance optimizations introduce security risks.
- Soft isolation is possible.

Co-location is not accidental — it is engineered.

Thank You!

Q&A