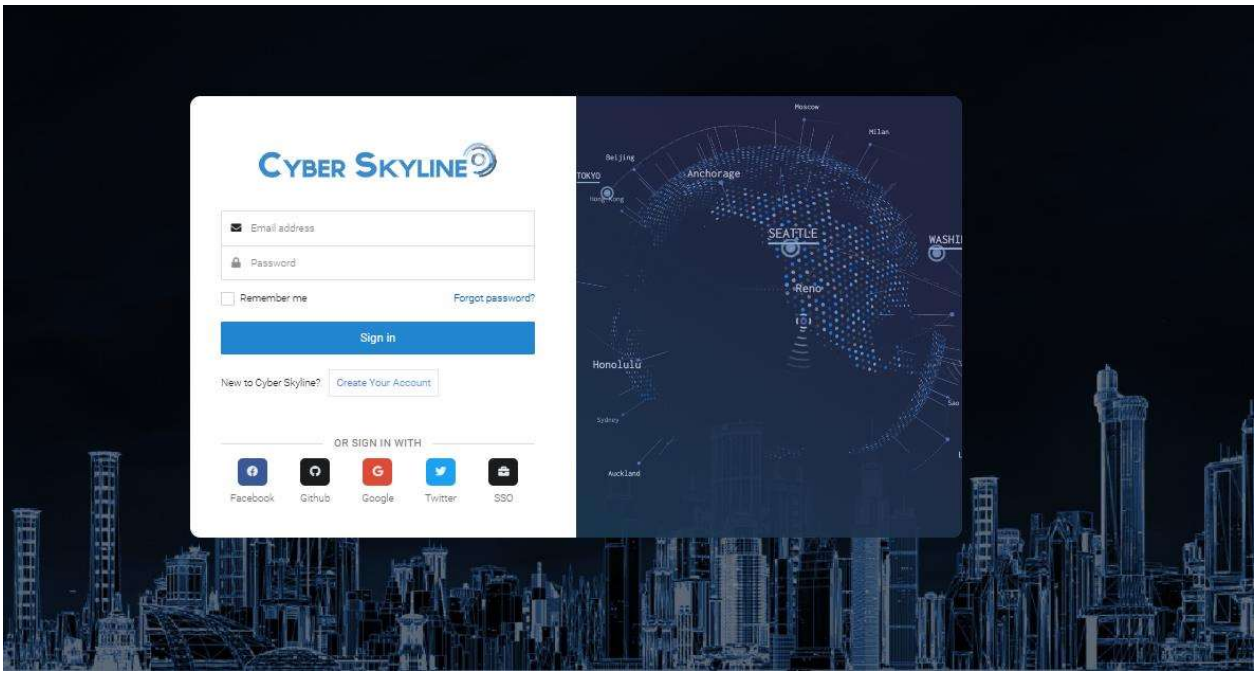
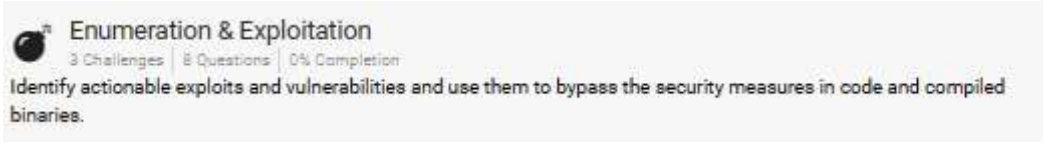


National Cyber League Spring 2024

ENUMERATION AND EXPLOITATION

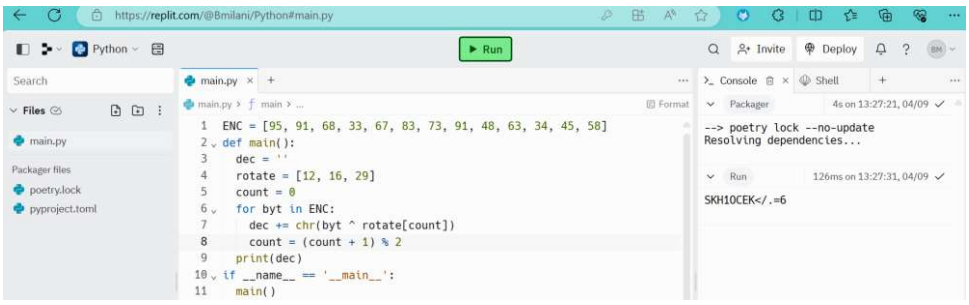


Challenge #1:

This challenges provides us with a code written in Python. When I run the code in a compiler (replit.com) we get a garbled output that is clearly not the hidden flag.

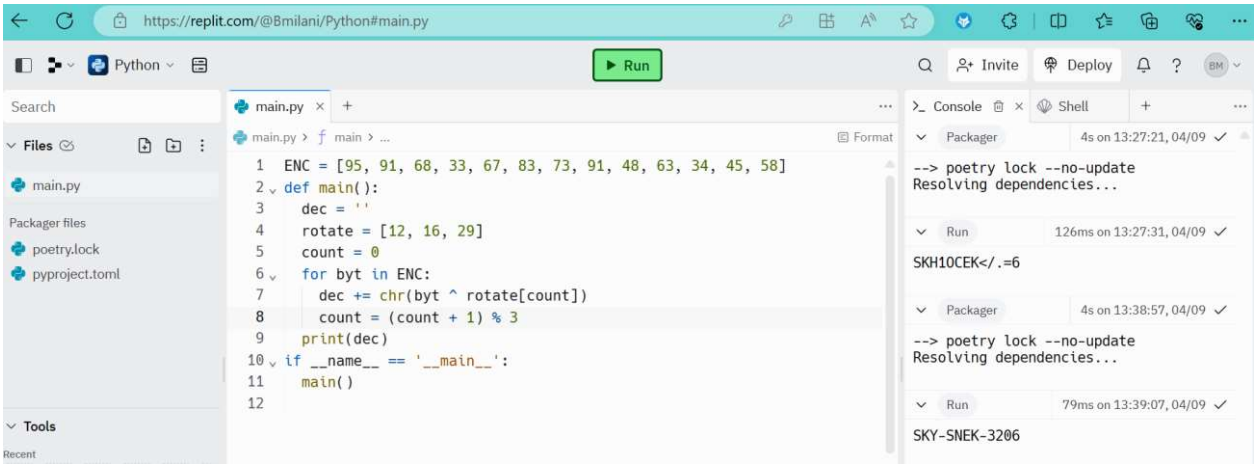
Python Code:

```
ENC = [95, 91, 68, 33, 67, 83, 73, 91, 48, 63, 34, 45, 58]
 def main():
    dec = ''
    rotate = [12, 16, 29]
    count = 0
    for byt in ENC:
        dec += chr(byt ^ rotate[count])
        count = (count + 1) % 2
    print(dec)
if __name__ == '__main__':
    main()
```



Answer:

The code attempts to decrypt the hidden flag encoded in a list (ENC) using an XOR operation with a rotating key. There's an error in the code. The 'rotate' list should have 3 elements because the modulo operation is '(count + 1) % 3 and not '(count + 1) % 2'.



Correct Answer: SKY-SNEK-3206

Challenge #2:

You will need this file(crosslock.dll) to solve this challenge:

https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/EQu8XkycoG1FgZ5U7-yYGeoBsbEwnQcM5YDMDTcIexTv9A?e=jlB6kA

Questions 1: What runtime does this program use?

Answer:

First examine the file after downloading it by running a couple Linux commands:

└─$ file crosslock.dll

crosslock.dll: PE32 executable (console) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections

This indicates that it is a Windows driver file. But even the file name itself tells us that it is a typical windows driver file with the extension (.dll).

Based on this result we can conclude that its a windows .NET runtime

Correct answer: .NET

PASSWORD CRACKING:

Challenge #1:

Question 1: What is the MD5 hash of the password thelastofus3091

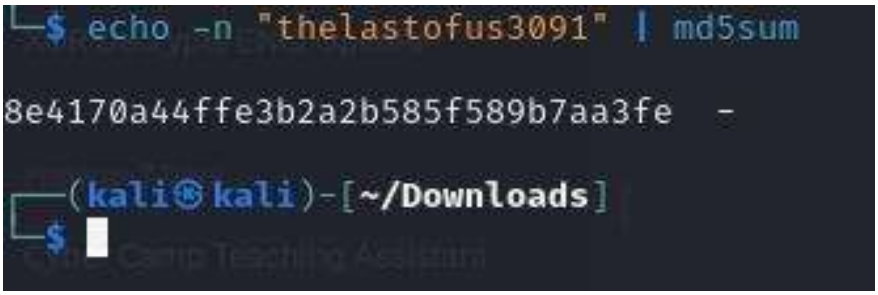
Answer:

to generate the MD5 hash of the string "thelastofus3091" using the command line in Kali Linux, you can use the md5sum command. Here's how you can do it:

echo -n "thelastofus3091" | md5sum

This command will generate the MD5 hash of the string "thelastofus3091" and display it in the terminal.

The -n option for the echo command is used to omit the trailing newline character from the string, ensuring that only the string itself is hashed without any additional characters.



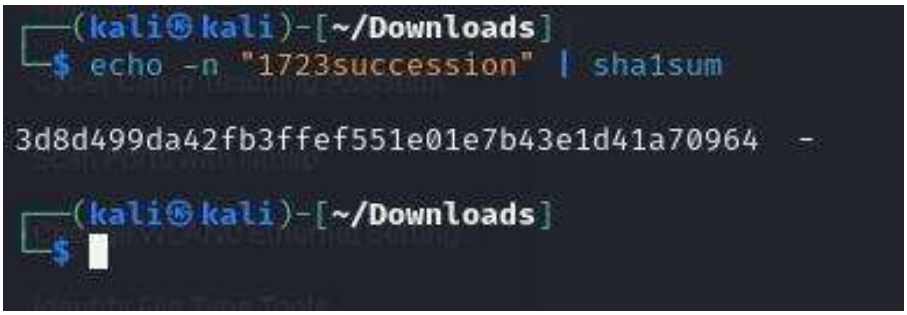
Question 2 : **What is the SHA1 hash of the password 1723succession**

Answer:

*To generate the SHA-1 hash of the password **(1723succession)** using the command line, you can use the echo command in combination with sha1sum. Here's the command:*

echo -n "1723succession" | sha1sum

*This command will output the SHA-1 hash of the password **(1723succession)**.*

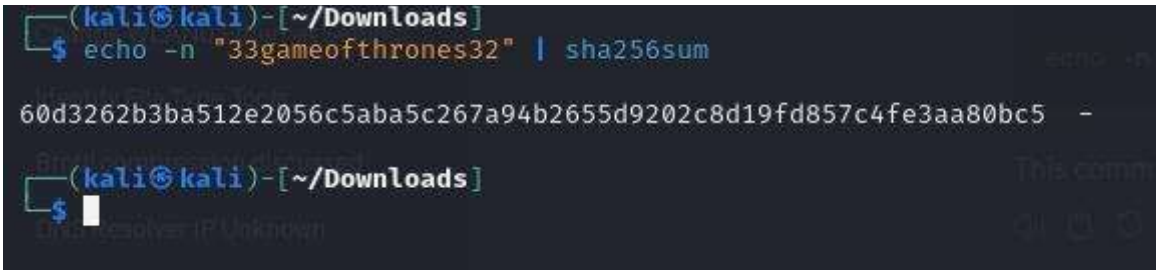


Question 3 : What is the SHA256 hash of the password (33gameofthrones32)
Answer:

*To generate the SHA-256 hash of the password **(33gameofthrones32)** using the command line, you can use the echo command in combination with sha256sum. Here's the command:*

echo -n "33gameofthrones32" | sha256sum

*This command will output the SHA-256 hash of the password **(33gameofthrones32)**.*



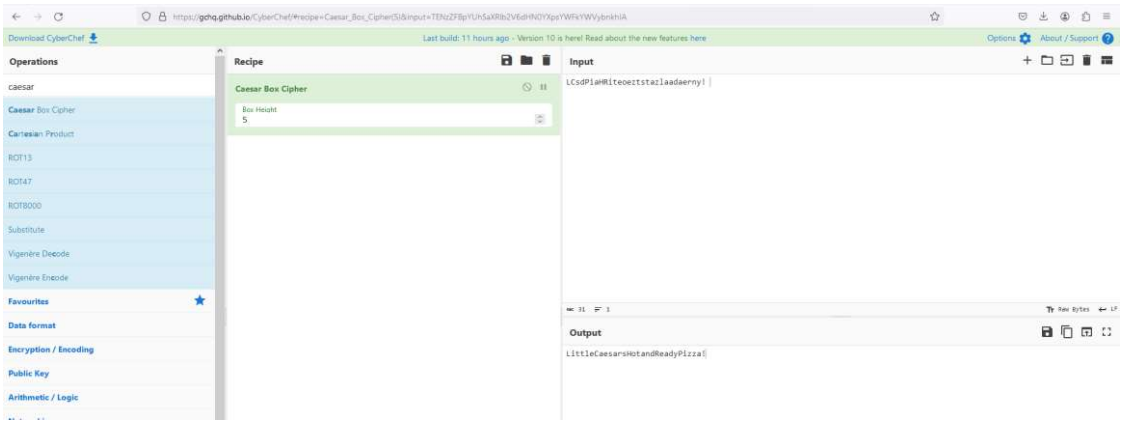
CRYPTOGRAPHY

Challenge #1: Boxed in

We intercepted an envelope from a courier who was making a delivery to a known Liber8tion agent. Inside the envelope was what appears to be an encrypted message that we believe contains the location for one of Liber8tion's upcoming meetings. Written on the back of the envelope was, "order 6 caesar salads". See if you can decrypt this message so we can identify their next meeting location.

Answer:

There seems to be a hint in the challenge message ("order 6 caesar salads".). From this we can take a vaild guess that it might be encrypted with caesar's cipher). Let's use **(https://gchq.github.io/CyberChef/)**



After adding the cipher to the recipe and placing the encrypted message in the input field I intuitively change the Box Height in the cipher to 6. But, in the process I noticed that when i put 5 I get a plaintext message.

Correct Ans: LittleCaesarsHotandReadyPizza!

Challenge 2: Validation

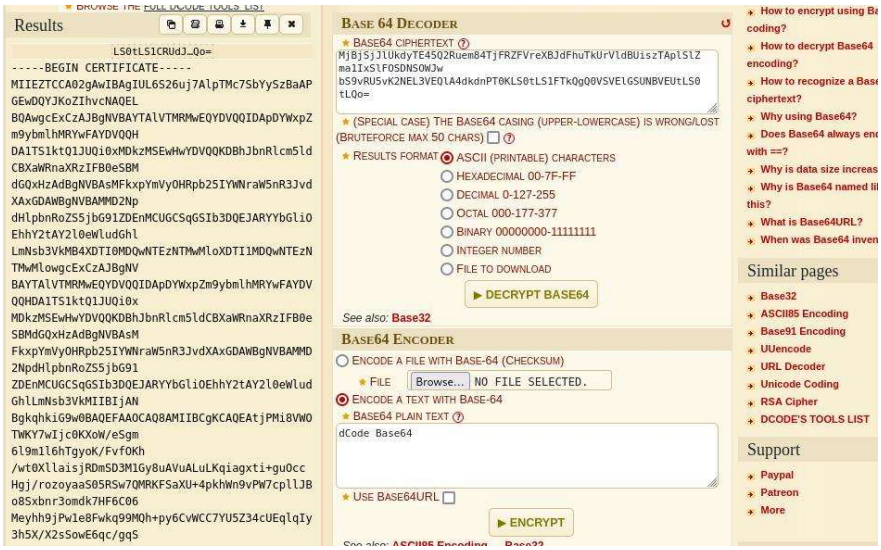
You will need this to solve the challenge (encoded.pem): https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/Eb3Zg1e7VjZDhQD31pAOP_EBRbSbY0dH0IukYYyTi6v7oA?e=4xLqoX

a. **What is the serial number of the certificate (in hex)?**

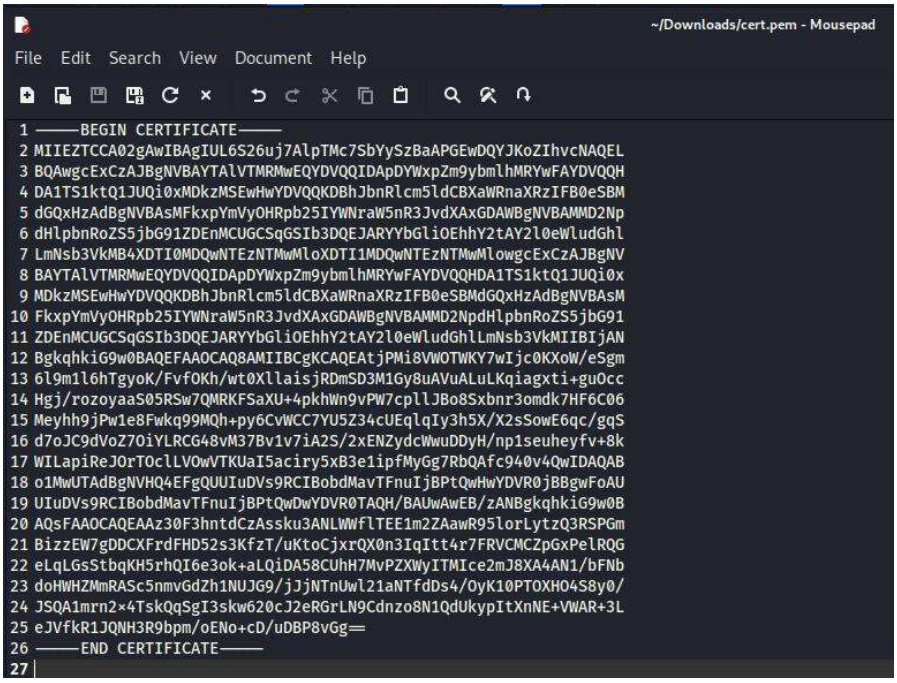
Answer:

Lets run some command line tools to get an idea of what we have:

Looks like it is Base64 encoded. run it through https://www.dcode.fr/cipher-identifier



We then need to make a new file from our decryption which is an x509 certificate and we will name the file 'cert.pem'. Simply copy the output from Base64 decrypter on dcode to any text editor (in my case its Mousepad), and save it.



*when you run the command ' **openssl x509 -in cert.pem -noout –serial** ', it reads the X.509 certificate from the file "cert.pem", extracts the serial number from it, and prints it to the terminal. This is useful for obtaining specific information about a certificate without having to parse through the entire certificate structure.*

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl x509 -in cert.pem -noout -serial
serial=2FA4B6EAE8FB025A5331CED26D8C92CC16803C61
```

But we want to find the serial number in hex:

OpenSSL provides a comprehensive set of tools and libraries for working with SSL/TLS protocols, certificates, and cryptographic operations. This includes commands for generating, managing, and examining X.509 certificates. X.509 is the standard format for public key certificates, and OpenSSL provides tools to handle certificates in this format. This makes OpenSSL a natural choice for tasks related to X.509 certificates. OpenSSL offers a wide range of options and parameters for certificate operations, allowing users to customize their actions according to their specific requirements. This flexibility is valuable in various scenarios, such as certificate generation, signing, verification, and examination. OpenSSL's command-line interface (CLI) allows for scripting and automation of certificate-related tasks. This is particularly useful in scenarios where batch processing or integration with other tools and scripts is necessary.

in summary, the entire command takes the serial number of the X.509 certificate, which is initially provided in a format like serial=XXXXXX, and then formats it into a hexadecimal representation with each byte separated by a colon. This format is commonly used for displaying serial numbers in certificates.

Correct answer : 2F:A4:B6:EA:E8:FB:02:5A:53:31:CE:D2:6D:8C:92:CC:16:80:3C:61

b. What is the organization that issued this certificate?

Answer:

*when you run the command ' **openssl x509 -in cert.pem -noout –subject** ' , OpenSSL reads the X.509 certificate from the file "cert.pem", extracts the subject information from it, and prints it to the terminal. This allows you to see the subject of the certificate, which often includes details about the organization that issued the certificate, among other information.*

Correct answer : O = Internet Widgits Pty Ltd

c. What is the email address associated with the certificate?

Answer:

when you run the command ' **openssl x509 -in cert.pem -noout –subject '** , OpenSSL reads the X.509 certificate from the file "cert.pem", extracts the subject information from it, and prints it to the terminal. This allows you to see the subject of the certificate, which often includes details about the organization that issued the certificate, among other information.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl x509 -in cert.pem -noout -subject

subject=C = US, ST = California, L = SKY-CRTB-1093, O = Internet Widgits Pty Ltd, OU = Liber8tionHackingGroup, CN = cityinthe.cloud, emailAddress = lib8Hack@cityinthe
.cloud
```

Correct answer: emailAddress = lib8Hack@cityinthe.cloud

d. What is the flag?

Answer:

*when you run the command ' **openssl x509 -in cert.pem -noout –subject '** , OpenSSL reads the X.509 certificate from the file "cert.pem", extracts the subject information from it, and prints it to the terminal. This allows you to see the subject of the certificate, which often includes details about the organization that issued the certificate, among other information.*

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl x509 -in cert.pem -noout -subject

subject=C = US, ST = California, L = SKY-CRTB-1093, O = Internet Widgits Pty Ltd, OU = Liber8tionHackingGroup, CN = cityinthe.cloud, emailAddress = lib8Hack@cityinthe
.cloud
```

Correct answer: L = SKY-CRTB-1093

SCANNING & RECONNAISSANCE

Challenge 1: Port Scan

We have identified what we believe to be one of Liber8tion's servers. We need you to conduct a TCP scan on the server.

The hostname of the server is 'target'

Question 1 : What is the 1st lowest open port?

Question 2 : What is the 2nd lowest open port?

Question 3 : What is the 3rd lowest open port?

Question 4 : What is the 4th lowest open port?

Answer:

nmap --top-ports 2000 --open target

Question 5: What is the version of the service running on the lowest open port?

Answer:

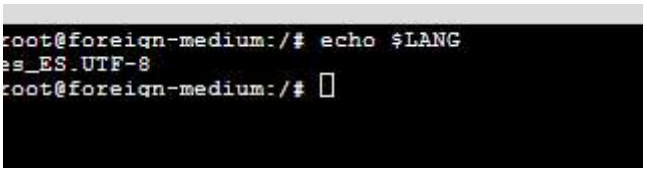
nmap -p 21 -sV target

Challenge 2: Foreign

Question 1: What language is the server set to use?

Answer:

To determine the default language/locale configured on the server, you can check the value of the LANG environment variable. Here's how you can do it in the command line:



The output es_ES.UTF-8 indicates that the default language and locale settings on the server are configured for Spanish (es) as spoken in Spain (ES), using the UTF-8 character encoding.

Correct answer : Spanish

Question 2: What time zone is the server using? (Use the tz db format)

Answer:

To determine the time zone the server is using in the tz database format, you can check the value of the TZ environment variable or directly inspect the contents of the /etc/timezone file.

```
root@foreign-medium:/# echo $TZ
Europe/Madrid
root@foreign-medium:/#
```

The output Europe/Madrid indicates that the server is using the Central European Time (CET) time zone, which corresponds to the Madrid time zone in the tz database format.

Correct answer: Europe/Madrid

Question 3: What country is the server located in?

Answer:

The server appears to be located in Spain, as indicated by the time zone setting Europe/Madrid, which corresponds to the time zone used in Madrid, Spain.

Correct answer: Spain

Question 4: What is flag 1 on the server?

Answer:

If you have a specific directory where you suspect the hidden message might be located, you can replace / in the command above with the path to that directory. For example: grep -r 'SKY' /path/to/directory

```
root@foreign-medium:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  li
root@foreign-medium:/# qrep -r -m 1 'SKY' /dev
root@foreign-medium:/# qrep -r -m 1 'SKY' /etc
/etc/default/locale:Flag 1 : SKY-LOCL-8520
root@foreign-medium:/#
```

Here we can see that we have a flag in the /etc folder

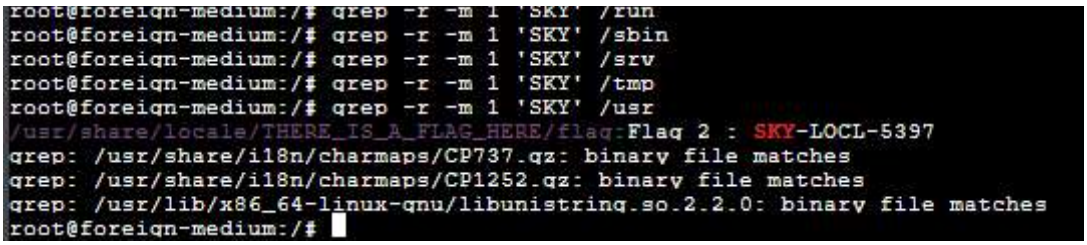
Correct answer: SKY-LOCL-8520

Question 5: What is flag 2 on the server?

Answer:

If you have a specific directory where you suspect the hidden message might be located, you can replace / in the command above with the path to that directory. For example:

grep -r 'SKY' /path/to/directory



Here we can see that we have a flag in the /usr folder

Correct Answer: SKY-LOCL-5397

LOG ANALYSIS

Challenge 1: Entry

You will need this to solve the challenge (access.log)

https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/EbRwwW9RiqRKtxhp4fGJD2QBsJIW4WyNniTwCdKv2m6dNA?e=8mQRWy

Question 1: What is the most common type of request?

Answer:

From just looking at the log data we can infer that it is ‘ GET ‘

Correct Answer: GET

Question 2: How many POST requests were sent to the server?

Answer:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ grep -c '"POST' access.log
268
```

Correct Answer: 268

Question 3: What IP address sent the most requests?

Answer:

awk '{print $1}' access.log | sort | uniq -c | sort -nr | head -1

```
┌──(kali㉿kali)-[~/Downloads]
└─$ awk '{print $1}' access.log | sort | uniq -c | sort -nr | head -1
     87 66.135.18.206
```

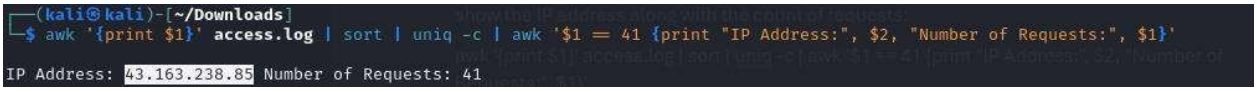
Correct Answer: 66.135.18.206

Question 4: What IP address sent exactly 41 requests?

Answer:

show the IP address along with the count of requests:

awk '{print $1}' access.log | sort | uniq -c | awk '$1 == 41 {print "IP Address:", $2, "Number of Requests:", $1}'



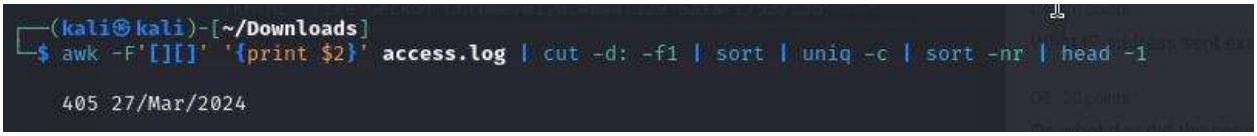
Correct Answer: 43.163.238.85

Question 5: On what day did the server receive the most traffic?

Answer:

To find out on what day the server received the most traffic from the provided access.log file snippet, we need to extract the dates from the log entries and then count the occurrences of each date. We can achieve this using awk and some text processing. Here's the command:

awk -F'[][]' '{print $2}' access.log | cut -d: -f1 | sort | uniq -c | sort -nr | head -1



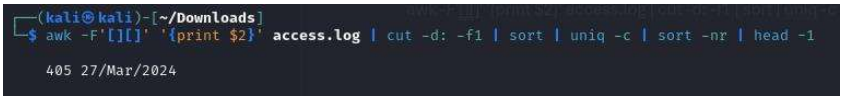
Correct Answer: 03/27/2024

Question 6: How many requests were sent on the day with the most traffic?

Answer:

To determine the number of requests sent on the day with the most traffic, we can extend the previous command to extract the count along with the date. Here's the modified command:

awk -F'[][]' '{print $2}' access.log | cut -d: -f1 | sort | uniq -c | sort -nr | head -1



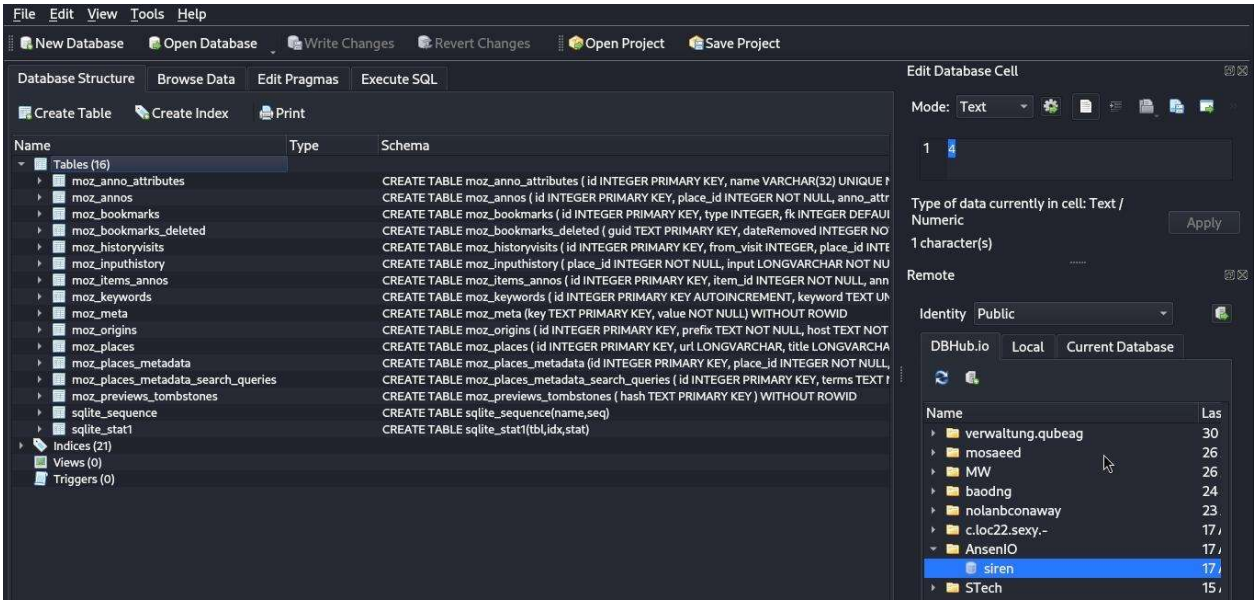
Correct Answer: 405

Challenge 2: Places

You will need this to solve the challenge (places.sqlite) :

https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/ESbjPn3bUftHkfQhWMpzWcEBeFdYxdYm8YjKOH_f1EfYqg?e=mk20dl

Note: since this file is a .sqlite file, we can open the file in DB Browser sqlite.

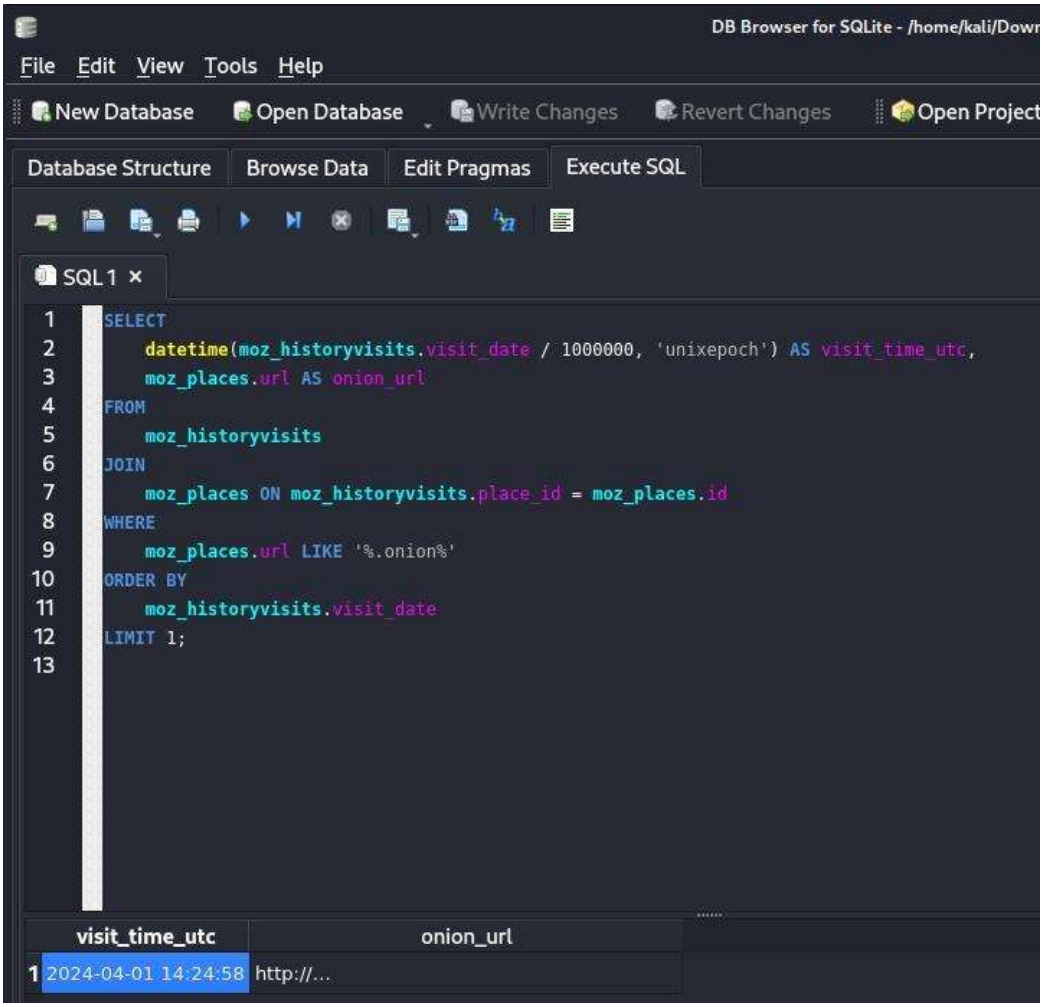


Questions 1: What time (in UTC) was the first page viewed using the Tor browser? (Round up to the nearest minute)

Answer:

We can run some sql queries directly in the DB Browser sqlite application

This query will return the URL of the .onion address along with the visit time in UTC. Please execute this query in your SQLite tool to find the URL of the .onion search engine that was used.

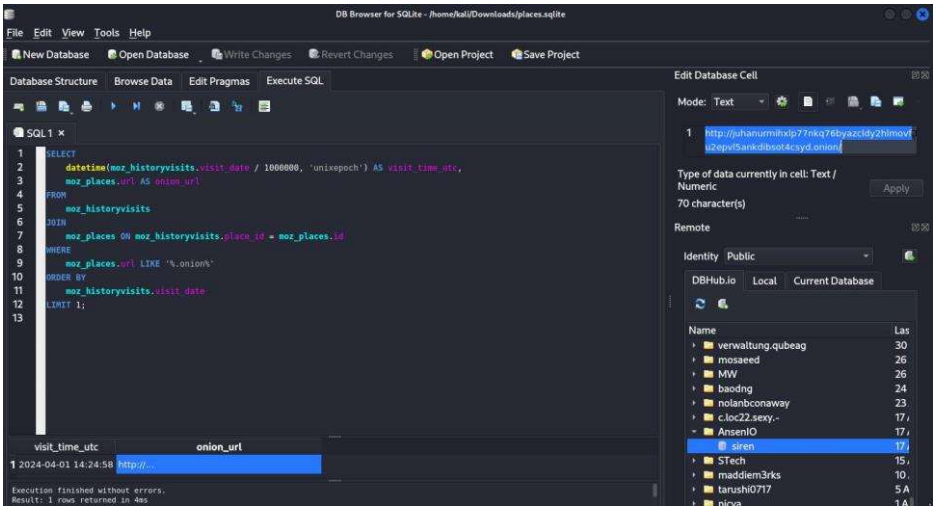


Correct Answer : 2024-04-01 14:24:58 (UTC) == 2024-04-01 14:25

Question 2: What is the URL of the .onion search engine that was used?

Answer:

For this question we can use this query:



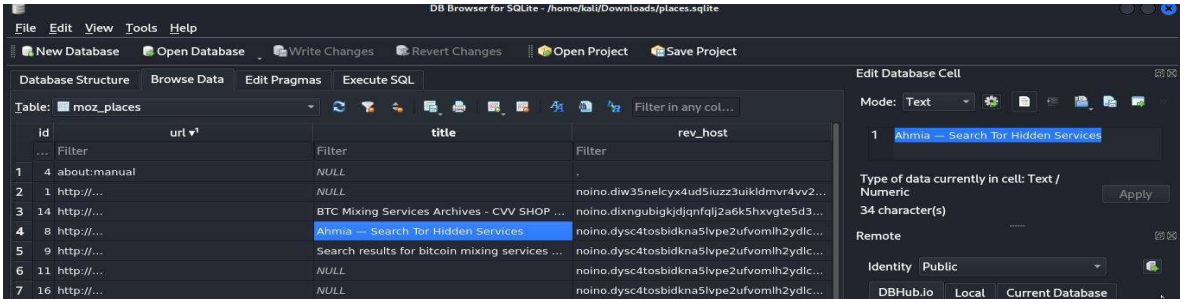
Correct Answer :
http://juhanurmihxlp77nkq76byazcldy2hlmovfu2epvl5ankdibsot4csyd.onion/

Question 3: What is the name of the .onion search engine that was used?

Answer:

The query provided earlier retrieves the URL of the .onion address that was visited, but it doesn't directly give the name of the .onion search engine. In most cases, the name of the search engine may not be explicitly stored in the browser's history.

However, based on the URLs provided earlier, if the .onion address corresponds to a known .onion search engine, you might recognize it by inspecting the URL. Common .onion search engines include Ahmia, Candle, and not Evil, among others. For example, Ahmia URLs typically contain "ahmia" in the domain name, while Candle URLs might contain "gjobqjj7wyczbqie" or "gjobqjj7wyczbqie.onion" in the domain name.



Correct Answer : Ahmia

Challenge 3 : Buffed

You will need this to solve the challenge (messages.log) :

https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/EYhSV8v1F9lKjXwy9eQit3gByJ8kDetlgYpjnzuHgrnxvA?e=pu34QH

We have found a log file containing chats between two Liber8tion hackers. We need to read their conversation; however, it is in a format we are unfamiliar with. Can you

analyze the file and answer some questions about it? We found this text comment in the same folder as the log: user, ??, ??, epoch, id, content.

Question 1: How many users are listed in the log?

Answer:

After examining the log we can infer that its encoded with Base64 and from the description it seems that there's 6 fields in the log file. Lets see if we can display the first field by copying the data in the first field into a new file 'users.txt'

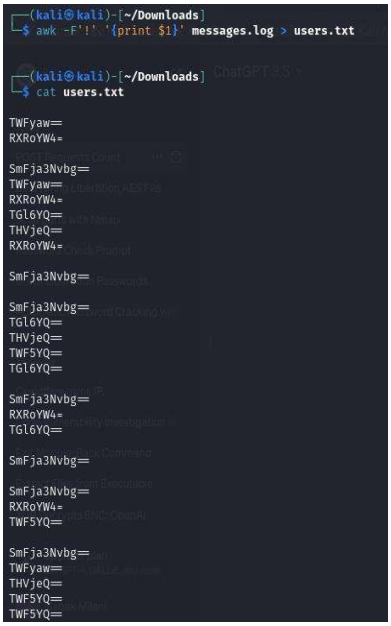
```
┌──(kali㉿kali)-[~/Downloads]
└─$ awk '{print $3}' messages.log | sort | uniq > users.txt
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat users.txt
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ awk '{print $1}' messages.log | sort | uniq > users.txt
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat users.txt

RXRoYW4=!
RXRoYW4=!O◆◆xB)@g@28VG90YW0gaXBzYSBjb25zZXF1YXR1ciBxdWlhIHZvbHVwdGFzIGVuaW0uBREAK
M◆◆xB)@f@20RXQgb2ZmaWNpYSBibGFuZGl0aWlzIHF1aSBvbW5pcyBxdWkuBREAK
RXRoYW4=!L◆◆xB)◆e@2,VmVsaXQgbWF4aW1lIHRvdGFtIG9jY2FlY2F0aSBhdC4=BREAK
RXRoYW4=!▒6◆◆xB)U@20Vm9sdXB0YXRlbSBlYSBldW0gc2ludCByZXBlbGxlbmR1cy4=BREAK
RXRoYW4=!J◆◆xB)◆d@2
RXRoYW4=!%I◆◆xB)@d@20RXVtIHF1aSBleHBsaWNhYm8gdm9sdXB0YXRlbSBwb3Jyby4=BREAK
```

So from the commands its clear that its only one field with all the data. I noticed that every line has the symbol '!' so that is an indication of some unique part of the encryption so I want to run a command to only copy every sentence up to the first '!'.

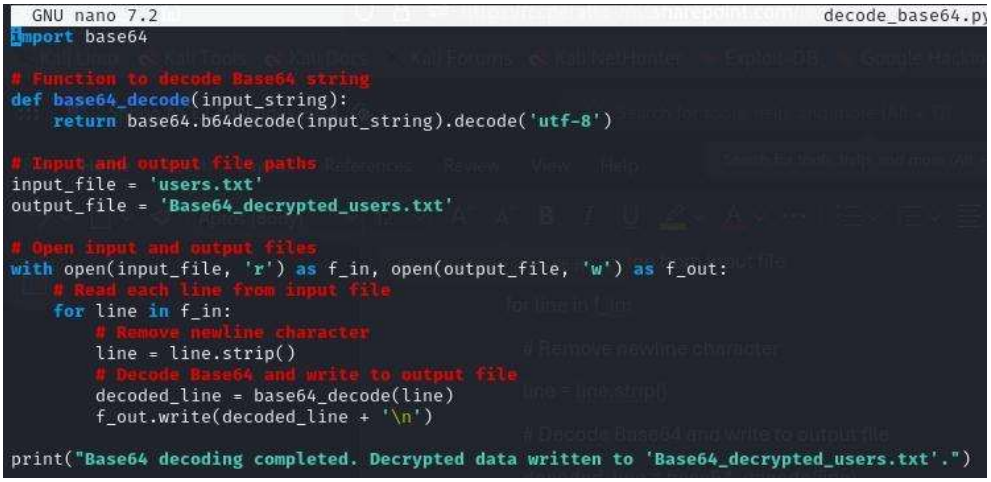


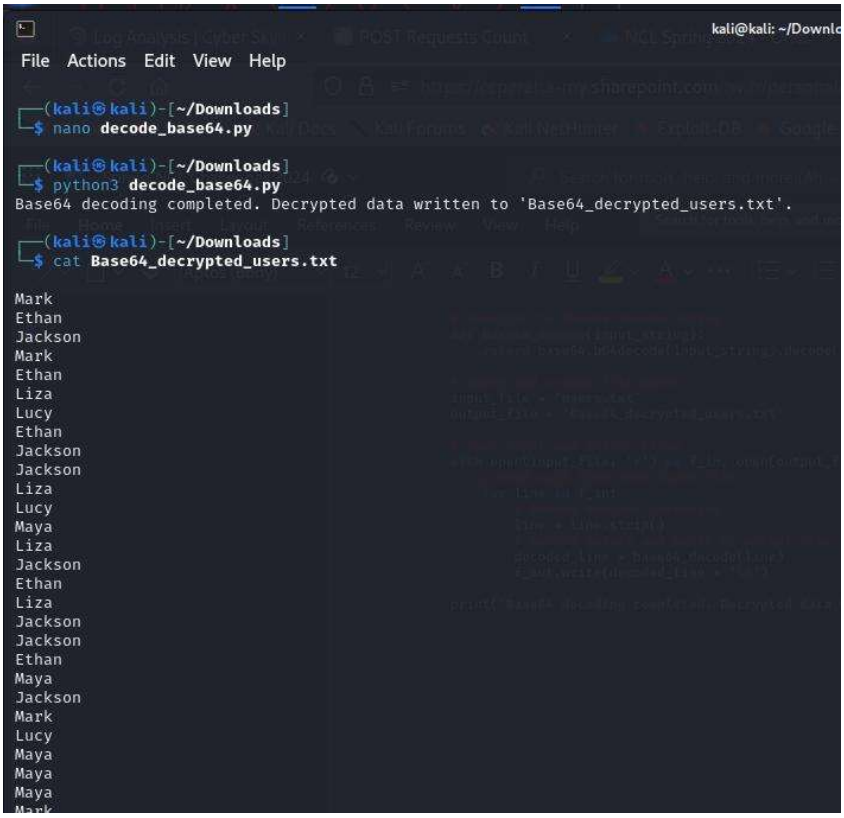
Let's clean the new file : remove inconsistencies and anamolies (eg : any lines that dont match the majority of the data).

To count how many listed users in the log, we must convert the Base64 to plaintext, and I developed a python script to do that. I will name the file (decode_Base64.py):

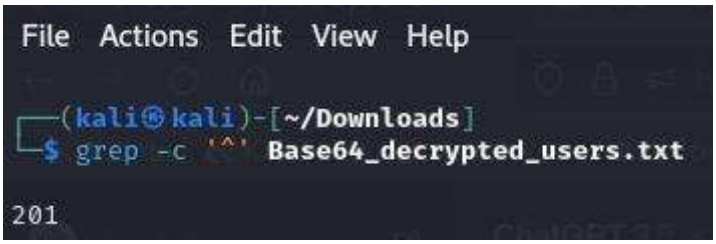
```
#Python Script

import base64

# Function to decode Base64 string

def base64_decode(input_string):

    return base64.b64decode(input_string).decode('utf-8')

# Input and output file paths

input_file = 'users.txt'

output_file = 'Base64_decrypted_users.txt'

 # Open input and output files

with open(input_file, 'r') as f_in, open(output_file, 'w') as f_out:

    # Read each line from input file

    for line in f_in:
```

```
    # Remove newline character
    line = line.strip()
    # Decode Base64 and write to output file
    decoded_line = base64_decode(line)
    f_out.write(decoded_line + '\n')
print("Base64 decoding completed. Decrypted data written to 'Base64_decrypted_users.txt'.")
```

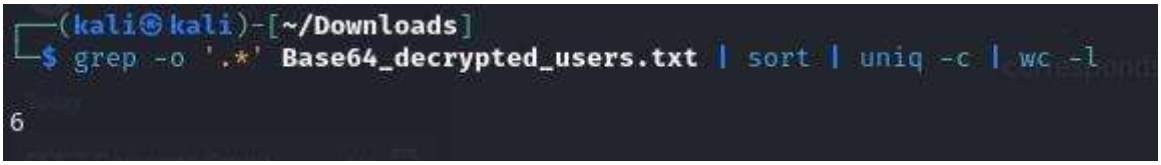




As you can see, we can decode the data to plaintext. Now we want to count how many users are in the log. To count the total number of lines in the file 'Base64_decrypted_users.txt' using grep, you can use the following command:



To count how many unique users are listed in the file 'Base64_decrypted_users.txt' using grep, you can use the following command:



Correct Answer : 6

FORENSICS

Challenge 1 : Lost

You will need this to solve the challenge (fs.img):

https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/EaVPCa9yoNBKijQGC8M-Af8Bj3QChV0bI_1rWkL_wGHp8A?e=N3MRwn

Question 1: What is the filesystem that is contained in this image?

Answer:

We run a simple command after downloading the file to our system.

```
└─$ file fs.img
fs.img: Linux rev 1.0 ext4 filesystem data, UUID=ec8ac828-88eb-449b-b6b6-78eed438ab9e (needs journal recovery) (extents) (64bit) (large files) (huge files)
```

Correct Answer : Linux rev 1.0 ext4 filesystem data

Question 2: What is the filetype of the deleted file?

Answer:

For this we want to use the Foremost tool in Kali. If you dont have already installed, go ahead and install it.

```
sudo apt-get install foremost
```

Use foremost to analyze the image file and recover deleted files:

```
foremost -i fs.img
```

This will create a folder with the content of that disk image

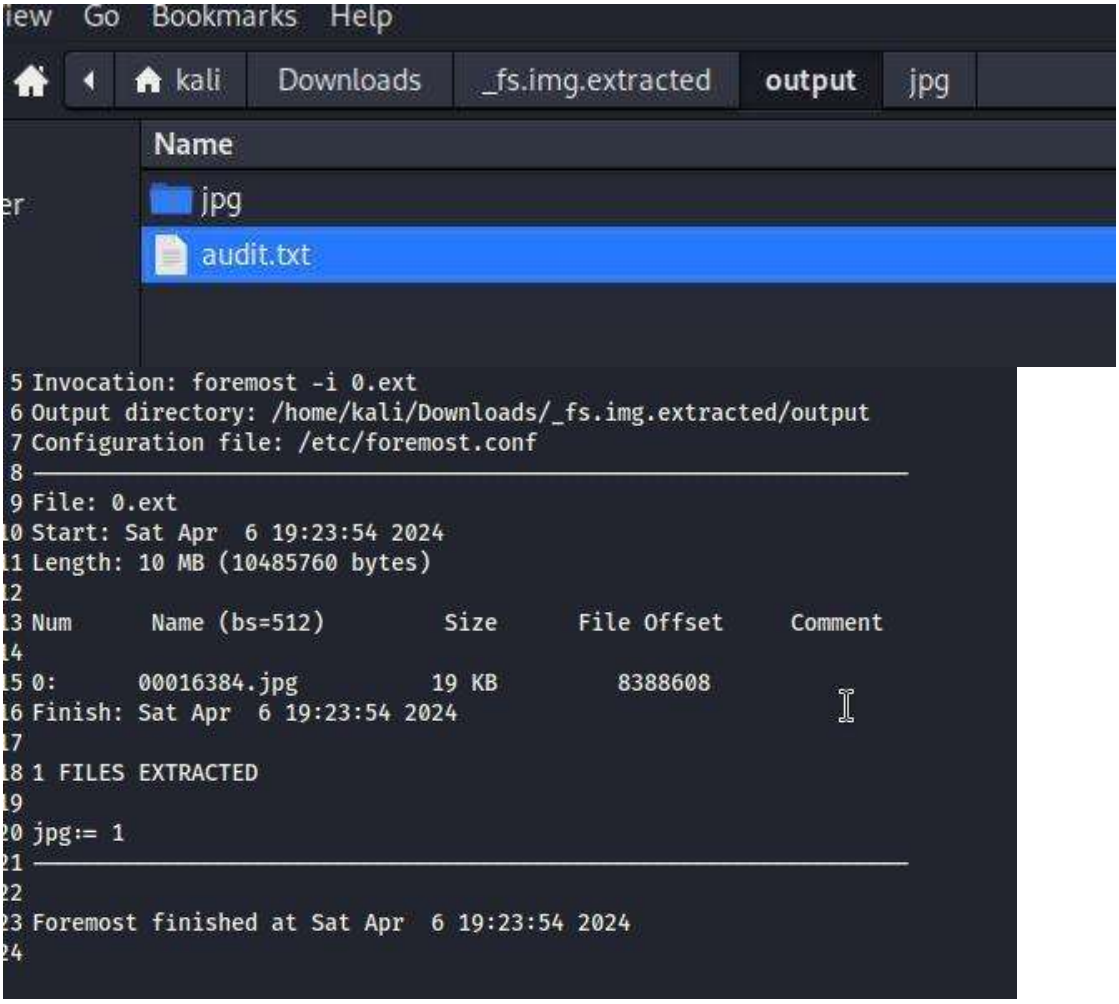
Now change directory to the new folder and look whats in there

```
┌──(kali㉿kali)-[~/Downloads/_fs.img.extracted]
└─$ file 0.ext
0.ext: Linux rev 1.0 ext4 filesystem data, UUID=ec8ac828-88eb-449b-b6b6-78eed438ab9e (needs journal recovery) (extents) (64bit) (large files) (huge files)

┌──(kali㉿kali)-[~/Downloads/_fs.img.extracted]
└─$ strings 0.ext
/home/ubuntu/tmp
lost+found
lost+found
flag.jpeg
/home/ubuntu/tmp
lost+found
/home/ubuntu/tmp
JFIF
```

We have to extract the files from the file '0.ext' we found with the foremost tool again

```
┌──(kali㉿kali)-[~/Downloads/_fs.img.extracted]
└─$ foremost -i 0.ext

Processing: 0.ext
|*|
```

We change directory again into that new folder



Correct Answer : jpg

Question 3: what is the flag

Answer:

Open the flag.jpg file

Correct Answer : SKY-RCVR-8755

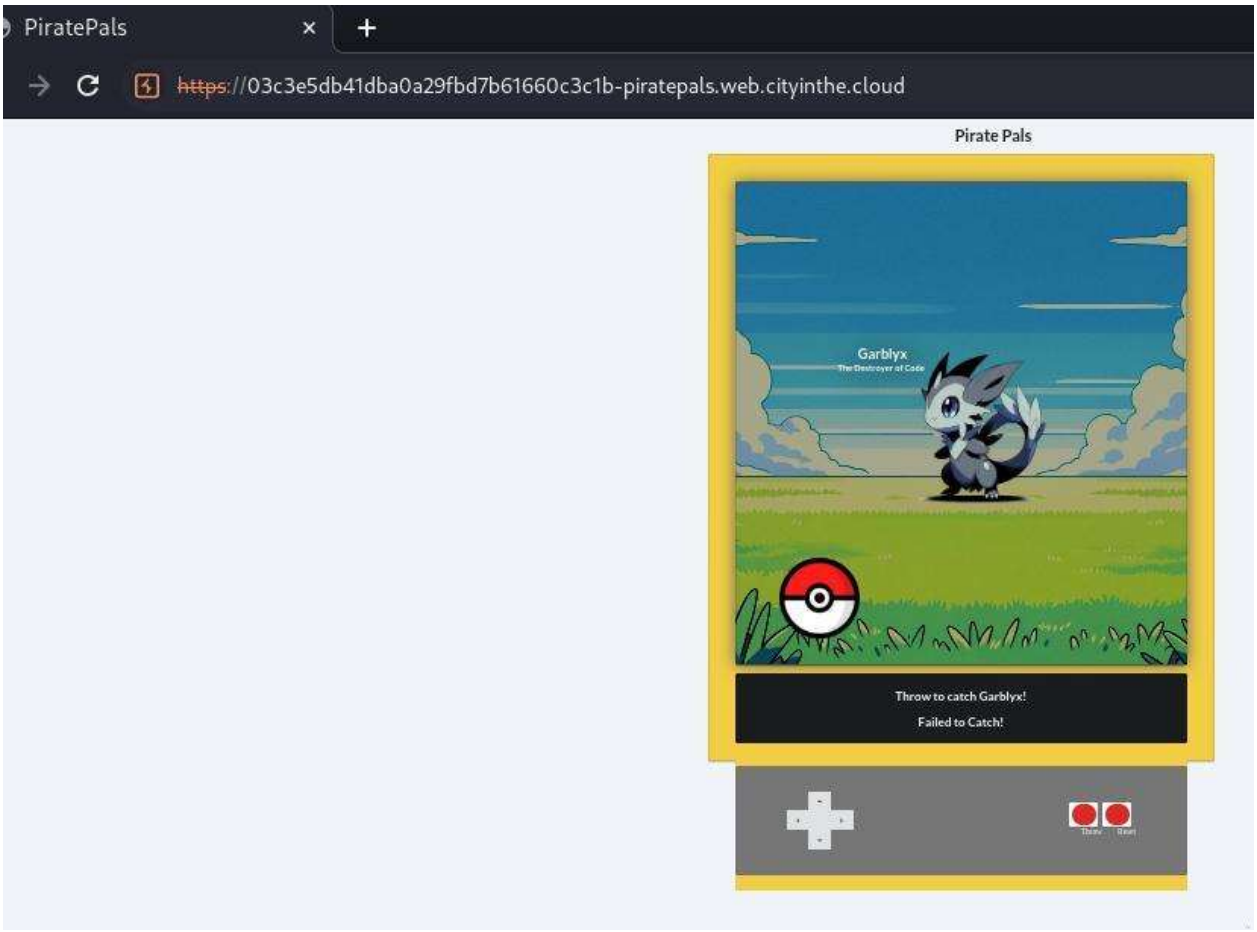
WEB APPLICATION AND EXPLOITATION

Challenge 1: PiratePals

You encountered a wild Garblyx at PiratePals, try to catch it and in exchange I'll give you a flag!

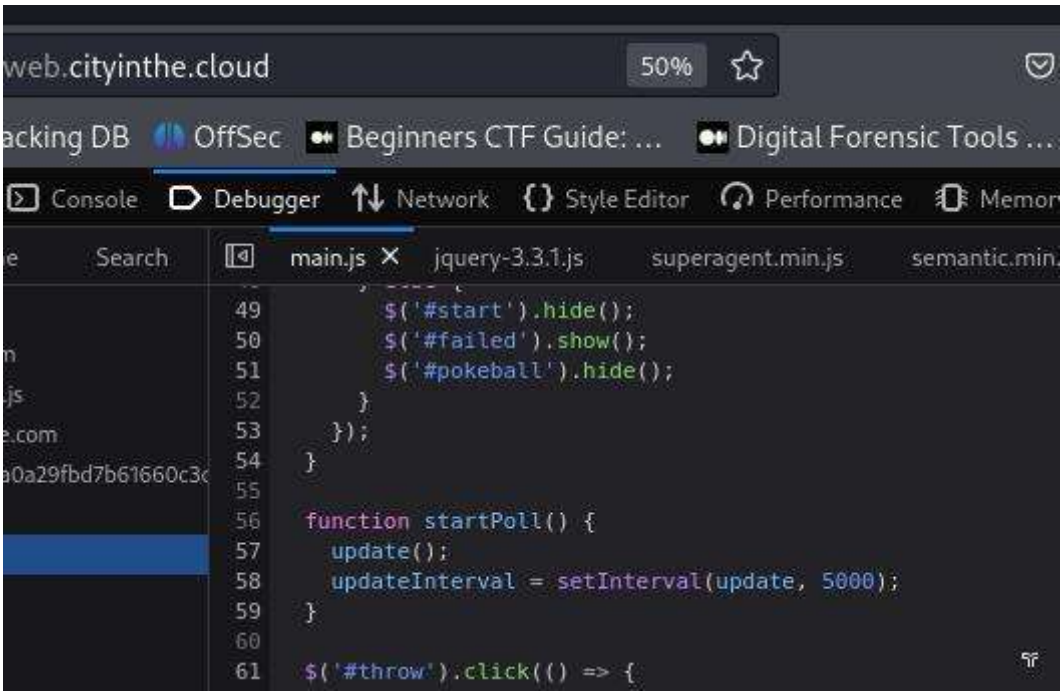
Note: Your scope is limited to HTTPS. You may use automated tools that make educated guesses for this challenge, but blind automated brute force is not permitted.

Answer:



Questions 1 : How many seconds are in between each update interval?

Opening the inspection feature in the Firefox browser reveals the code behind the web application where we can find information regarding the algorithm for every interval at every event the user engages the web application.



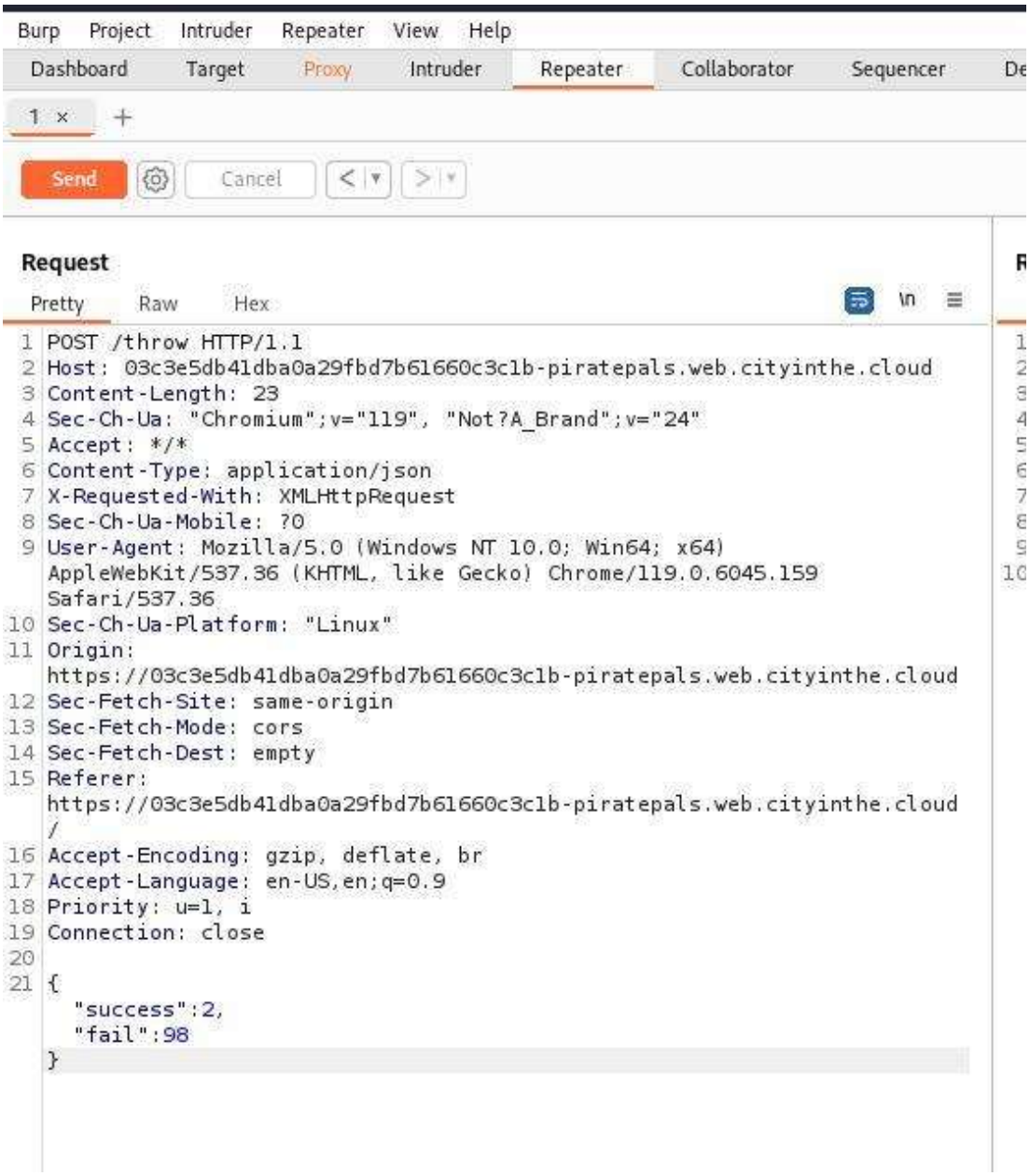
To answer the first question we can see here that the interval is set.

The setInterval() function is set to call the update() function every 5000 milliseconds (which is equivalent to 5 seconds). Therefore, the update interval between each call to the update() function is 5 seconds.

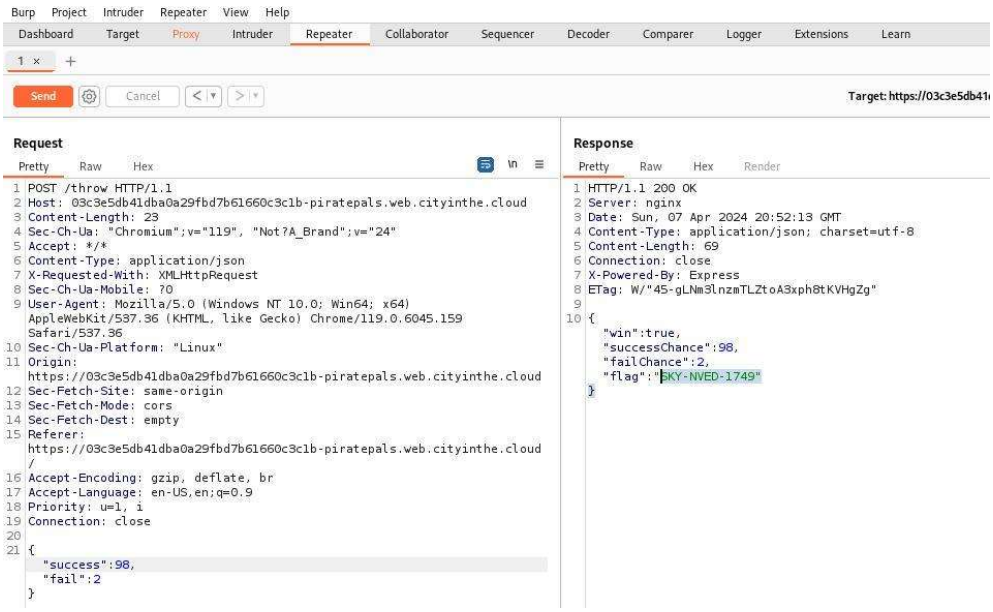
Correct Answer is : 5 Seconds

Questions 2: What is the URL path of the vulnerable endpoint on the server?

Here we want to use a proxy (Burpsuite to manipulate the backend algorithm). We intercept the url in Burpsuite and send the request to the repeater function in Burpsuite and manipulate the code.



As you can see the code is set to fail from the start so the user have a minimal chance of winning. Here we will reverse the success and fail rate and see what happens.

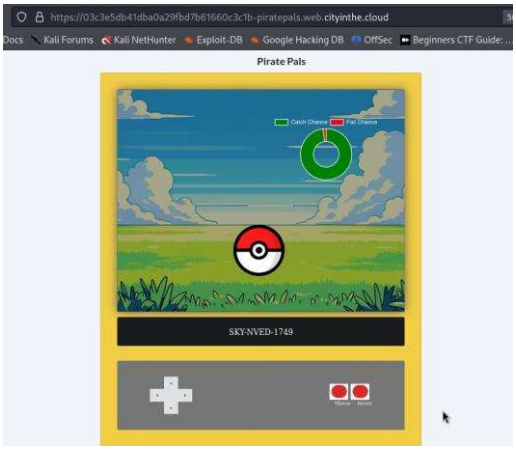


As you can see we are able to retrieve the hidden flag as highlighted in the screenshot.

So for answer two the answer is actually in line 1 under the pretty tab (first line : POST /throw HTTP/1.1)

Correct answer : /throw

Question 3 : What is the flag?



Correct answer: SKY-NVED-1749

CHALLENGE 2: Pierre's Store

You will need this to solve the challenge (pierre.har) : https://ccperalta-my.sharepoint.com/:u:/g/personal/bami9269_cc_peralta_edu/EZeC5rZmA2NAr4Kc3N9z8LQB3ShAjAkpADGMoZu9H6jnPw?e=duX4zM

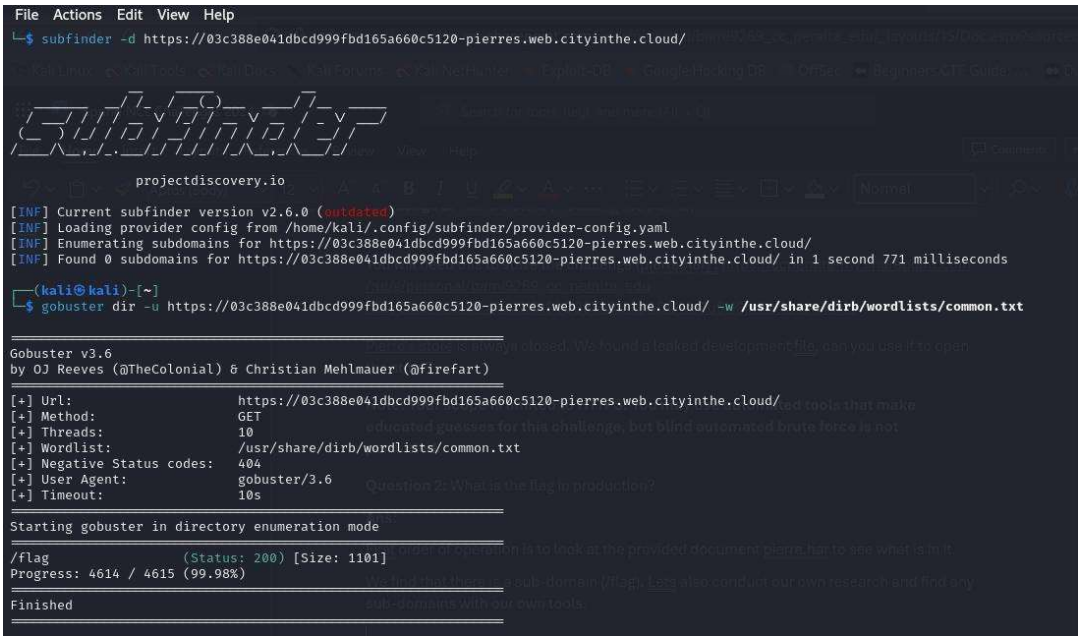
Pierre's store is always closed. We found a leaked development file, can you use it to open the store?

Note: Your scope is limited to HTTPS. You may use automated tools that make educated guesses for this challenge, but blind automated brute force is not permitted.

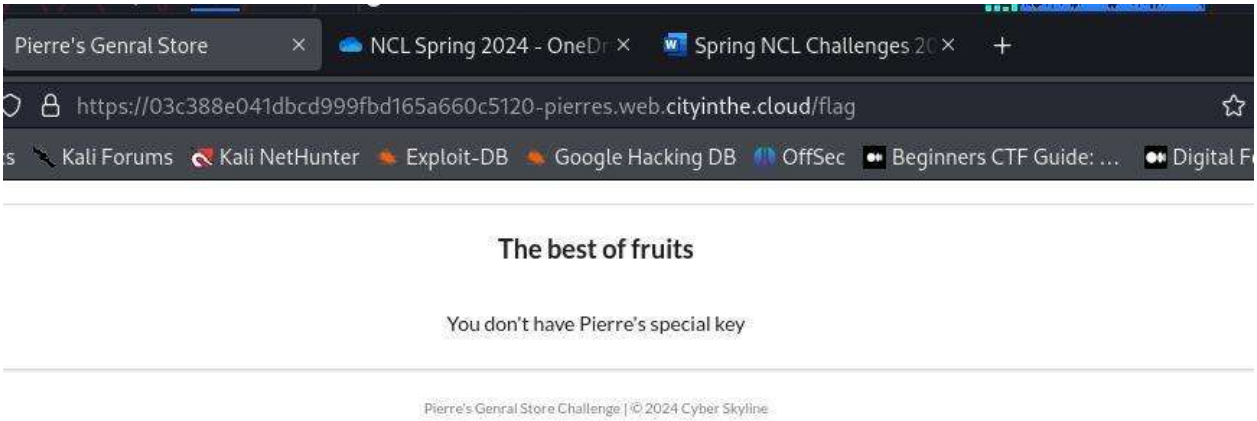
Question 2: What is the flag in production?

Answer:

First order of operation is to look at the provided document pierre.har to see what is in it. We find that there is a sub-domain (/flag). Lets also conduct our own research and find any sub-domains with our own tools.

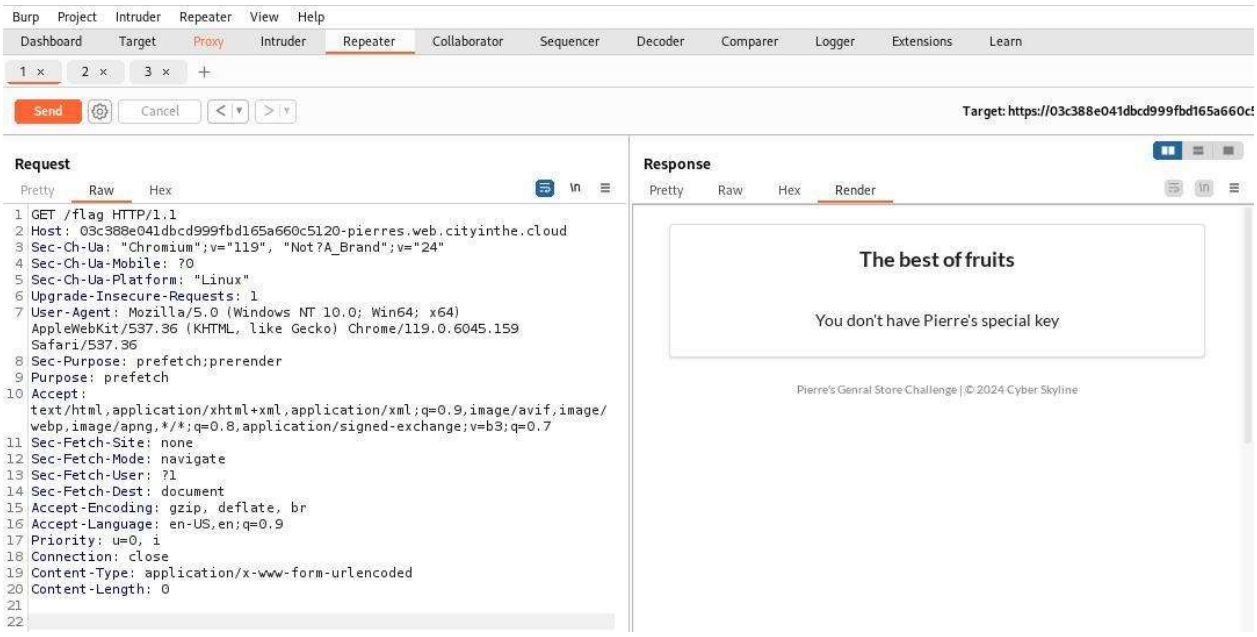


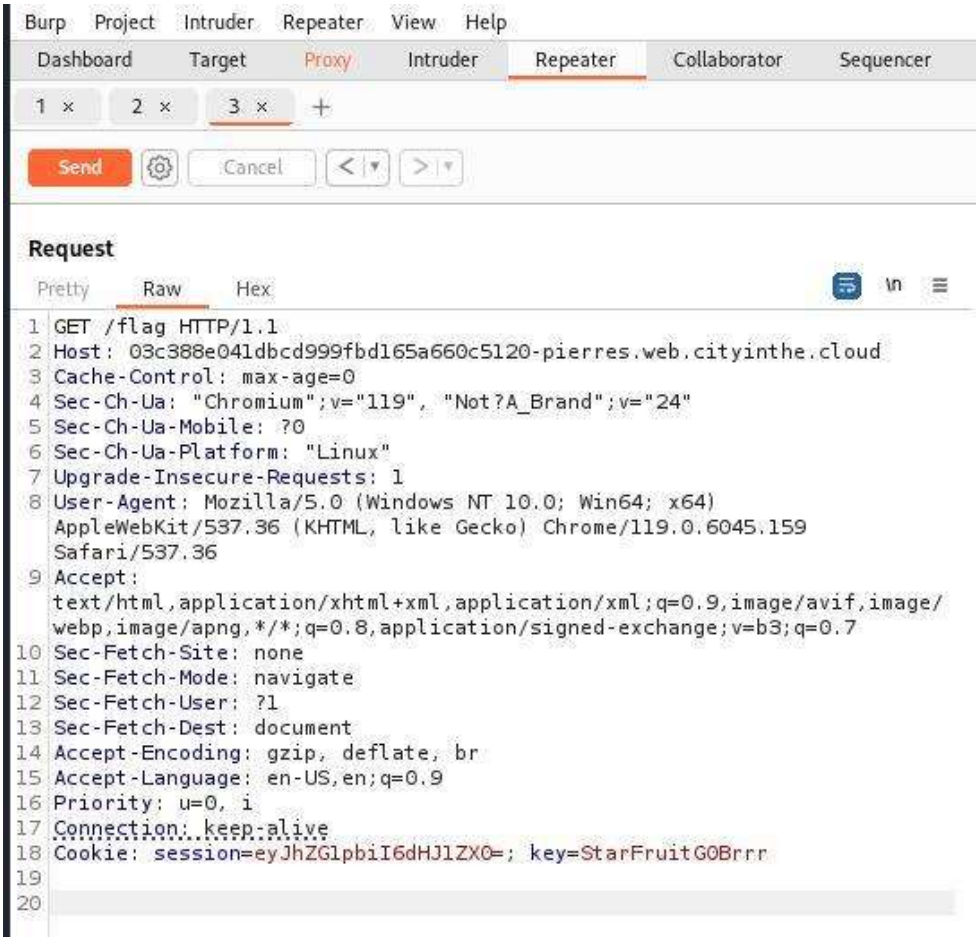
Our first command line tool execution is not successful so we try another one and in fact we find a sub-domain as we found in the document provided (/flag)



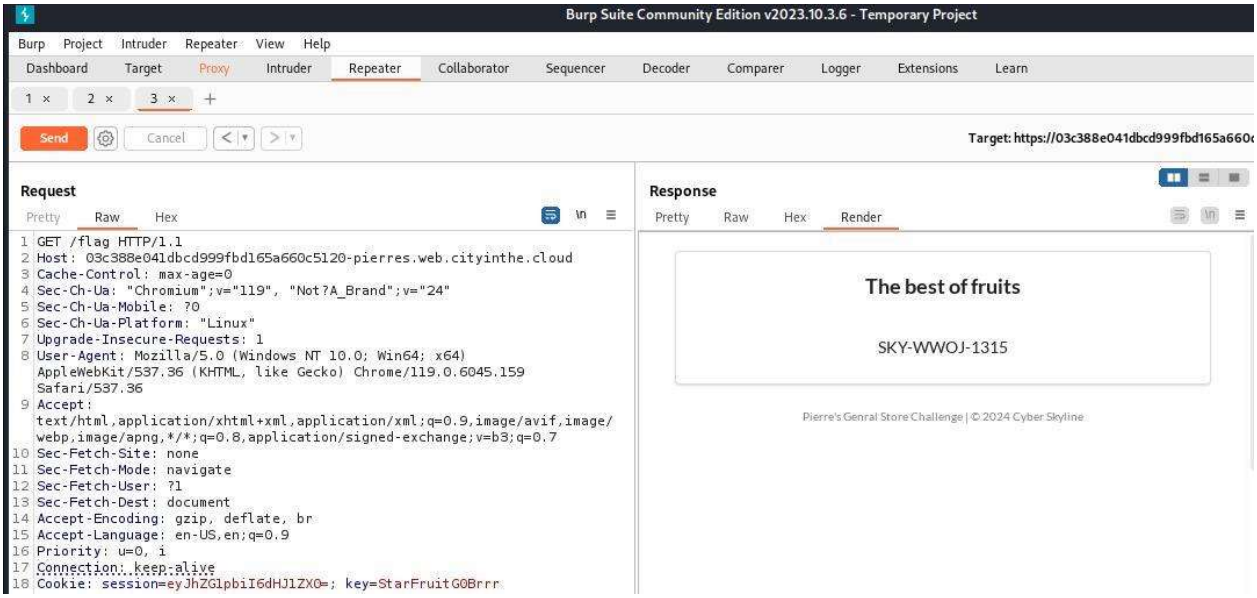
Looking further into the document, we see there is a key with a value in the cookie session.

We need to figure out how we can add and where to add the cookie with the value we found in the document. This is commonly added in the request header.





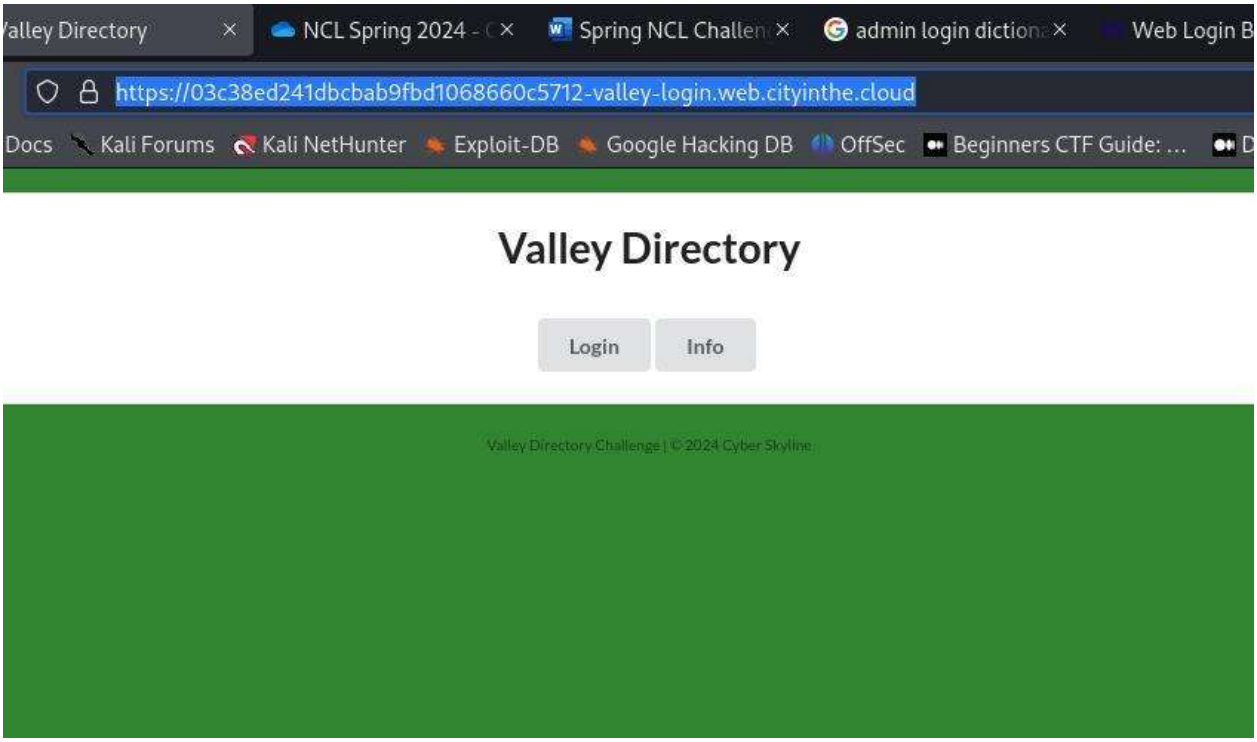
Turns out it was the correct path to finding the hidden flag.



Correct Answer : SKY-WWOJ-1315

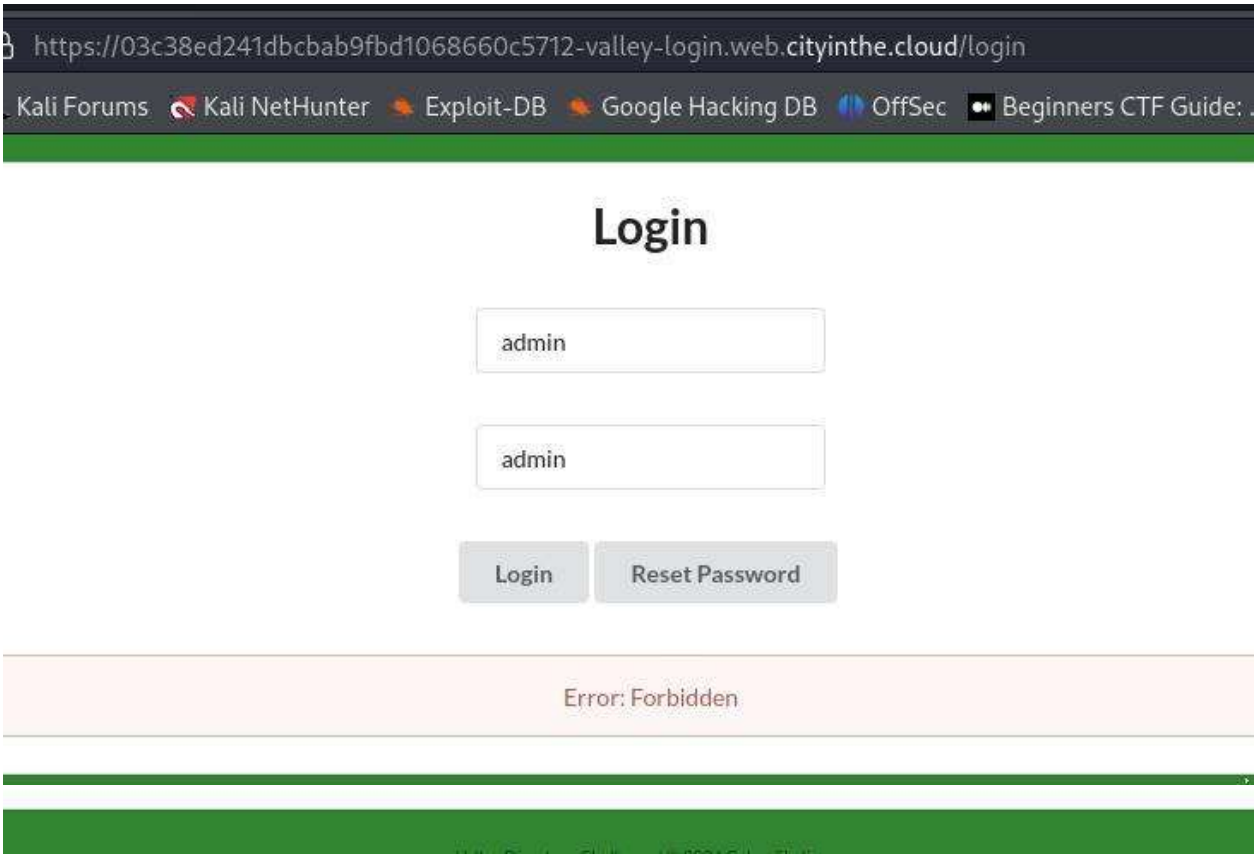
Challenge 3: Valley Directory

We received an anonymous report that there is a security vulnerability on our admin account login. Can you conduct a security evaluation to find the vulnerability?

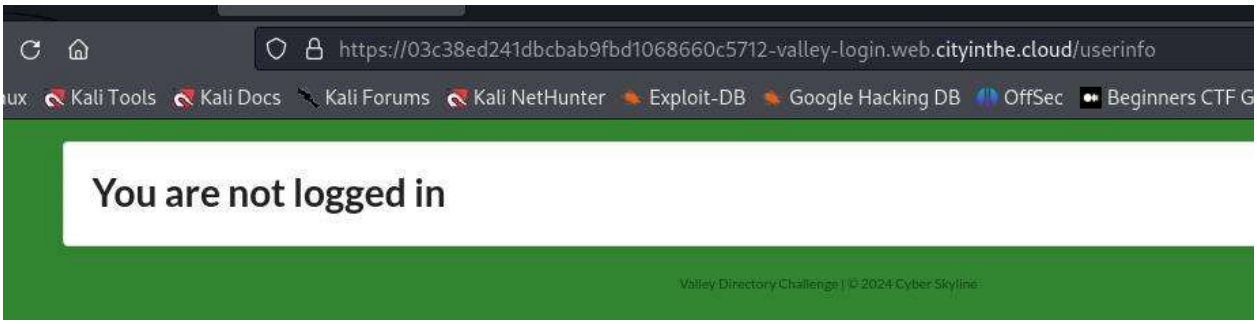


Lets run a command line tool against the url

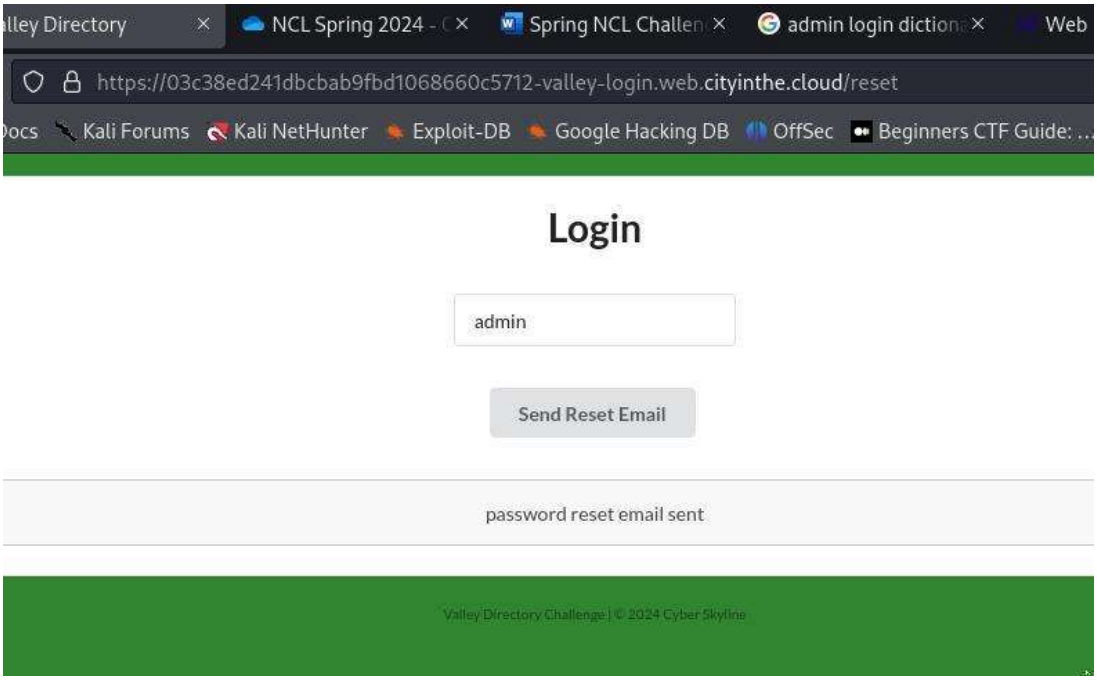
We can see that we have found some subdomains. Just for simple purposes lets just test our web application and see how it works.



Seems as if both the login pages are the same /login and /Login.



/userinfo apparently is for after we login



We move to simply check the reset page and try admin user. And then we go back to /userinfo page and find the hidden flag.