Kludge Core Selections 2025

EE24BTECH11005 : Arjun Pavanje

May 25, 2025

Writeup

Bachelor of Technology

1 Miscellaneous

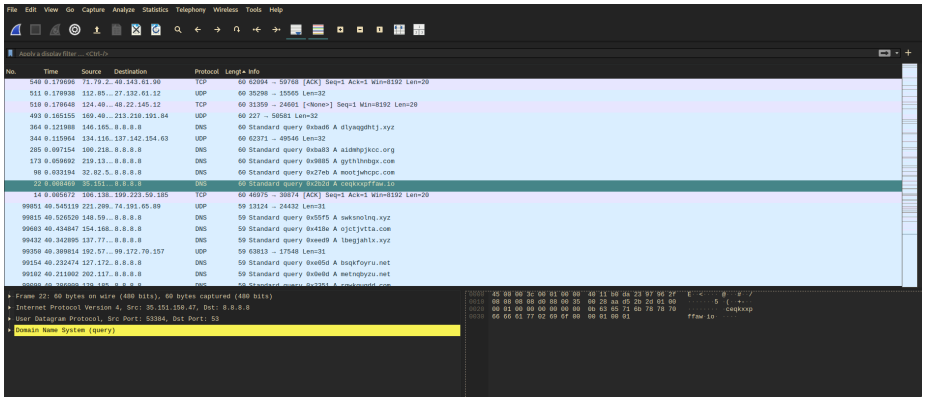
1.1 I am not MID

Tried the flag given in the question, and it worked :)

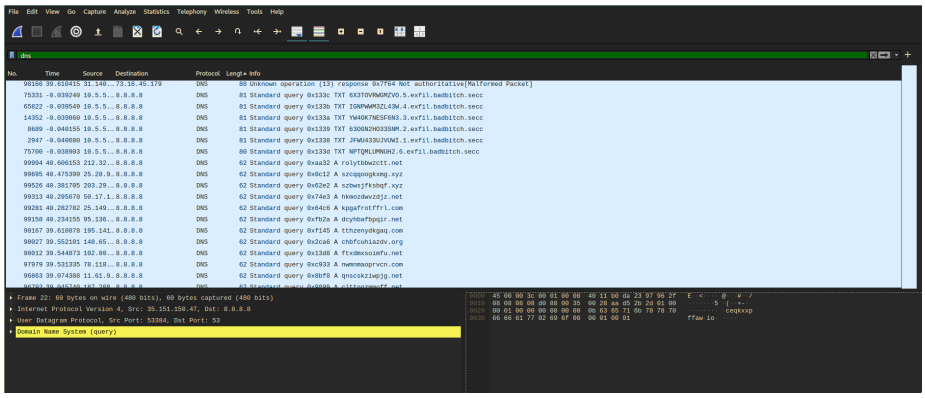
2 Forensics

2.1 Chatty Network

We were given a packet capture file with a suspicion that malware maybe stealing data during look ups.
Analysing using Wireshark,,,



Viewing all communication made using DNS protocol,



On sorting messages in descending order (size), we see 6 messages to the slightly suspicious domain *badbitch.secc*. To my observation, those 6 messages were the

only ones that had the same source and destination location. So I guessed that the message may have been split into 6 parts, so I took the text from each image and tried to decode them.

String obtained, $FWU433UJVUWI63OGN2HO33SNMYW4OK7NESF6$
$N3IGNPWWM3ZL43W6X3TOVRWGMZVONPTQMLUMNUH2$

This looked to be base-32 code, so I wrote a python code to decode it by making use of pythons *base*64 library. Python code,

```
import base64
# The three '=='s have been added as padding
encoded = "JFWU433UJVUWI63OGN2HO33SNMYW4OK7NESF6
N3IGNPWWM3ZL43W6X3TOVRWGMZVONPTQMLUMNUH2==="
decoded = base64.b32decode(encoded)
print(decoded)
```

This gives us the flag,

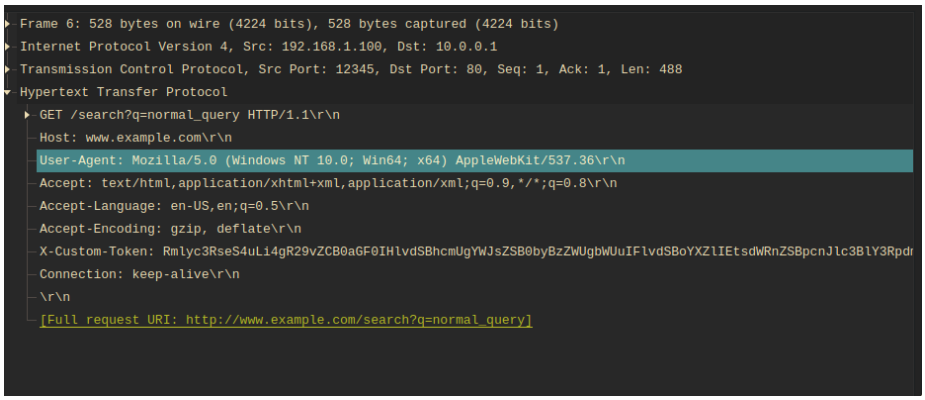
`ImNotMid{n3twork1n9_i$_7h3_k3y_7o_succ35s_81tch}`

2.2 Sniffer

I found the three parts of interest,

No.	Time	Source	Destination	Protocol	Length	Info
2	0.342366	16.16.5…	39.165.40.13	HTTP	127	GET / HTTP/1.1
6	2.031567	192.168…	10.0.0.1	HTTP	528	GET /search?q=normal_query HTTP/1.1
8	5.025936	172.16…	8.8.8.8	HTTP	288	GET /api/data HTTP/1.1
15	8.971056	229.67.…	11.66.187.240	HTTP	124	GET / HTTP/1.1
16	9.323643	33.21.7…	222.123.86.149	HTTP	124	GET / HTTP/1.1
19	9.744027	188.228…	18.26.216.58	HTTP	125	GET / HTTP/1.1
26	14.345860	10.10.1…	203.0.113.42	HTTP	386	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
29	15.593987	31.251.…	32.230.151.110	HTTP	125	GET / HTTP/1.1
32	19.956622	82.189.…	237.98.29.35	HTTP	127	GET / HTTP/1.1
34	20.433169	127.153…	38.1.165.144	HTTP	126	GET / HTTP/1.1
37	23.775562	248.75.…	181.50.249.164	HTTP	127	GET / HTTP/1.1
42	31.420079	106.67.…	19.133.7.47	HTTP	124	GET / HTTP/1.1
47	35.906528	102.219…	43.114.225.83	HTTP	125	GET / HTTP/1.1
52	46.056903	195.246…	161.184.94.89	HTTP	124	GET / HTTP/1.1
64	63.190671	204.46.…	224.150.48.189	HTTP	124	GET / HTTP/1.1
70	72.446951	144.5.2…	46.215.205.197	HTTP	124	GET / HTTP/1.1
77	82.086116	215.221…	245.47.178.118	HTTP	127	GET / HTTP/1.1
78	82.356154	90.6.13…	190.89.249.129	HTTP	127	GET / HTTP/1.1
80	84.565764	254.190…	9.169.64.235	HTTP	124	GET / HTTP/1.1

On analysing the first part, we find a message (aHR0cHM6Ly93d3cuYW5vbmZpbGUu
GEvMjM5ZGY1) which is in base64. On decoding we get the url `https://www.anonfile.la/239df5` from where we get a zip file which has the flag but is password locked. On analyzing the second part we get,

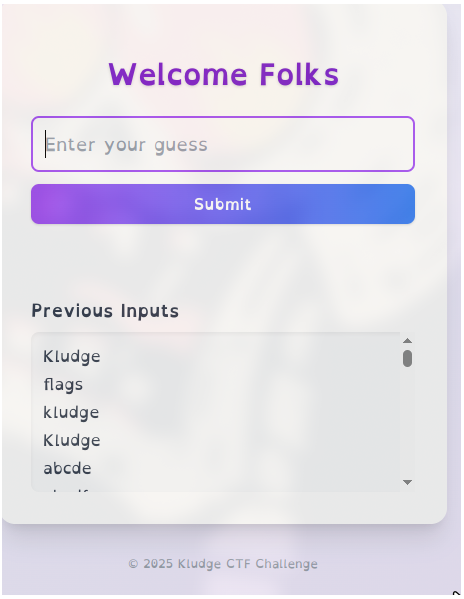


On decoding the X-Custom-Token (base64 encoded) we get, "Firstly.... Good that you are able to see me. You have Kludge irrespective of your selection. Now go search more with –SUSANO–".

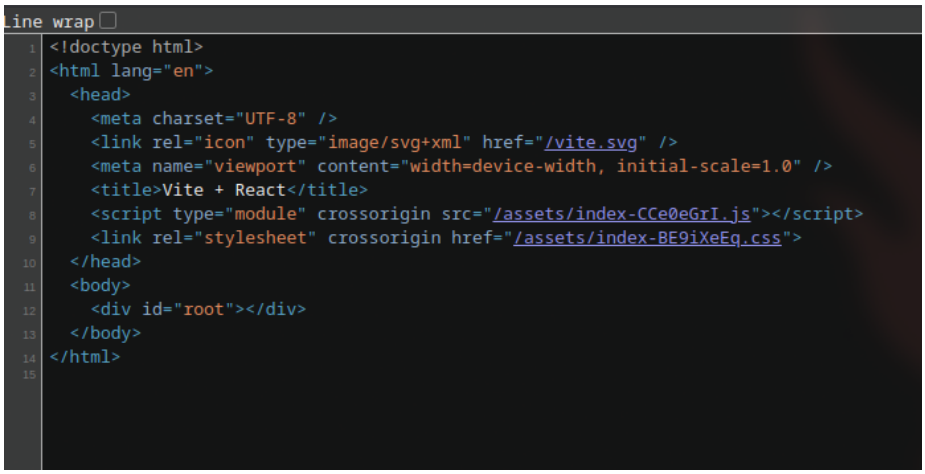
3 Cryptography

3.1 Wordle

Opened the website `https://core-ctf.vercel.app/`,



Initially I thought that it would be a substitution or rot cypher, but after a long while of guessing words I realized that wasn't the case. I tried to find the individual number of each alphabet but realized that the number of an alphabet would be constant only for a given word size (for example the number corresponding to 'A' in PLANT and SAD would be different, but would be the same in case of 'PLANT' and 'ABCDE'). Then I abandoned such ideas and tried to check the source code by pressing $ctrl + U$,



Then went to `https://core-ctf.vercel.app/assets/index-CCe0eGrI.js` (after realizing that there was nothing on the second link), where on searching for flag (using $ctrl + F$ I got the flag

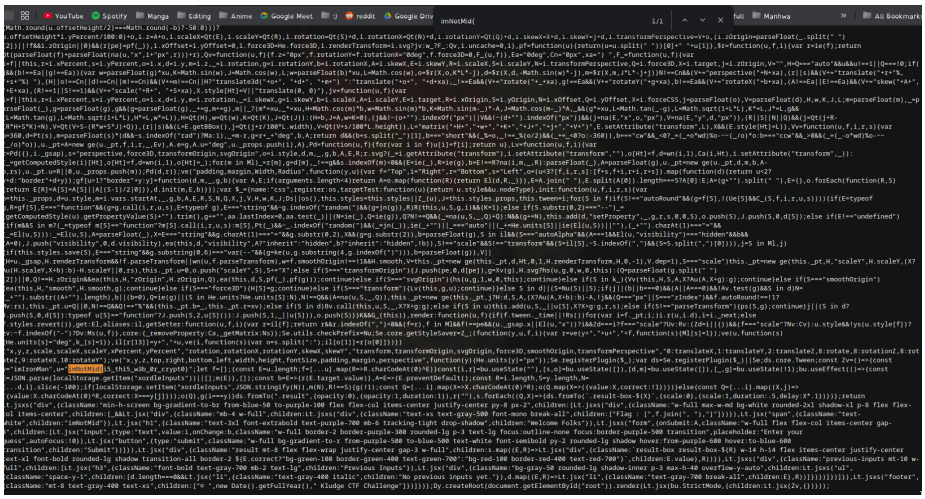
`imNotMid{i5\_thi5\_w3b\_0r\_crypt0}`

3.2 Crypto Misstep

Here, we are given two values of N used in RSA and the standard $e = 65537$ and the cypher text. We are required to obtain the plaintext flag. Usually, it would be extremely difficult to obtain the private key, (which is given by the relation $e * d \equiv 1 mod \varphi(n)$, whre φ is Euler Totient function) due N being an extremely large prime number (hence it is extremely difficult to calculate two prime numbers p, q which satisfy $p * q = N$ ($\varphi(N)$ is given be $(p-1)(q-1)$).

In this case, we have two N values (N_1, N_2) so if they have a GCD we have found p, q for N_1 and N_2, using which we can calculate Euler Totient function, using which we can calculate private key d.

Python code,



```python
from Crypto.Util.number import inverse, long_to_bytes
from math import gcd

N1 = 14974374473875195262766359759714498503053089849989142350615906604375075459239382918
114868001552711501100264166512783155193243369424875166451614237308755177787627661679153
407311577363068061643400744101086494410390726961463235329017993736316198284311340752922

N2 = 11377303457963693647634556693776352306798805099525254752031901804678648636881088711
926477688508021301745074012078147126558540520284378771368173461658671664570694525146838
466511586254342858049935950702182387012147757215640939517729687181123762690670586402035

e = 65537
c = 50620835015367606193759766262236780012280328339774890594865485759145875724074933467
583037165585878007146816194270009178435075843990690783561214662462610032711751223235548
884959208420406373053914635940047237068268327521369299948583610143571896073590221095836

p = gcd(N1, N2)
if p == 1:
    print("N1 and N2 are coprime")
    exit()

q = N1 // p  # Floor division
phi = (p - 1) * (q - 1) # Euler Totient Function

d = inverse(e, phi) # Private Key

m = pow(c, d, N1) # Calculating coded message (using cypher text, N, private key)
plaintext = long_to_bytes(m) # decoding coded message

print(plaintext)
```

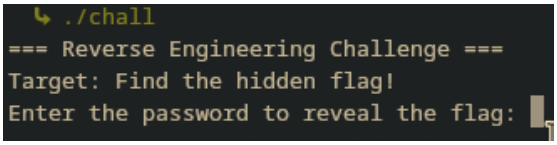
On running the code we get the flag,

`ImNotMid{r54_!s_n0t_50_c00l_4nym0r3_n1994}`

4 Reverse Engineering

4.1 JJK

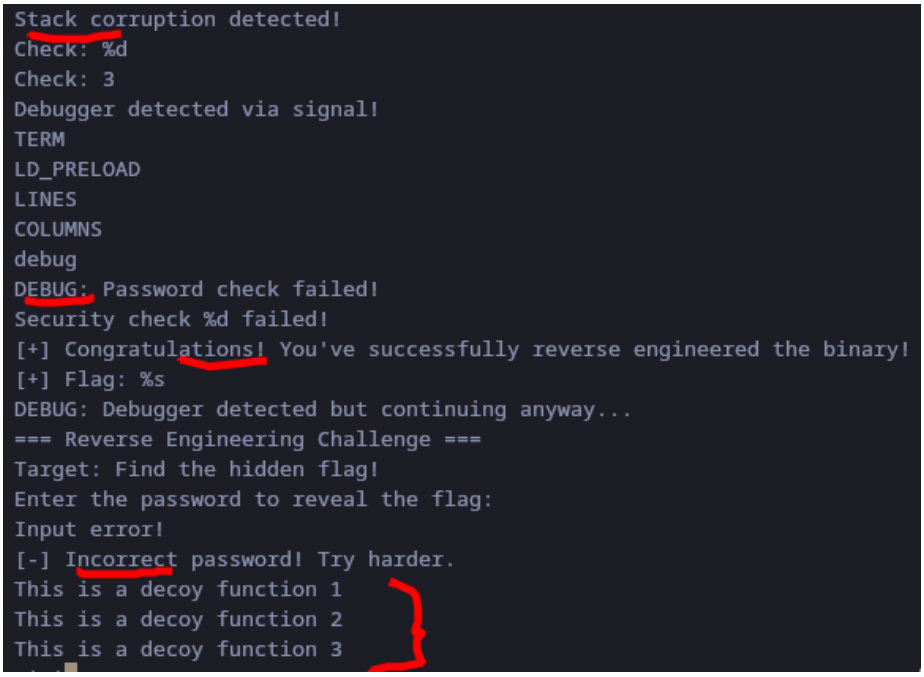
We are given only a binary executable, so on running it we get,



Also on running the command

`strings chall > jjk.txt`

we can observe a few lines of interest,



So we can infer that we will obtaiin the flag upon entering the right password. Since we are given nothing else, we must obtain the passphrase from the binary executable. Firstly, I disassembled the binary executable into assembly code using the command,

```
objdump -D chall > chall.asm
```

We can analyze it in even more detail using a tool like Ghidra (which even gives us the c-code behind the assembly function). We can tell on analyzing the assembly code that there mainly a few functions of interest,

- main
- verify_password: returns 1 if input matches password
- compare: compares user input and decoded obfuscated key
- decoded_string: obfuscated key is encoded

We observe that obfuscated key is held at the memory location 00104080 and holds the value 25 2c 2e 26 20 28 7c 79 7e 00 00 00 00 00 00 00. Encoding scheme is to XOR each 4-bit hexadecimal number with 0$x4d$ i.e. 77 (in decimal). On decoding we get the passphrase to be, "hackme143".

```
  ↳ ./chall
=== Reverse Engineering Challenge ===
Target: Find the hidden flag!
Enter the password to reveal the flag: hackme143

[+] Congratulations! You've successfully reverse engineered the binary!
[+] Flag: imNotMid{i_4l0n3_4m_th3_h0n0ur3d_0n3}M
```

Figure 1: "Throughout Heaven and Earth, I Alone Am The Honored One", Satoru Gojo

5 Stegnography

5.1 99.9 % truth

After heeding the hint of "Queen" and "rockyou", I figured that the password to unlock the text hidden in the image (when steghide is used) would be a part of "rockyou.txt" which contains a list of commonly used passwords. When I tried running stegseek (with rockyou.txt), I got nothing.

5.2 OSINT

5.2.1 Dora

We were given an image, which has text hidden in it. On running the "file" command on the image,

```
file chall.jpg
```

We get, "chall.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, comment: "imNotM", baseline, precision 8, 800x600, components 3 ".
We see that entering "imNotM" as a passphrase to (steghide extract -sf chall.jpg) works.

We get a textfile (secret.txt) with one part of the flag and a pastebin link. On clicking on the link, I recieved another part of the flag and another link from where I recieved a zip file. On unzipping I found a README file with two lines of poetry and a folder called _neverTEL_ which yielded nothing. I was unable to continue from there. I got parts of the flag, " imNotMid051nt_1 " and a file _neverTEL_ and a README and no idea what to do after.
All codes and files used can be found here,
`https://github.com/ArjunPavanje/Kludge2025_CoreSel`