The Dark Side of Flexibility: Detecting Risky Permission Chaining Attacks in Serverless Applications

Xunqi Liu[1], **Nanzi Yang[2]**, Chang Li[1], Jinku Li[1],Jianfeng Ma[1], Kangjie Lu[2]

[1]Xidian University, China; [2]University of Minnesota

I am seeking for a faculty position

Research focus: AI system security—from cloud infrastructure to Agent systems.

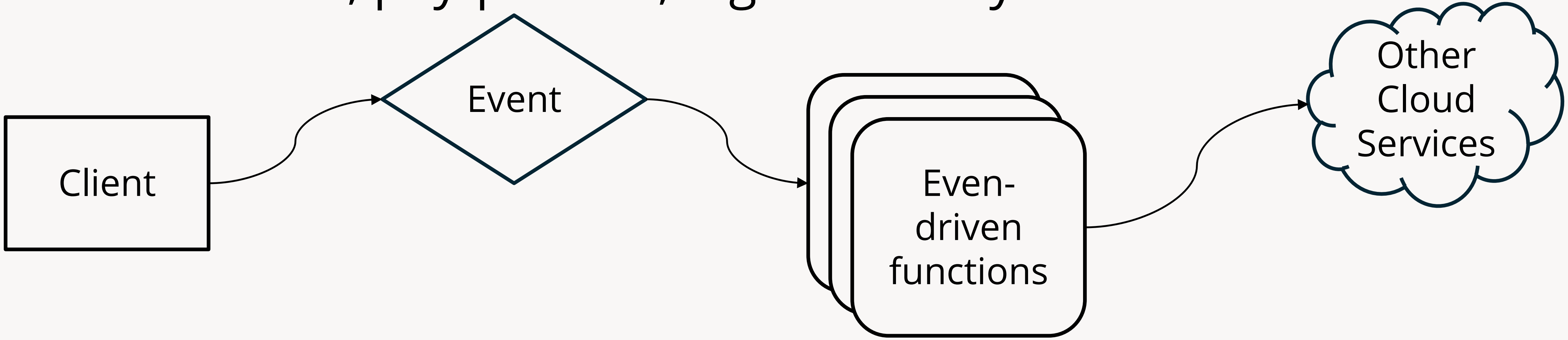
1. Research highlights: 5 first-author papers at top-tier security venues (CCS, USENIX Security, NDSS) .
2. Industry impact: 10+ critical CVEs, 2 Google Security bounties.
3. Community impact: Led the first ecosystem-scale MCP security audit.

Personal website: https://younaman.github.io

Email address: yang9467@umn.edu

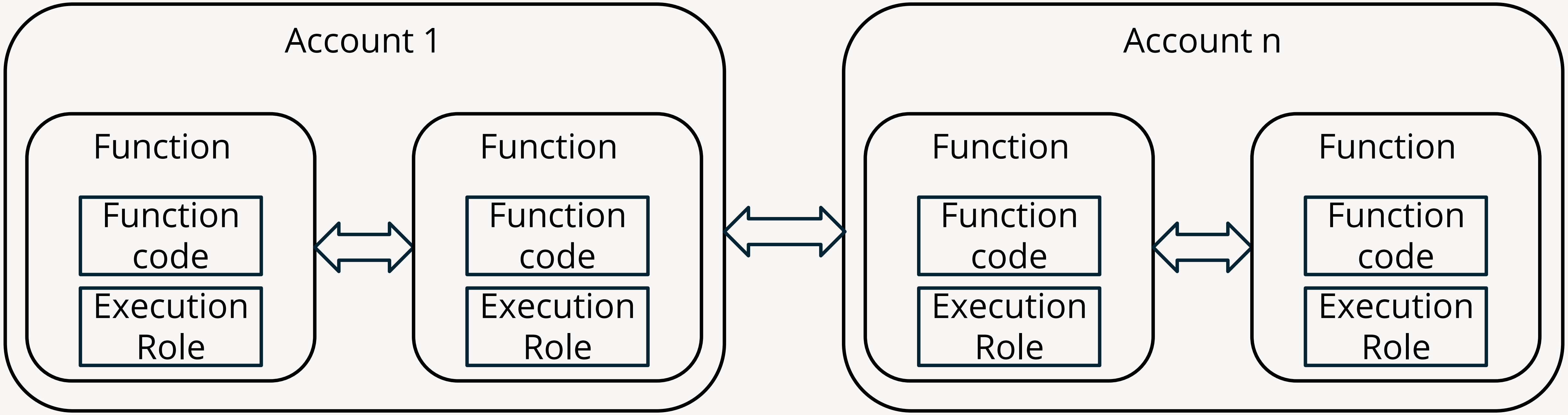
Serverless: function-centric computing

Event-driven, pay-per-use, high flexibility

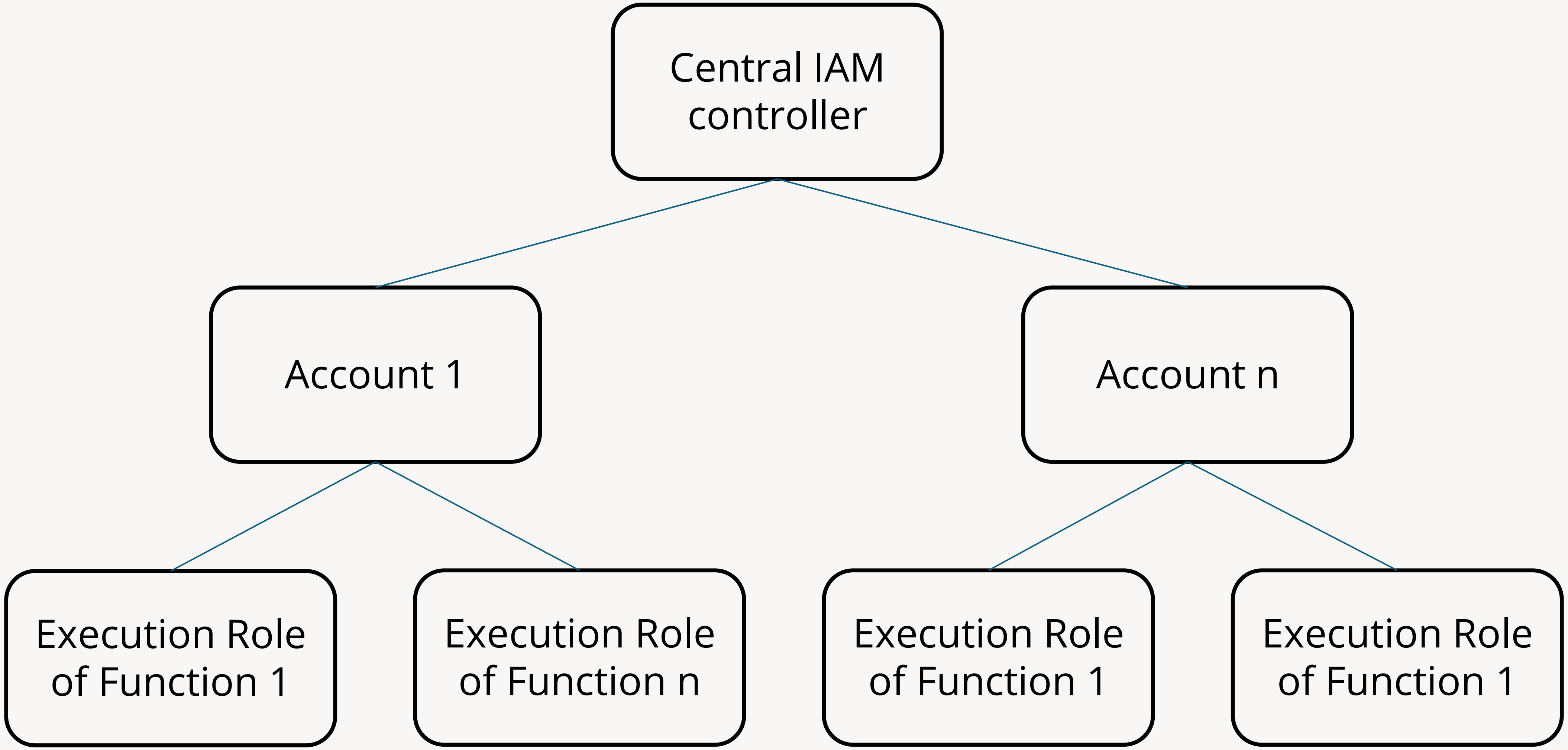


Widely adopted platforms: AWS Lambda · Azure Functions · Google Cloud Functions · Alibaba Function Compute

Serverless: decentralized execution



Permission: centralized authorization

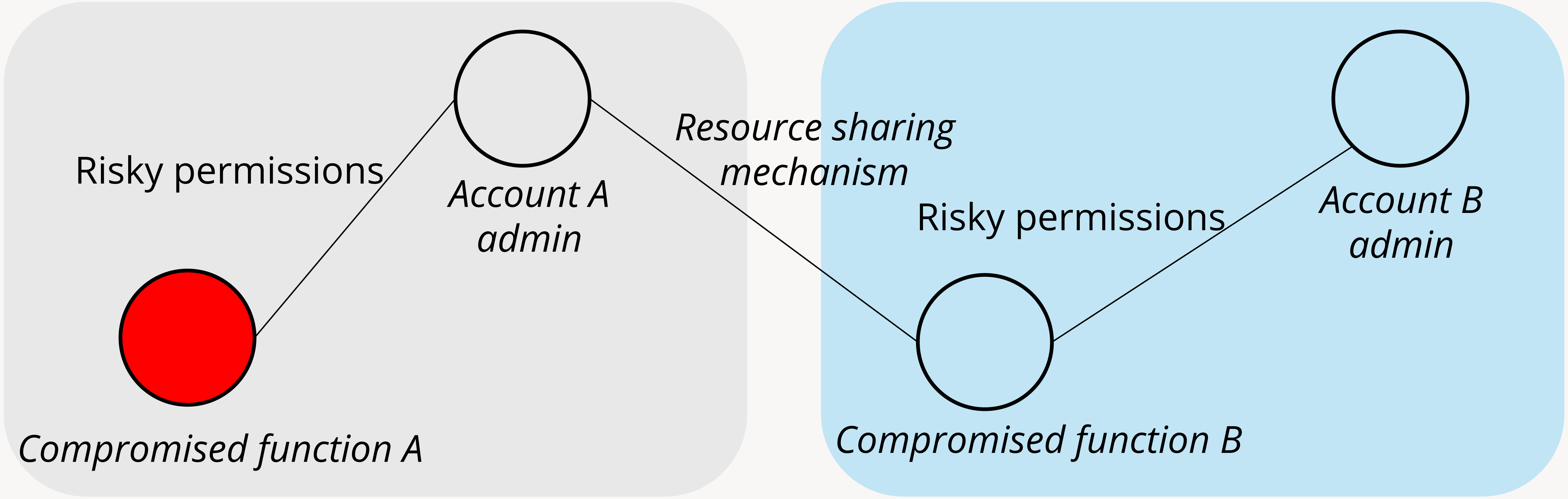

Central IAM controller
Account 1
Account n
Execution Role of Function 1
Execution Role of Function n
Execution Role of Function 1
Execution Role of Function 1

My Key Observation: compatibility at a cost

Centralized IAM policies fail to reason about decentralized serverless execution, enabling cross-function permission chaining risks

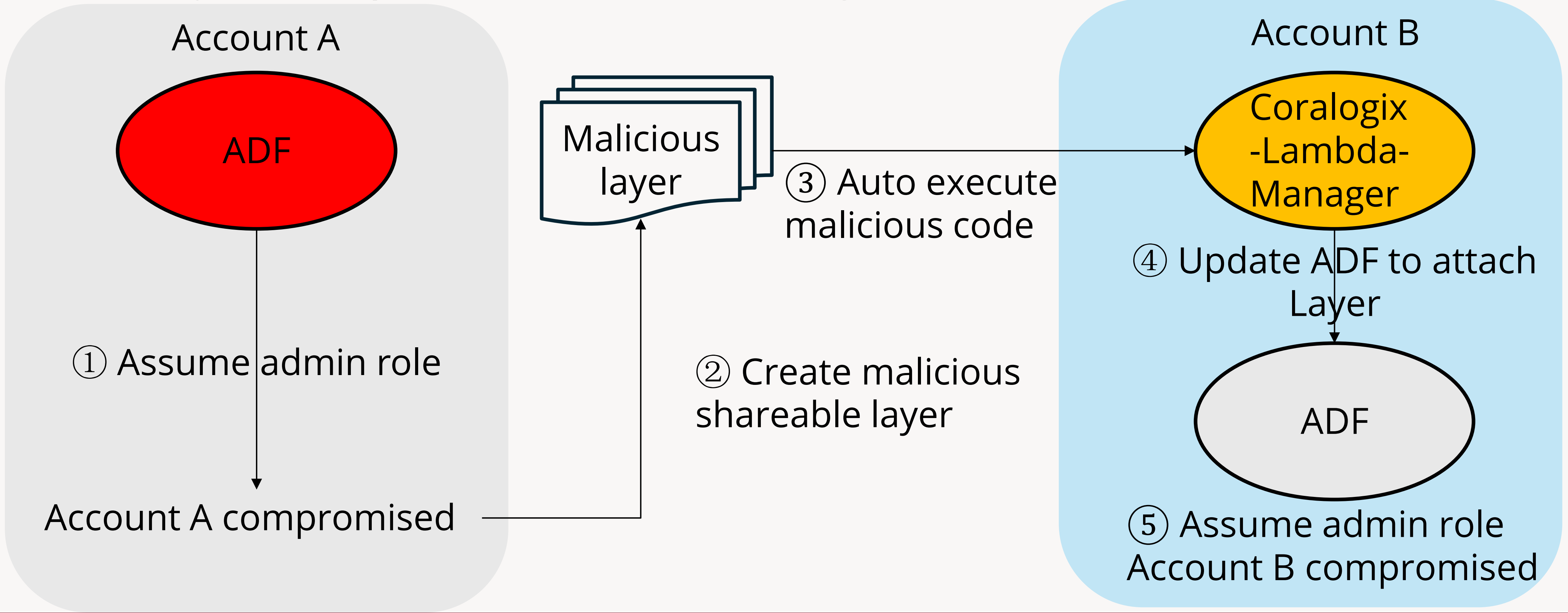
Chaing attack = risky permissions + cross function sharing

Key insight: Cross-function sharing enables attackers to escalate across accounts by chaining risky permissions

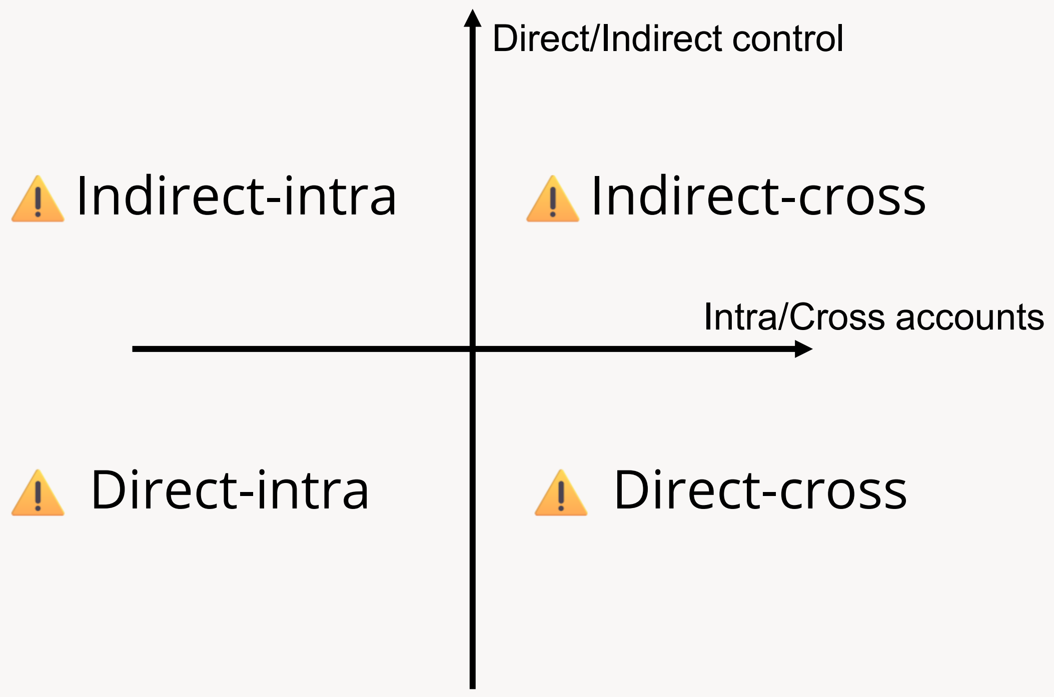


A real example: CVE-2024-37293

Attack Chain: AssumeRole(A)→ Publish shared Lambda layer→ UpdateFunctionConfiguration(B)→ AssumeRole(B)

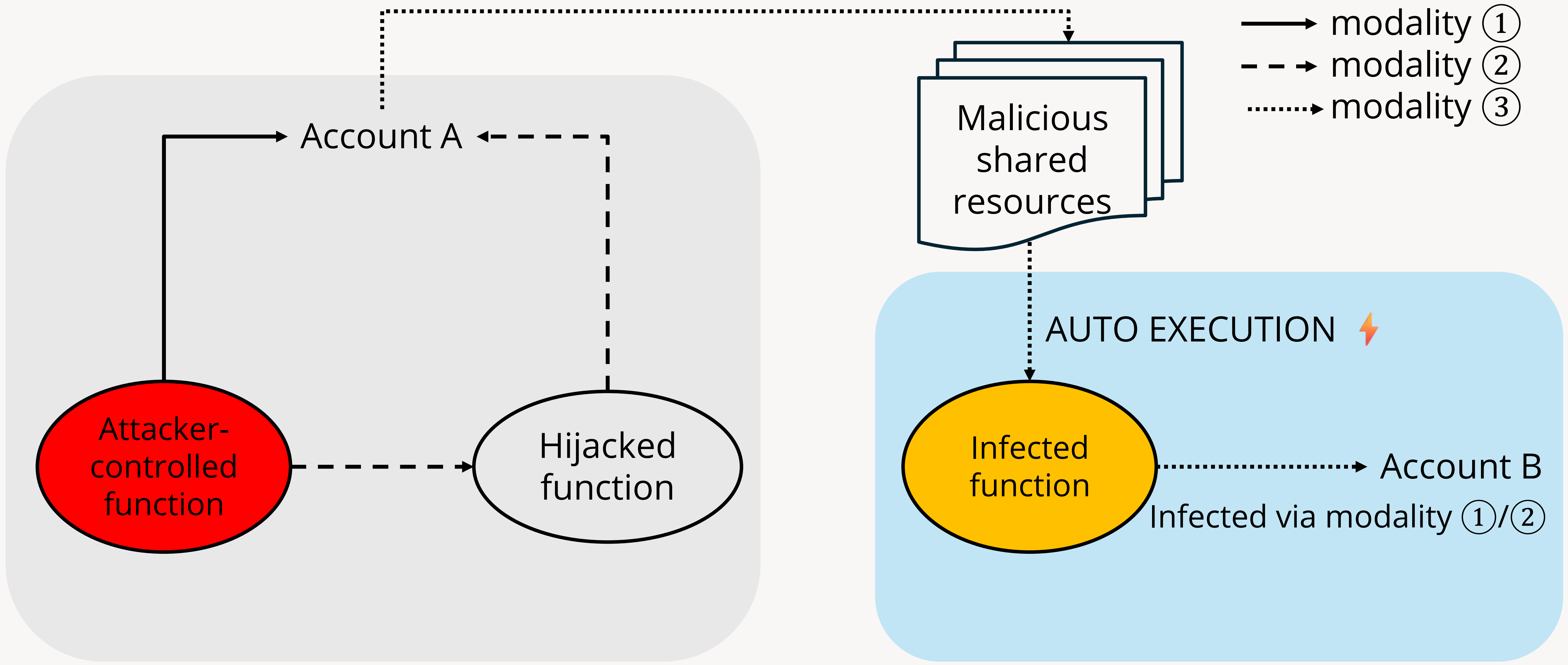


Attack space = capability × topology



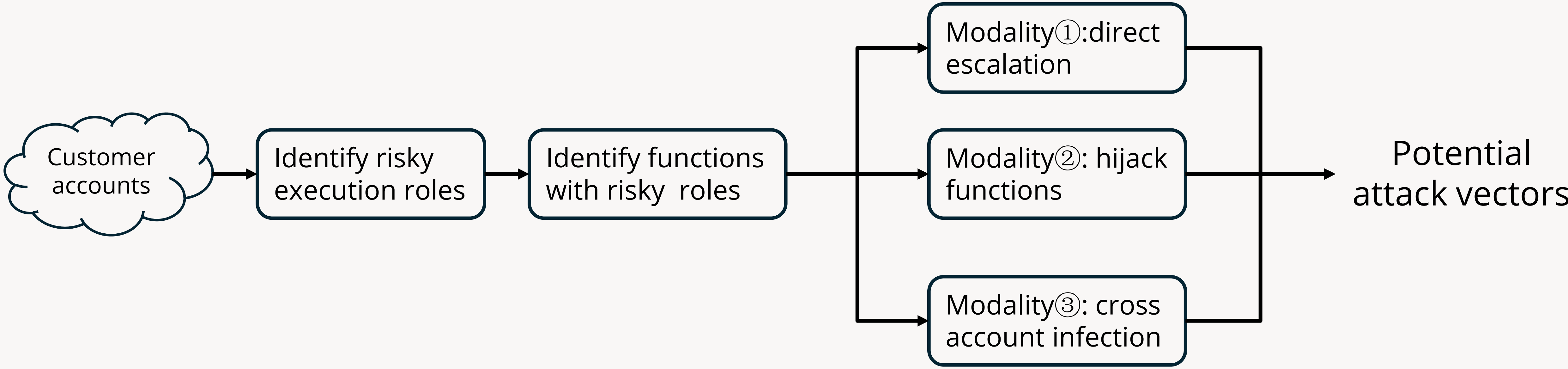
- Modality 1: Attacker can directly control an account
- Modality 2: Attacker can indirectly control an account
- Modality 3: After control the account, the attacker can access function in other account and reuse ½ to control other accounts.

Attack space = capability × topology



System building: modality-based chaining detection

Key insight: Chaining detection is a modality-constrained graph pruning and path reasoning problem.


Customer accounts
Identify risky execution roles
Identify functions with risky roles
Modality①:direct escalation
Modality②: hijack functions
Modality③: cross account infection
Potential attack vectors

Tool evaluations: performance tests

Tool	Risky Permission Detection	Attack Chain Detection	Cross-Account Detection
IAMGraph	✗	✗	✗
IAMSpy	✗		
PMapper	✓		
Cloudsplaining	✓		
Red-Shadow	✗		
AWS Access Analyzer	✓		
AWS Policy Simulator	✗		
This Paper's Tool	✓	✓	✓

Scenario	Apps Deployed	Tool Runtime (s)
Risky Only	1	22.55
	5	24.50
	10	25.06
	15	28.34
	20	31.47
	26	34.71
Mixed	50 (26+24)	39.92
	100 (26+74)	93.26
	200 (26+174)	146.46

Cloud Provider	Serverless Apps	Risky Apps Identified	Confirmed Security Issues
AWS	308	26	10
Alibaba Cloud	55	2	2

Our tool can detect in-the-wild vulnerabilities with acceptable cost

Tool evaluations: real security impacts

Provider	Serverless Apps	Functions	Permissions	Modality	CVE-ID
AWS	aws-deployment-framework	StackWaiter	sts:AssumeRole of *	①	CVE-2024-37293
	measure-cold-start	Loop	lambda:UpdateFunction Configuration of *	③	CVE-2025-45471
	autodeploy-layer	DeployToExisting Functions	lambda:UpdateFunction Configuration of *	③	CVE-2025-45472
LoadZilla	LoadLogic	LogicLoadEc2Deploy Lambda	sts:AssumeRole of *	①	CVE-2024-46511
Alibaba Cloud	fc-stable-diffusion-plus	sd	ram:PassRole and fc:* of *	②	CVE-2025-45468

Takeaway

Novel attack surface: The mismatch between decentralized execution and centralized IAM enables permission chaining.

Modality based detection: Attack modalities emerge from capability × topology. Attack detection via modality-based reasoning.

Real security impact: We identify multiple in-the-wild attacks with multiple CVEs/bounty.