

National Cyber League Spring 2024

ENUMERATION AND EXPLOITATION

Enumeration & Exploitation
3 Challenges | 8 Questions | 0% Completion
Identify actionable exploits and vulnerabilities and use them to bypass the security measures in code and compiled binaries.

Challenge #1:

This challenge 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()
```

The screenshot shows the Replit IDE interface. The left sidebar displays files: main.py, poetry.lock, and pyproject.toml. The main workspace shows the Python code. The top navigation bar includes a 'Run' button. The right side features a 'Console' tab showing the command poetry lock --no-update and the output SKH10CEK<./=6. Below the console is a 'Run' section showing a successful execution with a duration of 126ms.

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'.

The screenshot shows the Replit Python environment. On the left, the file tree shows 'main.py' and 'poetry.lock'. The code editor contains the following Python script:

```
1 ENC = [95, 91, 68, 33, 67, 83, 73, 91, 48, 63, 34, 45, 58]
2 def main():
3     dec = ''
4     rotate = [12, 16, 29]
5     count = 0
6     for byt in ENC:
7         dec += chr(byt ^ rotate[count])
8         count = (count + 1) % 3
9     print(dec)
10 if __name__ == '__main__':
11     main()
12
```

The right side of the interface shows the execution logs. It starts with 'poetry lock --no-update Resolving dependencies...', followed by a successful run command: 'SKH10CEK<./=6'. Then it runs again with 'poetry lock --no-update Resolving dependencies...', and finally, the output 'SKY-SNEK-3206' is displayed.

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-yYGeoBsbEwnQcM5YDMDTclexTv9A?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:



Password Cracking

5 Challenges | 17 Questions | 0% Completion

Identify types of password hashes and apply various techniques to efficiently determine plaintext passwords.

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.

```
└$ echo -n "thelastofus3091" | md5sum
8e4170a44ffe3b2a2b585f589b7aa3fe  -
└─(kali㉿kali)-[~/Downloads]
└$
```

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**).*

```
(kali㉿kali)-[~/Downloads]
$ echo -n "1723succession" | sha1sum
3d8d499da42fb3fef551e01e7b43e1d41a70964  -
(kali㉿kali)-[~/Downloads]
$
```

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**).*

```
(kali㉿kali)-[~/Downloads]
$ echo -n "33gameofthrones32" | sha256sum
60d3262b3ba512e2056c5aba5c267a94b2655d9202c8d19fd857c4fe3aa80bc5  -
(kali㉿kali)-[~/Downloads]
$
```

CRYPTOGRAPHY

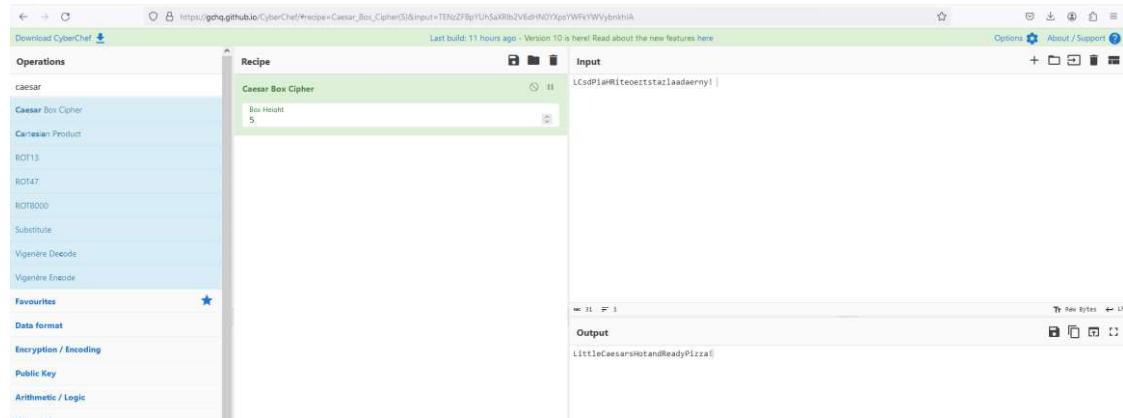
 Cryptography
7 Challenges | 10 Questions | 0% Completion
Identify techniques used to encrypt or obfuscate messages and leverage tools to extract the plaintext.

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 valid 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/Eb3Zg1e7VjZDhQD31pAO_P_EBRbSbY0dH0lukYYyTi6v7oA?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:

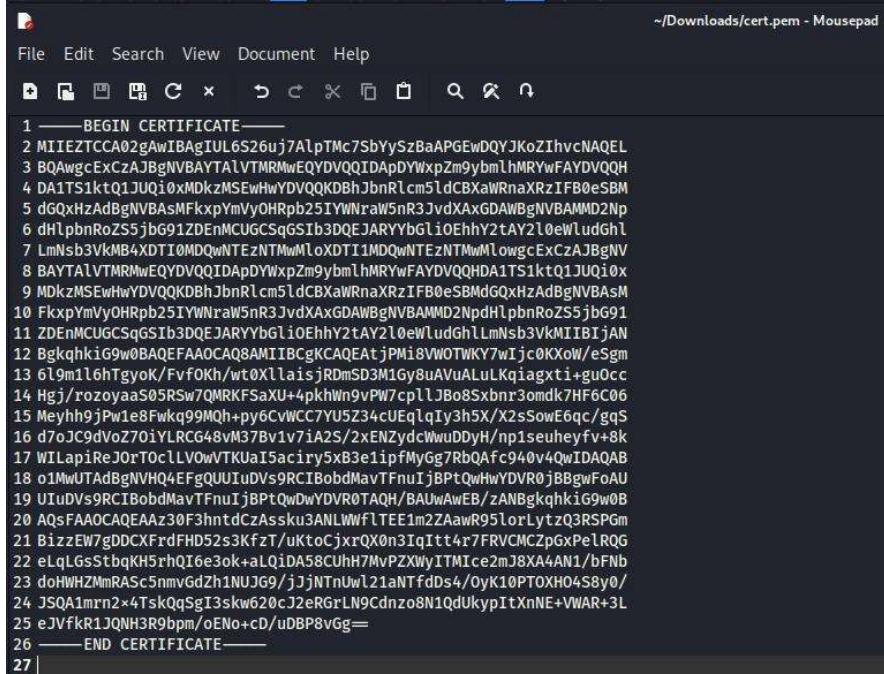
```
(kali㉿kali)-[~/Downloads]
$ file encoded.pem
encoded.pem: ASCII text

(kali㉿kali)-[~/Downloads]
$ strings encoded.pem
LS0tLS1CRUdtJt1bDRVJUSUZJQ0FURS0tLS0tCk1JSUVaVENDQTAYz0F3SUJBZ01VTDZTMjZ1ajdB
bHBUTWM3U2JZeVN6QmFBUEdFd0RRRUwLb1pJaHZjTkFRRUwKQlFBd2djRXhDekFKQmdOVkJBWWRB
bFZUTVJNd0VRWURWUVFJREFwRFxExhBabTl5Ym1saE1SWXdGQVlEVlFRSApEQTFUuzFrdFExSlVR
aTB4tURrek1TRxdId1EVlFRS0RCAEpibljsY201bGRDQlhvh1JuYhSekLGQjbLU0JNCmrHUxhI
ekFkQmdOVkJBc01Ga3hwW1WeU9iUnBiMjVJWVdOcmFXNW5SM0p2ZFhBeEdEQVdcZ05WQkFNTUQy
TnAkZehscGJuUm9aUzVqYkc5MvpERW5NQ1VHQ1NxR1NjYjNEUUVKQVJZWWJhbGLPRWhoWTJ0QVky
bDBlV2x1ZEdobApMbU5zYjNwa01CNFhEVEkwTURRd05URXpOVE13TwvWERUSTfNRFF3TlRFek5U
TxidNbG93Z2NFeEN6QUpCZ05WckJBWVRBbFZUTVJNd0VRWURWUVFJREFwRFxExhBabTl5Ym1saE1S
WXdQVlEVlFRSERBMVRTMwt0UTFKVVFpMhgKTURrek1TRxdId1EVlFRS0RCAEpibljsY201bGRD
QlhhV1JuYvhSekLGQjbLU0JNZEddReEh6QWRCZ05WQkFzTQpGa3hwW1WeU9iUnBiMjVJWVdOcmFX
NW5SM0p2ZFhBeEdEQVdcZ05WQkFNTUQyTnBkSGxwYm5Sb1pTNWpiRzkxClpERW5NQ1VHQ1NxR1Nj
YjNEUUVKQVJZWWJhbGLPRWhoWTJ0QVkybDBlV2x1ZEdobExtTnNiMzrTULjQklqQU4KQmdrcWhr
aUc5dzBCQVFFRkfBT0NBUTHBTU1JQkNnS0MBUUVBdGp0TWk4YldPVFdLWTd3SWpjMEtYb1cvZVnn
bQo2bDltMWw2aFRneW9LL0Z2Zk9LaC1dBNyQkRtU0QzTTFHeTh1QVZlQQux1TeTxawFn
eHRpk2d1T2NjCkhna19yb3pveWFhJuZA1U1N3N1FNuuktGU2FYVss0cGtoV245dlBXN2NwbGxKQm84
U3hibnIzb21kazdIRjZMDYKTWV5aG5aLB3MWU4RndrcTk5TVFoK3B5NKn2V0NDN1lVNvozNGNV
RXFscUl5M2g1WC9YMnNTb3dFnNfjL2dxUwpkN29KQzlkVm9aN09pWUxSQ0c00HZNMzdCdjF2N2lB
MLMvMnhFTlp5ZGNxd3VERHIL25wMXNldwHleWZ2KzhrCldJTGFWaVjlSk9yVE9jbExWT3dWVeTV
YUK1YWNPcnk1eIzzTFpcGZNeUdn1NjIuUFmYzk0MHY0UXdJREFRQIKbzfNd1VUQWRCZ05WSFE0
RUZnUVVVSVXEVnM5UkNJQm9iZE1hd1RGbnVJakJQdFF3SHdZRFZSMGpCQmd3Rm9BVQpVSXEVnM5
UkNj0m9iZE1hd1RGbnVJakJQdFF3RHdZRFZSMFRBUUgvQkFVd0F3RU1ivekFOQmdrcWhraUc5dzBC
CkFrC0ZBQ9DQVFQUF6MzBGM2hudGRDekFzc2t1M0F0TfdXzmxURUUbTjaQWF3Ujk1bG9pTHl0
elEZuLNQR20KQml6ekVXN2dERENYRnJkRkhENTJzM0tmelQvdut0b0NqeHJRWDBuM0lxSXReHHI3
RLJWQ01DWnBHeFB1JFRRwpLTHFMRTNTdGJxS0g1cmhRSTZLM29rK2FMuWLEQTU4Q1VsDdNdlBa
WFd55VRNSWNlMm1KOFhBNEFOMs91Rk5icMrvSFd1Wk1tUkFTYzVubXHZFpoMU5VSkc5L2pKak5U
blV3bDIxyU5UzREczQvT31LMTBQVE9YSE80Uzh5MC8KSLNROTFcm4yeDRUc2tRcVNnSTNza3c2
Mjb1sjJlUkdyTE45Q2Ruem84TjFRZFVreXB1JfhuTkUrVldBuIsztAp1SlZma1IxSlFOSDNSOWJw
bS9vRU5vK2NEL3VEQla4dkdnPt0KLS0tFtkQg0VSVElGSUNBVEutLS0tLQo=
(kali㉿kali)-[~/Downloads]
$
```

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

The screenshot shows a web interface for decoding Base64 encoded data. On the left, there is a large text area containing the decrypted X.509 certificate content. On the right, there are several sections: 'BASE 64 DECODER' with a note about uppercase/lowercase casing, 'RESULTS FORMAT' (radio buttons for ASCII, Hexadecimal, Decimal, Octal, Binary, Integer Number, or File Download), and a 'DECRYPT BASE64' button. Below these are sections for 'BASE64 ENCODER' (with options to encode a file or text) and 'Similar pages' (links to Base32, ASCII85 Encoding, Base91 Encoding, URL Decoder, Unicode Coding, RSA Cipher, and Decode's Tools List). There is also a 'Support' section with links to Paypal, Patreon, and more.

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.

```
(kali㉿kali)-[~/Downloads]
$ openssl x509 -in cert.pem -noout -serial | awk -F '=' '{print $2}' | sed 's/.. /:&:g;s/:$/::'
```

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

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.

```
(kali㉿kali)-[~/Downloads]
$ openssl x509 -in cert.pem -noout -subject
subject=C = US, ST = California, L = SKY-CRTB-1093, O = Internet Widgit Pty Ltd, OU = LiberationHackingGroup, CN = cityinthe.cloud, emailAddress = lib8Hack@cityinthe.cloud
```

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



Scanning & Reconnaissance

3 Challenges | 14 Questions | 71.42% Completion

Identify and use the proper tools to gain intelligence about a target including its services and potential vulnerabilities.

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:

```
root@foreign-medium:/# echo $LANG  
es_ES.UTF-8  
root@foreign-medium:/#
```

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:/# grep -r -m 1 'SKY' /dev  
root@foreign-medium:/# grep -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
```

```
root@foreign-medium:/# grep -r -m 1 'SKY' /run
root@foreign-medium:/# grep -r -m 1 'SKY' /sbin
root@foreign-medium:/# grep -r -m 1 'SKY' /srv
root@foreign-medium:/# grep -r -m 1 'SKY' /tmp
root@foreign-medium:/# grep -r -m 1 'SKY' /usr
/usr/share/locale/THERE_IS_A_FLAG_HERE/flag:Flag 2 : SKY-LOCL-5397
grep: /usr/share/i18n/charmaps/CP737.qz: binary file matches
grep: /usr/share/i18n/charmaps/CP1252.qz: binary file matches
grep: /usr/lib/x86_64-linux-gnu/libunistring.so.2.2.0: binary file matches
root@foreign-medium:/# █
```

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

Correct Answer: SKY-LOCL-5397

LOG ANALYSIS



Log Analysis

3 Challenges | 21 Questions | 0% Completion

Utilize the proper tools and techniques to establish a baseline for normal operation and identify malicious activities using log files from various services.

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}'
```

```
(kali㉿kali)-[~/Downloads] $ awk '{print $1}' access.log | sort | uniq -c | awk '$1 == 41 {print "IP Address:", $2, "Number of Requests:", $1}'  
IP Address: 43.163.238.85 Number of Requests: 41
```

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
```

```
(kali㉿kali)-[~/Downloads] $ awk -F'[]]' '{print $2}' access.log | cut -d: -f1 | sort | uniq -c | sort -nr | head -1  
405 27/Mar/2024
```

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
```

```
(kali㉿kali)-[~/Downloads] $ awk -F'[]]' '{print $2}' access.log | cut -d: -f1 | sort | uniq -c | sort -nr | head -1  
405 27/Mar/2024
```

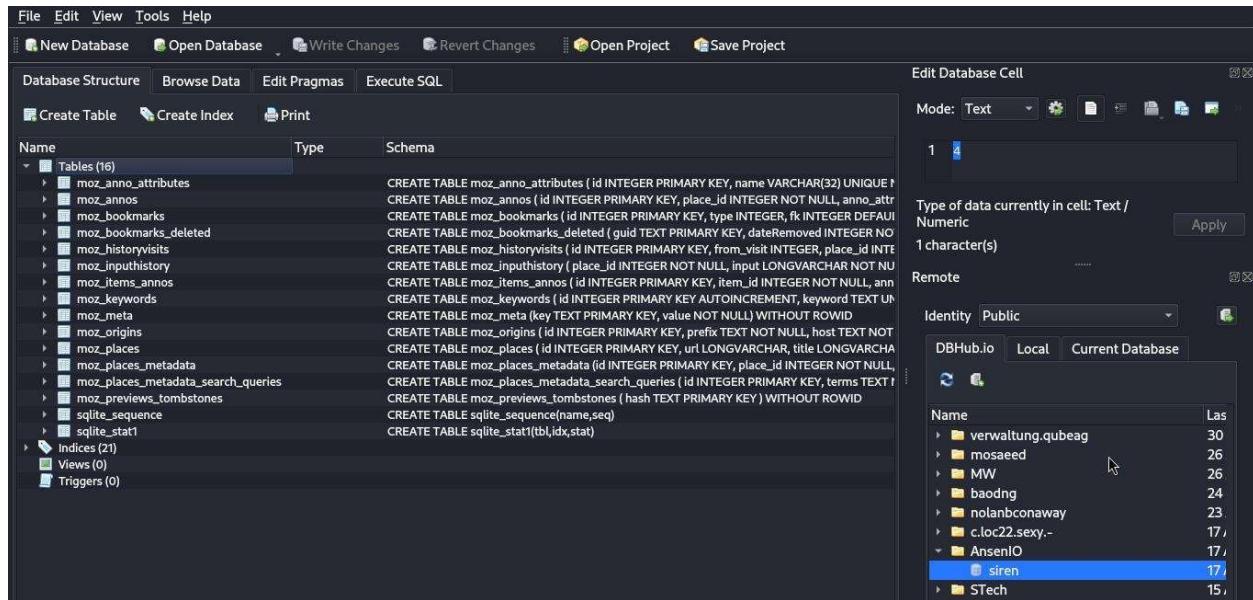
Correct Answer: 405

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/ESbjPn3bUftHkfOhWMpzWcEBeFdYxdYm8YjKOH_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.

The screenshot shows the DB Browser for SQLite interface. The title bar reads "DB Browser for SQLite - /home/kali/Downloads/places.sqlite". The menu bar includes File, Edit, View, Tools, Help, New Database, Open Database, Write Changes, Revert Changes, and Open Project. Below the menu is a toolbar with icons for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The Execute SQL tab is selected. A SQL editor window titled "SQL1" contains the following query:

```
1 SELECT
2     datetime(moz_historyvisits.visit_date / 1000000, 'unixepoch') AS visit_time_utc,
3     moz_places.url AS onion_url
4 FROM
5     moz_historyvisits
6 JOIN
7     moz_places ON moz_historyvisits.place_id = moz_places.id
8 WHERE
9     moz_places.url LIKE '%.onion'
10 ORDER BY
11     moz_historyvisits.visit_date
12 LIMIT 1;
```

The results pane shows two columns: "visit_time_utc" and "onion_url". The first row has the value "1 2024-04-01 14:24:58" under "visit_time_utc" and "http://..." under "onion_url".

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:

The screenshot shows the DB Browser for SQLite interface with a context menu open over the "onion_url" column. The menu is titled "Edit Database Cell" and includes options like Mode: Text, Apply, and a dropdown for Identity. The "onion_url" cell contains the value "http://julianumixp77ekq78byazclzyzhm0v27eprvzankdhs04cyc.onion". The results pane shows the same query and results as the previous screenshot.

Correct Answer :

<http://juhanurmihxlp77nkq76byazcldy2hlmovfu2epvl5ankdibsot4csyd.onion/>

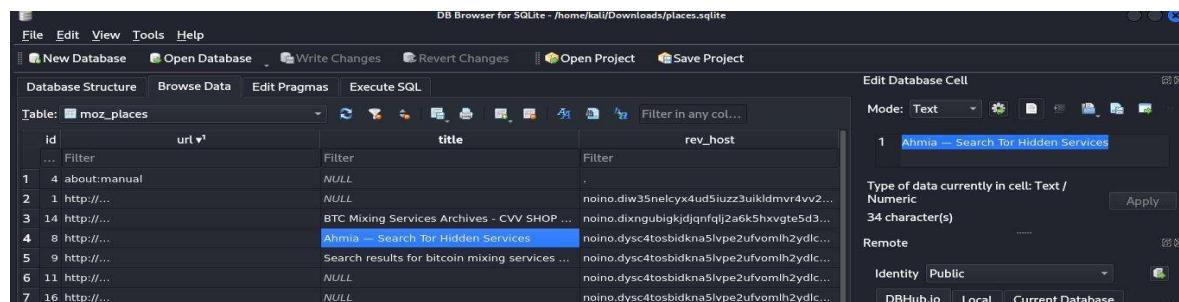
Question 3: What is the name of the .onion search engine that was used?

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 "gjobqjj7wyczqbqie" or "gjobqjj7wyczqbqie.onion" in the domain name.



id	url	title	rev_host
1	4 about:manual	Filter	Filter
2	1 http://...	NULL	noino.diw35nelcyx4ud5uzz3uikldmvr4vv2...
3	14 http://...	BTC Mixing Services Archives - CVV SHOP ...	noino.dixngubigkjdjqnqfqlj2a6k5hxvgte5d3...
4	8 http://...	Ahmia -- Search Tor Hidden Services	noino.dysc4tosbidkna5lvppe2ufvomlh2ydlc...
5	9 http://...	Search results for bitcoin mixing services ...	noino.dysc4tosbidkna5lvppe2ufvomlh2ydlc...
6	11 http://...	NULL	noino.dysc4tosbidkna5lvppe2ufvomlh2ydlc...
7	16 http://...	NULL	noino.dysc4tosbidkna5lvppe2ufvomlh2ydlc...

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/EYhSV8v1F9IKjXwy9eQit3gByJ8kDetlgYpjnzHgrnxvA?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=!0***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 '!'.
[

```
(kali㉿kali)-[~/Downloads]
$ awk -F'!`{print $1}' messages.log > users.txt

(kali㉿kali)-[~/Downloads] ChatGPT 4.5
$ cat users.txt

TWFyaw==
RXRoYw4==

SmFja3Nvbg==
TWFyaw==
RXRoYw4==
TG16YQ==
THVjeQ==
RXRoYw4==

SmFja3Nvbg== Passwords
SmFja3Nvbg==

TG16YQ==
THVjeQ==
TWFSYQ==
TG16YQ==

SmFja3Nvbg==

RXRoYw4== Vulnerability investigation
TG16YQ==

SmFja3Nvbg== Back Command
SmFja3Nvbg==

RXRoYw4== Data Breach, Compliance
TWFSYQ==

SmFja3Nvbg==

TWFyaw==
THVjeQ==
TWFSYQ==
TWFSYQ==
```

Let's clean the new file : remove inconsistencies and anomalies (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'.")
```

```
GNU nano 7.2 decode_base64.py

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() # Remove newline character
        # Decode Base64 and write to output file
        decoded_line = base64_decode(line)
        line = line.strip()
        f_out.write(decoded_line + '\n') # Decode Base64 and Write to output file

print("Base64 decoding completed. Decrypted data written to 'Base64_decrypted_users.txt'.")
```

```
(kali㉿kali)-[~/Downloads]$ nano decode_base64.py
(kali㉿kali)-[~/Downloads]$ python3 decode_base64.py
Base64 decoding completed. Decrypted data written to 'Base64_decrypted_users.txt'.
(kali㉿kali)-[~/Downloads]$ cat Base64_decrypted_users.txt
Mark
Ethan
Jackson
Mark
Ethan
Liza
Lucy
Ethan
Jackson
Jackson
Liza
Lucy
Maya
Liza
Jackson
Ethan
Liza
Jackson
Jackson
Ethan
Maya
Jackson
Mark
Lucy
Maya
Maya
Maya
Mark
```

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:

```
File Actions Edit View Help
(kali㉿kali)-[~/Downloads]
$ grep -c '^' Base64_decrypted_users.txt
201
```

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

```
(kali㉿kali)-[~/Downloads]
$ grep -o '.*' Base64_decrypted_users.txt | sort | uniq -c | wc -l
6
```

Correct Answer : 6

FORENSICS



Forensics

3 Challenges | 6 Questions | 0% Completion

Utilize the proper tools and techniques to analyze, process, recover, and/or investigate digital evidence in a computer-related incident.

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/EaVPCa9yoNBKijQGC8MAf8Bj3QChV0bl_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



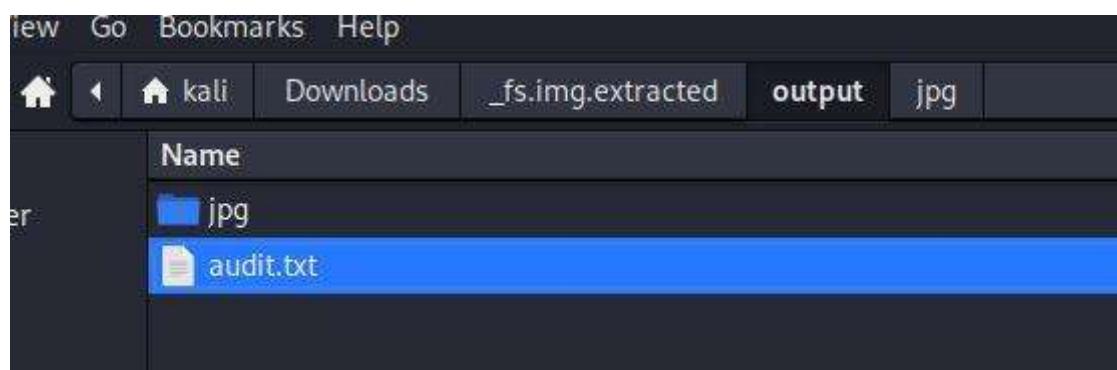
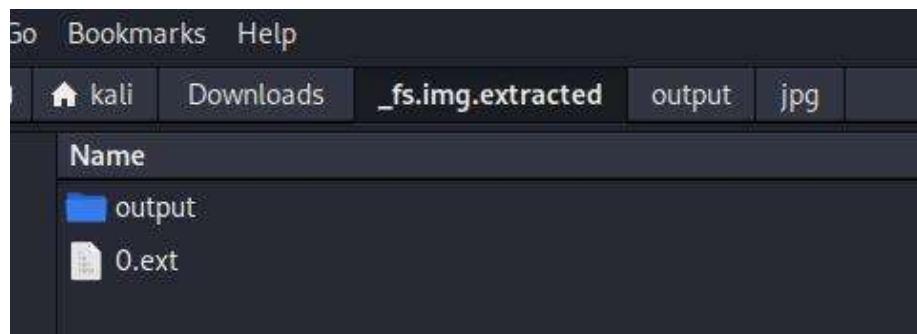
Now change directory to the new folder and look whats in there

```
(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



```
5 Invocation: foremost -i 0.ext
6 Output directory: /home/kali/Downloads/_fs.img.extracted/output
7 Configuration file: /etc/foremost.conf
8 -
9 File: 0.ext
10 Start: Sat Apr  6 19:23:54 2024
11 Length: 10 MB (10485760 bytes)
12
13 Num      Name (bs=512)      Size      File Offset      Comment
14
15 0:      00016384.jpg       19 KB      8388608
16 Finish: Sat Apr  6 19:23:54 2024
17
18 1 FILES EXTRACTED
19
20 jpg:= 1
21
22
23 Foremost finished at Sat Apr  6 19:23:54 2024
24
```

Correct Answer : jpg

Question 3: what is the flag

Answer:

Open the flag.jpg file

SKY-RCVR-8755

Correct Answer : SKY-RCVR-8755

WEB APPLICATION AND EXPLOITATION



Web Application Exploitation

3 Challenges | 7 Questions | 0% Completion

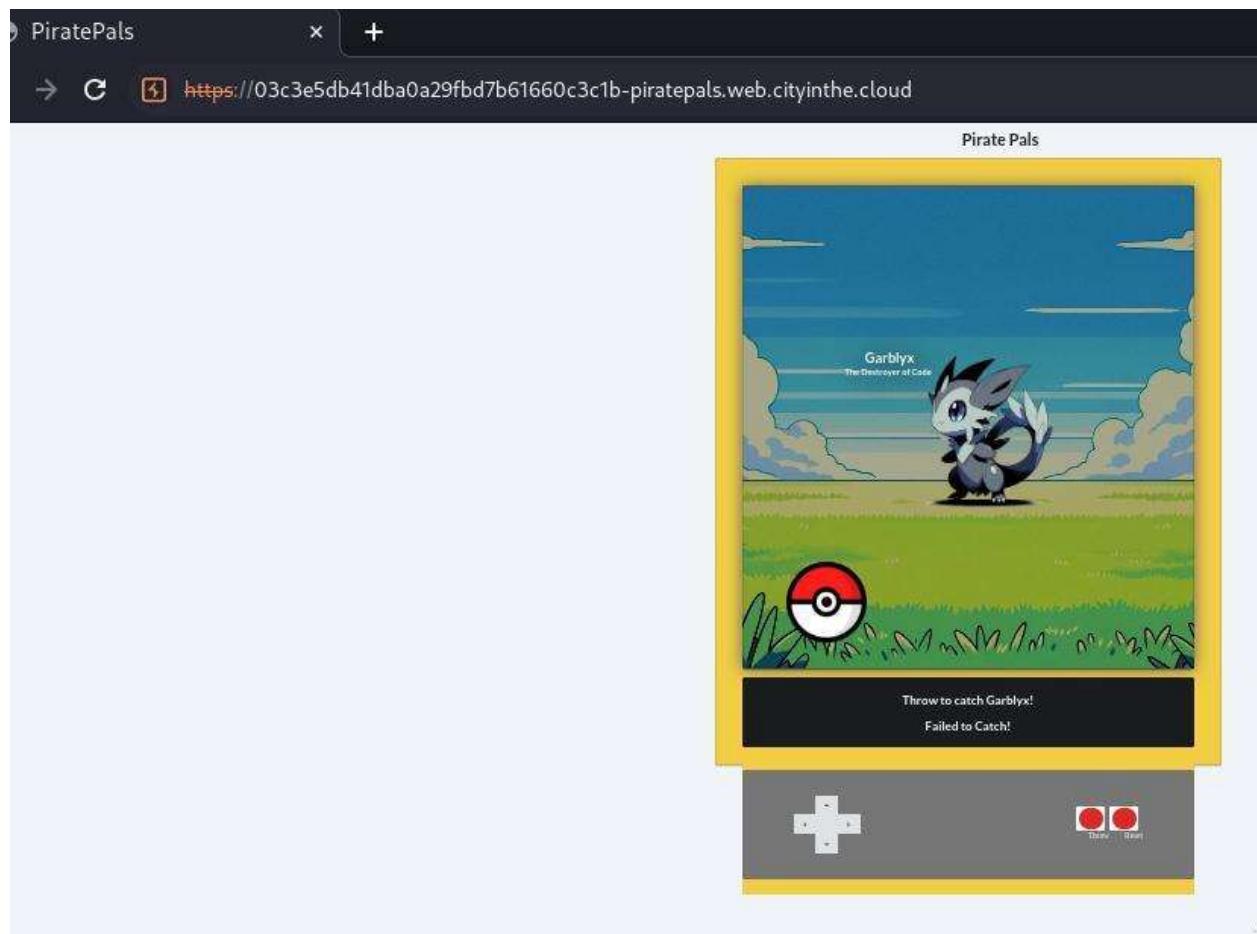
Identify actionable exploits and vulnerabilities and use them to bypass the security measures in online services.

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.

The screenshot shows a browser developer tools window with the 'Debugger' tab selected. The code editor displays a portion of the 'main.js' file:

```
49     $('#start').hide();
50     $('#failed').show();
51     $('#pokeball').hide();
52   }
53 });
54 }
55
56 function startPoll() {
57   update();
58   updateInterval = setInterval(update, 5000);
59 }
60
61 $('#throw').click(() => {
```

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.

The screenshot shows the Burp Suite interface in the Repeater tab. The request pane displays a POST request to the endpoint '/throw'. The request body is a JSON object with two fields: 'success' set to 2 and 'fail' set to 98. The response pane on the right shows a single line of text starting with '1 POST /throw HTTP/1.1'.

```
1 POST /throw HTTP/1.1
2 Host: 03c3e5db41dba0a29fb7b61660c3c1b-piratepals.web.cityinthe.cloud
3 Content-Length: 23
4 Sec-Ch-Ua: "Chromium";v="119", "Not ?A_Brand";v="24"
5 Accept: */*
6 Content-Type: application/json
7 X-Requested-With: XMLHttpRequest
8 Sec-Ch-Ua-Mobile: ?0
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
   Safari/537.36
10 Sec-Ch-Ua-Platform: "Linux"
11 Origin:
   https://03c3e5db41dba0a29fb7b61660c3c1b-piratepals.web.cityinthe.cloud
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
   https://03c3e5db41dba0a29fb7b61660c3c1b-piratepals.web.cityinthe.cloud
/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Priority: u=1, i
19 Connection: close
20
21 {
  "success":2,
  "fail":98
}
```

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.

```

1 POST /throw HTTP/1.1
2 Host: 03c3e5db41dba0a29fdb7b61660c3c1b-piratepals.web.cityinthe.cloud
3 Content-Length: 23
4 Sec-Ch-Ua: "Chromium";v="119", "Not ?A_Brand";v="24"
5 Accept: */*
6 Content-Type: application/json
7 X-Requested-With: XMLHttpRequest
8 Sec-Ch-Ua-Mobile: ?
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
  Safari/537.36
10 Sec-Ch-Ua-Platform: "Linux"
11 Origin: https://03c3e5db41dba0a29fdb7b61660c3c1b-piratepals.web.cityinthe.cloud
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://03c3e5db41dba0a29fdb7b61660c3c1b-piratepals.web.cityinthe.cloud/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US, en;q=0.9
18 Priority: u=1, i
19 Connection: close
20
21 {
  "success":98,
  "fail":2
}

```

The response tab shows the JSON response from the server:

```

1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Sun, 07 Apr 2024 20:52:13 GMT
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 69
6 Connection: close
7 X-Powered-By: Express
8 ETag: W/"45-gLNn3lnzmTLZtoA3xph8tKVHgZg"
9
10 {
  "win":true,
  "successChance":98,
  "failChance":2,
  "flag": "SKY-NVED-1749"
}

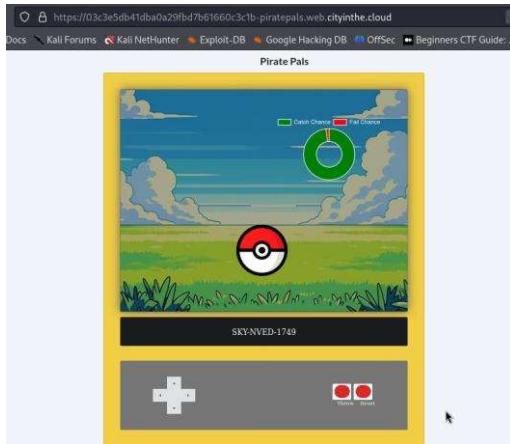
```

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

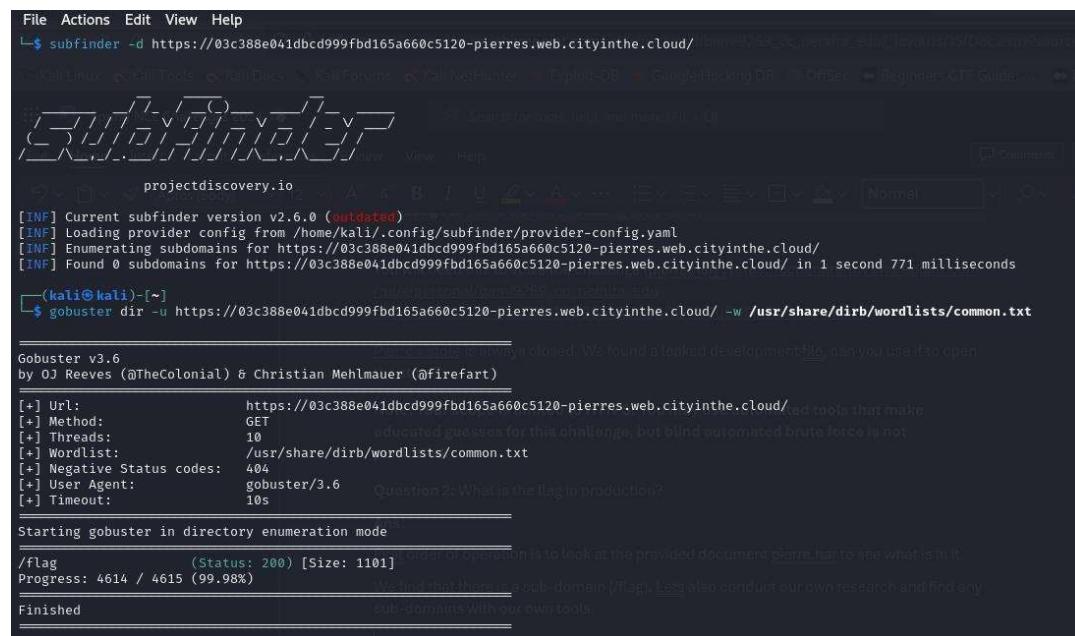
Pierre's store is always closed. We found a leaked development file, can you use it to open the store?

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.



The screenshot shows a terminal window with several command-line tool executions:

- `subfinder -d https://03c388e041dbc999fdb165a660c5120-pierres.web.cityinthe.cloud/` (Output: Found 0 subdomains)
- `gobuster dir -u https://03c388e041dbc999fdb165a660c5120-pierres.web.cityinthe.cloud/ -w /usr/share/dirb/wordlists/common.txt` (Output: /flag (Status: 200) [Size: 1101])

The terminal window has a dark background and light-colored text. It includes a navigation bar with links like "File", "Actions", "Edit", "View", "Help", "Kali Linux", "Kali Tools", "Kali Distro", "Kali Forums", "Kali Newsletter", "ExploitDB", "Google Hacking DB", "Offsec", "Beginner-GTF-Guide", and "Documentation". The bottom of the window shows a status bar with icons and the text "Normal".

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)

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.

Burp Project Intruder Repeater View Help

Dashboard Target Proxy **Repeater** Collaborator Sequencer Decoder Comparer Logger Extensions Learn

1 × 2 × 3 × +

Send Cancel < > Target: https://03c388e041dbcd999fdb165a660c

Request

Pretty Raw Hex

```

1 GET /flag HTTP/1.1
2 Host: 03c388e041dbcd999fdb165a660c5120-pierres.web.cityinthe.cloud
3 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
   Safari/537.36
8 Sec-Purpose: prefetch/prerender
9 Purpose: prefetch
10 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
   webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11 Sec-Fetch-Site: none
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-User: ?1
14 Sec-Fetch-Dest: document
15 Accept-Encoding: gzip, deflate, br
16 Accept-Language: en-US,en;q=0.9
17 Priority: u=0, i
18 Connection: close
19 Content-Type: application/x-www-form-urlencoded
20 Content-Length: 0
21
22

```

Response

Pretty Raw Hex Render

The best of fruits

You don't have Pierre's special key

Pierre's General Store Challenge | © 2024 Cyber Skyline

Burp Project Intruder Repeater View Help

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer

1 × 2 × 3 × +

Send Cancel < >

Request

Pretty Raw Hex

```

1 GET /flag HTTP/1.1
2 Host: 03c388e041dbcd999fdb165a660c5120-pierres.web.cityinthe.cloud
3 Cache-Control: max-age=0
4 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159
   Safari/537.36
9 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
   webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Priority: u=0, i
17 Connection: keep-alive
18 Cookie: session=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.sVXWzDwvLJLkPjyfCzHxOOGdRrBrrr
19
20

```

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

Burp Suite Community Edition v2023.10.3.6 - Temporary Project

Request

Target: <https://03c388e041dbcd999fb165a660c5120-pierres.web.cityinthe.cloud>

Response

The best of fruits

SKY-WWOJ-1315

Pierre's Genral Store Challenge | © 2024 Cyber Skyline

Correct Answer : SKY-WWOJ-1315

Challenge 3: Valley Directory

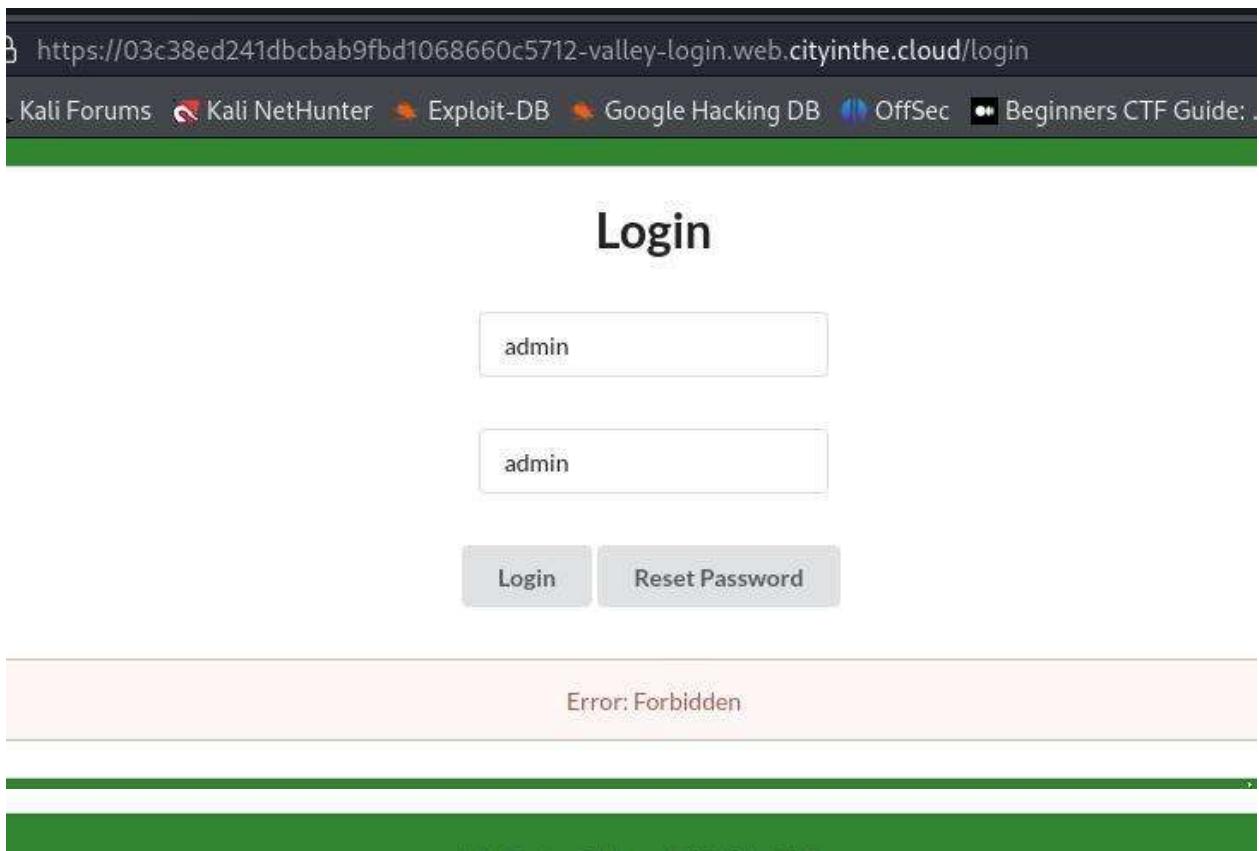
We received an anonymous report that there is a security vulnerability on our [admin account login](#). Can you conduct a security evaluation to find the vulnerability?

Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Beginners CTF Guide: ... D

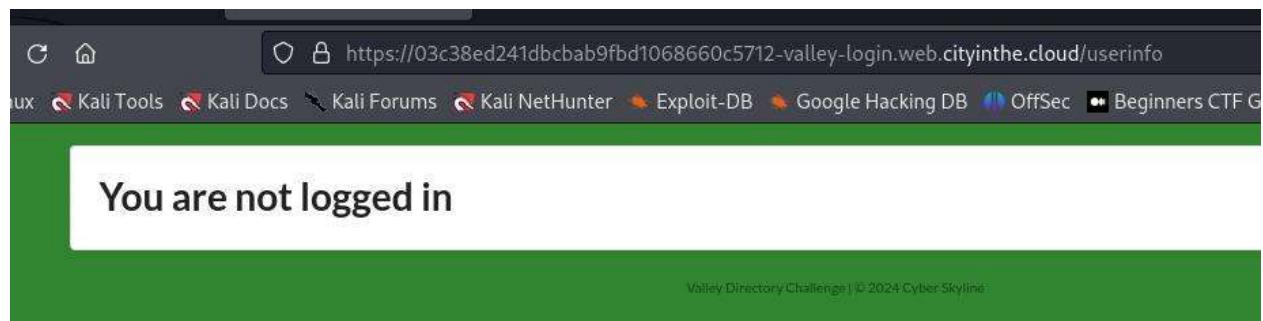


Lets run a command line tool against the url

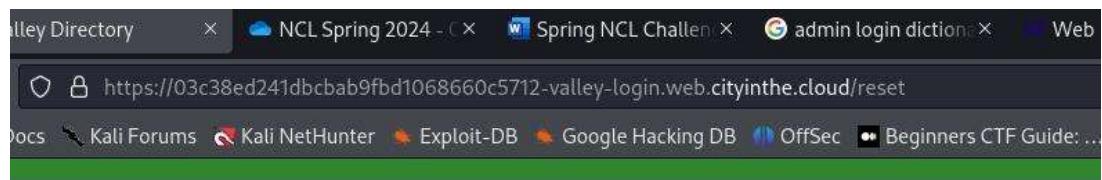
We can see that we have found some subdomains. Just for simple purposes lets just test our web application and see how it works.



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



/userinfo apparently is for after we login



Login

admin

Send Reset Email

password reset email sent

Valley Directory Challenge | © 2024 Cyber Skyline

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.

