Once we have found some Amazon S3 Buckets, we can then start searching through them for valuable information using AWS's CLI tool.

Anonymous Access

Many s3 buckets can be accessed using anonymous rights, which enables an attacker to more stealthy interact with resources within s3 buckets.

List Files

We can try to list files within the "cdn2.lizardblue.com" s3 bucket we found previously using anonymous privileges by leveraging the "--no-sign-request" switch with the AWS CLI:

```
root@ip-10-0-1-215:/shared# aws s3 --no-sign-request ls s3://cdn2.lizardblue.com
2018-07-03 23:22:22 99 .boto
2018-06-01 00:55:02 382985 gecko.jpg
2018-06-05 23:59:33 11 index.html
2018-06-01 17:08:08 75 lizardssns.txt
2018-06-01 16:58:52 22838 pricing.pdf
2018-06-11 21:19:15 1194298 salamander.jpg
```

File Download

We can try to download files from the "cdn2.lizardblue.com" s3 bucket we found previously:

```
root@ip-10-0-1-215:/shared# mkdir -p /shared/s3 && cd /shared/s3

root@ip-10-0-1-215:/shared/s3# aws s3 --no-sign-request cp s3://cdn2.lizardblue.com/lizardssns.txt .
download: s3://cdn2.lizardblue.com/lizardssns.txt to ./lizardssns.txt

root@ip-10-0-1-215:/shared/s3# cat lizardssns.txt
Tammy - 851024955
Eric - 518428535
John - ...redacted...
Salamantha - 850195385
```

We can also try to download all the files using the sync command:

```
root@ip-10-0-1-215:/shared# mkdir -p /shared/s3/sync && cd /shared/s3/sync

root@ip-10-0-1-215:/shared/s3/sync# aws s3 --no-sign-request sync s3://cdn2.lizardblue.com .
download failed: s3://cdn2.lizardblue.com/index.html to ./index.html An error occurred (AccessDenied) when calling the GetObject operation: Access Denied
download: s3://cdn2.lizardblue.com/.boto to ./.boto
download: s3://cdn2.lizardblue.com/pricing.pdf to ./pricing.pdf
download: s3://cdn2.lizardblue.com/gecko.jpg to ./gecko.jpg
download: s3://cdn2.lizardblue.com/salamander.jpg to ./salamander.jpg

root@ip-10-0-1-215:/shared/s3/sync# ls -alF
...
-rw-r--r-- 1 root root 99 Jul 3 23:22 .boto
-rw-r--r-- 1 root root 382985 Jun 1 00:55 gecko.jpg
-rw-r--r-- 1 root root 75 Jun 1 17:08 lizardssns.txt
-rw-r--r-- 1 root root 22838 Jun 1 16:58 pricing.pdf
-rw-r--r-- 1 root root 1194298 Jun 11 21:19 salamander.jpg

root@ip-10-0-1-215:/shared/s3/sync# cat .boto
[Credentials]
aws_access_key_id = the_access_key_here
aws_secret_access_key = the_secret_key_here
```

File Upload

We can try to upload files to the "cdn2.lizardblue.com" s3 bucket we found previously:

```
root@ip-10-0-1-215:/shared/s3/sync# echo "hi" > test.txt

root@ip-10-0-1-215:/shared/s3/sync# aws s3 cp --no-sign-request test.txt s3://cdn2.lizardblue.com
upload failed: ./test.txt to s3://cdn2.lizardblue.com/test.txt An error occurred (AccessDenied) when calling the PutObject operation: Access Denied
```

But we can see that the anonymous user does not have access to upload files to this bucket.

"Authenticated Users" Access

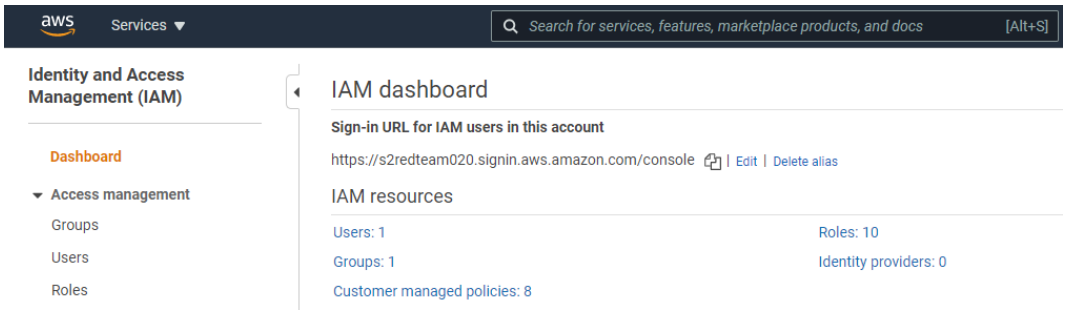
Sometimes s3 buckets have been mistakenly configured to enable any user from any AWS account to authenticate to resources within the s3 bucket. In this manner resources seem secure as the anonymous attempts will fail but if an attacker has an AWS account of there own, they may be able to leverage their own account to access the resources within another account.

Grants to "Authenticated Users" should only be done to charge storage consumers for access to public objects in S3 buckets.

Creating Access Keys

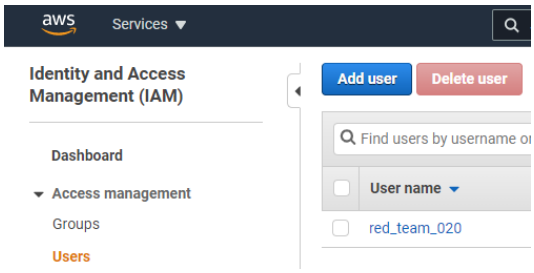
When leveraging the AWS CLI to explore S3 bucket, it's best to create an access keys within our own AWS account, encase bucket permissions are mis-configured to let other AWS accounts access sensitive files.

Navigate to: https://console.aws.amazon.com/iam/home

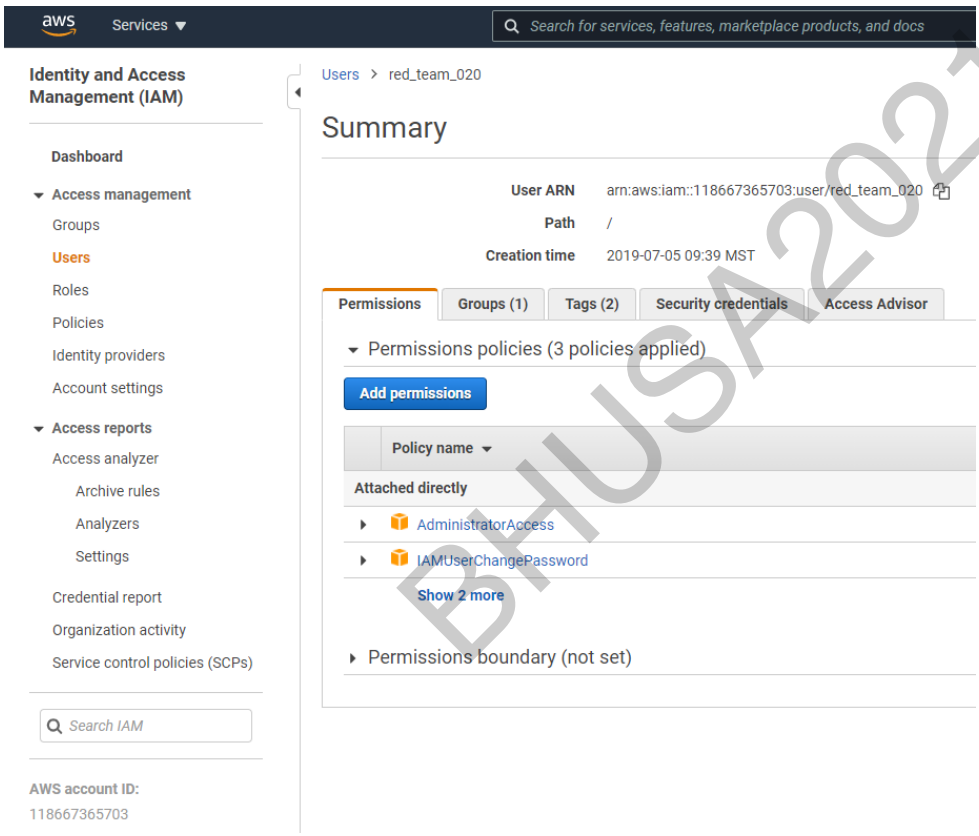


We are going to generate some access keys for our IAM user (e.g. red_team_020 in this example)

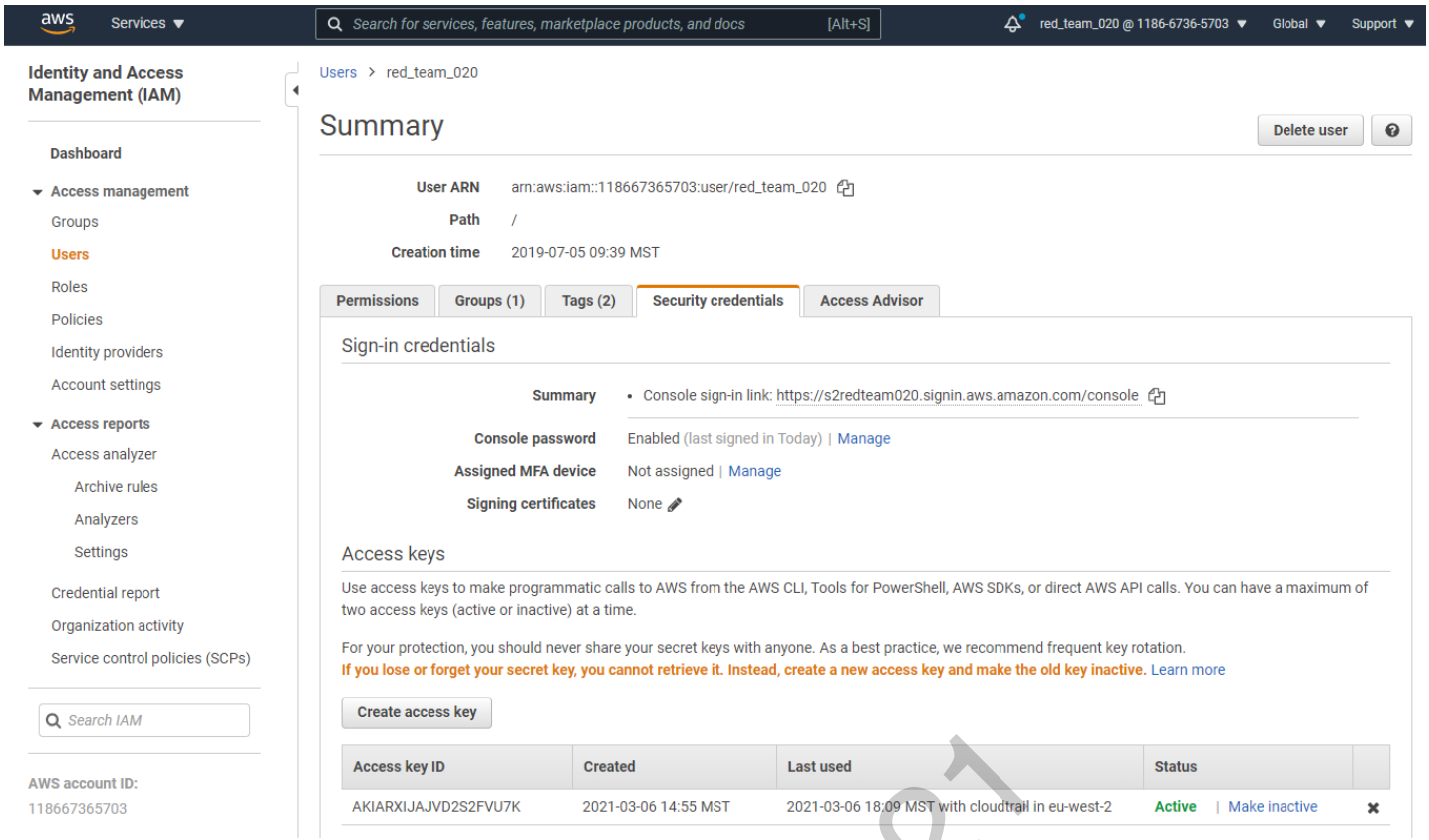
Click on the "Users" link



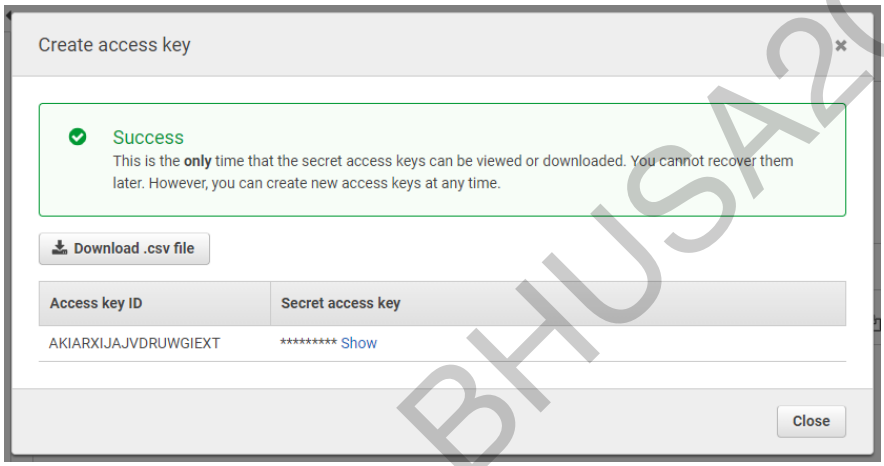
Click on your student username e.g. red_team_### (replace ### with your student number)



Click on the "Security credentials" tab



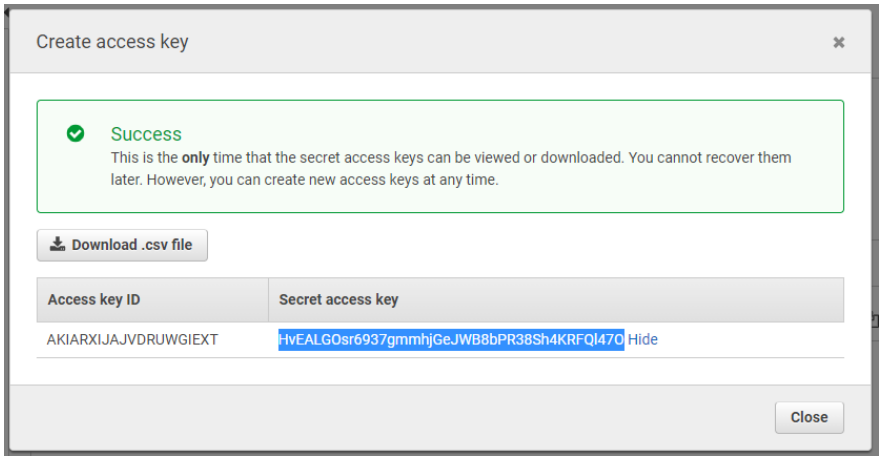
Click on the "Create access key" button



Click on the "Download .csv" file to retain a copy of the secret access key.

Note: Once you close this box, you will not be able to get the secret access key again via AWS.

Click on the "Show" link to see the secret access key



After your access key has been created, download it, and store it somewhere safe.

We will need to use it with the aws cli to interact with the aws control plane APIs.

Configuring Access Keys

Now that we have an access key, we next need to configure the aws cli to use them. When it asks for your Access Key ID and your Access Key they are inside the Access Key file you downloaded previously.

```
root@ip-10-0-1-215:/shared# cd /shared

root@ip-10-0-1-215:/shared# aws configure
AWS Access Key ID [None]: AKIAzzzEXAMPLEzzzEXT
AWS Secret Access Key [None]: HvEzzzEXAMPLEzzzEXAMPLEzzzEXAMPLEzzzz47O
Default region name [None]:
Default output format [None]:

root@ip-10-0-1-215:/shared# cat ~/.aws/config
[default]

root@ip-10-0-1-215:/shared# cat ~/.aws/credentials

[default]
aws_access_key_id = AKIAzzzEXAMPLEzzzEXT
aws_secret_access_key = HvEzzzEXAMPLEzzzEXAMPLEzzzEXAMPLEzzzz47O
```

Regions with the CLI

We can try to list files within the "soundslike" s3 bucket we found previously using both the anonymous user privileges, as well as your AWS account:

```
root@ip-10-0-1-215:/shared# aws s3 --no-sign-request ls s3://soundslike

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

root@ip-10-0-1-215:/shared# aws s3 ls s3://soundslike
2018-07-02 05:30:12 771534 ...redacted...002.png
2018-08-01 13:57:54 3 test.txt
```

Note: Previously, AWS CLI would complain that we need to specify the region before running the command, depending on the version of the cli being used, and the AWS API being called, but now days this is automatically taken care of for users within this AWS CLI experience.

Bucket Region

We can in most cases easily discover the region a bucket is located within using the nslookup tool:

```
root@ip-10-0-1-215:/shared# nslookup soundslike.s3.amazonaws.com
Server: 10.0.0.2
Address: 10.0.0.2#53

Non-authoritative answer:
soundslike.s3.amazonaws.com canonical name = s3-w.us-east-2.amazonaws.com.
Name: s3-w.us-east-2.amazonaws.com
Address: 52.219.88.220
```

You can see in this output that the bucket is located within the "us-east-2" region of AWS.

Listing Objects within Bucket

Let's try the command again this time using the "--region" option:

```
root@ip-10-0-1-215:/shared# aws s3 ls --region=us-east-2 s3://soundslike
2018-07-02 05:30:12 771534 ...redacted...002.png
2018-08-01 13:57:54 3 test.txt
```

We can set "us-east-2" as the default region by running the "aws configure" command again.

```
root@ip-10-0-1-215:/shared# aws configure
AWS Access Key ID [****************IEXT]:
AWS Secret Access Key [****************l47O]:
Default region name [None]: us-east-2
Default output format [None]:

root@ip-10-0-1-215:/shared# aws s3 ls s3://soundslike
2018-07-02 05:30:12 771534 ...redacted...002.png
2018-08-01 13:57:54 3 test.txt
```

Known Techniques for Identifying S3 Buckets & S3 Bucket Names

Some of the known techniques for identifying S3 buckets on engagements are:

- HTTP response contains a "Server" header which contains the string "AmazonS3"
- Access a resource that does not exist (e.g. http://cdn2.lizardblue.com/i_just_made_this_up) and the HTTP response may contain strings:
  - "S3-404",
  - "Access Denied" and/or,
  - "NoSuchKey"
- Access the root of a domain (http://cdn2.lizardblue.com/), if listing is enabled it will contain the string "ListBucketResult".
- Nslookup the domain ($ nslookup cdn2.lizardblue.com), if it was CNAME'd to the bucket we will see a result like <bucket name>.s3.amazonaws.com (e.g. cdn2.lizardblue.com.s3.amazonaws.com)
- Frequently, the bucket name will be the same as the FQDN, so we take the FQDN (cdn2.lizardblue.com) and append .s3.amazonaws.com to it, to see if it resolves (e.g. cdn2.lizardblue.com.s3.amazonaws.com)
- Append a %C0 to the end of a valid resource (http://cdn2.lizardblue.com/gecko.jpg%C0), sometimes the bucket name will be return within the resulting error message.
- Append a ?torrent to the end of a valid resource (http://cdn2.lizardblue.com/gecko.jpg?torrent), download the torrent file & parse it (https://github.com/7sDream/torrent_parser), it will contain the bucket name.
- Brute-force the bucket names (e.g. gobuster), preferably prepending/appending permutations (e.g. "prod-", "stage-", etc...).
- Spider the website(s), frequently some of the linked resources are for directly accessible S3 bucket objects ¯_(ツ)_/¯
  - Also check out "AWS Security Checks" for Burp Suite Pro: https://portswigger.net/bappstore/f078b9254eab40dc8c562177de3d3b2d

Exercise

The Plan:

- Collect sensitive files from the S3 Buckets as anonymous user

- Create our own AWS secret

-- Configure the AWS CLI to use the secret

--- Collect sensitive files from the S3 Buckets as authenticated user

Practical Example:

GrayHatWarFare has identified & indexed almost 50,000 S3 buckets with Unauthenticated ListObject permissions, produce a shodan-like inventory of buckets. Buckets may have been identified via spidering/google dorking or a dirby-type bruteforce enumeration. It's claimed that they refresh their database every 2 weeks. Likely to continue to provide opportunities to identify data exposure for years to come.

Website: https://buckets.grayhatwarfare.com/

References:

Check out the following references for more information:

- AWS CLI Reference - https://docs.aws.amazon.com/cli/latest/reference/
- 3 Things to Know About AWS S3 Security to Stay Out of the Headlines - https://blog.rackspace.com/3-things-aws-s3-security-stay-headlines
- GrayHatWarFare Buckets - https://buckets.grayhatwarfare.com/
- Unveiling Amazon S3 bucket names - https://labs.detectify.com/2017/07/13/a-deep-dive-into-aws-s3-access-controls-taking-full-control-over-your-assets/
- A deep dive into AWS S3 access controls – taking full control over your assets - https://medium.com/@localh0t/unveiling-amazon-s3-bucket-names-e1420ceaf4fa
- %C0 Error on S3 - https://twitter.com/0xmdv/status/1065581916437585920