fwd:cloudsec 2024

GCP and AWS identity federation Lessons learned from the field as well as cross-cloud forensics and incident response

Attila Dulovics
Senior Security engineer
Spotify

Marcus Hallberg
Senior Security engineer
Spotify

Security | Spotify

Who we are?

Attila Dulovics
Senior Security engineer
Spotify

- Engineer, who is a fan of codified solutions, passionate about multi cloud. Secret skill: Hungarian language.

Marcus Hallberg
Senior Security engineer
Spotify

- Passion for cloud security, forensics and automation. Secret skill: Swedish folk dancing.

5 GCP Regions

4096 nodes/cluster

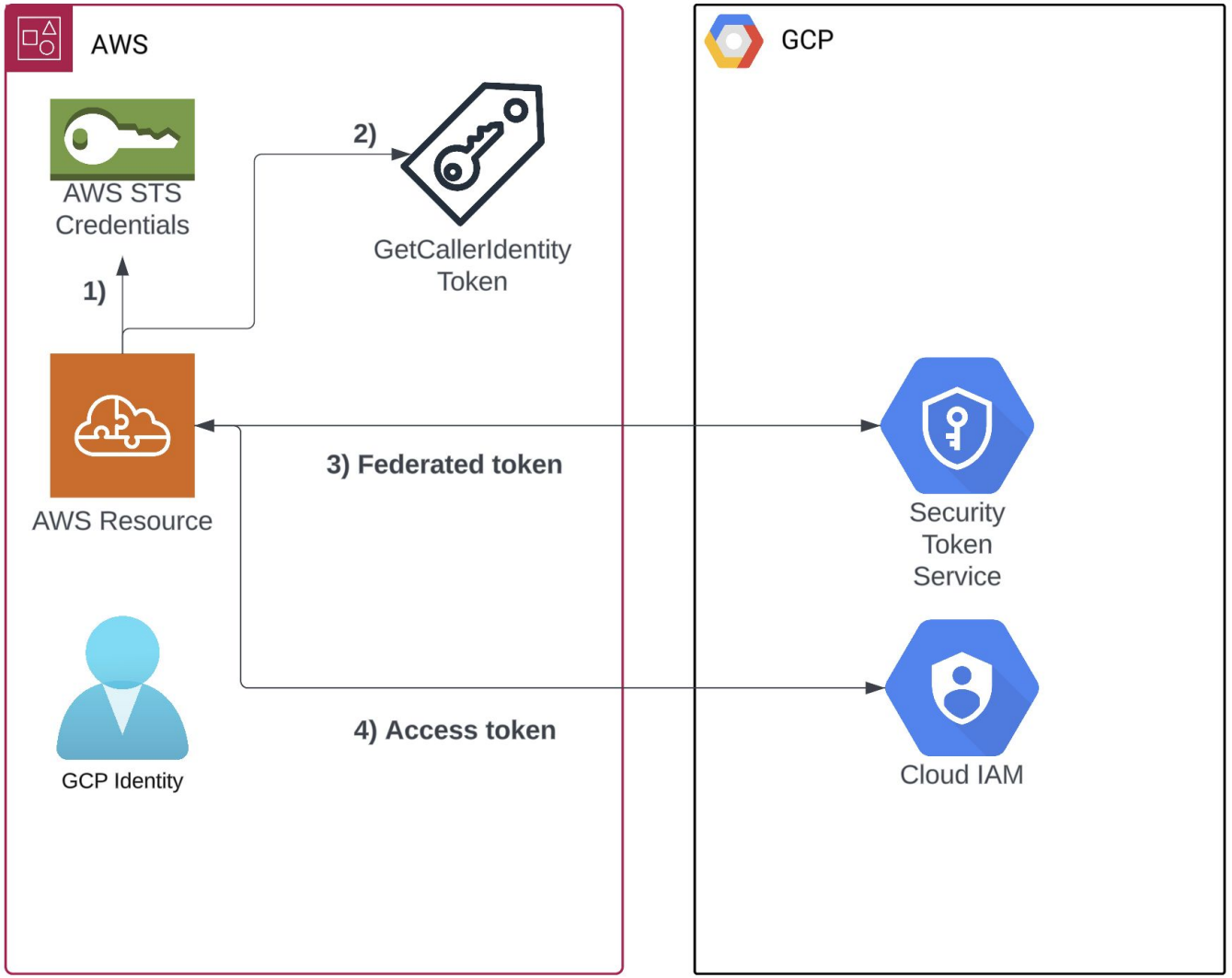
+4000 services

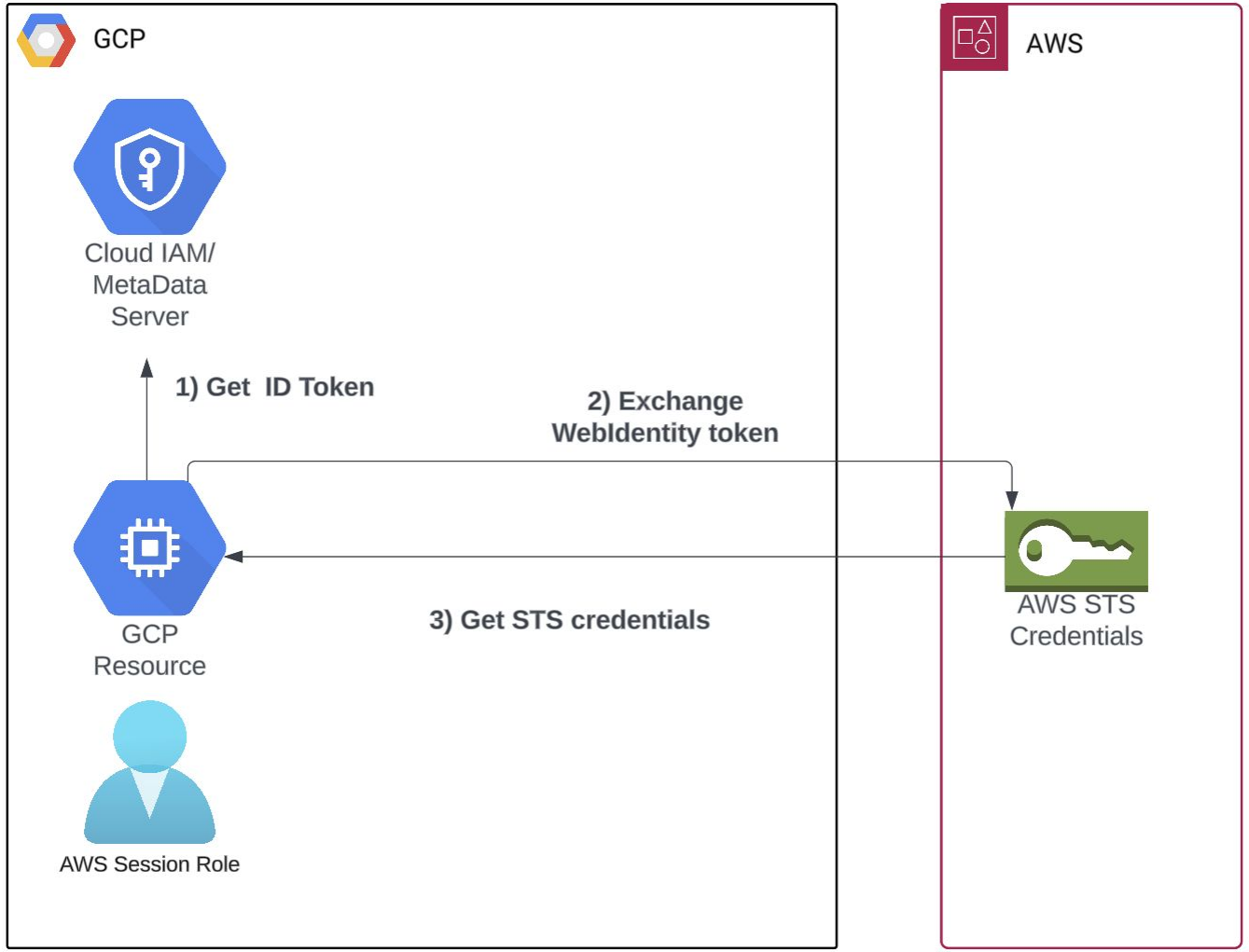
+60 000 service accounts

Agenda

1. Why did we look into Cross Cloud Identity Federation?
2. What strategy did we follow?
3. Highlights and Hiccups (Unexpected Road Bumps)
4. Identity federation for cross cloud incident response and forensics

Why did we look into Cross Cloud identity Federation?


AWS
GCP
AWS STS Credentials
2)
GetCallerIdentity Token
1)
AWS Resource
3) Federated token
Security Token Service
GCP Identity
4) Access token
Cloud IAM


GCP
AWS
Cloud IAM/
MetaData
Server
1) Get ID Token
2) Exchange
WebIdentity token
GCP
Resource
3) Get STS credentials
AWS STS
Credentials
AWS Session Role

What was our motivation?

- Reduce the risk of unnecessary static credential usage in our environment
  - GCP projects >5000
  - AWS Account >80
  - Active AWS access key >100
  - Service account >60000
    - Service Account keys ~40000
- Enable a keyless authn/z for Spotify's service-to-service authentication in a cross-cloud environment

What strategy did we follow?

Our “playlist” for introducing Identity Federation

1. Acquiring Knowledge
2. Build Capability
3. Make Adaptation Appealing
4. Ensure Support

1. Acquiring knowledge

Questions to answer

- What do we need to configure?
- What is the actual developer pain points?
- What threats should we consider?

Threats to consider

- Spoofing
- Privilege escalation
- Non-repudiation

AWS Trust policy snippet

```
"Condition": {
       "StringEquals": {
      "accounts.google.com:oaud":
GCP_SERVICE_ACCOUNT_OAUD_DEFINED_AUDIENCE,
      "accounts.google.com:email":
GCP_SERVICE_ACCOUNT_EMAIL,
      "accounts.google.com:sub":
GCP_SERVICE_ACCOUNT_EMAIL_UNIQUE_ID
       }
```

Workload Identity configuration decisions

- Mapping IAM role as “service identity”
- Dedicated Identity provider and pools for each AWS account

2. Build capability

Requirements about our solution

- Focus on developer pain points
- Wide Applicability
- Low Maintenance
- Smooth Implementation

idxchange & whippet

idxchange support GCP->AWS

- Python CLI tool and library
  - Used as part of base image,
  - Import as a library support application
- Relay on GCP service Identity (SA)

whippet support AWS-> GCP

- Simple CLI tool to generate blueprint with baked best practices of IAMWorkloadIdentityPool and IAMWorkforcePool Provider in Kubernetes Config Connecter

3. Adaptation & 4. Support

Make Adaptation Appealing

- Eliminate the need of Credential Rotation
- Reduced Configuration Overhead
- Simplified IAM Management for federated Identities

Ensure Support

- Clear documentation
- Tool offering for supporting teams and developers
- Enhanced Logging

Highlights and Hiccups (Unexpected Road Bumps)

Hiccups 😟

A.k.a Unexpected Road Bumps

- **Moderate Adaptation**
  - GCP to AWS: 212 federated roles, AWS to GCP: 3 federations.
- **Unexpected challenges:**
  GCP to AWS:
  - Pre-existing OpenID Connect
  - Custom CredentialProvider issue in Apache Beam
  - JIB Usage
- **AWS to GCP:**
  - AWS Fargate Workload compatibility with google auth library

Highlights

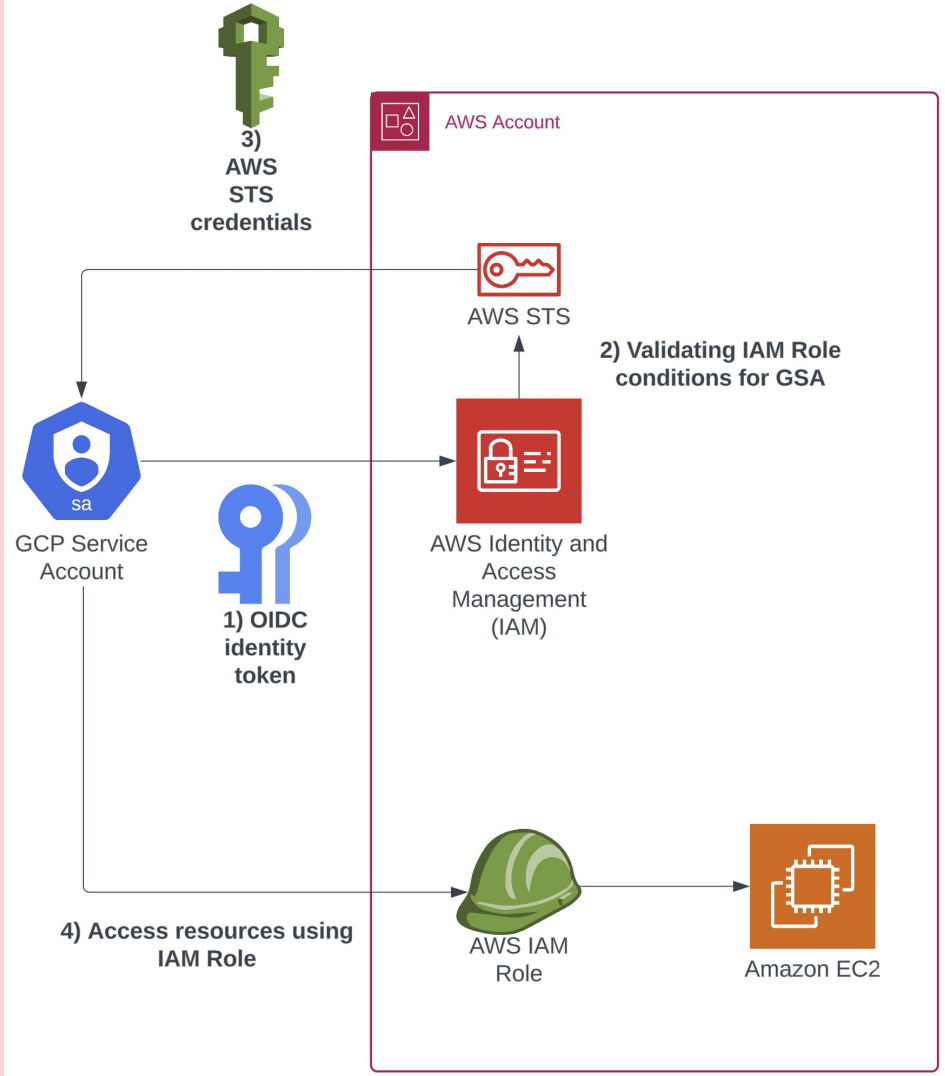
- **Simplified AWS organization wide access from GCP**
  - Significantly simplified the process for accessing AWS accounts from GCP environment for supporting teams (1 steps instead of 3)

- **Proven Value Over Time**
  - Demonstrated long-term benefits, enhancing our Service-to-Service authentication in multi cloud environment

Identity federation for cross cloud incident response and forensics

Setup

1. GSA with identity pool or static keys.
2. Dedicated IAM roles.
3. AssumeRoleWithWebIdentity.



AWS Trust policy example

```
{
    "Effect": "Allow",
    "Principal": {
         "Federated": "accounts.google.com"
     },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
       "StringEquals": {
     "accounts.google.com:oaud": GCP_SERVICE_ACCOUNT_OAUD_DEFINED_AUDIENCE,
     "accounts.google.com:email": GCP_SERVICE_ACCOUNT_EMAIL,
     "accounts.google.com:sub": GCP_SERVICE_ACCOUNT_EMAIL_UNIQUE_ID
       }
  }
},
```

Identity federation methods

1. APIs with HTTP requests

```
$ curl -v \
      --header "Content-Type: application/x-www-form-urlencoded" \
        --data "Action=AssumeRoleWithWebIdentity" \
        --data "Version=2011-06-15" \
        --data "DurationSeconds=3600" \
        --data "RoleSessionName=YOUR_AWS_SESSION_NAME_THAT_YOU_DECIDE" \
        --data "RoleArn=AWS_ROLE_ARN_HERE" \
        --data "WebIdentityToken=YOUR_WEB_IDENTITY_TOKEN \
      POST https://sts.amazonaws.com
```

2. Cloud SDKs

3. Cloud CLIs

```
$ aws sts assume-role-with-web-identity \
    --role-arn YOUR_AWS_ROLE_ARN \
    --role-session-name YOUR_AWS_SESSION_NAME_THAT_YOU_DECIDE \
    --web-identity-token $(gcloud auth print-identity-token
--impersonate-service-account=YOUR_GCP_SA_EMAIL
--audiences=THE_OAUD_AUDIENCE --include-email)
```

https://github.com/spotify/gcp-aws-iam-federation-webidentity

Thank you!