

**Ghulam Ishaq Khan Institute of engineering & sciences**

# CEH Module 19 - Cloud Computing

### Lab Report

**Submitted By:**

Rubab Ali Qureshi

Sofia Faisal

Yashfa Fatima

### Roll Numbers:

2023614

2023519

2023762

### Course:

CY201 - Cyber Security Principles and Concepts

### Instructor:

Mr. Abdullah Bin Zarshaid

Submission date : May 15, 2025

This module focuses on the security aspects of cloud computing environments,

particularly AWS services. The objective is to identify and exploit common misconfigurations in cloud infrastructure, demonstrate various enumeration techniques for discovering vulnerable cloud resources, and explore privilege escalation methods in AWS environments. Through these labs, we aim to understand attack vectors related to cloud infrastructure and develop countermeasures to protect against such threats.

* **Attacker OS:** Kali Linux / Parrot Security OS
* **Target:** AWS Cloud Infrastructure (Free Tier Account)
* **Virtualization Tool:** VirtualBox
* **Network Settings:** NAT connection established in vm

#### Additional Tools:

* + AWS CLI
  + Various S3 Enumeration Tools (LazyS3, S3Scanner, aws cli command)
  + Pip3
* **AWS CLI v2** - Used to interact with AWS services through command line
* **LazyS3** - Used for S3 bucket enumeration through permutation techniques
* **S3Scanner** - Used for detailed S3 bucket access permission analysis
* **Git** - Used for version control and project submission
* **Terminal/Bash** - Used for executing commands and scripts

# Lab 1: S3 Bucket Enumeration using Various Tools

## Objective of Task

To identify publicly accessible AWS S3 buckets using multiple enumeration techniques and tools, demonstrating how cloud storage misconfigurations can lead to potential data exposure.

## Task 1: Setting Up the Environment

### Command Used:

1 sudo apt install -y python3 - pip python -is - python3 git firefox

### Screenshot:

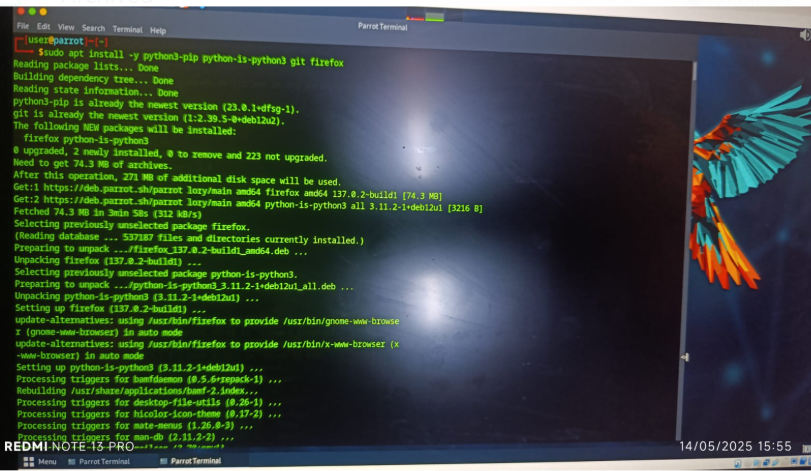
### identifier:

### Note : Identifier is that photo was taken from my phone as mentioned below redmi note

### 13 pro with date ,after this next proceeding are the screenshot directly taken from laptop

### But the screenshot doc in git hub will contain photos from my phone such as this .

### (ignore first screenshot it is an identifier)



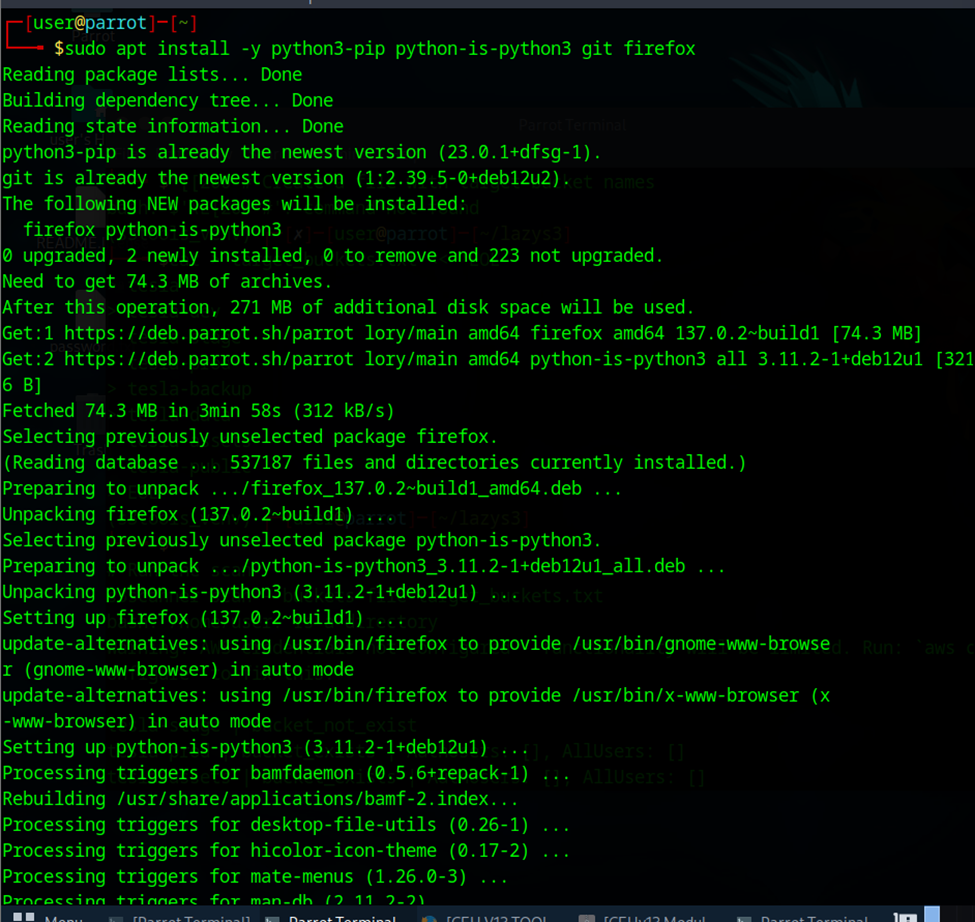


Figure 4.1: Installation of required dependencies

### Result/Output:

Successfully installed required dependencies for the S3 enumeration tools.

### Interpretation:

These packages are necessary prerequisites for the various tools we’ll be using to enumer- ate S3 buckets. Python packages are essential for S3Scanner, while Git is needed to clone the LazyS3 repository.

## Task 2: Installing and Using LazyS3

### Command Used:

# Clone the repository

git clone https://github.com/nahamsec/lazys3.git

cd lazys3

# Make the script executable

chmod +x lazys3.rb

# Make the script executable

chmod +x lazys3.rb

# Install required Ruby gem

sudo gem install colorize

### Screenshot:



Figure 4.2: LazyS3 installation

### Result/Output:

Successfully cloned the LazyS3 repository and installed the required Ruby gem.

### Interpretation:

LazyS3 is a Ruby script that creates common permutations of S3 bucket names based on a company name input. It helps in identifying potentially exposed S3 buckets by testing various naming conventions.

## Task 3: Using LazyS3 for Bucket Enumeration

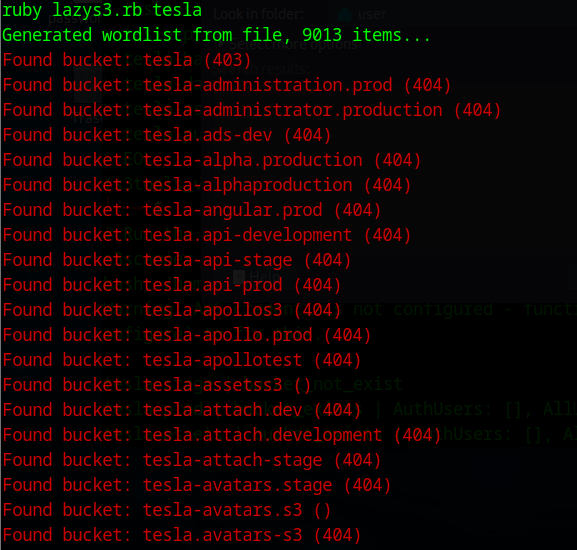
### Command Used:

# Install required Ruby gem

sudo gem install colorize

### Screenshot:

**example 1**



Example 2:

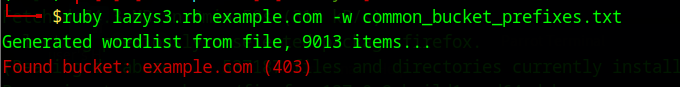


Figure 4.3: LazyS3 execution results

### Result/Output:

The tool generated several permutations of potential bucket names for ”tesla” and example.com and tested their accessibility. Some buckets were found to be publicly accessible.

### Interpretation:

LazyS3 successfully identified multiple S3 buckets related to the target name ”tesla”and example. The color-coded output showed which buckets were accessible (green) and which were not (red). This demonstrates how easily attackers can discover misconfigured buckets using simple enumeration techniques.

## Task 4: Installing and Using S3Scanner

### Command Used:

1

python 3 -m venv ~/ s3 tools\_venv

source ~/ s3 tools\_venv / bin / activate pip install s3 scanner

2

3

### Screenshot:

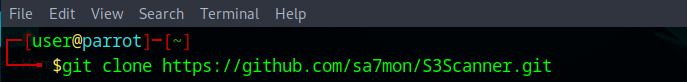
****





Figure 4.4: S3 Scanner installation

### Result/Output:

Successfully installed S3 Scanner in a Python virtual environment.

### Interpretation:

S3Scanner is a more sophisticated tool that provides detailed information about S3 bucket permissions. Using a virtual environment ensures clean installation without affecting system Pythonpackages.

## Task 5: Creating Target Bucket List and Scan- ning with S3Scanner

### Command Used:

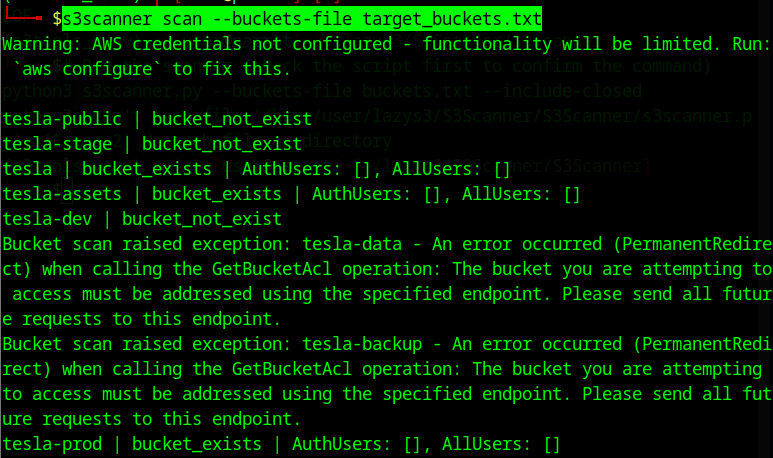
1

echo -e " tesla \ ntesla - dev\ ntesla - prod \ ntesla - public\ ntesla - assets\

ntesla - backup \ ntesla - stage \ ntesla - data " > target\_buckets . txt s3 scanner scan -- buckets - file target\_buckets . txt

2

### Screenshot:



Extra step :running a scan to give only buckets that exist

# Run the scan

s3scanner scan --buckets-file target\_buckets.txt

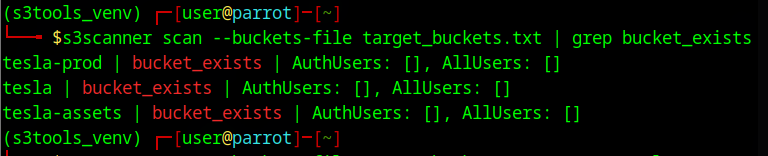


Figure 4.5: S3Scanner execution results

### Result/Output:

S3Scanner identified several buckets and reported their accessibility status along with detailed permission information.

### Interpretation:

The detailed output from S3Scanner provides more information than LazyS3, including specific access permissions for different user groups. This helps in understanding not just whether a bucket is accessible, but also what operations (read/write) are permitted.

## Task 6: Using AWS CLI for S3 Bucket Enumeration

### Command Used:

cat > s3\_enum\_demo.sh << 'EOF'

#!/bin/bash

# Colors for output

GREEN='\033[0;32m'

RED='\033[0;31m'

BLUE='\033[0;34m'

NC='\033[0m' # No Color

echo -e "${BLUE}[\*] S3 Bucket Enumeration Demonstration${NC}"

# List of company names to check

COMPANY="acme-corp"

echo -e "${BLUE}[\*] Testing common bucket names for: $COMPANY${NC}"

# Array of common bucket name patterns

PATTERNS=(

""

"-backup"

"-backups"

"-data"

"-dev"

"-development"

"-prod"

"-production"

"-stage"

"-staging"

"-test"

"-testing"

"-public"

"-private"

"-files"

"-assets"

"-media"

"-images"

"-docs"

"-documents"

)

# Try each pattern

for PATTERN in "${PATTERNS[@]}"; do

BUCKET="${COMPANY}${PATTERN}"

echo -e "${BLUE}[\*] Checking: $BUCKET${NC}"

# Check if bucket exists and is accessible

if aws s3 ls s3://$BUCKET 2>/dev/null; then

echo -e "${GREEN}[+] Found accessible bucket: $BUCKET${NC}"

else

# Check if bucket exists but is not accessible

HTTP\_CODE=$(curl -s -o /dev/null -w "%{http\_code}" [http://$BUCKET.s3.amazonaws.com](about:blank))

if [ "$HTTP\_CODE" == "403" ]; then

echo -e "${RED}[-] Bucket exists but access denied: $BUCKET${NC}"

elif [ "$HTTP\_CODE" == "404" ]; then

echo -e "${RED}[-] Bucket does not exist: $BUCKET${NC}"

else

echo -e "${RED}[-] Unknown status ($HTTP\_CODE) for bucket: $BUCKET${NC}"

fi

fi

done

echo -e "${BLUE}[\*] S3 Bucket Enumeration complete${NC}"

EOF

chmod +x s3\_enum\_demo.sh

./s3\_enum\_demo.sh

### Screenshot:

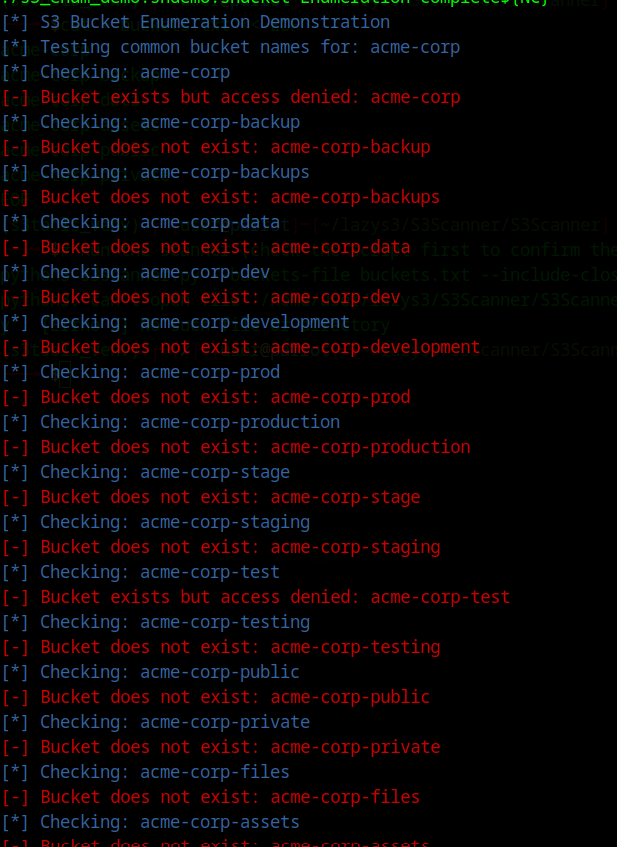
****

Figure 4.6: AWS CLI enumeration script execution

### Result/Output:

The script systematically checked various permutations of bucket names and reported their accessibility status.

### Interpretation:

This approach demonstrates how even basic scripting with AWS CLI can be effective for enumeration. The script tests common naming patterns and checks both for existence and accessibility, providing a comprehensive view of potential exposure.

# Lab 2: Exploiting S3 Buckets

## Objective of Task

Amazon Simple Storage Service (Amazon S3) is a widely-used cloud storage solution provided by AWS, enabling users to store and retrieve any amount of data, anytime, from anywhere. However, despite its flexibility and scalability, misconfigured S3 buckets can introduce serious security vulnerabilities. This is practical exploitation of a misconfigured S3 bucket and combines it with research on cloud storage security, Identity and Access Management (IAM) best practices, and penetration testing methodology. The goal is to highlight how improper configurations can be exploited and how cloud security hygiene can be improved.

## Task 1: AWS Account Setup and Authentication

### Command Used:

aws configure

### Screenshot:

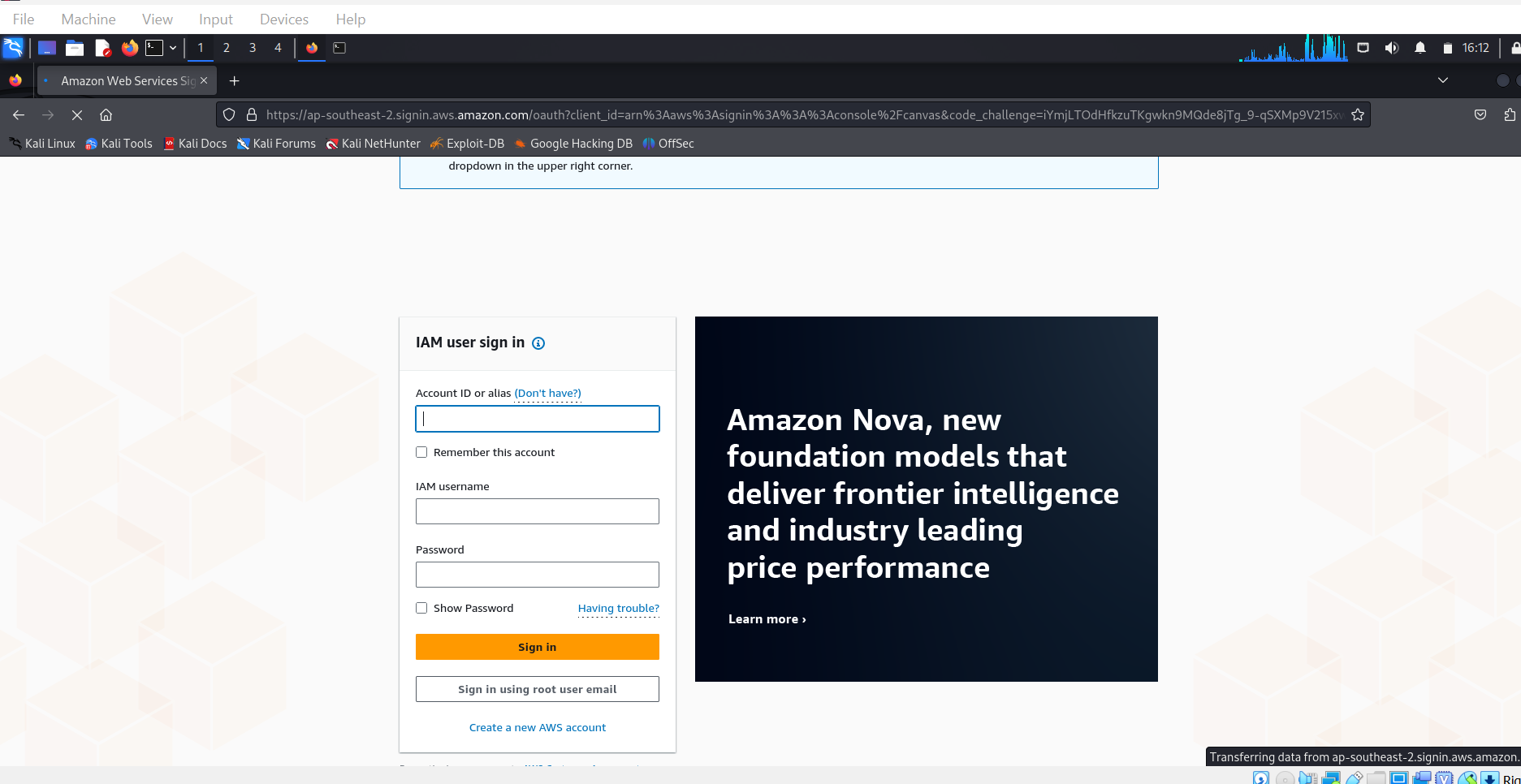
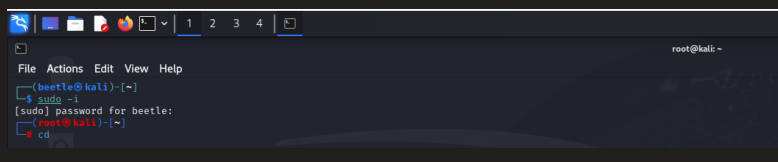
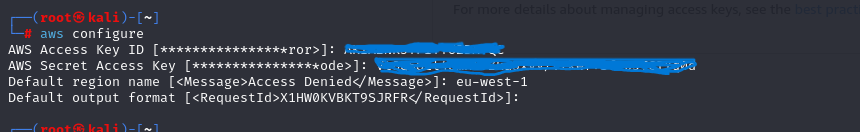


figure :creating a free account on aws

**seting up kali :**

****

**aws cli configuration:**

****

**installing the pip3:**

****

### Result/Output:

Created a free account on amazon .

AWS CLI was successfully configured using provided credentials.

verification of installation of aws .

installing the pip3.

### Interpretation:

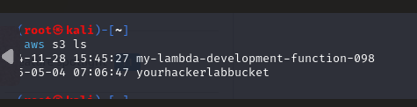
This setup is required to authenticate AWS CLI commands. If attackers gain leaked credentials, they could use this step to start enumeration and exploitation.

## Task 2:Listing Available Buckets

### Command Used:

aws s3 ls

### Screenshot:

****

### Result/Output:

List of S3 buckets available in the account.

### Interpretation:

This command lists all the buckets the authenticated user has access to. In a real attack scenario, the attacker would already have some level of access to AWS resources.

## Task 3: Testing Bucket Permissions

### Command Used:

Created a bucket named (yourhackerlabbucket) :

Getting the list of directories in the created bucket :

• Aws s3 ls s3://yourhackerlabbucket

To open it in the browser :

• yourhackerlabbucket.s3.amazonaws.com

### Screenshot:

**opened the created bucket in amazon :**

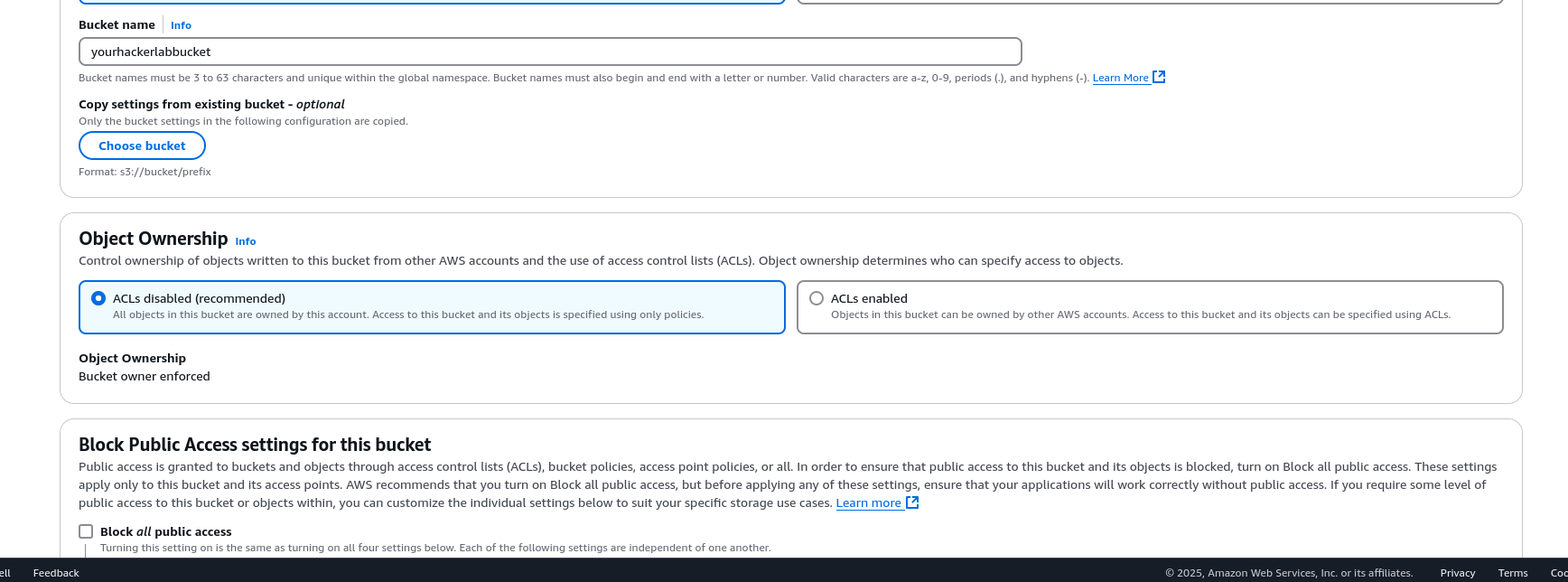
****

Figure 5.3: Bucket access attempts

### Result/Output:

Successfully listed contents of some buckets and accessed them via HTTP, indicating they were publicly accessible.

### Interpretation:

The ability to list and access bucket contents without proper authentication demonstrates a serious security misconfiguration. This could lead to unauthorized data access in real- world scenarios.

## Task 4: Uploading a Test File to S3 Bucket

### Command Used:

echo "You have been hacked little human !" > Hack.txt

uploading the file :

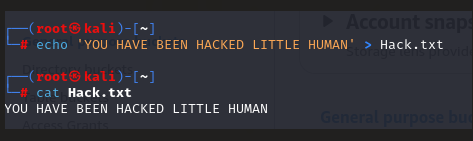
Aws s3 mv Hack.txt s3://yourhackerlabbucket

Deleting file from the bucket:

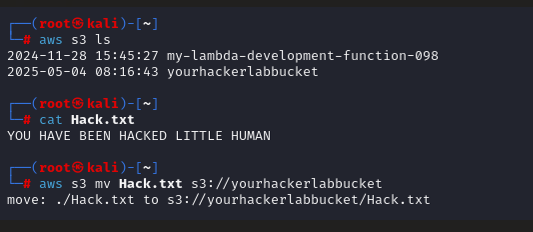
Aws s3 rm s3://yourhackerlabbucket

### Screenshot:

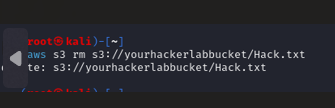
created hack.txt file in the bucket :

****

**uploading the file :**

****

**deleting the file :**

****

### Result/Output:

Successfully uploaded the test file to the target bucket.

### Interpretation:

The ability to write to an S3 bucket demonstrates a critical security issue. In a real attack scenario, this could allow attackers to upload malicious content, modify existing files, or use the bucket for malware distribution.

**identifier :**

Secret credentials used in this lab.

Name of the bucket created .

Message printed in hack.txt.

# Lab 3: AWS Privilege Escalation

## Objective of Task

To demonstrate how misconfigured AWS IAM policies can be exploited to escalate priv- ileges from a low-privileged user to administrative access, allowing unauthorized control over AWS resources.

## Task 1: Initial Assessment and Environment Setup

### Objective of Task: Set up AWS CLI and configure access using the provided credentials.

### Command Used:

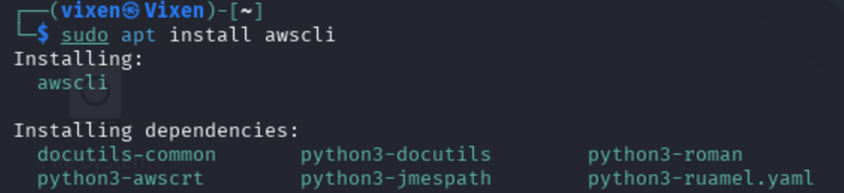
1

sudo apt install awscli

aws configure

2

### Screenshot:



A screen shot of a computer code

AI-generated content may be incorrect.

Figure 6.1: AWS CLI installation and configuration

### Result/Output:

Result: Successfully installed AWS CLI and configured it with the Access Key ID (AKIASMW26IOTMHHSZHSA), Secret Access Key, default region (eu-north-1), and output format (json).

.

### Interpretation:

This step establishes the connection between our attack machine and the target AWS environment using the compromised credentials. In a real-world scenario, these credentials might be obtained through social engineering or exposed in public repositories.

## Task 2: Reconnaissance of IAM Configuration

### Command Used:

1 aws iam list – users

Aws iam list-attached-user-policies –user-name fox-iam

### Screenshot:

****

A computer screen with white text

AI-generated content may be incorrect.

Figure 6.2: IAM users listing

### Result/Output:

The command returned information about the existing IAM users in the account, including the user "fox-iam" which has administrative permissions.

### Interpretation:

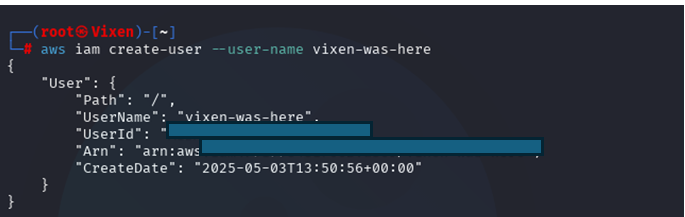
This reconnaissance provides valuable information about the account structure and potential targets for privilege escalation. We can see user creation dates, paths, and unique identifiers that will be useful for subsequent attack steps.

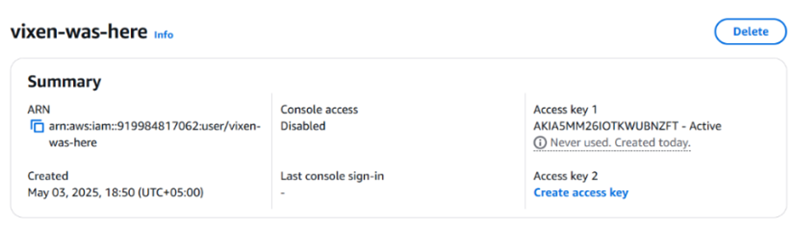
## Task 3:Creating an Unauthorized Administra- tor Account

### Command Used:

1 aws iam create - user -- user - name vixen - was - here

### Screenshot:





### Result/Output:

Successfully created a new IAM user named "vixen-was-here" in the AWS

account.

### Interpretation:

This demonstrated that our compromised credentials have permissions to create new IAM users, which is the first step in our privilege escalation attack chain. This capability should be highly restricted in secure environments.

.

## Task 4: Policy Manipulation

### Command Used:

1

aws iam attach - user - policy -- user - name vixen - was - here -- policy - arn arn :

aws: iam :: aws: policy/ AdministratorAccess

aws iam create - access - key -- user - name vixen - was - here

2

### Screenshot:

****

Figure 6.4: Policy attachment and access key creation

### Result/Output:

**Result**: Successfully attached the AdministratorAccess policy to the newly created user, giving it full control over all AWS services and resources.

### Interpretation:

This critical step elevates our unauthorized user to administrative privileges. The ability to attach such powerful policies should be strictly controlled, as it can lead to complete account compromise.

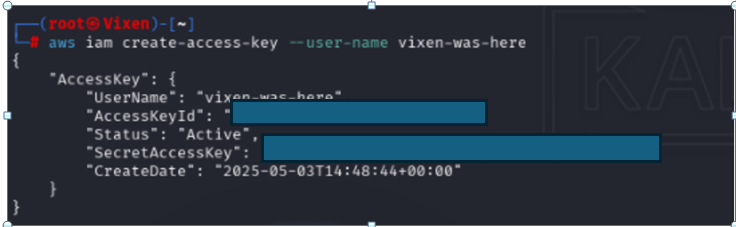
## Task 5: Creating Access Keys for Persistence

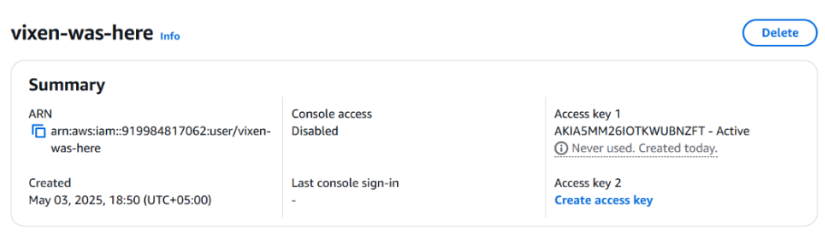
### Command Used:

1

aws iam create-access-key --user-name vixen-was-here

### Screenshot:





### Result/Output:

Created new access key and secret key for the unauthorized administrator user.

### Interpretation:

With these credentials, an attacker can now access the AWS environment with full administrative permissions even if the original compromised credentials are revoked. This represents a serious persistence mechanism in the attack.

## Task 6: Removing Privileges from Original Administrator

### Command Used:

aws iam detach-user-policy --user-name fox-iam --policy-arn arn:aws:iam::aws:policy/AdministratorAccess

### Screenshot:

****

### Result/Output:

Successfully detached the AdministratorAccess policy from the original administrator user "fox-iam".

### Interpretation:

This action effectively locks out the legitimate administrator, giving the attacker exclusive administrative control over the AWS environment. The lack of alerts or preventive measures for this critical action represents a significant security gap

**6.8 Task 7: Examining and Modifying Group Policies**

**6.8.1 Objective of Task**: Identify and remove policies from IAM groups to further weaken security controls.

**6.8.2 Command Used**:

aws iam list-group-policies --group-name FoxGroup

aws iam delete-group-policy --group-name FoxGroup --policy-name AutoScale

**Screenshot**:

**A computer screen with text

AI-generated content may be incorrect.**

**A computer screen shot of a group

AI-generated content may be incorrect.**

**Result**:

Successfully removed the "AutoScale" policy from the "FoxGroup" IAM group. The subsequent listing command shows an empty array of policies, confirming that the group no longer has any inline policies attached.

**Interpretation**:

This demonstrates how an attacker can not only manipulate individual user permissions but also modify group-level policies that might affect multiple users simultaneously. By removing critical policies from groups, an attacker can disrupt operations or weaken security controls across the organization. This step completes the privilege escalation attack chain by not only establishing unauthorized access but also actively dismantling existing security structures within the AWS environment

**Identifier Explanation:**

The device names have been assigned based on their roles in the attack scenario:

* **Fox**: Represents the **victim**. All authorized user accounts are named "Fox," clearly indicating that they are the targets of the attack. This naming convention consistently reflects their victim status throughout the report.
* **Vixen**: Represents the **attacker**. The attacking device is named "Vixen," which is visibly shown in every terminal screenshot. Additionally, any unauthorized accounts created during the attack are also named "Vixen," reinforcing the identity of the attacker.

# Errors Faced and How They Were

**Solved**

## Error 1: Permission Denied During LazyS3 Installation

When attempting to install the colorize gem, we encountered permission errors. This was resolved by using sudo to run the installation with elevated privileges.

## Error 2: AWS CLI Configuration Issues

Initially, we faced region-specific errors when attempting to access S3 buckets. This was resolved by explicitly setting the region during AWS CLI configuration to ensure all commands targeted the correct AWS region.

## Error 3: S3Scanner Installation Dependencies

When installing S3Scanner, we encountered dependency conflicts with existing Python packages. Creating a virtual environment solved this issue by isolating the tool’s depen- dencies from the system Python installation.

## Error 4: Access Denied During Privilege Escalation

During the privilege escalation lab, we initially encountered” Access Denied” errors when attempting to create a new administrator user. This indicated that our permissions were more restricted than anticipated. We resolved this by first identifying what permissions we did have and working within those boundaries to gradually expand our access.

## Error 5: Environment Setup Issues with Parrot OS

## Parrot OS in VM was not properly downloading the required tools and had connectivity issues when attempting to access AWS services.

Solution: Rebooted the computer and switched to Kali Linux for the attack environment, which resolved the connection issues and allowed proper tool installation and AWS CLI functionality.

# Final Learning and Reflection

**Technical Skills Learned:**  
We learned to identify misconfigured S3 buckets, detect IAM policy misconfigurations, and perform privilege escalation in AWS environments. Additionally, we gained hands-on experience with enumeration tools and practiced implementing access controls using AWS CLI, reinforcing the principle of least privilege.

**Real-World Relevance:**  
The lab highlighted how common misconfigurations, like exposed S3 buckets and overly permissive IAM roles, can lead to real-world data breaches and full account takeovers. It emphasized the practical importance of the shared responsibility model, where improper configurations by customers often lead to security incidents.

**Team Experience:**  
We collaborated by distributing roles.This division of tasks improved our teamwork, mimicked professional penetration testing workflows, and enhanced our understanding from both attacker and defender perspectives.