KludgeCTF

Arjun Pavanje

May 2025

1 Miscellaneous

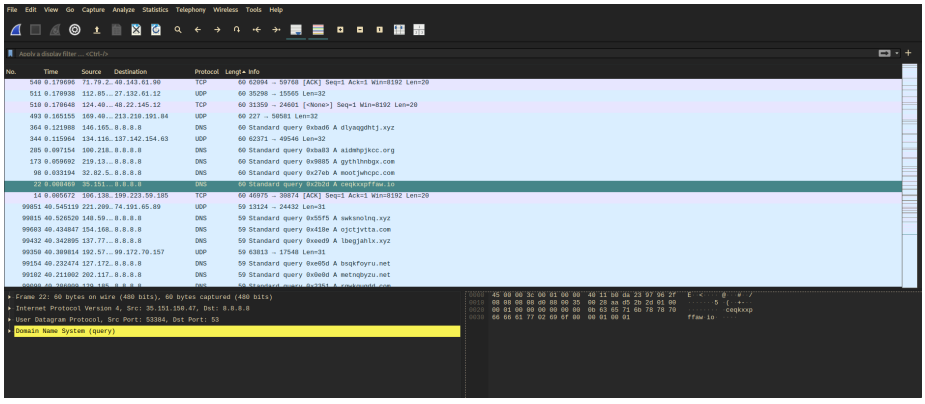
1.1 I am not MID

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

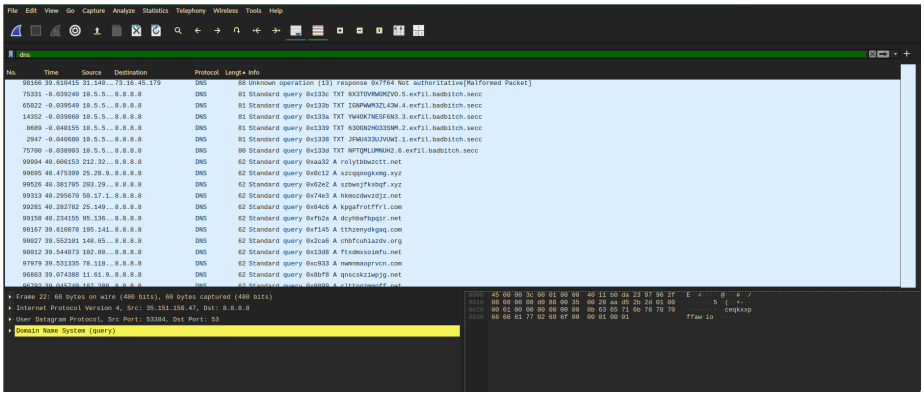
2 Forensics

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, $FWU433UJVUWI63OGN2HO33SNMYW4OK7NESF6N3IGNPWWM3ZL43W6X3TOVRWGMZVONPTQMLUMNUH2$

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)
```

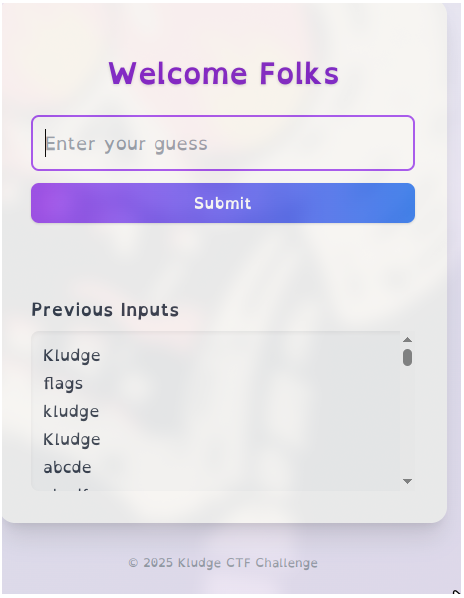
This gives us the flag,

```
ImNotMid{n3twork1n9_i$_7h3_k3y_7o_succ35s_81tch}
```

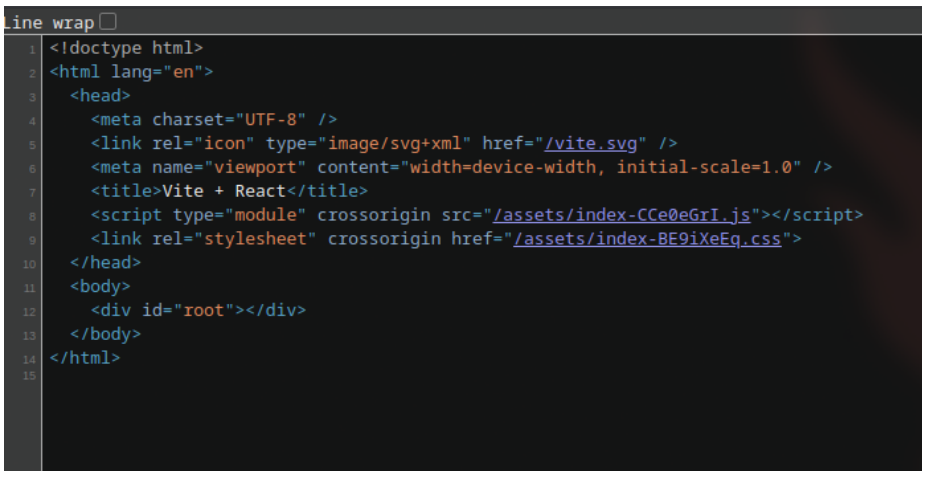
3 Cryptography

3.1 Wordle

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



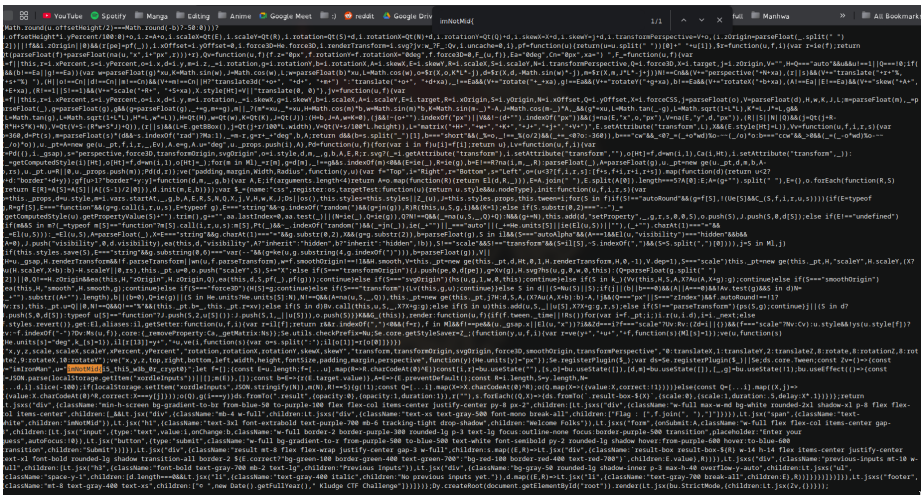
Initially I thought that it would be a substitution or rot cypher, but after a long while of guessing words I realized that wasn't the case. I tried to find the individual number of each alphabet but realized that the number of an alphabet would be constant only for a given word size (for example the 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` (af-

ter 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
11486800155271150110026416651278315519324336942487516645161423730875517778762766167915390
40731157736306806164340074410108649441039072696146323532901799373631619828431134075229224

N2 = 11377303457963693647634556693776352306798805099525254752031901804678648636881088711
92647768850802130174507401207814712655854052028437877136817346165867166457069452514683868
46651158625434285804993595070218238701214775721564093951772968718112376269067058640203591

e = 65537
c = 50620835015367606193759766262236780012280328339774890594865485759145875724074933467417
58303716558587800714681619427000917843507584399069078356121466246261003271175122323554850
88495920842040637305391463594004723706826832752136929994858361014357189607359022109583684

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

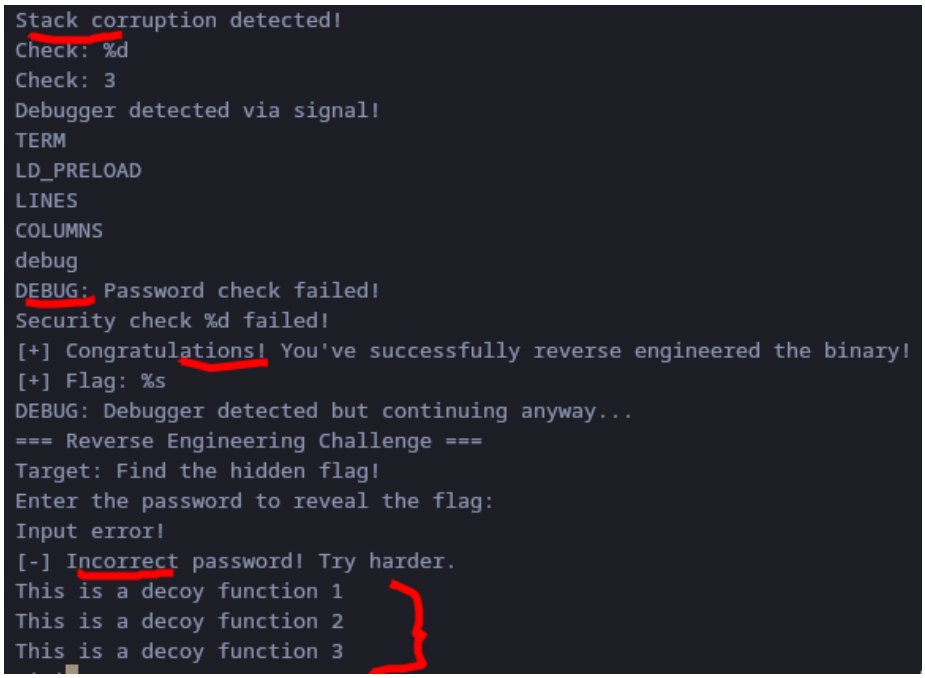
We are given only a binary executable, so on running it we get,

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

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".

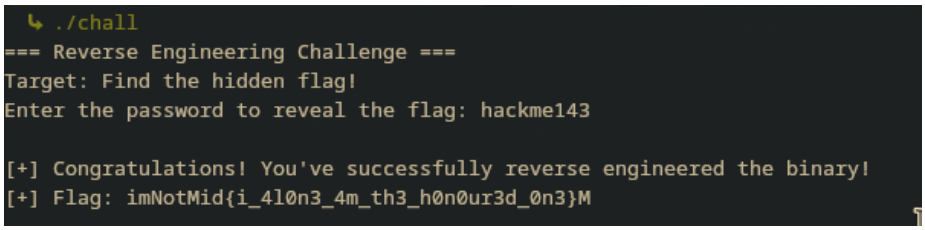


Figure 1: "Throughout Heaven and Earth, I Alone Am The Honored One", Kit-Kat