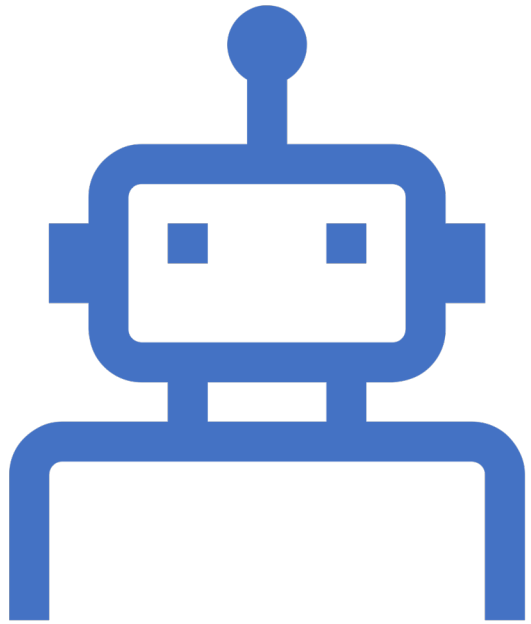
Privilege Escalation in AWS

Jay Kalsi



About Me

- Background in Network and SysAdmin
- Penetration Testing Consulting
- Currently look after a Penetration Testing team in a Bank

Agenda

- AWS IAM & Policies
- Understanding Permissions
- High Risk AWS IAM Vulnerabilities
- IAM Privilege Escalation
- Tools and Resources

Goal

- Penetration Testers - How to attack AWS environments
- Red Teamers - What to do with AWS credentials
- Blue Teamers - What to monitor

Setup – Install awscli

- Install awscli (AWS recommends python 3)
  - Python - pip3 install awscli
  - macos - brew install awscli
  - Linux, virtualenv, windows - https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
- Configure the cli with IAM account with Administrator Permissions
- Install Rhino Security Lab's Pacu

Overview of IAM

Core service behind access management in AWS

Easy to misconfigure hence main source of vulnerabilities

Amazon Resource Number (ARN)

- Uniquely identify all AWS resources
- Example formats:
  - IAM user name
    - arn:aws:iam::123456789012:user/David
  - Amazon RDS instance used for tagging
    - arn:aws:rds:eu-west-1:123456789012:db:mysql-db
  - Object in an S3 bucket
    - arn:aws:s3:::my_corporate_bucket/exampleobject.png

Overview of IAM

USERS

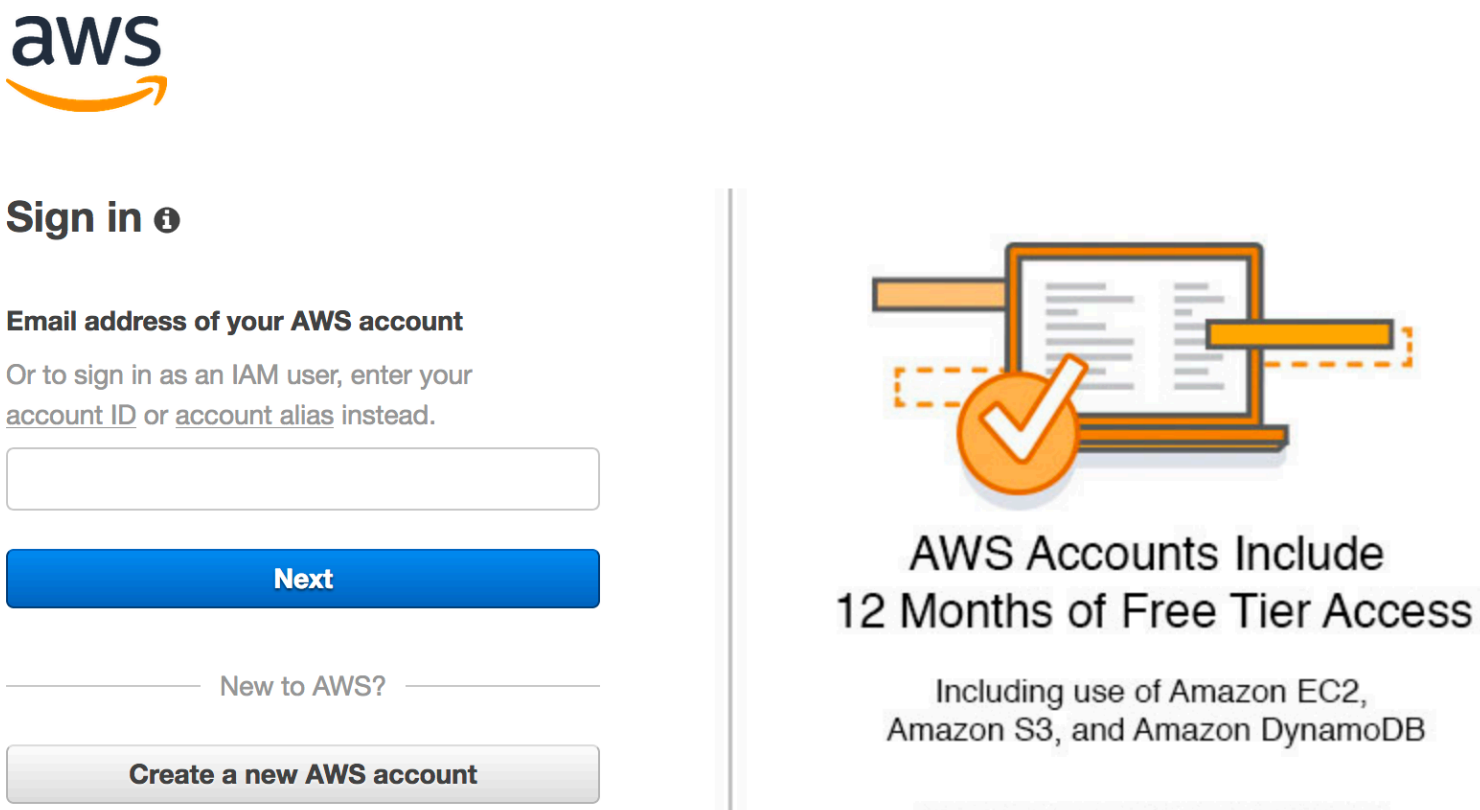
GROUPS

ROLES

IAM Users

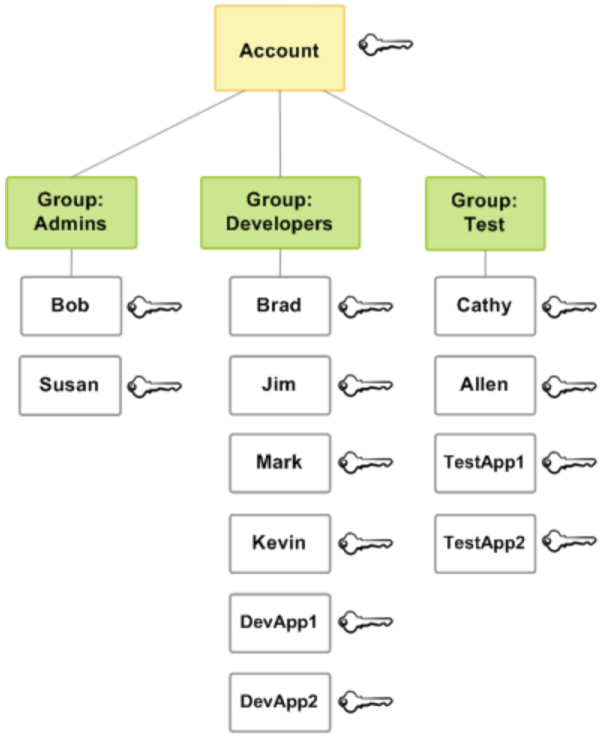
- Users created in your AWS account
- Need a root account to create the first IAM account
- IAM accounts can login using custom URL:
  - `https://496000092283.signin.aws.amazon.com/console`

Root Account Login



IAM Groups

- Collection of IAM users
- A group can contain many users
- A user can belong to multiple groups
- Groups can't be nested

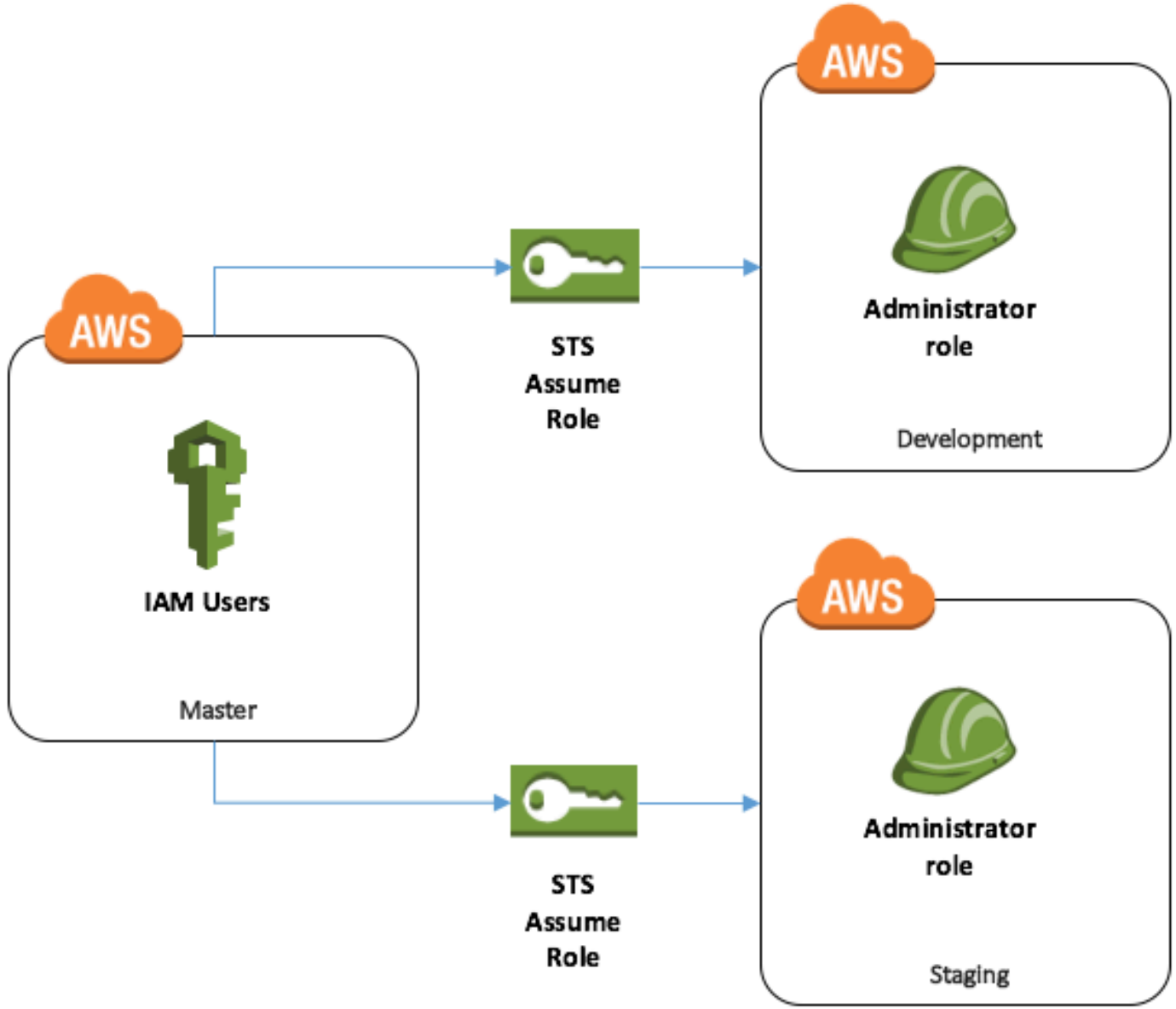


IAM Groups

- Collection of IAM users
- A group can contain many users
- A user can belong to multiple groups
- Groups can't be nested

IAM Roles

- Similar to IAM Users, however Roles need to be “Assumed” by users
- Restrict access by trust policy
- Many users can share a role

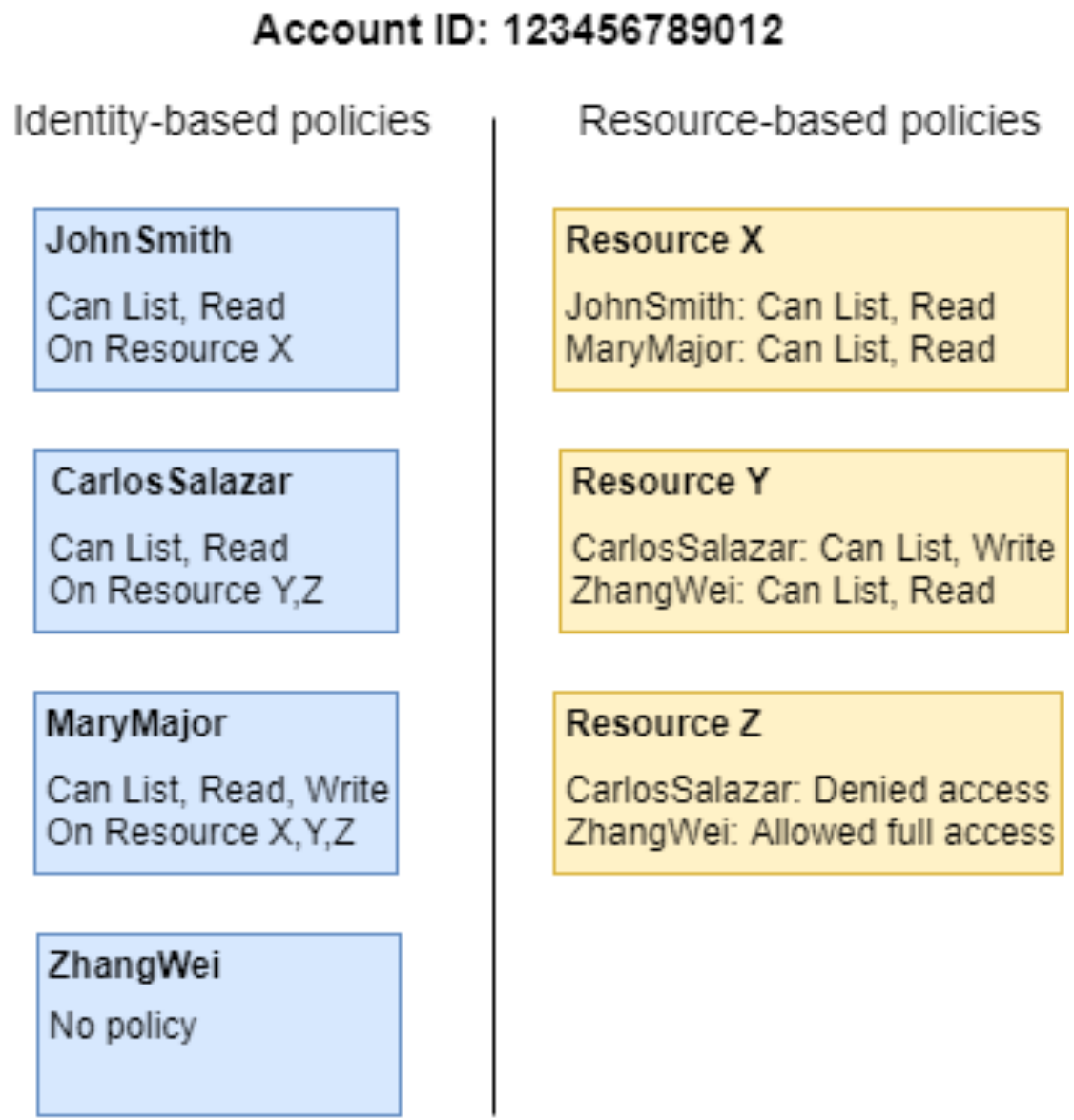

AWS
IAM Users
Master
STS
Assume
Role
AWS
Administrator
role
Development
STS
Assume
Role
AWS
Administrator
role
Staging

IAM Policies

- JSON policy document
- Consist of the following elements:
  - **Effect** – Allow or Deny access to the resource
  - **Action** — Specific to each service (“iam:CreateUser”, “s3:DeleteObject”, etc.).
  - **Resource** — Resource names (like “arn:aws:s3:::conf-* “)
  - **Condition** (Optional) — Grant conditions (like “aws:RequestedRegion”: “ap-south-1”)

IAM Policies

- Identity-based Policies (Attach to users, groups, roles)
- Resource-based Policies - Attach inline policies to resources (S3 buckets or IAM Role trust policies)


Account ID: 123456789012
Identity-based policies
Resource-based policies
John Smith
Can List, Read
On Resource X
Carlos Salazar
Can List, Read
On Resource Y,Z
MaryMajor
Can List, Read, Write
On Resource X,Y,Z
ZhangWei
No policy
Resource X
JohnSmith: Can List, Read
MaryMajor: Can List, Read
Resource Y
CarlosSalazar: Can List, Write
ZhangWei: Can List, Read
Resource Z
CarlosSalazar: Denied access
ZhangWei: Allowed full access

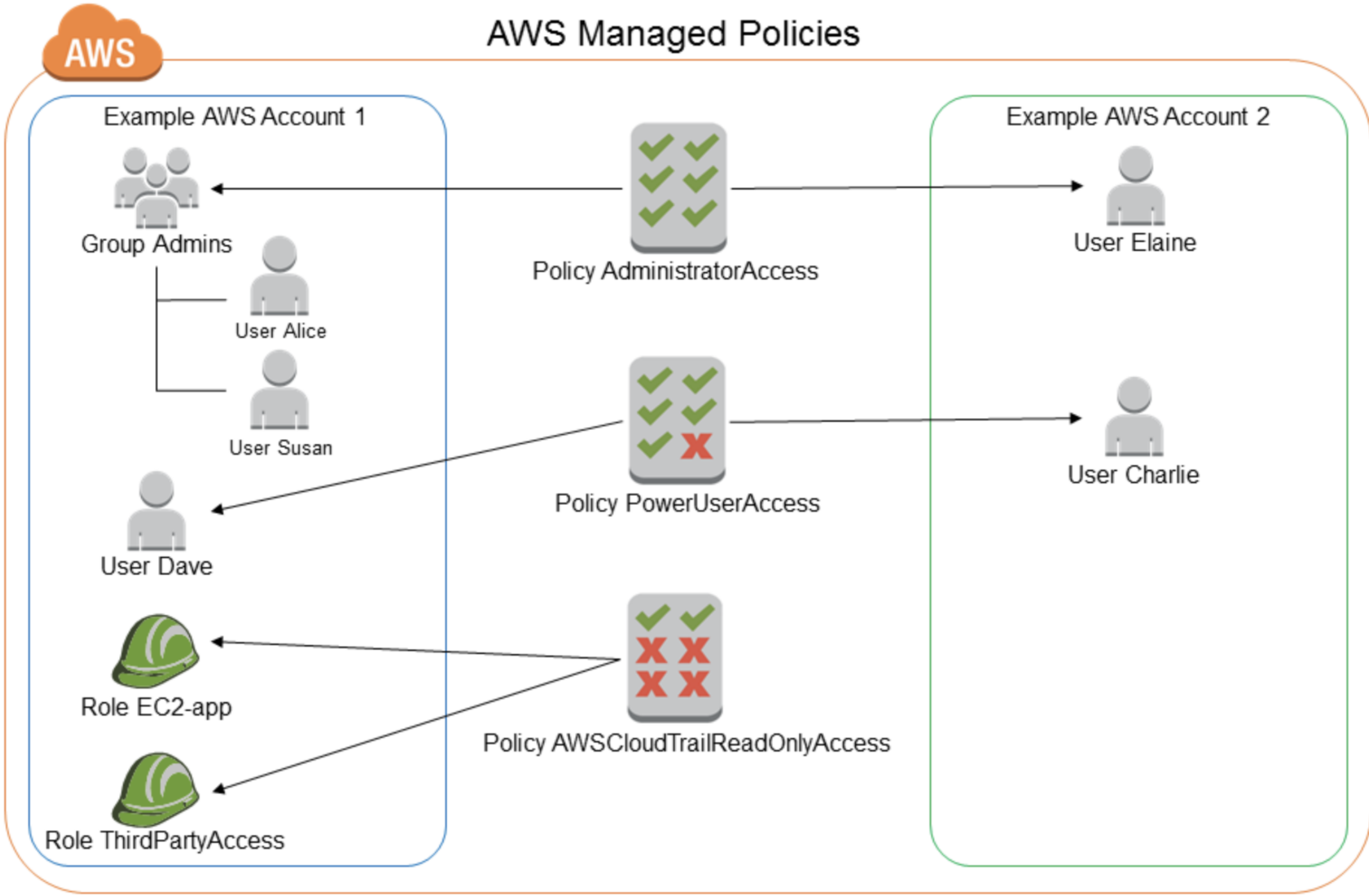
IAM – Identity-based Policies

1. AWS Managed Policies
2. Customer Managed Policies
3. Inline Policies

- Can be attached to a principal (or identity), such as
  - IAM users, groups and roles

IAM AWS Managed Policies

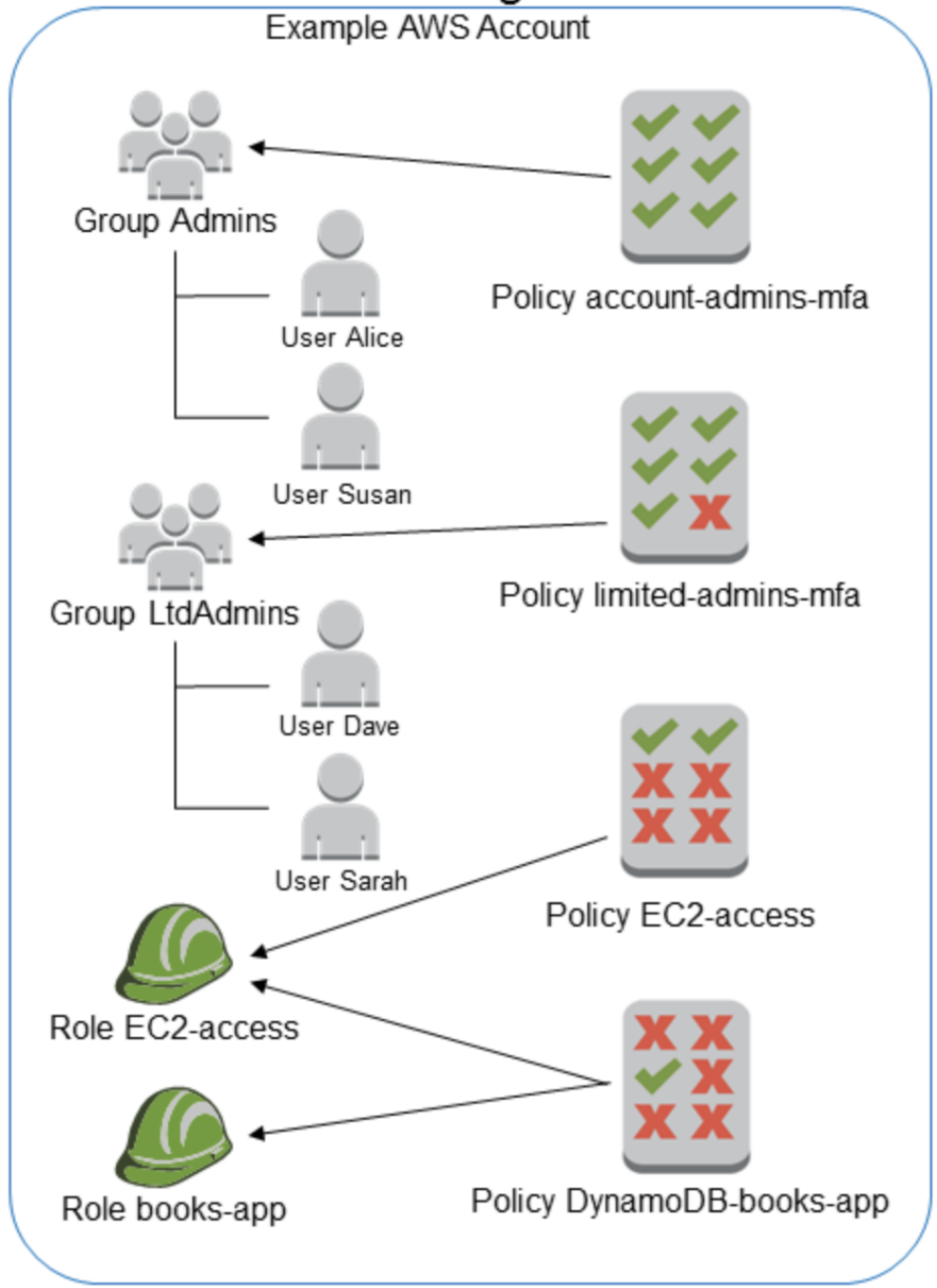
- Created and Managed by AWS
- Can be attached to:
  - Users
  - Groups
  - Roles
- Has its own Amazon Resource Name (ARN)
- Can be access by multiple AWS accounts


AWS
AWS Managed Policies
Example AWS Account 1
Example AWS Account 2
Group Admins
User Alice
User Susan
User Dave
Role EC2-app
Role ThirdPartyAccess
Policy AdministratorAccess
Policy PowerUserAccess
Policy AWSCloudTrailReadOnlyAccess
User Elaine
User Charlie

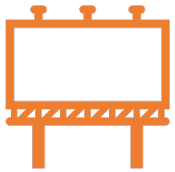
IAM Customer Managed Policies

- Created and Managed by AWS Customers
- Can be attached to:
  - Users
  - Groups
  - Roles
- Has its own Amazon Resource Name (ARN)

Customer Managed Policies



IAM Inline Policies



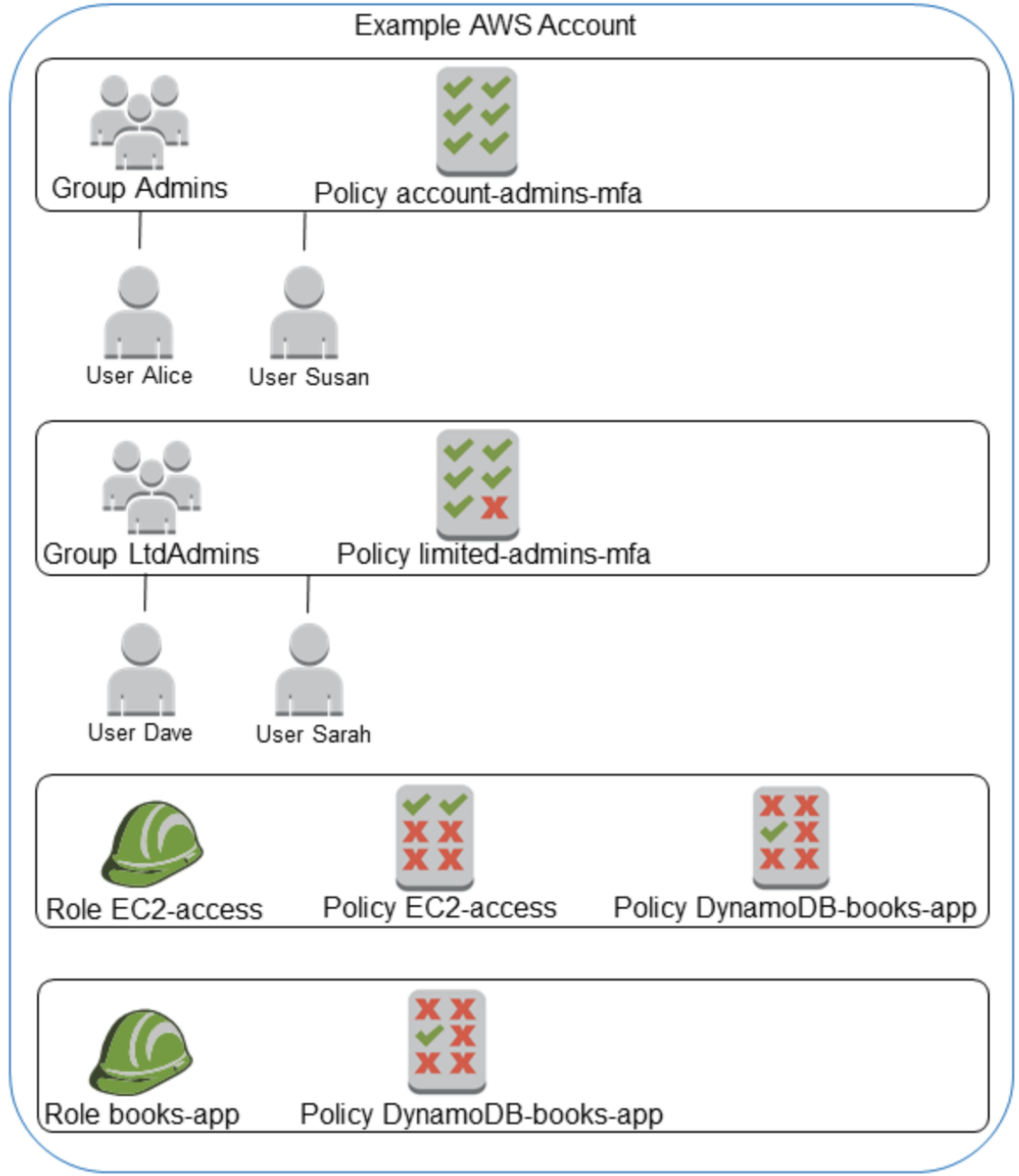
Created by Customers

Embedded in a principal entity (User, Group or Role)

Cannot be reused between multiple principal entities

Deleted when the principal is deleted

Inline Policies



IAM Resource Based Policies

- Can only be attached to resources, such as an S3 bucket
- Are Inline Policies
- Control what a specific principal can perform on the resource

IAM Policy Evaluation

- If a permission isn't mentioned, it's not given
- If a permission is given, that action is allowed UNLESS any other policy explicitly denies that permission.
- the order of how the rules are applied does not matter

Attacking AWS Deployments

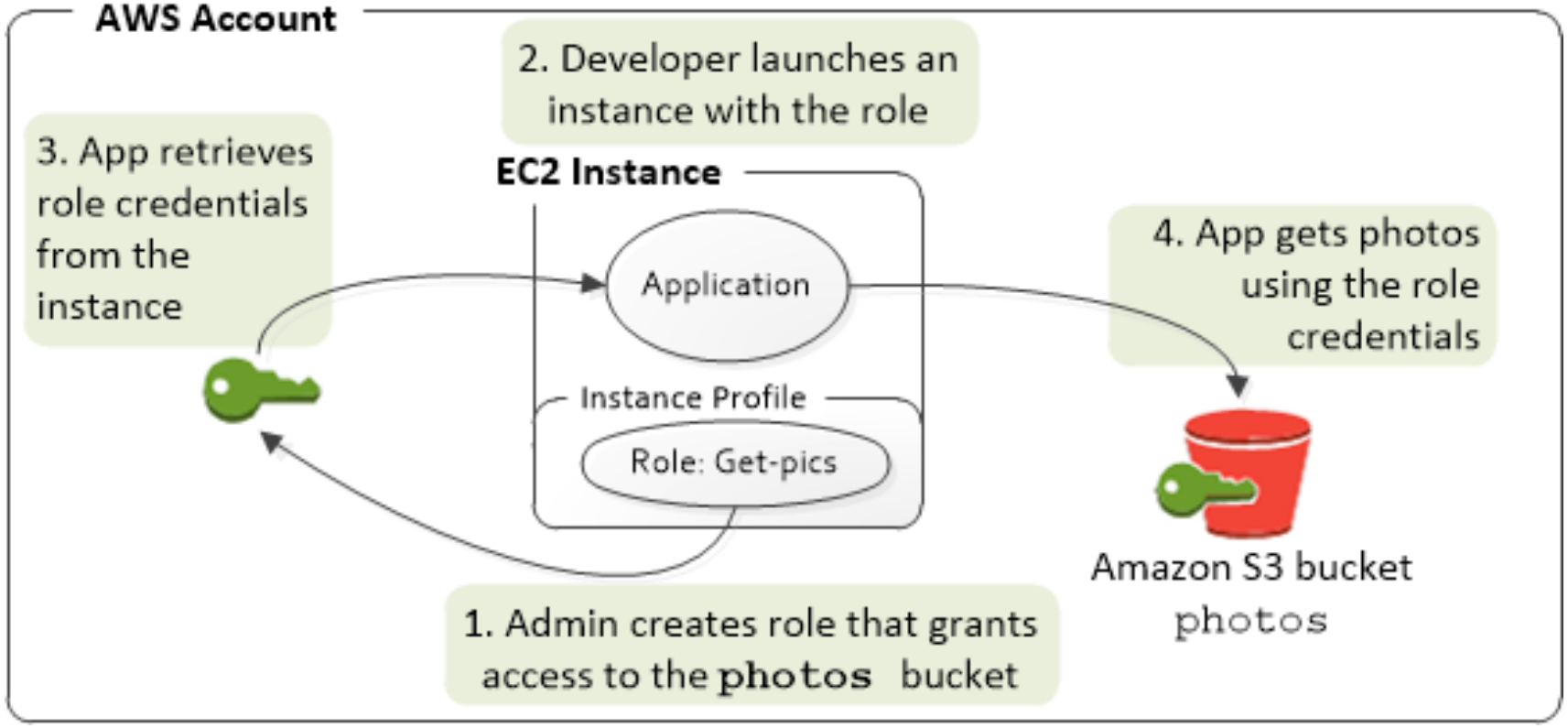
Common High Risk Vulnerabilities

- Public access to resources (S3, EBS Volumes, DB Snapshots, etc.)
- EC2 Metadata – Role security tokens accessible from EC2 instances
- Excessive Trust - Anonymous Access to IAM Roles
- AWS Policies allowing excessive privileges

Public Access to Resources

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddPerm",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    }
  ]
}
```

EC2 Metadata – Role security tokens



EC2 Instance Metadata

- EC2 Instance profile's Security Credentials can be accessed via an internal IP address, same for all instances
- http://169.254.169.254/latest/meta-data/iam/security-credentials/role-name

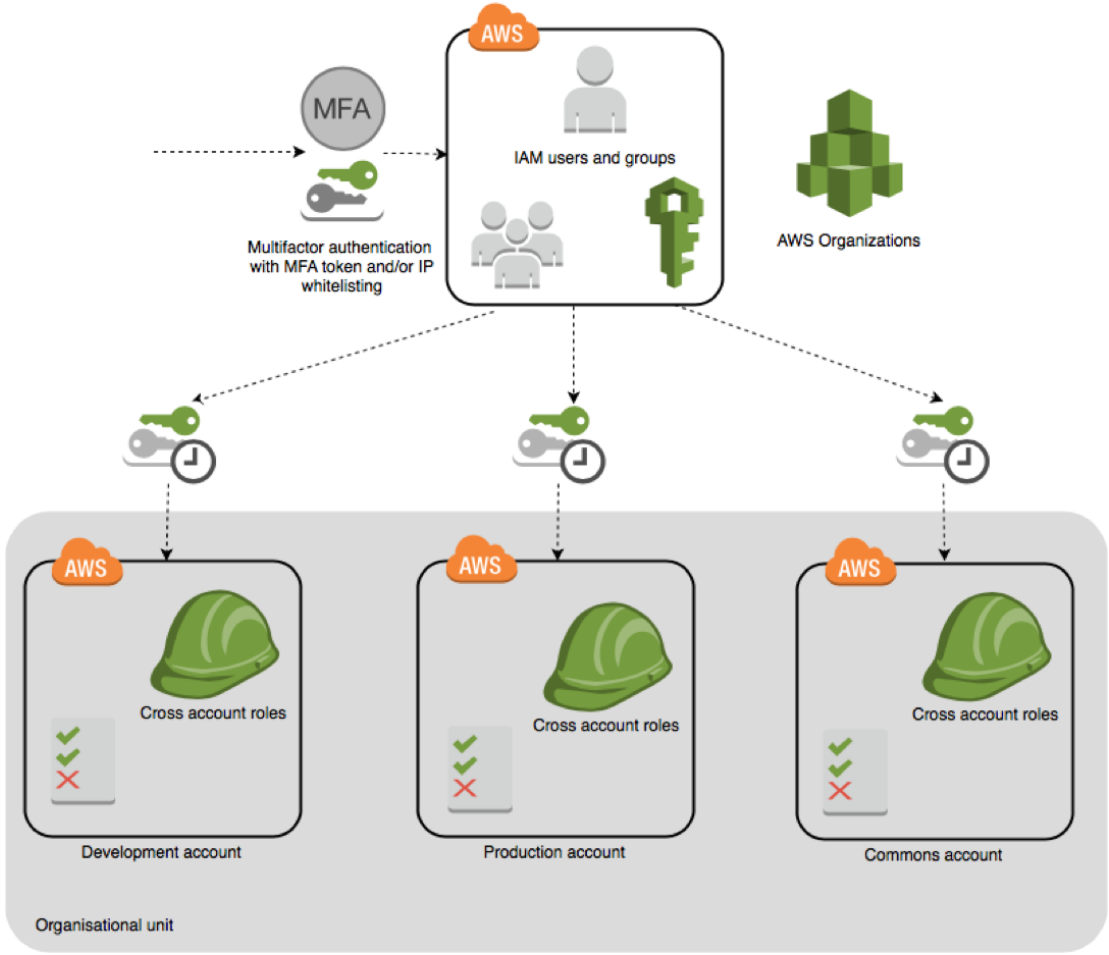
```
{ "Code" : "Success", "LastUpdated" : "2012-04-
26T16:39:16Z", "Type" : "AWS-HMAC",
"AccessKeyId" : "ASIAIOSFODNN7EXAMPLE",
"SecretAccessKey" :
"wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
"Token" : "token", "Expiration" : "2017-05-
17T15:09:54Z" }
```

EC2 Instance Metadata

- Vulnerabilities such as SSRF could allow access to the keys:
  - `window.location="http://169.254.169.254/latest/meta-data/iam/security-credentials/role-name"`

Excessive Trust

- Anonymous trusted entity configured for AWS Roles



Edit Trust Relationship

You can customize trust relationships by editing the following access control policy document.

Policy Document

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

AWS Policies allowing excessive privileges

Sample Customer Managed Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Admin Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

IAM Privileged Permissions

- Permission to modify policy versions
  - iam:CreatePolicyVersion
  - iam:SetDefaultPolicyVersion

IAM Privileged Permissions

- Permission to attach different policies
  - iam:PutGroupPolicy
  - iam:PutRolePolicy
  - iam:PutUserPolicy
  - iam:AttachGroupPolicy
  - iam:AttachRolePolicy
  - iam:AttachUserPolicy

IAM Privileged Permissions

- Permission to change group management
  - iam:AddUserToGroup
- iam:AddUserToGroup
  - iam:AddUserToGroup

IAM Privileged Permissions

- Modify Role Trust Policy
  - iam:UpdateAssumeRolePolicy
  - sts:AssumeRole
- Pass roles to EC2 instances/Lambda functions/cloudformation stacks/Sagemaker/codestar to gain evelated permissions
  - iam:PassRole
  - ec2:RunInstances
  - sagemaker:CreatePresignedNotebookInstanceUrl
  - sagemaker:CreateNotebookInstance
  - lambda:UpdateFunctionConfiguration
  - codestar:CreateProject

Checking for Insecure IAM Configurations

- Check for IAM user permissions
- User's Group memberships
- Group permissions
- Identify what Roles are available to assume

Checking for Insecure IAM Configurations

- Manually check each user, group & role in IAM
- Use the aws cli:
  - `aws iam help`
  - `for i in `aws iam list-users --output text | awk '{print $7}' | awk 'NF'`; do (echo "Inline Policies for $i"; aws iam list-user-policies --user-name $i; echo "Managed Policies for $i"; aws iam list-attached-user-policies --user-name $i); done`
- Tools are available to make this simpler

Checking for Insecure IAM Configurations

- Least privileges not configured for users, roles, and groups
- Get All Inline & Managed Policies Associated with All IAM Users
- ```
  $ for i in `aws iam list-users --output text | awk '{print $7}' | awk 'NF'`; do (echo "Inline Policies for $i"; aws iam list-user-policies --user-name $i; echo "Managed Policies for $i"; aws iam list-attached-user-policies --user-name $i); done
  ```
- Get All Inline & Managed Policies Associated with All IAM Groups
- ```
  $ for i in `aws iam list-groups --query 'Groups[*].{GroupName:GroupName}' --output text`; do (echo "Inline Policies for $i"; aws iam list-group-policies --group-name $i; echo "Managed Policies for $i"; aws iam list-attached-group-policies --group-name $i); done
  ```

IAM Confused

Labs

- Intro to IAM
- Manual privilege escalation scenarios
- Privilege escalation using tools

Lab URL - https://github.com/nemo-wq/privilege_escalation

Intro to IAM/ pre-requisites

- Log into AWS Console & setup IAM account with Admin permissions
- Run `aws configure` using the security tokens

Lab – Create IAM User & Named Profile

- Admin IAM user required to complete the Lab
- Use the aws cli to create a new user
  - `aws iam create-user --user-name brucon1`
- Create access keys, note the SecretAccessKey & AccessKeyId
  - `aws iam create-access-key --user-name brucon1`
- Setup awscli with a new profile, type in the newly generated AccessKey
  - `aws configure --profile brucon1`
    - `region = ap-southeast-2`
    - `SecretAccessKey = Generated above`
    - `AccessKeyId = Generated above`

Lab - Attach AWS Managed Policy

- Attach the SecurityAudit AWS Managed policy to the brucon1 user:
  - `aws iam attach-user-policy --user-name brucon1 --policy-arn arn:aws:iam::aws:policy/SecurityAudit`
- Check attached policy:
  - `aws iam list-attached-user-policies --user-name brucon1` **`--profile brucon1`**

```
jay@mb02:~$ aws iam list-attached-user-policies --user-name brucon1
--profile brucon1
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ]
}
```

Lab - Detach Managed Policy

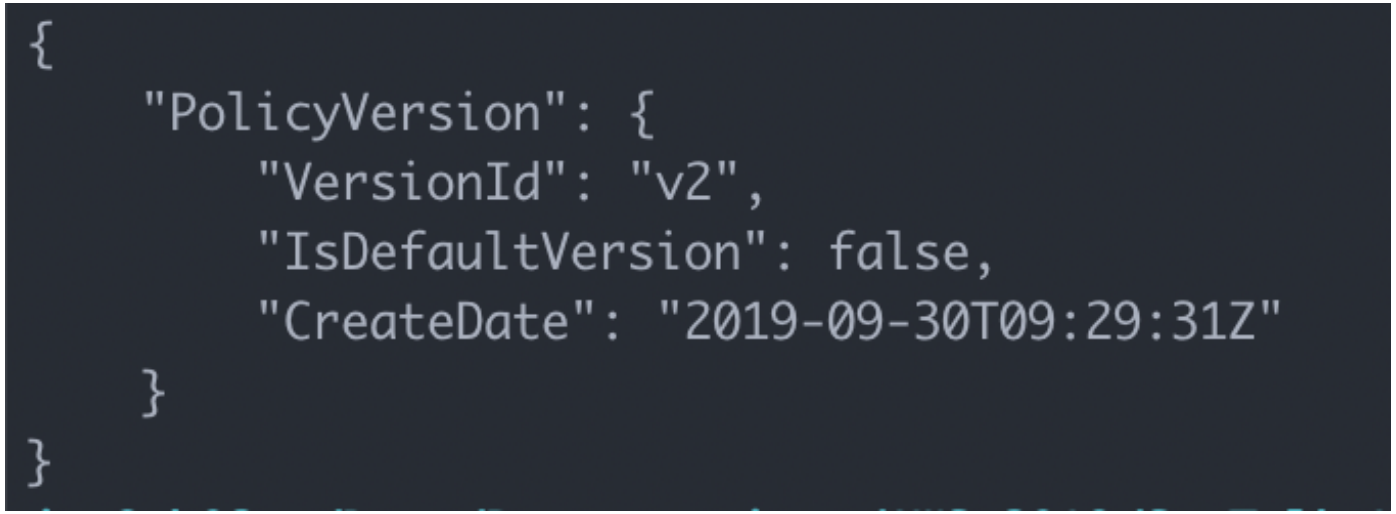
- `aws iam detach-user-policy --user-name brucon1 --policy-arn arn:aws:iam::aws:policy/SecurityAudit`

Lab - Create new Customer Managed Policy

- Add new Managed Policy to your account:
  - `aws iam create-policy --policy-name brucon-policy --policy-document file://policy1.txt`

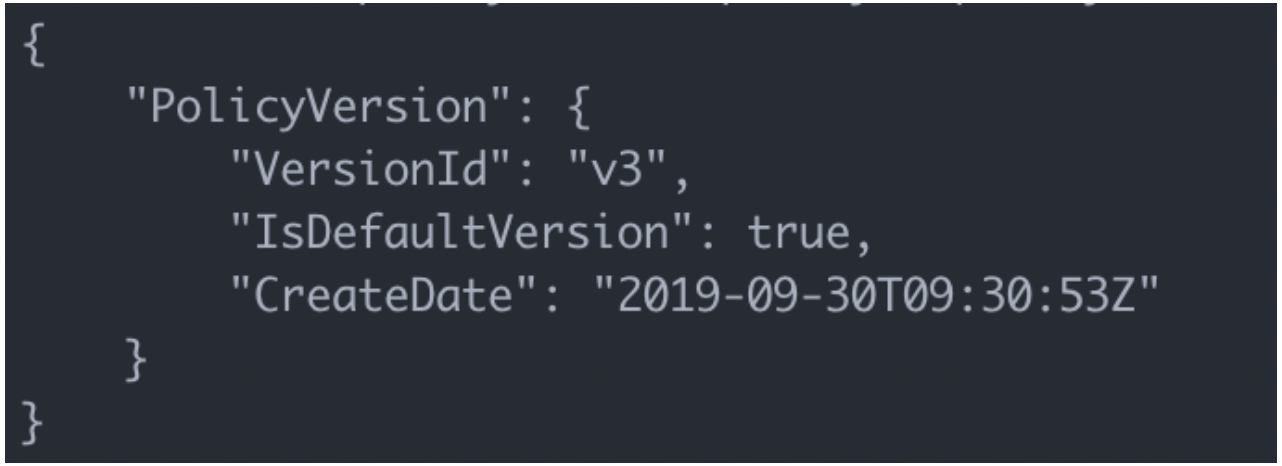
Lab - Create version 2 of brucon-policy

- `aws iam create-policy-version --policy-arn arn:aws:iam::<your-account-number>:policy/brucon-policy --policy-document file://policy2.txt`



Lab - Create version 3 & set as default

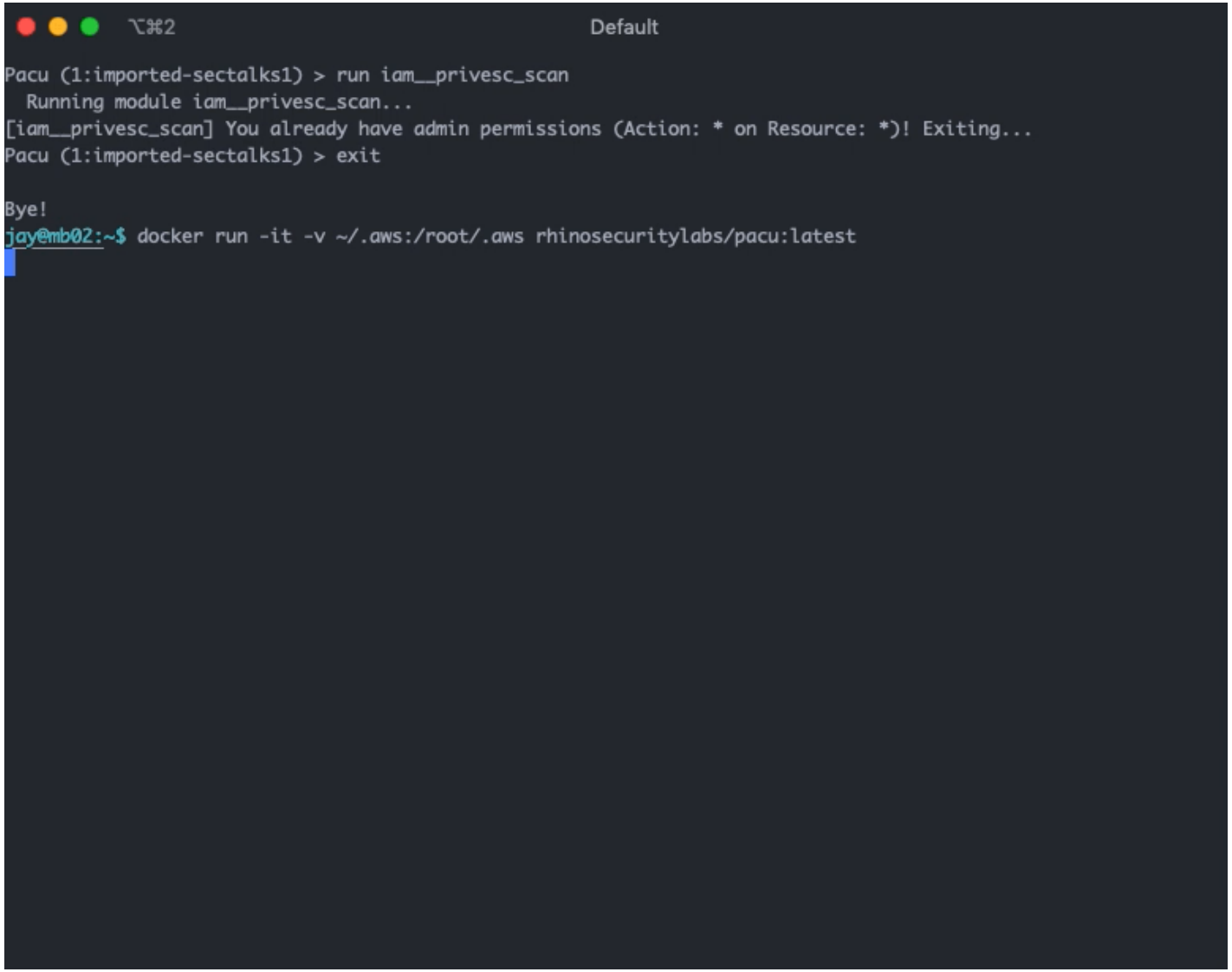
- `aws iam create-policy-version --policy-arn arn:aws:iam::<your-account-number>:policy/brucon-policy --policy-document file://policy3.txt --set-as-default`



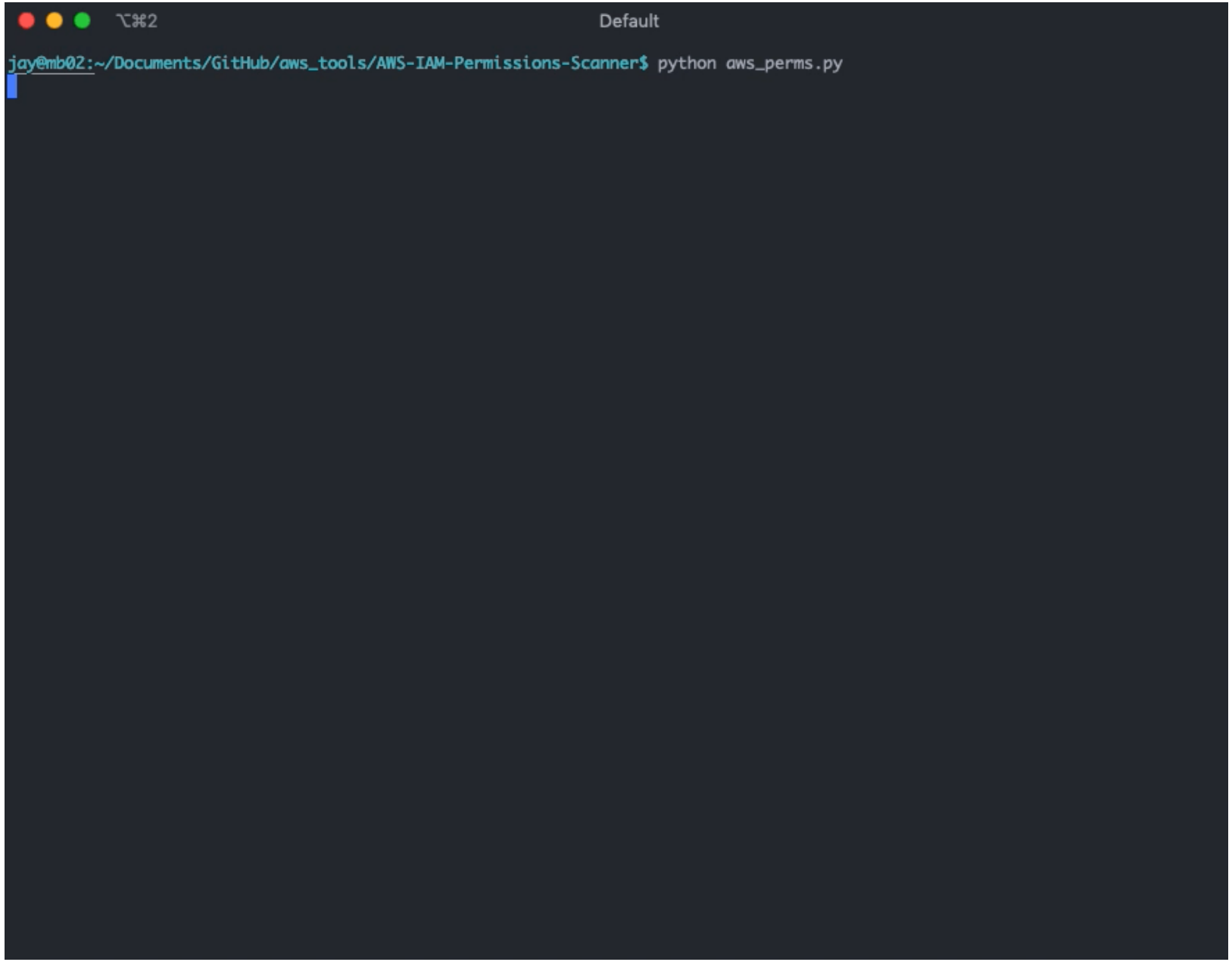
Lab - Attach the BruConPolicy to brucon1 user

- `aws iam attach-user-policy --user-name brucon1 --policy-arn arn:aws:iam::<your-account-number>:policy/BruConPolicy`

Rhino Security's Pacu - https://github.com/RhinoSecurityLabs/pacu



AWS-IAM-Permissions-Scanner - https://github.com/nemo-wq/AWS-IAM-Permissions-Scanner



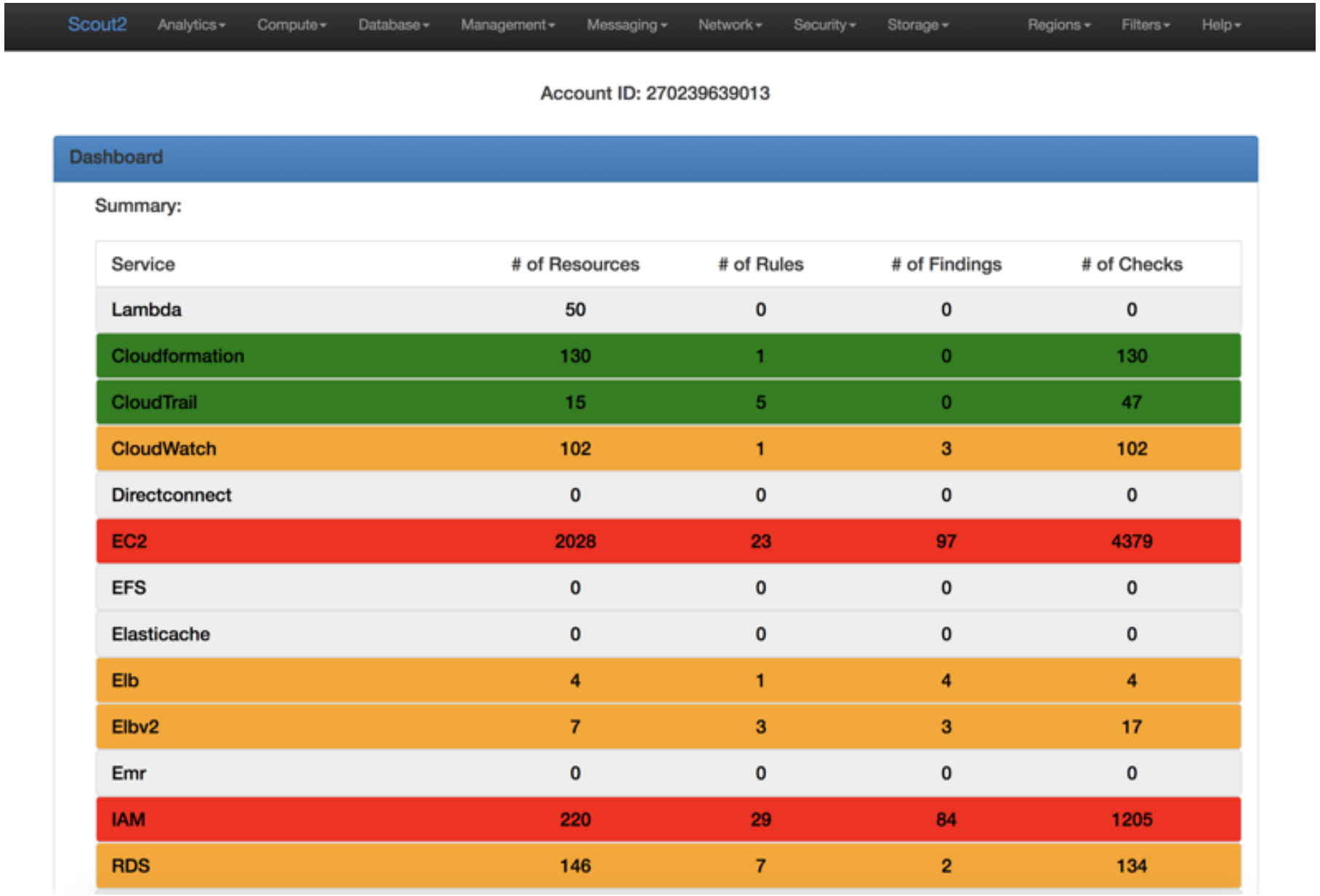
Tools & Resources - IAM

- Audit AWS Permissions
  - AWS-IAM-Permissions-Scanner - https://github.com/nemo-wq/AWS-IAM-Permissions-Scanner
  - SkyArk - https://github.com/cyberark/SkyArk
  - Rhino Security's AWS_Escalate - https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools/aws-pentest-tools/aws_escalate.py
  - Rhino Security's Pacu - https://github.com/RhinoSecurityLabs/pacu
  - NCC Group's Pmapper - https://github.com/nccgroup/PMapper
  - NCC Group's ScoutSuite - https://github.com/nccgroup/ScoutSuite

Rhino Security's Pacu - https://github.com/RhinoSecurityLabs/pacu

```
  backdoor_assume_role
  privesc_scan

[Category: logging_monitoring]

  dl_cloudtrail_event_history
  disrupt_monitoring
  dl_cloudwatch_logs
  enum_monitoring

[Category: persistence]

  backdoor_users_password
  backdoor_users_keys

[Category: post_exploitation]

  cloudtrail_csv_injection
  add_ec2_startup_sh_script
  backdoor_ec2_sec_groups
  sysman_ec2_rce
  download_lightsail_ssh_keys

[Category: recon_enum_no_keys]

  s3_finder

[Category: recon_enum_with_keys]

  inspector_report_fetcher
  enum_ebs_volumes_snapshots
  download_ec2_userdata
  enum_ec2_termination_protection
  get_credential_report
  enum_ec2
  confirm_permissions
  enum_glue
  enum_codebuild
  enum_lightsail
  enum_users_roles_policies_groups
  s3_bucket_dump
  enum_lambda
  enum_elb_logging

Pacu (TestSession:TestUser) >
```

NCC Group's ScoutSuite - https://github.com/nccgroup/ScoutSuite



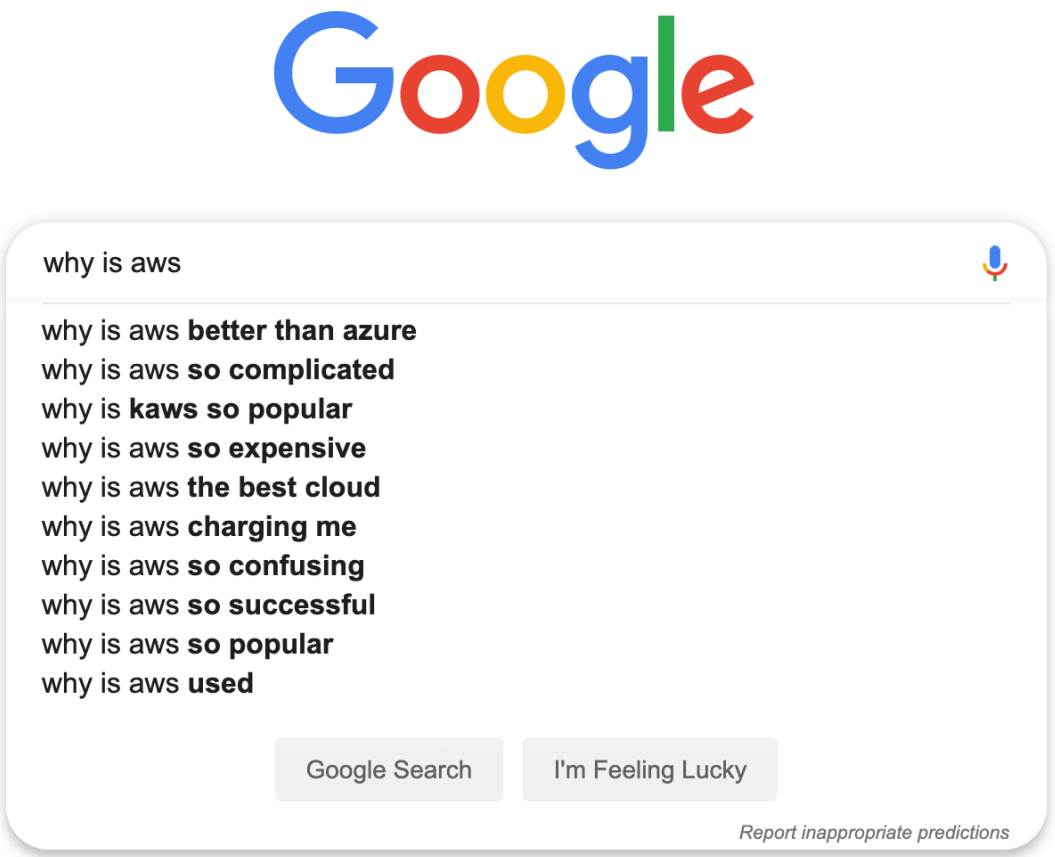
Summary

- High complexity of IAM can lead to higher likelihood of misconfigurations
- Use managed or customer managed policies where possible, avoiding Inline policies
- Perform regular audits of IAM users, roles and permissions

Credits

- Rhino Security Labs - https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation/
- NCC Labs - https://www.owasp.org/images/c/cc/AWS_Security_-_Staying_on_Top_of_the_Cloud.pdf
- CyberArk - https://github.com/cyberark/SkyArk

Questions?


Google
why is aws
why is aws better than azure
why is aws so complicated
why is kaws so popular
why is aws so expensive
why is aws the best cloud
why is aws charging me
why is aws so confusing
why is aws so successful
why is aws so popular
why is aws used
Google Search
I'm Feeling Lucky
Report inappropriate predictions