IAM The One Who Knocks

Igal Gofman, Noam Dahan

Igal Gofman

@IgalGofman

- Head of Research, Ermetic
- Microsoft MSTIC
- Microsoft security research
- Active Directory expert

Noam Dahan

@NoamDahan

- Cloud security researcher
- Love/hate relationship with embedded devices
- Offensive background

Why are we here?

IAM best practices

- AWS - Apply least-privilege permissions
- AWS - Use IAM Access Analyzer to generate least-privilege policies based on access activity
- AWS - Regularly review and remove unused users, roles, permissions, policies, and credentials
- AWS - Use conditions in IAM policies to further restrict access

- GCP - Basic roles include thousands of permissions across all Google Cloud services. In production environments, do not grant basic roles unless there is no alternative. Instead, grant the most limited predefined roles or custom roles that meet your needs.
- GCP - Treat each component of your application as a separate trust boundary.
- GCP - Grant roles at the smallest scope needed.

- Azure - Treat identity as the primary security perimeter
- Azure - Use role-based access control
- Azure - Lower exposure of privileged accounts

https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html
https://cloud.google.com/iam/docs/using-iam-securely
https://docs.microsoft.com/en-us/azure/security/fundamentals/identity-management-best-practices

Agenda

- IAM Crash Course
- Cloud IAM weak spots (permissions landscape)
- Things are not always what they seem
- Defense & Monitoring techniques
- Demo

IAM Crash Course

Federation Services

(Who?)

Security Context

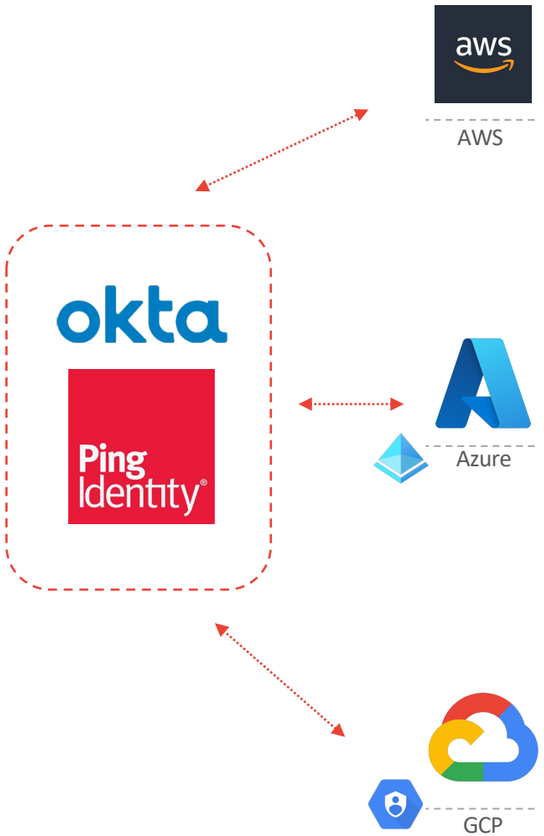
- Users
- Roles
- Services account
- Managed Identity
- Service principal

(Who?)

Access policy

- Policy
- Roles
- Permissions
- Scope

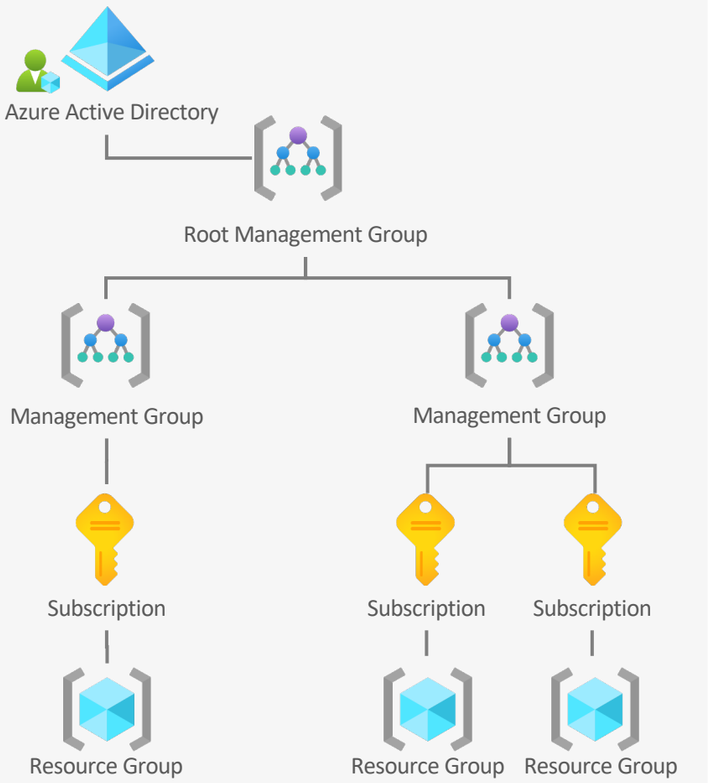
(Under which conditions?)



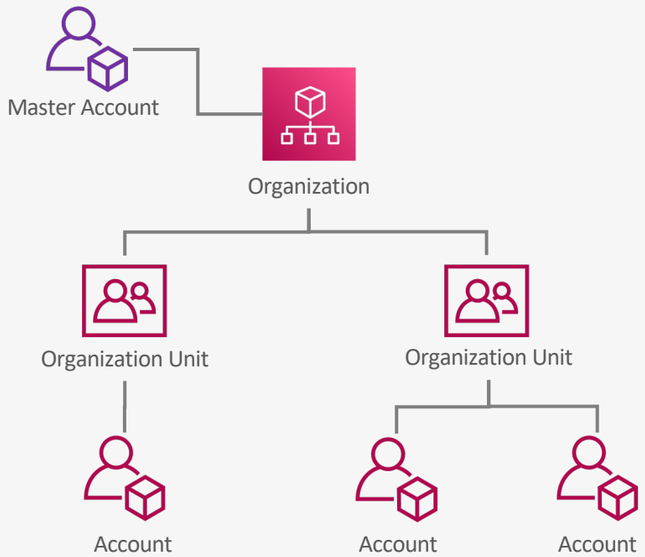
Identity	Permissions	Scope
Users Groups IAM Role AWS SSO directory Federated Identities	Attach	AWS Policy Customer Managed Inline policy
Users Groups Managed identity Service Principal Federated Identities	Assign	Role (Built-In | Custom)
Users Groups Service Account Pre-Defined Groups | Special groups Federated Identities	Bind	Role (Basic | Predefined Custom)

Scope
Scope

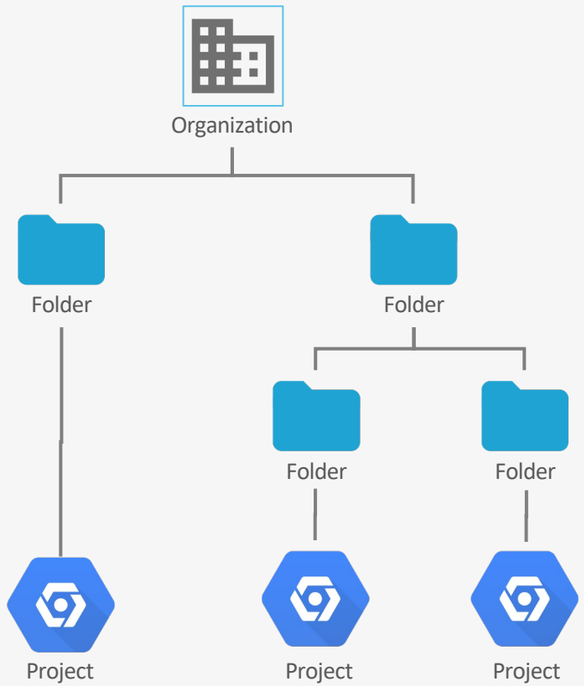
Azure



AWS



GCP



IAM Crash Course: Multi-Cloud Organizational Structure

Cloud IAM weak spot

Cloud IAM weak spot

- Non-human Identities
- Assignment of new permissions
- Code Execution | Task | Template 😇
- Grants and Delegation
- New credentials | secrets
- Encryption & Cryptographic key management
- Organizational policies

Cloud IAM weak spot

Non-human Identities

AWS

Role attachment

Azure

Managed Identities

GCP

Service account

Permissions Landscape

Assignment | Code Execution | Grants and Delegation | New credentials

Assignment

- Azure - Microsoft.Authorization/roleAssignments/write
- Azure - Microsoft.Authorization/roleDefinitions/write
- GCP - iam.roles.update
- GCP - orgpolicy.policy.set
- GCP - resourcemanager.projects.setIamPolicy
- AWS - lambda:AddPermission
- AWS - iam:AttachUserPolicy
- AWS - iam:AttachGroupPolicy
- AWS - iam:AttachRolePolicy

Code Execution

- AWS - lambda:CreateFunction
- AWS - lambda:InvokeFunction
- AWS - lambda:UpdateFunctionConfiguration
- AWS - cloudformation:CreateStack
- GCP - cloudscheduler.jobs.create
- GCP - cloudbuild.builds.create
- GCP - cloudfunctions.functions.create
- GCP - cloudfunctions.functions.update
- GCP - run.services.create

Grants and Delegation

- GCP - iam.serviceAccounts.implicitDelegation
- GCP - deploymentmanager.deployments.create
- GCP - iam.serviceAccounts.actAs
- AWS - iam:PassRole
- Azure - Microsoft.ManagedIdentity/userAssignedIdentities/*/assign/action

New Credentials

- AWS - iam:CreateLoginProfile
- AWS - iam:UpdateLoginProfile
- AWS - iam:CreateAccessKey
- GCP - iam.serviceAccountKeys.create
- GCP - iam.serviceAccounts.signJwt
- GCP - serviceusage.apiKeys.create
- GCP - iam.serviceAccounts.getAccessToken

Things are not always what they seem

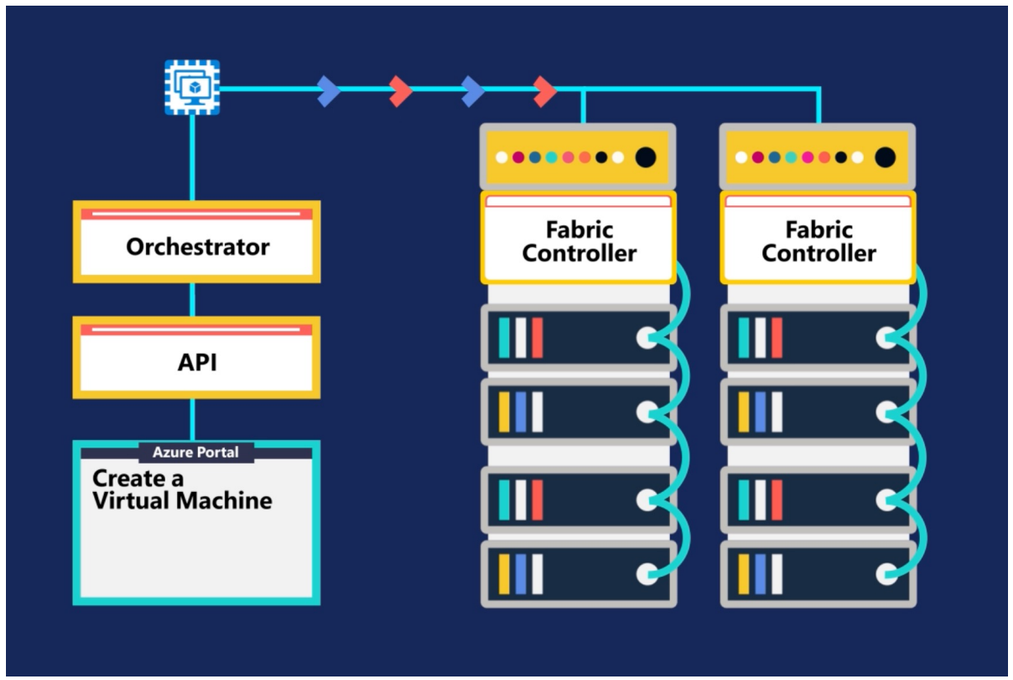
Lesson #1: Beware of non-human identities

- How cloud providers handle non-human credentials (Certificates)
- How cloud consumers handle non-human credentials (Short-lived tokens)
- The Instance metadata, local addresses, and environment variables
- Beware of hybrid Instance metadata

Azure use case: Managed identities

Lesson #1: Beware of non-human identities

- The **Fabric Controller** (**FC**) is a distributed program that manages the hardware and applications in a cluster internally used by Azure.




```
PS C:\Windows\system32> route print
===========================================================================
Interface List
  6...00 0d 3a 9e ed 35 ......Microsoft Hyper-V Network Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.0.1        10.0.0.4      5
         10.0.0.0    255.255.255.0         On-link         10.0.0.4    261
         10.0.0.4  255.255.255.255         On-link         10.0.0.4    261
       10.0.0.255  255.255.255.255         On-link         10.0.0.4    261
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
    168.63.129.16  255.255.255.255         10.0.0.1        10.0.0.4      6
  169.254.169.254  255.255.255.255         10.0.0.1        10.0.0.4      6
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         10.0.0.4    261
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link         10.0.0.4    261
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  6    261 fe80::/64                On-link
  6    261 fe80::30d3:10ba:e29a:858d/128
                                    On-link
  1    331 ff00::/8                 On-link
  6    261 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
PS C:\Windows\system32>
```

```
PS C:\Windows\system32> curl http://168.63.129.16/?comp=versions

StatusCode        : 200
StatusDescription : OK
Content           : <?xml version="1.0" encoding="utf-8"?>
                    <Versions>
                      <Preferred>
                        <Version>2015-04-05</Version>
                      </Preferred>
                      <Supported>
                        <Version>2015-04-05</Version>
                        <Version>2012-11-30</Version...
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 510
                    Content-Type: text/xml; charset=utf-8
                    Date: Sun, 24 Jul 2022 13:30:55 GMT
                    Server: Microsoft-IIS/10.0

                    <?xml version="1.0" encoding="utf-8"?>
                    <Versions>
                      <P...
Forms             : {}
Headers           : {[Content-Length, 510], [Content-Type, text/xml; charset=utf-8], [Date, Sun, 24 Jul 2022 13:30:55 GMT], [Server, Microsoft-IIS/10.0]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 510


PS C:\Windows\system32>
```

Azure use case: Managed identities

Lesson #2: Study implementation details

- Serverless code – Are AWS lambda and GCP functions the same?
- Versioning and revision
- Who can access my function code?
- Privilege escalation

Lesson #3: Defaults are a hacker's best friend

- Why do we need default policies?
- Can we rely on custom policies? (Limitations)
- Service providers best practices?

Lesson #3: Defaults are a hacker's best friend

- **AWS**
- Inherently broad permissions
- "Temporary fix" that becomes permanent
- **Look for:** ReadOnlyAccess, CloudTrailReadOnlyAccess, PassRole, Network modifiers, Permission modifiers, AssumeRole escalations

Lesson #3: Defaults are a hacker's best friend

- **Azure**
- Built-in roles… but oh so many of them
- Custom role limits
- Inherited permissions
- **Look for:** Read permissions, Assignment permissions (self-assignment)

Lesson #3: Defaults are a hacker's best friend

- **GCP**
- Inherited permissions by scope
- Legacy roles have strong and broad permissions (Viewer)
- Legacy mechanism: Access Scopes

Practical Practices for Defenders

Clay or Marble

- Two approaches
- Bottom-up or Top-Down
- Clay is hard → have to know exactly what you need
- Marble is hard → have to prove a negative
- Most people choose Marble, and then never cut down permissions

Limit the effect of mistakes

- AWS Account/GCP Project/Azure resource group **separation per workload**
- Avoid permanent credentials when possible
- Secure human identities

Log more, audit better

- Log whatever you can
- Use audit to build a stronger security policy
- **Challenges:** opaque APIs and distributed logging

Limits of logging

- The unlogged and the un-loggable:
- **Azure** read actions, distributed logging
- **AWS** cross-account actions & failures, passive recon, some data actions, session name manipulation, CloudTrail manipulation

DEMO

Takeaways

Questions?