A Multi-Cloud Framework for Zero Trust Workload Authentication

Saurabh Deochake, Ryan Murphy, Jeremiah Gearheart
SentinelOne

Saurabh Deochake
Senior Staff Engineer

Agenda

- Authors
- The Problem
- The Solution
- The Mechanism
- Case Study
- The Impact
- Future Work

Chapter I

The Problem: Insecure Static Credentials at Scale

The Problem

Insecure Credentials Storage

Static private keys stored on disk or in secret managers represent insecure static credentials

Supply Chain Vulnerability

Static keys are frequently leaked in code repositories, CI/CD logs, or compromised containers.

Operational Burden

Managing rotation for thousands of keys is manually impossible and error-prone.

We faced the challenge of securing thousands of workloads across AWS, GCP, and Azure.

Chapter II

The Solution: "Passwordless" Authentication

The Solution

Remove Skeleton Key Risk

Workloads authenticate using long-lived static keys (e.g., AWS IAM User Access Keys, GCP Service Account Keys).

A single compromised key grants persistent access with a massive impact radius.

Remove an ability to use persistent, long-lived private keys

Key Technologies

- Workload Identity Federation (WIF)[1]
- OpenID Connect (OIDC) Standard[2]

The Paradigm Shift

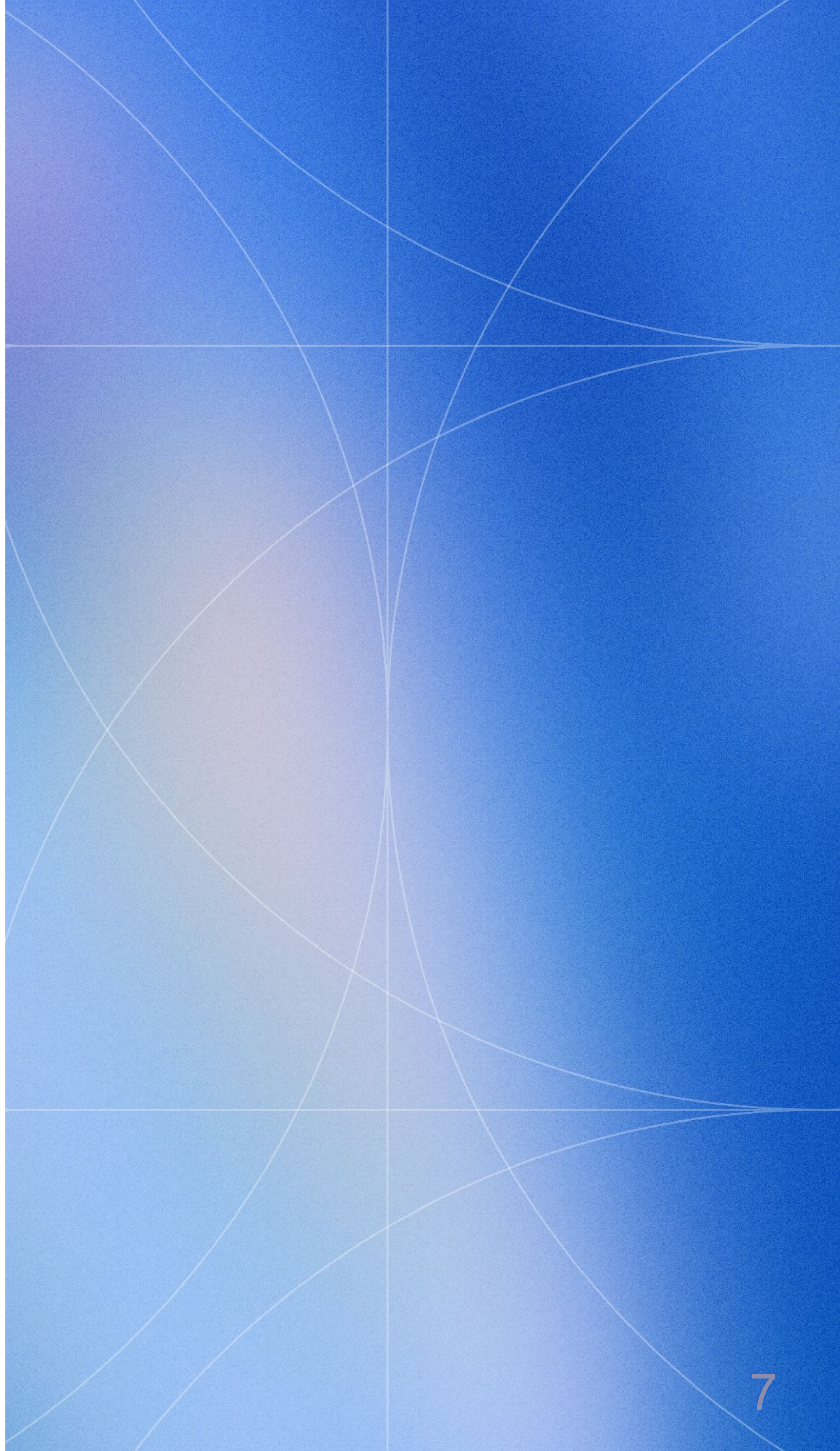
- From: "What you have" (A static credential file)
- To: "Who you are" (A signed identity attested by the platform)

Zero long-lived secrets to manage, rotate, or leak.

[1]: Workload Identity Federation, Google Cloud, https://docs.cloud.google.com/iam/docs/workload-identity-federation
[2]: OpenID Connect, https://openid.net/specs/openid-connect-core-1_0.html

Chapter III

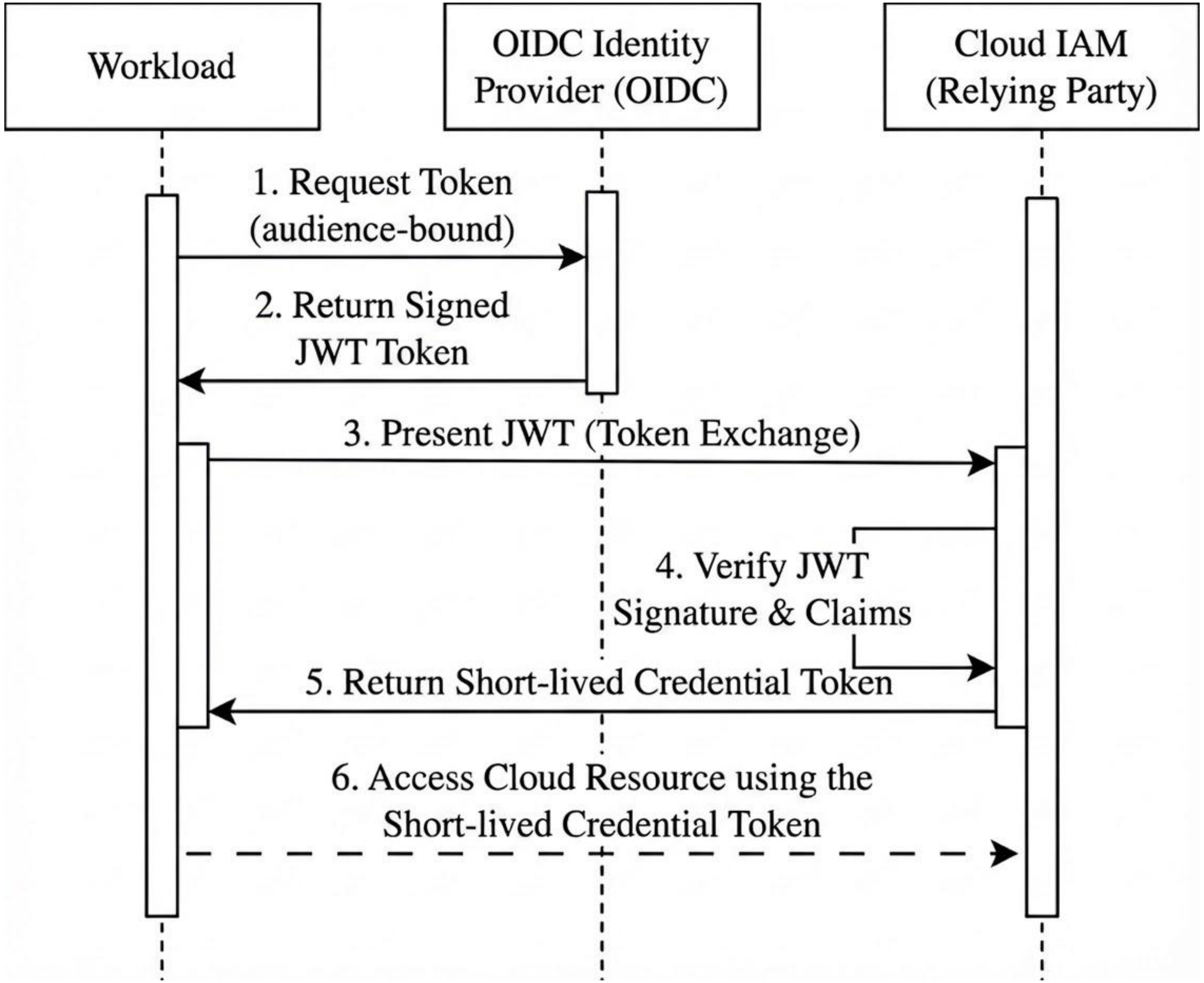
The Mechanism



The Mechanism

The “Who you are” mechanism

- IdP: identity provider
- RP: target service that validates tokens and grants the access
- sub: authenticated workload for token
- aud: token’s intended recipient
- exp: token’s expiration


Workload
OIDC Identity Provider (OIDC)
Cloud IAM (Relying Party)
1. Request Token (audience-bound)
2. Return Signed JWT Token
3. Present JWT (Token Exchange)
4. Verify JWT Signature & Claims
5. Return Short-lived Credential Token
6. Access Cloud Resource using the Short-lived Credential Token

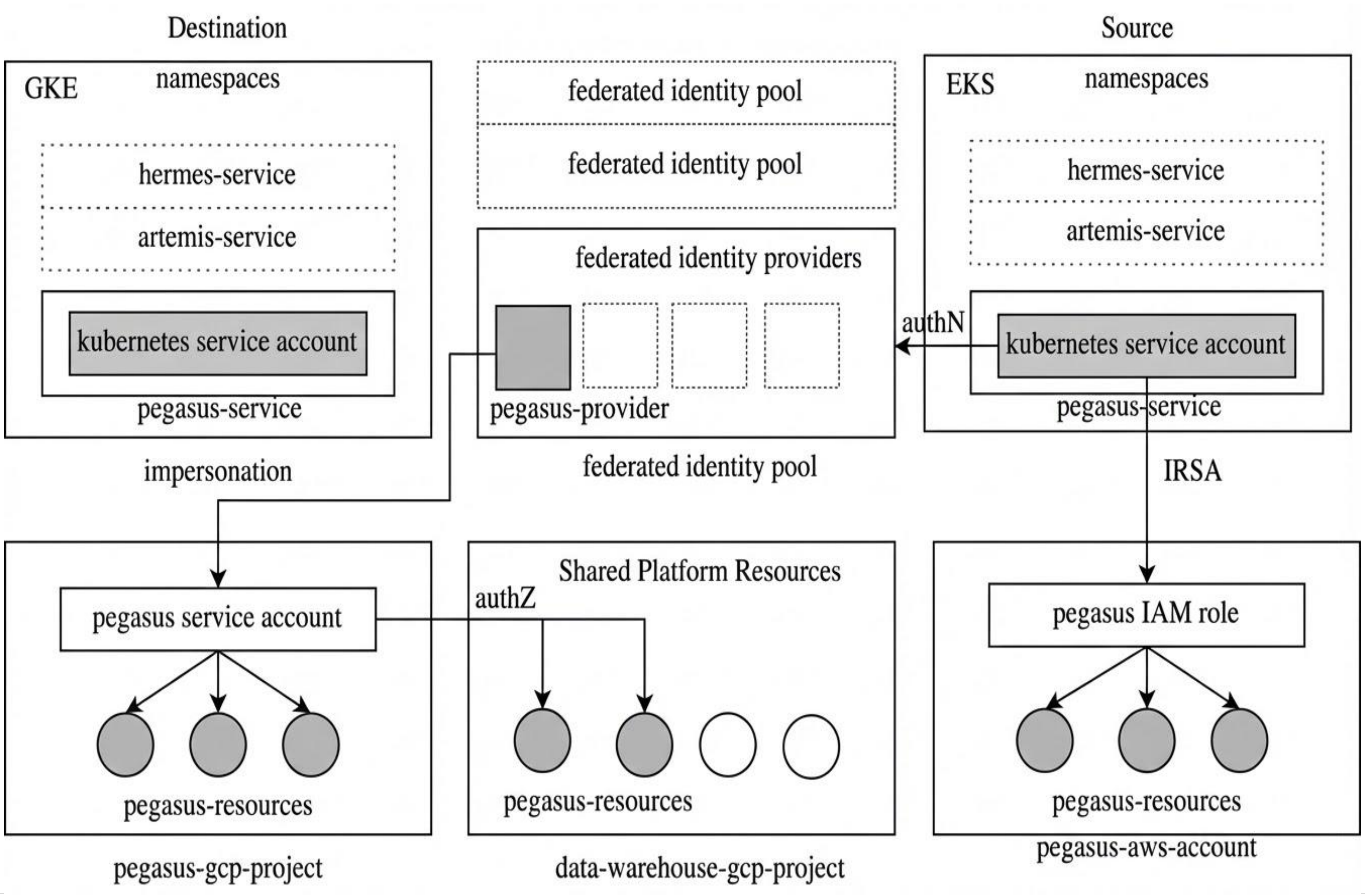
Chapter IV

Case Study

The Scale

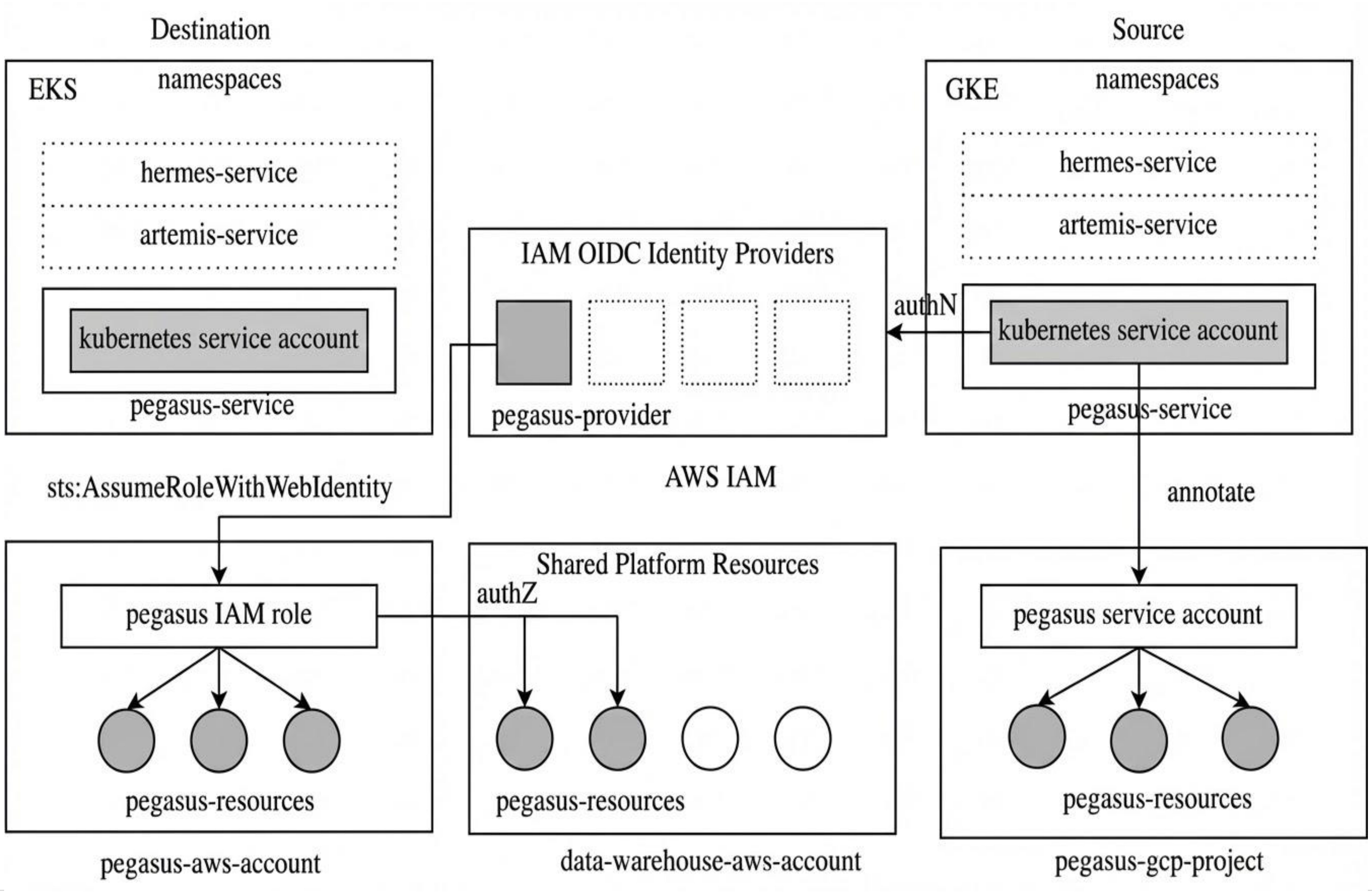
- AI-powered Enterprise Cybersecurity
- Footprint on all major public and private clouds
- 15+ cloud regions
- 100+ Kubernetes clusters, some of the largest in the industry
- Multi-tenancy
  - each service gets an account/project
  - each service gets an IAM role
  - each service gets a namespace
  - 500+ namespaces per cluster
- Hundreds of thousands of cloud resources

Architecture - AWS to GCP



```
# yaml
- provider_id: "eks-pegasus-provider"
  aws:
   account_id: "123456789"
  attribute_condition:
"assertion.arn.endsWith(':assumed-role/pe
gasus-iam-role/pegasus-sa')"
  attribute_mapping:
    google.subject: "assertion.arn"
```

Architecture - GCP to AWS



```
# json
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Federated":
"arn:aws:iam::123456789:oidc-provider/con
tainer.googleapis.com/..."
        },
        "Action":
"sts:AssumeRoleWithWebIdentity",
        "Condition": {
            "StringEquals": {

"container.googleapis.com/...:sub":
"system:serviceaccount:pegasus:pegasus-sa
",

"container.googleapis.com/...:aud":
"sts.amazonaws.com"
            }
        }
    }]
}
```

Chapter V

The Impact

The Impact (By the Numbers)

Scale

100+

- Kubernetes clusters secured
- Across GCP, AWS, Azure

Efficiency

>80%

- Reduction in Audit Overhead
- Eliminated manual verification of key rotation

Risk Reduction

~0

- The platform runs “keyless”
- Eliminated private keys for GitHub, CI/CD - Jenkins

Security Wins

Elimination of The Confused Deputy Problem[1]

Threat: Malicious service tricking a privileged workload.

Fix: Audience Binding (aud). Tokens are cryptographically stamped for one specific recipient (e.g., AWS) and cannot be replayed elsewhere.

Reduced Risk of Credential Theft & Exfiltration

Threat: Attackers scanning filesystems and Secret/Password Managers for private keys

Fix: Memory-Only Ephemerality. Credentials live only in RAM and expire in <60 minutes. Nothing to steal from disk or Secret/Password Manager.

Minimized Supply Chain Exposure

Threat: Third-party vendor retaining access indefinitely.

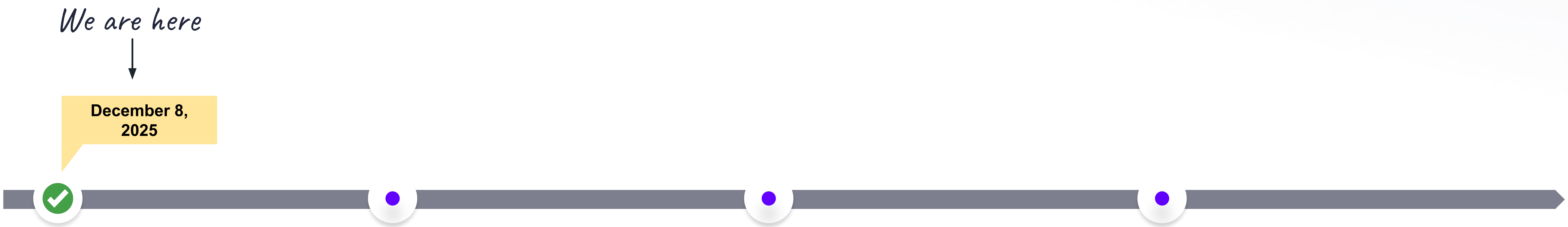
Fix: Policy-Based Trust. Access is revoked instantly by deleting the Terraform policy line. No key rotation ceremony required.

[1]: N. Hardy,The Confused Deputy: (or why capabilities might have been invented), https://dl.acm.org/doi/10.1145/54289.871709

Chapter VI

Future Work

Future Work



We are here

December 8, 2025

Current Implementation

Expanding Compatibility

What **Prove universality beyond AWS/GCP/Azure.**

How Validate patterns for other public cloud providers like Oracle Cloud (OCI).

Goal **A truly vendor-agnostic identity standard.**

Dynamic Authorization (ABAC)

What **Move beyond "Who are you?" to "What is your context?"**

How Inject rich attributes (Project ID, Cost Center, Environment) into the OIDC token claims.

Goal **Policies like "Only Dev workloads can write to Dev buckets" without managing separate roles.**

Just-in-Time (JIT) Access

What **Zero standing privileges.**

How Reduce token lifetime from 1 hour to the duration of a single request/job.

Goal **The "attack window" for a stolen token shrinks to milliseconds.**

Q&A

Thank you!