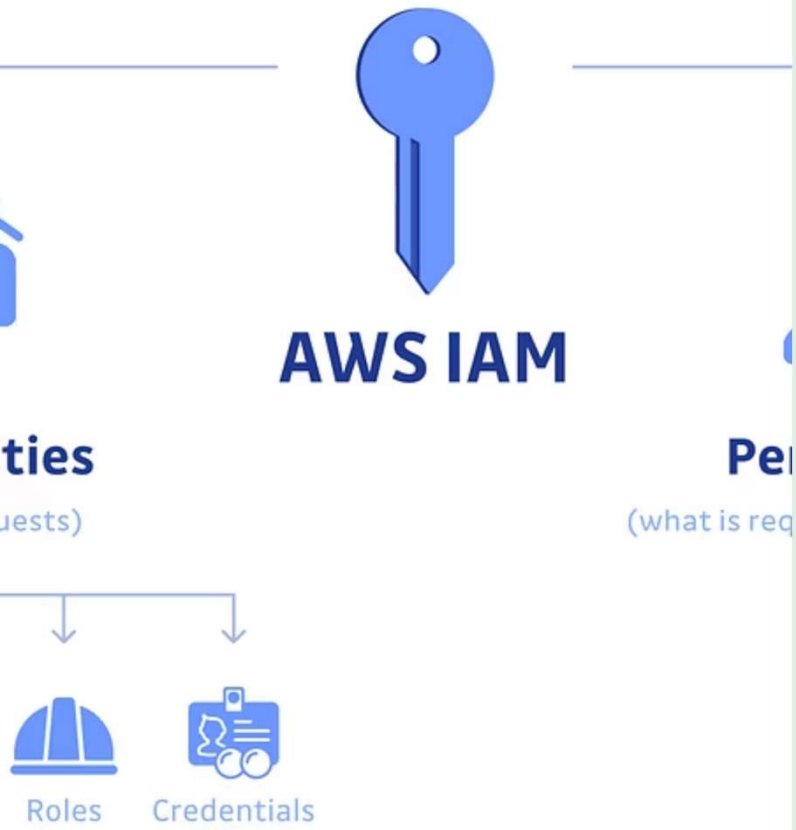
AWS IAM Policy Simulator & Best Practices + STS & Role Assumption

Duration: 60 min | Level: Intermediate

This presentation covers two essential AWS security topics: the IAM Policy Simulator for testing and debugging permissions, and STS for secure cross-account access. We'll explore best practices for both to help you implement robust security controls in your AWS environment.



Part 1

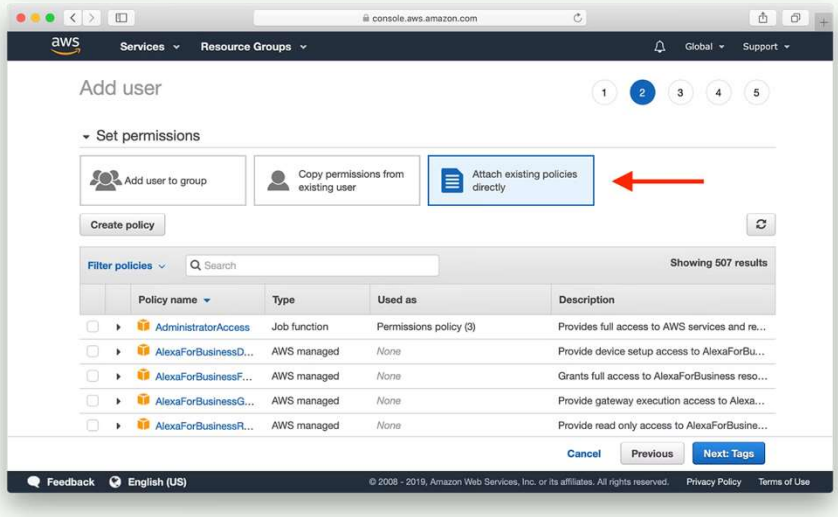
IAM Policy Simulator & Best Practices

Introduction to IAM Policy Simulator

The **AWS IAM Policy Simulator** is a service designed to test IAM and resource-based policies before applying them to your production environment.

This powerful tool helps you:

- Validate permissions before applying them
- Reduce trial-and-error in production environments
- Debug frustrating "AccessDenied" issues



Access the simulator at: https://policysim.aws.amazon.com

Key Features of Policy Simulator

Comprehensive Policy Testing

Evaluate IAM policies, Service Control Policies (SCPs), and resource-based policies in a safe environment

Entity Coverage

Test permissions for users, groups, and roles to ensure proper access control

API Simulation

Simulate specific API calls to verify if they would be allowed or denied

Clear Results

View detailed allow/deny results with explanations of which policy statements affected the decision

Multi-Service Support

Test permissions across multiple AWS services from a single interface

How to Use IAM Policy Simulator

Demo Workflow

1. ### Open Policy Simulator

   Access the simulator through the AWS Console or directly at policysim.aws.amazon.com

2. ### Select Identity

   Choose the User, Group, or Role whose permissions you want to test

3. ### Choose Service

   Select the AWS service you want to test (e.g., S3, EC2, DynamoDB)

4. ### Select Actions

   Choose specific API actions to test (e.g., s3:ListBucket, ec2:DescribeInstances)

5. ### Run Simulation

   Execute the test and check results (Allow/Deny + detailed explanation)

6. ### Iterate

   Modify the policy and re-simulate until you achieve the desired permissions

Common Use Cases

Debugging Permission Issues

Quickly identify why a user is receiving "AccessDenied" errors without modifying production policies

Testing New Policies

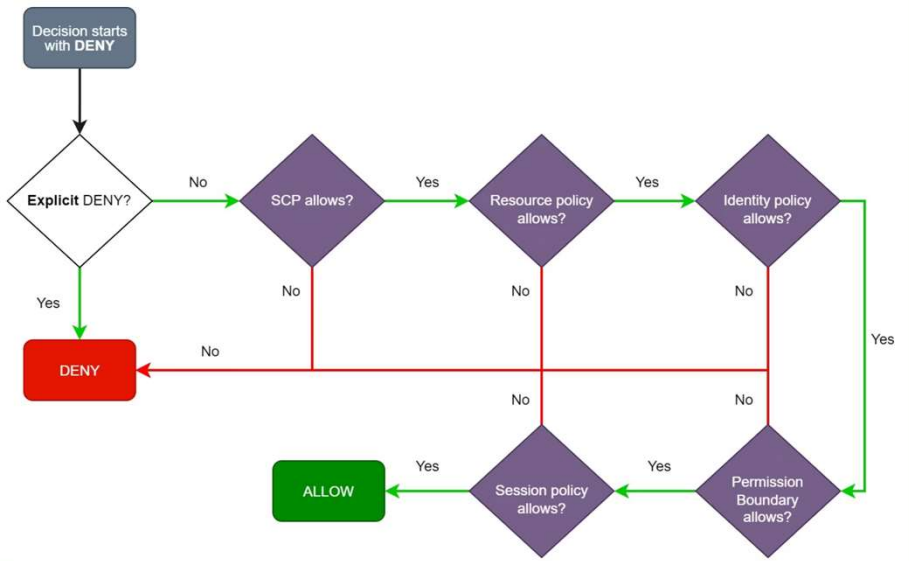
Validate new or updated policies before deployment to prevent accidental permission gaps or excessive access

Validating SCP Impacts

Understand how Service Control Policies in AWS Organizations affect permissions across your organization

Reviewing Third-Party Permissions

Verify the actual permissions granted to third-party roles before allowing them access to your resources



IAM Best Practices: Security & Management

Least Privilege Principle

Grant only the permissions required to perform a task - nothing more

Use Roles Instead of Keys

Prefer temporary role credentials over long-term access keys

Enable MFA

Require multi-factor authentication for all privileged accounts

Use IAM Groups

Assign permissions to groups rather than individual users

Rotate Access Keys

Regularly rotate access keys to minimize risk from compromised credentials

Implement SCPs

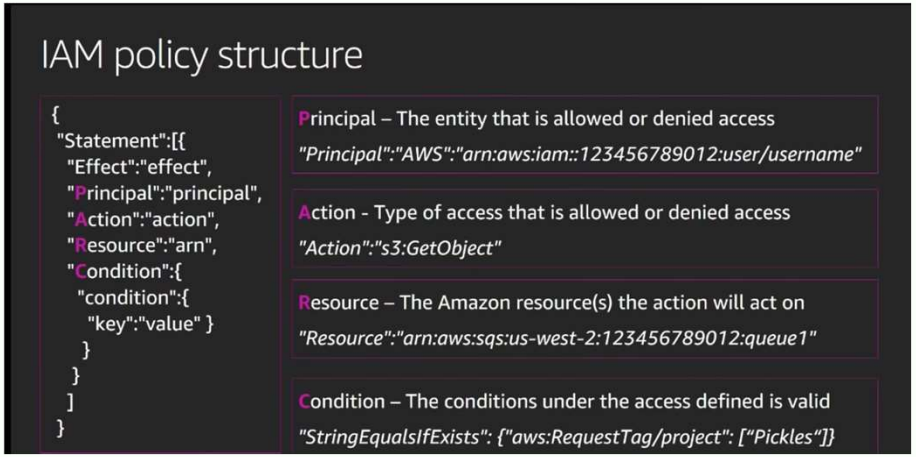
Use Service Control Policies for organization-level permission guardrails

Enable Logging

Log all IAM changes via CloudTrail for comprehensive audit trails

IAM Best Practices: Policy Management

- **Use AWS Managed Policies** when possible for standardized permission sets
- **Restrict wildcard (*) actions** to prevent unintended access
- **Apply Conditions** to limit when policies are effective:
  - aws:SourceIp - restrict by IP address
  - aws:MultiFactorAuthPresent - require MFA
- **Use tags** for attribute-based access control (ABAC)
- **Version control** your IAM policies via Infrastructure as Code (Terraform/CloudFormation)



Quick Demo Idea

5 minutes

Problem

User receives "AccessDenied" when attempting to delete an S3 object

Investigate

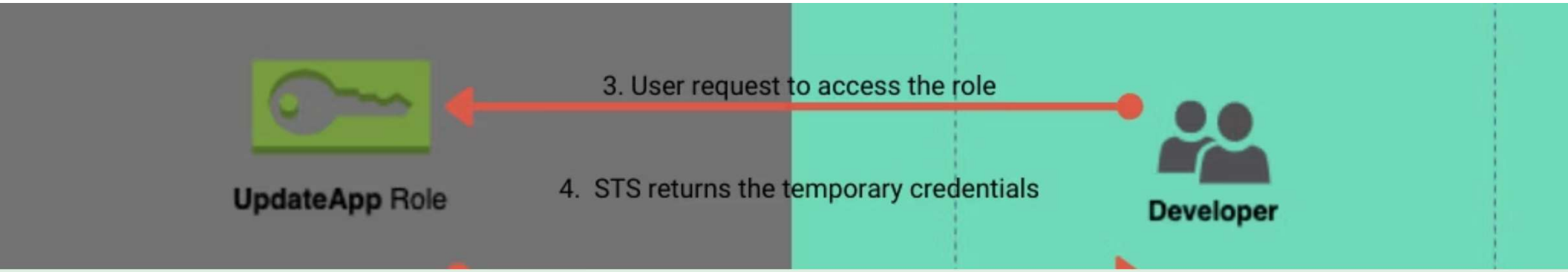
Use Policy Simulator to test s3:DeleteObject permission and identify the missing permission

Fix

Update the IAM policy to include the necessary permission

Verify

Re-test in Policy Simulator to confirm the fix works before applying to production



Part 2

STS & Role Assumption

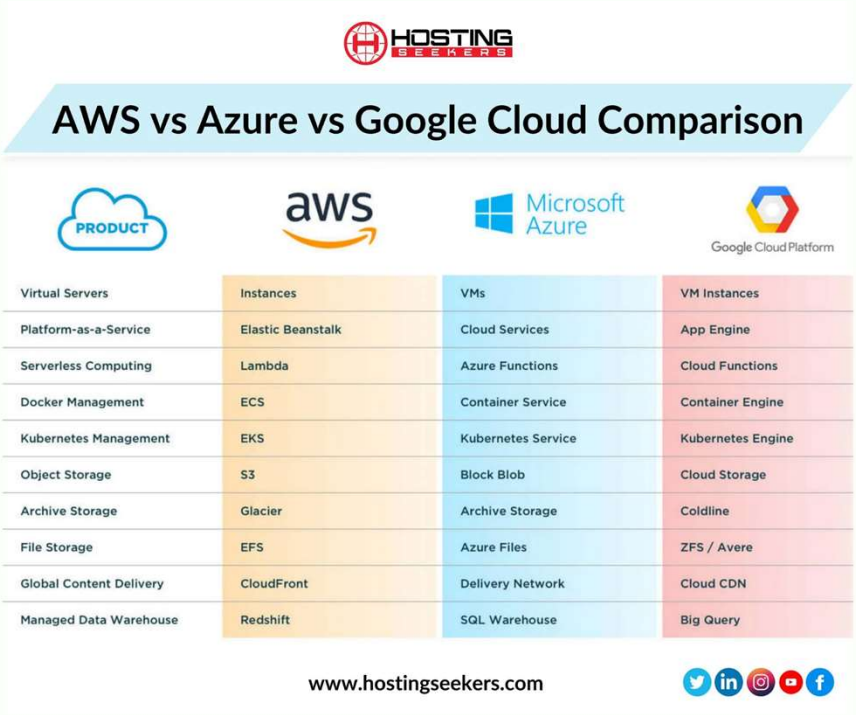
Cross-Account Basics (~30 min)

Introduction to AWS STS

AWS Security Token Service (STS) is a global web service that issues temporary security credentials for accessing AWS resources.

Why Use STS?

- **Cross-account access** - Access resources in different AWS accounts
- **Federated identity** - Integrate with external identity providers (Active Directory, Google, etc.)
- **Temporary privilege escalation** - Grant elevated permissions only when needed



STS credentials are temporary by design, enhancing security by limiting the exposure window if credentials are compromised.

Key STS API Calls

AssumeRole

Switch to a different IAM role within your account or in another AWS account

```
aws sts assume-role --role-arn ROLE_ARN --role-session-name SESSION_NAME
```

GetSessionToken

Get temporary credentials for an IAM user or AWS account root user, typically used with MFA

```
aws sts get-session-token --serial-number MFA_DEVICE --token-code TOKEN
```

AssumeRoleWithSAML

Returns temporary credentials for users authenticated with SAML-based federation

```
aws sts assume-role-with-saml --role-arn ROLE_ARN --principal-arn PRINCIPAL_ARN --saml-assertion SAML_ASSERTION
```

AssumeRoleWithWebIdentity

Returns temporary credentials for users authenticated with an OpenID Connect (OIDC) provider

```
aws sts assume-role-with-web-identity --role-arn ROLE_ARN --web-identity-token TOKEN
```

Role Assumption Basics

Role

A set of permissions that AWS can assign to an authenticated entity

Trust Policy

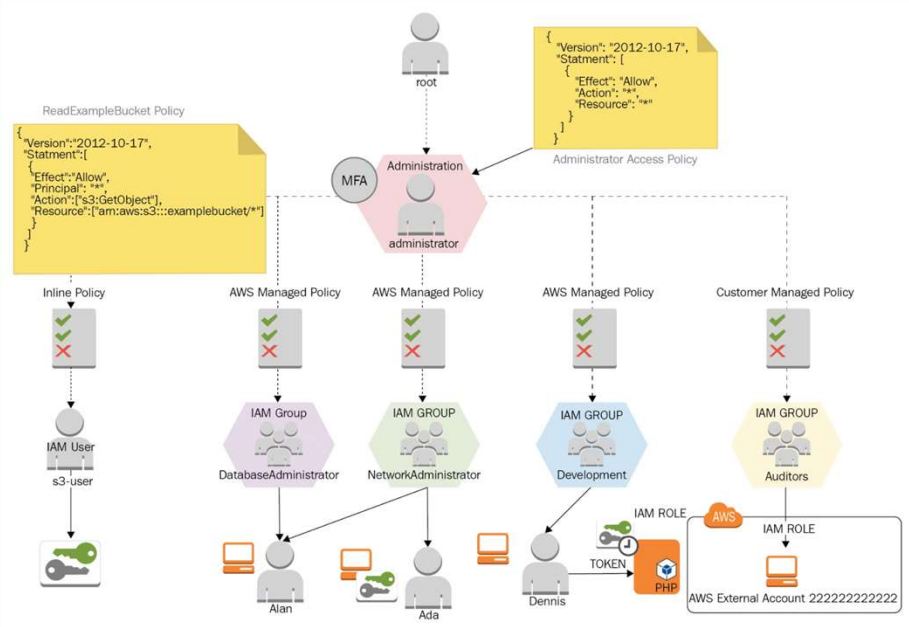
Defines who can assume the role (the trusted entities)

Permission Policy

Defines what actions the role can perform on which resources

Session Duration

15 minutes to 12 hours (default: 1 hour)



Roles are the foundation of temporary access in AWS. They separate *who* can access resources (trust policy) from *what* they can do (permission policy).

Cross-Account Access Flow

Account A (Resource Owner)

- Create IAM Role with trust policy for Account B
- Attach permission policy defining allowed actions

Account B (User Account)

- Use STS AssumeRole to get temporary credentials
- Access Account A's resources using these credentials

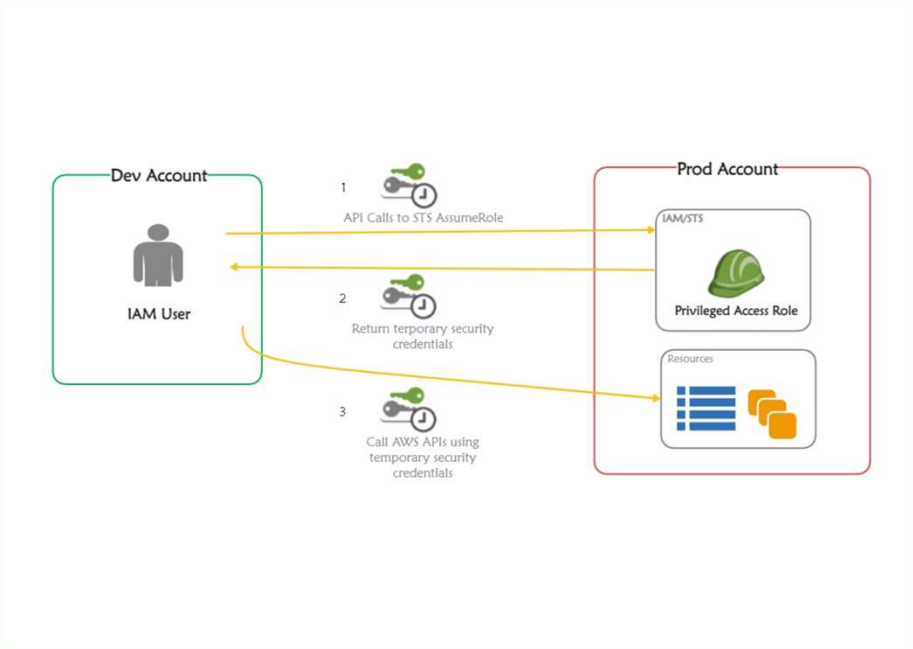
Example: Assume Role CLI Command

```
aws sts assume-role \--role-arn
arn:aws:iam::111122223333:role/CrossAccountS3Access \--
role-session-name MySession
```

The command returns temporary credentials:

- Access Key ID
- Secret Access Key
- Session Token

These credentials can be used with any AWS SDK or CLI by setting the appropriate environment variables or profile configuration.



The role-session-name parameter helps identify who assumed the role in CloudTrail logs, aiding in security auditing.

Security Best Practices for STS

Use Short Session Durations

Limit session duration to the minimum time needed to complete tasks, reducing the window of opportunity if credentials are compromised

Require MFA for Role Assumption

Add the aws:MultiFactorAuthPresent condition to trust policies for sensitive roles

Limit Trusted Principals

Be specific about which entities can assume roles; avoid wildcards in trust policies

Log AssumeRole Events

Monitor CloudTrail for suspicious role assumption patterns or unauthorized attempts

Avoid Using STS for Permanent Access

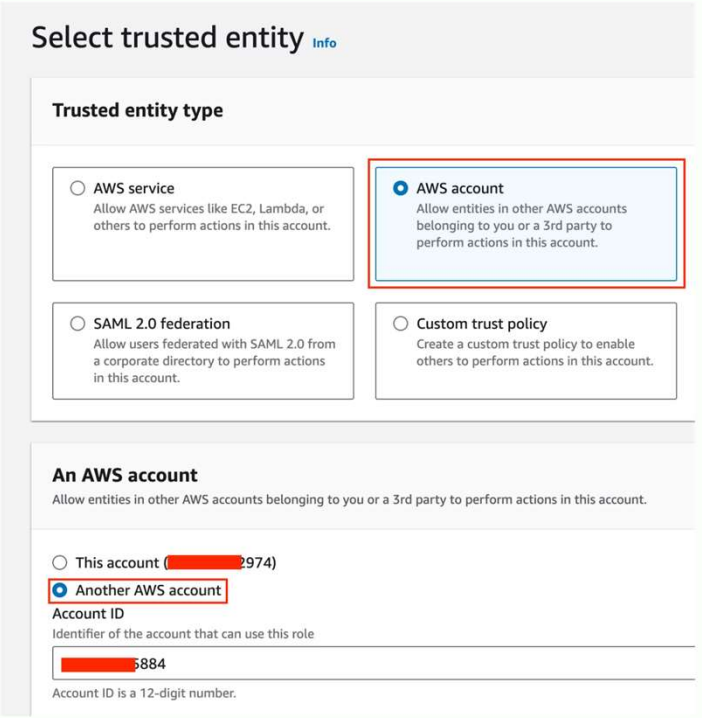
Don't script continuous role assumption as a workaround for proper cross-account access design

Demo Idea

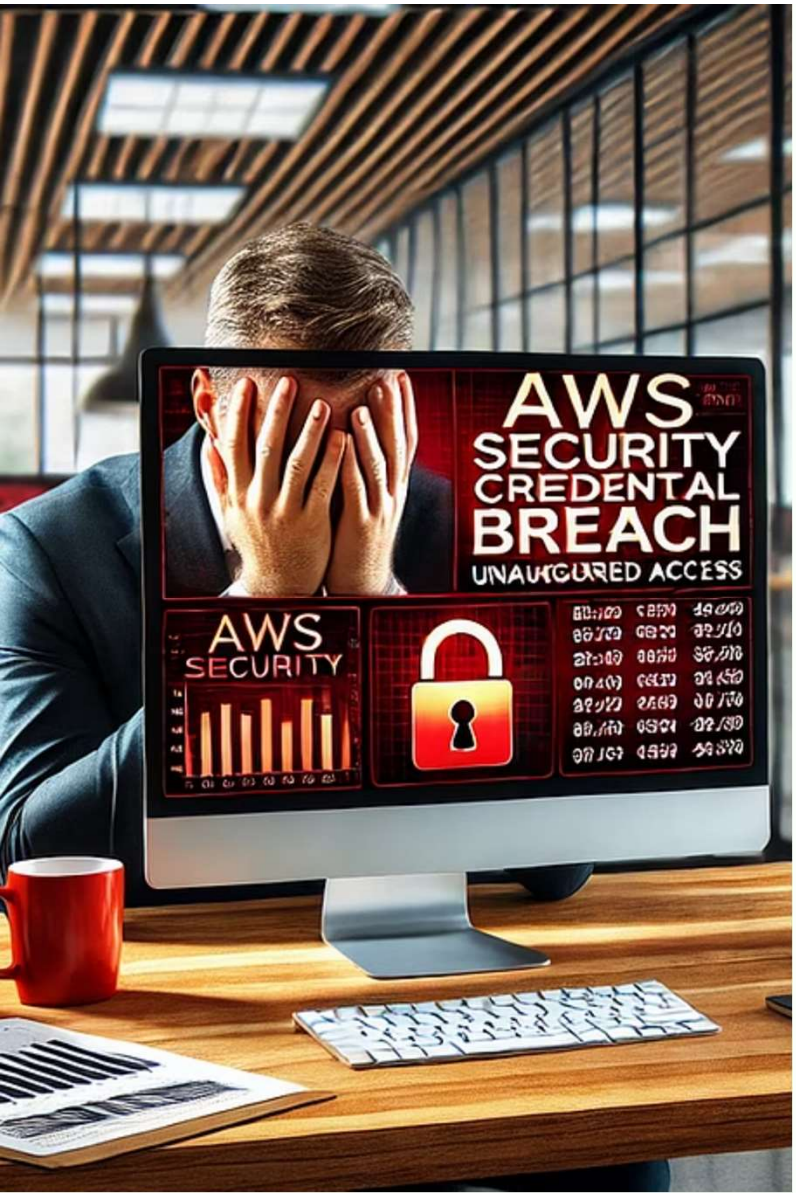
10 minutes

Setup:

1. Create Role in Account A with S3 read permissions
2. Configure trust policy to allow Account B's user to assume the role
3. From Account B, assume the role using STS
4. List S3 objects in Account A using the temporary credentials



This demo illustrates the complete cross-account access workflow from role creation to successful resource access.



Recap & Q&A

Policy Simulator

Test and debug IAM policies before applying them to production, reducing the risk of permission errors or excessive access

IAM Best Practices

Implement secure, least-privilege access controls with proper management processes for auditable security

STS & Roles

Enable secure temporary and cross-account access using role-based permissions and the Security Token Service

Questions? We'll now open the floor for discussion about implementing these practices in your environment.