Terlantarkan

Bigby

DarkAngel

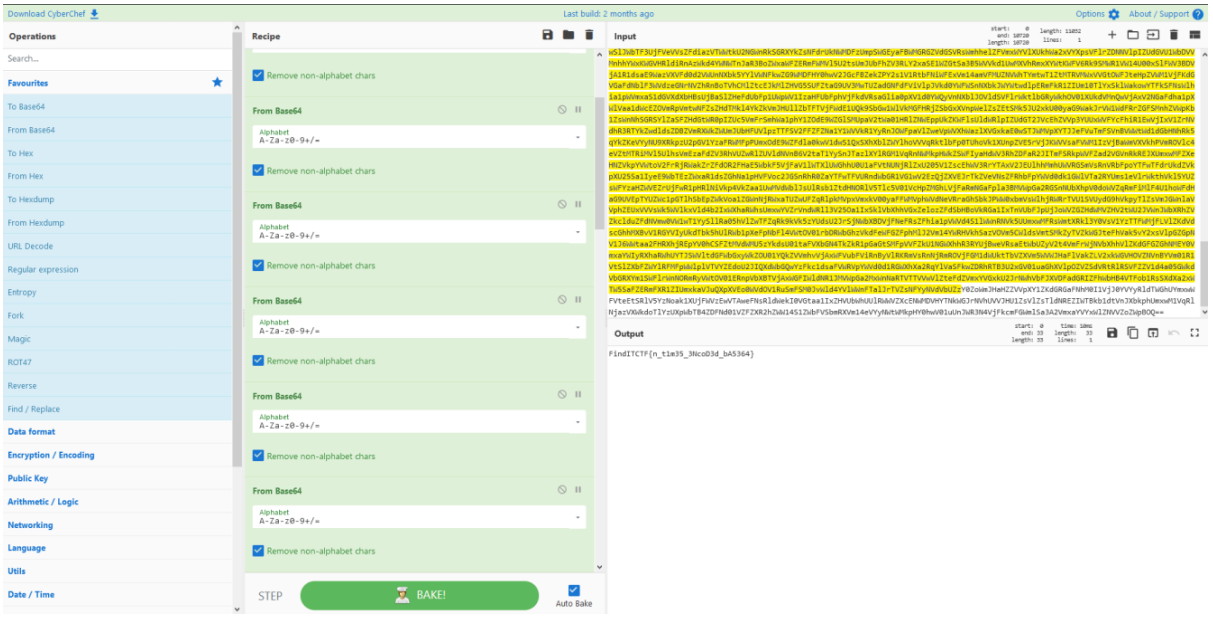
EternalBeats

Daftar Isi

Cryptography

64

Langkah Penyelesaian:



Just base64 a bunch of times

Flag: FindITCTF{n_t1m35_3NcoD3d_bA5364}

Solvable Random

Langkah Penyelesaian:

```
>>> import random,string
>>>
>>> decode_this = "MmypTSPBJ{q6k_e1s_q1Ld8I}"
>>> flag = ""
>>> random.seed("FINDIT")
>>> for c in decode_this:
...     if c.islower():
...         flag += chr((ord(c)-ord('a')-random.randrange(0,26))%26 + ord('a'))
...     elif c.isupper():
...         flag += chr((ord(c)-ord('A')-random.randrange(0,26))%26 + ord('A'))
...     elif c.isdigit():
...         flag += chr((ord(c)-ord('0')-random.randrange(0,10))%10 + ord('0'))
...     else:
...         flag += c
...
>>> print(flag)
FindITCTF{n0t_t0o_r4Nd0M}
```

Puter + random.randrange jadi - random.randrange done...

Code :

```
import random,string

decode_this = "MmypTSPBJ{q6k_e1s_q1Ld8I}"
flag = ""
random.seed("FINDIT")
for c in decode_this:
    if c.islower():
        flag += chr((ord(c)-ord('a')-random.randrange(0,26))%26 +
ord('a'))
    elif c.isupper():
        flag += chr((ord(c)-ord('A')-random.randrange(0,26))%26 +
ord('A'))
    elif c.isdigit():
        flag += chr((ord(c)-ord('0')-random.randrange(0,10))%10 +
ord('0'))
    else:
        flag += c

print(flag)
```

Flag: FindITCTF{n0t_t0o_r4Nd0M}

Baby sHA

Langkah Penyelesaian:

```
import hashlib

FLAG = 'FindITCTF{REDACTED}'

now = ""
ct = []

for c in FLAG:
    now += c
    ct.append(
        int(hashlib.sha512(now.encode()).hexdigest(), 16)>>256
    )

print(f"ct = {ct}")
```

Jadi ini menggunakan current message trus di sha512, current messagenya (now) itu mengambil dari panjang flag yang sesuai dengan iterasi loopingnya… Ini brute saja.

Code :

```
import hashlib
import string

charset = string.printable

now = ""
ct =
[9504009141604242246610997920624537785157996848964987635001850459043234110432 9, ...,
1139026499697285600010097970921716719013340856358207636014099972551420458150 59]
index = 0
while index != (len(ct)-1):
    for c in charset:
        tmp = now + c
        if int(hashlib.sha512(tmp.encode()).hexdigest(), 16)>>256 == ct[index]:
            now += c
            print(f"{index} : {now}")
            index += 1

    print(now)
```

Flag: FindITCTF{BAbY_S3cUr3_H4sh_4l90r17HmZ_256__}

Baby RSA

Langkah Penyelesaian:

```python
from Crypto.Util.number import getPrime, getRandomRange

FLAG = 'FindITCTF{REDACTED}'

p = getPrime(1024)
q = getPrime(1024)

N = p*q
e = 0x10001

N_masked = list(str(N))
for _ in range(6):
    N_masked[getRandomRange(0, len(N_masked))] = 'x'
N_masked = "".join(N_masked)

print(f"N_masked = {N_masked}")

C = [pow(ord(c), e, N)>>1 for c in FLAG]
print(f"C = {C}")
```

nilai N nya di masked… trus value masing-masing karakter di encrypt. Ini kita brute untuk value maskingnya, trus tinggal buat dictionary dapetin encrypted value -> char. tinggal di replace deh…

notes : '-' penyambung flagnya tidak sesuai ascii biasa… saya mengambilnya dari wikinya RSA dan benar :/

Code :

```
import re
import string

N_masked =
"21022850466621037769708677576925994323799557100753480004047244311247904448349530332497012095272352769059840683329533206682874110x28912684596936635262712109513369765379261556427785396808537445818749727001135701899152492274068256090927696842343223052502277341073262675904500340445384451485408905905216810610540830x708027127649766972774083229432233432553633811154328458408459412533053240161392440194546999106937853759994815436357300847907531247536803547215343156161582978526040439387197830x73397x464422421450886030601813580335318837037530x33743304492768817641596769191204027112388668915570x6041351548154537140694
```

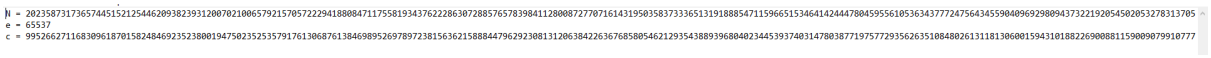
```
7204366503"
C =
[24301980541718133564685704192002712194490765766789773307021553846164229665364399841496527879043477862519917347609920634635501390878515267842242598236078864645688112788610606926296244175778361808539270376332928062154477794906730095965969335651737075001150863849529259647081160156442623747672276278070099029365138641776022278089223813428036705917070304223502871149606833719885150087877691166239028202707081425246506082870157913644918283423118729882296712750522382804481629234958684286689276340755446935712343789582288199905375667153496202580066249252990640033601578153579725815808013691712312351228022778304829009182 19, …,
11447850540397601916239520035236026462443137274484428891193784463383494128992898243041116549269662300597419232227954218542088003008471719574971887225604651622708783539151897229625678041003513329179729155320991076728981009741012451350280177472811786767937496034461831394553004332170425517269922623303642963471199690076482403260720568643420584801847118101024118886263527582326849783878238499560957922358946890753965686677760000038584784061247860647488262223846145065624794111519011465271363133727623167473836969431245744089792817691999740673940721580451162681899705691562496606806802110558565658136595612436911192078154]

maskedIndex = [m.start() for m in re.finditer('x', N_masked)]
e = 0x10001
known = "FindITCTF"

tmp = list(N_masked)
# for i in range(10):
#     for j in range(10):
#         for k in range(10):
#             for l in range(10):
#                 for m in range(10):
#                     for n in range(10):
#                         print(f"Trying : {i} {j} {k} {l} {m} {n}")
#                         # Trying : 1 3 3 8 6 1
#                         tmp[maskedIndex[0]] = str(i)
#                         tmp[maskedIndex[1]] = str(j)
#                         tmp[maskedIndex[2]] = str(k)
#                         tmp[maskedIndex[3]] = str(l)
#                         tmp[maskedIndex[4]] = str(m)
#                         tmp[maskedIndex[5]] = str(n)
#                         N = int("".join(tmp))
#                         # for a in range(known):
#                         c = [pow(ord(c), e, N)>>1 for c in known]
#                         for a in range(len(known)):
#                             if c[a] != C[a]:
#                                 break
#                             if a == (len(known)-1):
#                                 print(N)
```

```
#                                exit()

N =
210228504666210377697086775769259943237995571007534800040472443112479044483495303324970120952723527690598406833295332066828741101289126845969366352627121095133697653792615564277853968085374458187497270011357018991524922740682560909276968423432230525022773410732626759045003404453844514854089059052168106105408307370802712764976697277408322943223343255363381115432845840845941253305324016139244019454699910693785375999481543635730084790753124753680354721534315616158297852604043938719783373397846442242145088603060181358033531883703753633743304492768817641596769191204027112388668915571604135154815453714069472043665 03
charset = string.printable
convertDict = {}
for c in range(256):
    convertDict[pow(c, e, N)>>1] = chr(c)

for i in C:
    print(convertDict[i], end="")
    # FindITCTF{R1v357-5H4M1r-4dl3M4n_bF}
```

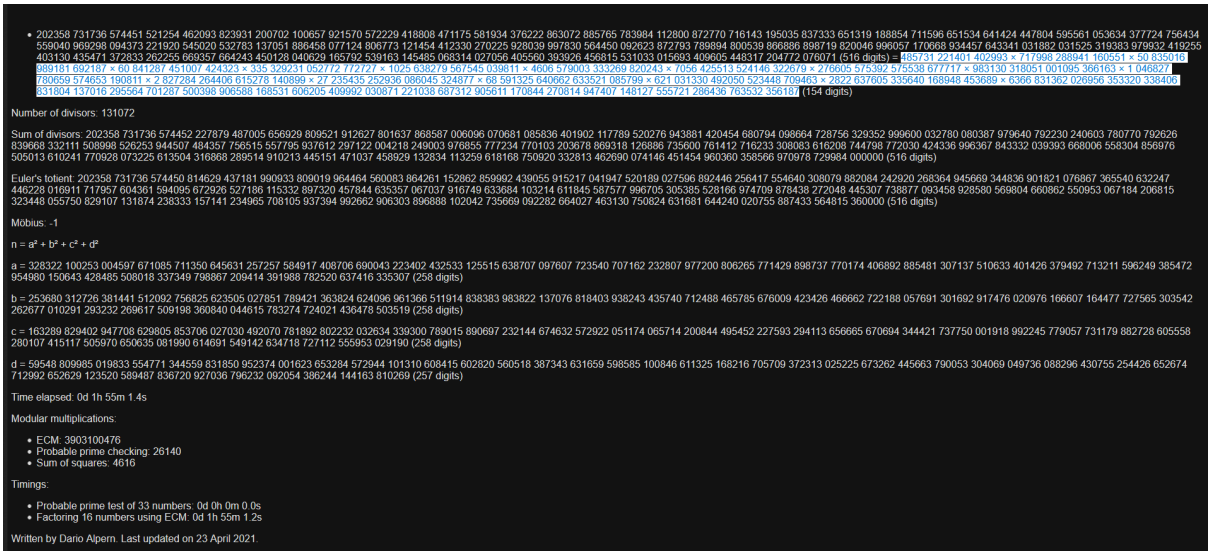
Flag: FindITCTF{R1v357–5H4M1r–4dl3M4n_bF}

Book RSA

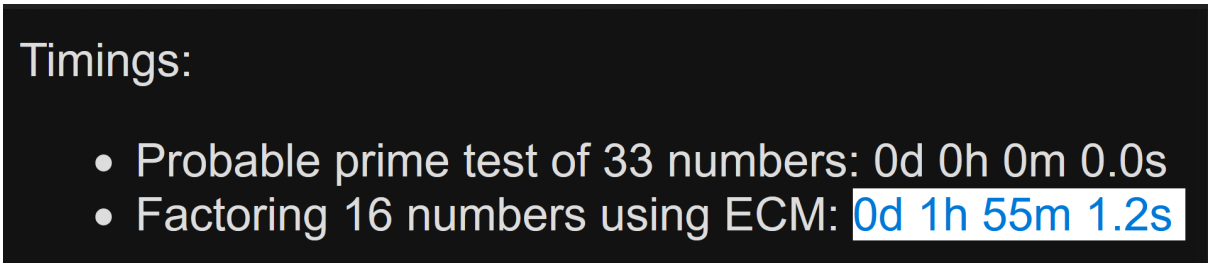
Langkah Penyelesaian:



Tidak diberikan hal lain selain public key RSA dan ciphernya… menggunakan beberapa factoring technique ga ketemu… lempar ke alpertron buat background factoring… ternyata dapat…



Dari sini tinggal pake multiprime selesai…



:)

Code :

```
from Crypto.Util.number import inverse, long_to_bytes
primes = [485731221401402993, 717998288941160551, 50835016989181692187,
608412874510074243 23, 335329231052772772727, 102563827956754503 9811,
460657900333326982 0243, 705642551352414632 2679, 276605575392575538677717,
983130318051001095366163, 104682778065957465319 0811,
282728426440661527 8140899, 272354352529360860453248 77,
685913256406626335210857 99, 621031330492050523448709463,
282263760533564016894845 3689,
6366831362026956353320338406831804137016295564701287500398906588168531606205
409992030871221038687312905611170844270814947407148127555721286436763532356187]

phi = 1
for p in primes:
```

```
    phi *= (p-1)

N =
202358731736574451521254462093823931200702100657921570572229418808471175581934376222863072885765783984112800872770716143195035837333651319188854711596651534641424447804595561053634377724756434559040969298094373221920545020532783137051886458077124806773121454412330270225928039997830564450092623872793789894800539866886898719820046996057170668934457643341031882031525319383979932419255403130435471372833262255669357664243450128040629165792539163145485068314027056405560393926456815531033015693409605448317204772076071
e = 65537
c =
995266271168309618701582484692352380019475023525357917613068761384698952697897238156362158884479629230813120638422636768580546212935438893968040234453937403147803877197577293562635108480261311813060015943101882269008811590090799107771997952490566208559007107325449001558305247092097701991303020118183866318842959548678028717221041345957729264679139834360624004726175255219978516191835173510328822566298928885879265596389073833359631542268351556998005759985426237417323955190981428894476749229654726037133754617663 18
d = inverse(e,phi)
print(long_to_bytes(pow(c,d,N)))
```

Flag: FindITCTF{MuLt1-pR1m3_R54_&_jU5t_F4cT0r_1T__}

Random XOR 1

Langkah Penyelesaian:

```
from Crypto.Util.number import *

FLAG = "REDACTED"
NBITS = len(FLAG)<<2

a = getPrime(NBITS)
c = getRandomNBitInteger(NBITS)
m = getPrime(NBITS<<1)
seed = getRandomNBitInteger(NBITS)
state = seed

ciphertext = []

for i,f in enumerate(FLAG):
    state = (state*a+c)%m
    ciphertext.append(state^i^ord(f))

print(f"ciphertext = {ciphertext}")
```

Soal ini menggunakan LCG untuk generate random number (state), trus menggunakan 'state' tersebut untuk "encyrpt" datanya dengan xor menggunakan iterasi looping dan flag pada posisi itu… Karena kita sudah tau posisi awal flagnya 'FindITCTF{' kita bisa reverse itu menjadi state sebelumnya… Dari sana tinggal lempar ke cracking LCG dapet semuanya (increment, multiplier, modulus). Tinggal ulangi processnya untuk dapetin flag…

Code :

```
from functools import reduce
from math import gcd

def egcd(a, b):
    if a == 0:
```

```
        return (b, 0, 1)
    else:
        g, x, y = egcd(b % a, a)
        return (g, y - (b // a) * x, x)

def modinv(b, n):
    g, x, _ = egcd(b, n)
    if g == 1:
        return x % n
    else:
        raise Exception("Modular inverse does not exist")

def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return modulus, multiplier, increment

def crack_unknown_multiplier(states, modulus, index=0):
    if index > (len(states)-1):
        raise Exception("Multiplier cannot be found")
    try:
        multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    except Exception:
        index += 2
        crack_unknown_multiplier(states, modulus, index)

    multiplier = (states[index + 2] - states[index + 1]) *
modinv(states[index + 1] - states[index], modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)

def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:],
diffs[2:])]
    modulus = abs(reduce(gcd, zeroes))
    return crack_unknown_multiplier(states, modulus)

ciphertext =
[570940291250001418516102164077172487131102045067335208648028098170552635313
768732630829466473065775458954082981919020596364453958 9, ...,
128978629339902401675225570549478385726214250123352952364450984256591507980 9
664470475438857415153537572006456809511851432046948 63,
571081710259626525723878211891098267904565648961926151442720411778012936690 0
038279818723528405663476646755507801735268266887028603]
known = "FindITCTF{"

states = []
```

```
for i,j in enumerate(known):
    states.append(ciphertext[i] ^ i ^ ord(j))
# print(states)
m, a, c = crack_unknown_modulus(states)
# print(m,a,c)
state = states[0]
flag = ""
for i in range(1,len(ciphertext)):
    state = (state*a+c)%m
    flag += chr(ciphertext[i] ^ state ^ i)
    print('F'+flag)
    # FindITCTF{l1N3aR_C0N9RU3NT1AL_93n3rat0r_903555_BrRrrr}
```

Flag: FindITCTF{l1N3aR_C0N9RU3NT1AL_93n3rat0r_903555_BrRrrr}

Random XOR 2

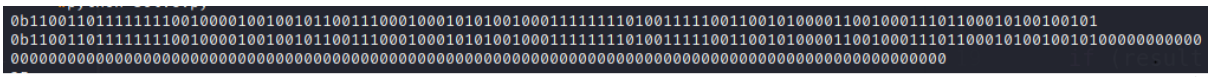
Langkah Penyelesaian:

Langkah pertama yang dilakukan adalah mencari state awal setelah dihitung,
Dengan menggunakan flag paling depan yaitu 'F' dan i yaitu 0.

setelah itu direverse.

setelah ketemu statenya yang masih dishift 116 kekiri, penulis akan dishift ke kanan 116.

```
print (bin(ciphertext[0]^0^ord('F')))
print (bin((ciphertext[0]^0^ord('F'))<<116))
```



Dilihat hasil diatas yang sebelum dan sesudah di shift ke kanan, ada bagian 0 setelah 101. Angka itu yang harus dicari.

```
start = (ciphertext[0]^0^ord('F'))<<116
end = ((ciphertext[0]^0^ord('F'))+1)<<116

print (len(str(end-start)))
```

Nanti variable start akan ditambah dengan angka brute force sepanjang 35(len(str(end-start)))

awal nya seperti dibawah

```
p = '0'*35
state = start + int(p)
```

Yang dibrute force ada bagian p yang paling kiri

```
p = angka_brute_force + '0'*(35-1)
```

angka_brute_force adalah 0 - 9

setelah ditambah state baru akan dicalculate dengan index flag ke 2

```
state = (state*a+c)%m
result = (state>>116)^1^ord('i')
```

jika hasil resultnya lebih kecil atau sama dengan hasil output encrypt ke 2 , maka akan dimasukan ke variable penampung

angka_brute_force yang diambil adalah yang memiliki hasil result yang mendekati atau sama dengan hasil output encrypt ke 2. selain itu fail.

Tidak bisa menggunakan index flag ke 2 sajah, index ke 3 dan selanjutnya akan digunakan. karena state yang dihasilkan hanya index flag ke 2 masih salah. Jadi saya akan mencoba lanjutkan dengan flag index ke 2 dan seterusnya.

```
└──╼ #python solve.py
0b1100110111111111001000010010010110011100010001010100100011111111010011111001100101000011001000110110001010010
0b1100110111111111001000010010010110011100010001010100100011111111010011111001100101000011001000110110001010010
0000000000000000000000000000000000000000
35
175765775000000000000000000000000
175765775028431421300000000000000
175765775028431436533704534700000000
175765775028431436533704539203054 05
FindITCTF{L477ice_4774ck_0N_lInE4R_c0n9rUeN7I4l_9eNeR470r}
```

Code :

solve.py

```
import string

ciphertext = [835607687794861790454672319937059 5,
660799266979132536417409697219500 35,
809930297491151992009566296527786 27,
377075817445491500291379885592934 78,
115237595635262089769089037452547 87,
539002771361093683459858219606137 6,
233461893819096032173023552777489 32,
352278541511265043960239627758303 47,
718328540214470583420472016715470 09,
656912884268263698419176644334654 63,
289035813986392378588336974809285 45,
763684730500411799395038374575395 20,
611536178735298956564187994062385 07,
110345318109770673470646469634809 60,
413294100238017961556641592279092 64,
226549406681371850597159814194238 00,
425119794146442430978528773844355 54,
815511879135866354606068387359609 16,
680099477257026545045561174296243 80,
290098559953831718236456699460000 25,
513631495242622675157262831989324 42,
821025852204431373413991699754508 35,
933521885204713807175774821312629 2,
401213951283245381501299031819784 30,
725580447238584326065539024645581 94,
```

```
26200882429889946050148231496032603,
27757196714211570347368413689309020,
45696313426756283444841573524697615,
35007525929490317519675322615536472,
15771380847896929864947129903497962,
16451620031463407219117649099779213,
27648553409945627779660381493675509,
58140947580028619442467052766700217,
62508224916737793035358860414319991,
26932791745466169635428614961739134,
85955781144138159049465981328407737,
45744915793733442853878349453822373,
33600028854644621978916871587550690,
22662456837881334819080362869941079,
41057863524442521804573408764833194,
67976856105925580307540544286911948,
92222185053052940596489230915329O,
77062958508660605861577392540963381,
56594300760704301019005906600337850,
26609680641122633695681107899696511,
33835178933051944792081006700715456,
56706076248303379958146622478628344,
19765584456602703691311637562982768,
24128205841694243808570064754450 97,
61467816436270309494524432611592431,
50929472183440533130281995847845363,
65962077835044307474510426262846146,
64065574557217732528522501878975151,
44990780333729864578949979695935 78,
25574626226310774198396316578902244,
39840927013488106149312604620612729,
71270307095631700756460419975628218,
29832445257034598053293551460100056]

a = 0xBAD2C0DE
c = 0x6969
lenFlag = len(ciphertext)
nbits = lenFlag<<2
m = 1<<nbits

to_flag = 'FindITCTF{'
def gene(state,idx,flag):
    temp = state
    for _ in range(idx):
        temp = (temp*a+c)%m
    result = (temp>>(nbits>>1))^idx^ord(flag)

    if (result <= ciphertext[idx]):
        return ciphertext[idx] - result
    return -1
```

```
start = (ciphertext[0]^0^ord('F'))<<116
end = ((ciphertext[0]^0^ord('F'))+1)<<116

print (len(str(end-start)))

state_temp =''
//looping pertama
for _ in range(35):
    tmp = ''
    comp_result= -1
    for i in range(10):
        p = state_temp + str(i)
        state = start+int(p.ljust(35,'0'))
        result_1 = gene(state ,1 ,to_flag[1])

        if result_1 == -1:
            pass
        elif (comp_result == -1):
            comp_result = result_1
            tmp = str(i)
        elif (result_1 < comp_result) :
            comp_result = result_1
            tmp = str(i)

    state_temp = state_temp + tmp

print (state_temp)
len_until_0 = len("175765775")
state_temp = state_temp[:len_until_0]

//looping kedua
for _ in range(35-len_until_0):
    tmp = ''
    comp_result= -1
    for i in range(10):
        p = state_temp + str(i)
        state = start+int(p.ljust(35,'0'))
        result_1 = gene(state ,2 ,to_flag[2])

        if result_1 == -1:
            pass
        elif (comp_result == -1):
            comp_result = result_1
            tmp = str(i)
        elif (result_1 < comp_result) :
            comp_result = result_1
            tmp = str(i)

    state_temp = state_temp + tmp
```

```
print (state_temp)
len_until_0 = len("1757657750284314")
state_temp = state_temp[:len_until_0]

//looping ketiga
for _ in range(35-len_until_0):
    tmp = ''
    comp_result= -1
    for i in range(10):
        p = state_temp + str(i)
        state = start+int(p.ljust(35,'0'))
        result_1 = gene(state ,3 ,to_flag[3])

        if result_1 == -1:
            pass
        elif (comp_result == -1):
            comp_result = result_1
            tmp = str(i)
        elif (result_1 < comp_result) :
            comp_result = result_1
            tmp = str(i)

    state_temp = state_temp + tmp

print (state_temp)
len_until_0 = len("17576577502843143653370453")
state_temp = state_temp[:len_until_0]

for _ in range(35-len_until_0):
    tmp = ''
    comp_result= -1
    for i in range(10):
        p = state_temp + str(i)
        state = start+int(p.ljust(35,'0'))
        result_1 = gene(state ,4 ,to_flag[4])

        if result_1 == -1:
            pass
        elif (comp_result == -1):
            comp_result = result_1
            tmp = str(i)
        elif (result_1 < comp_result) :
            comp_result = result_1
            tmp = str(i)

    state_temp = state_temp + tmp

print (state_temp)
print (start+int(state_temp))
```

```
state = start+int(state_temp)

flag = "F"
for i in range(1, len(ciphertext)):
    state = (state*a+c)%m
    for ch in string.printable:
        tmp2 = (state>>(nbits>>1)) ^ i ^ ord(ch)
        if tmp2 == ciphertext[i]:

            flag += ch
            print(flag)
            break
```

Flag:
FindITCTF{L477ice_4774ck_0N_lInE4R_c0n9rUeN7I4l_9eNeR470r}

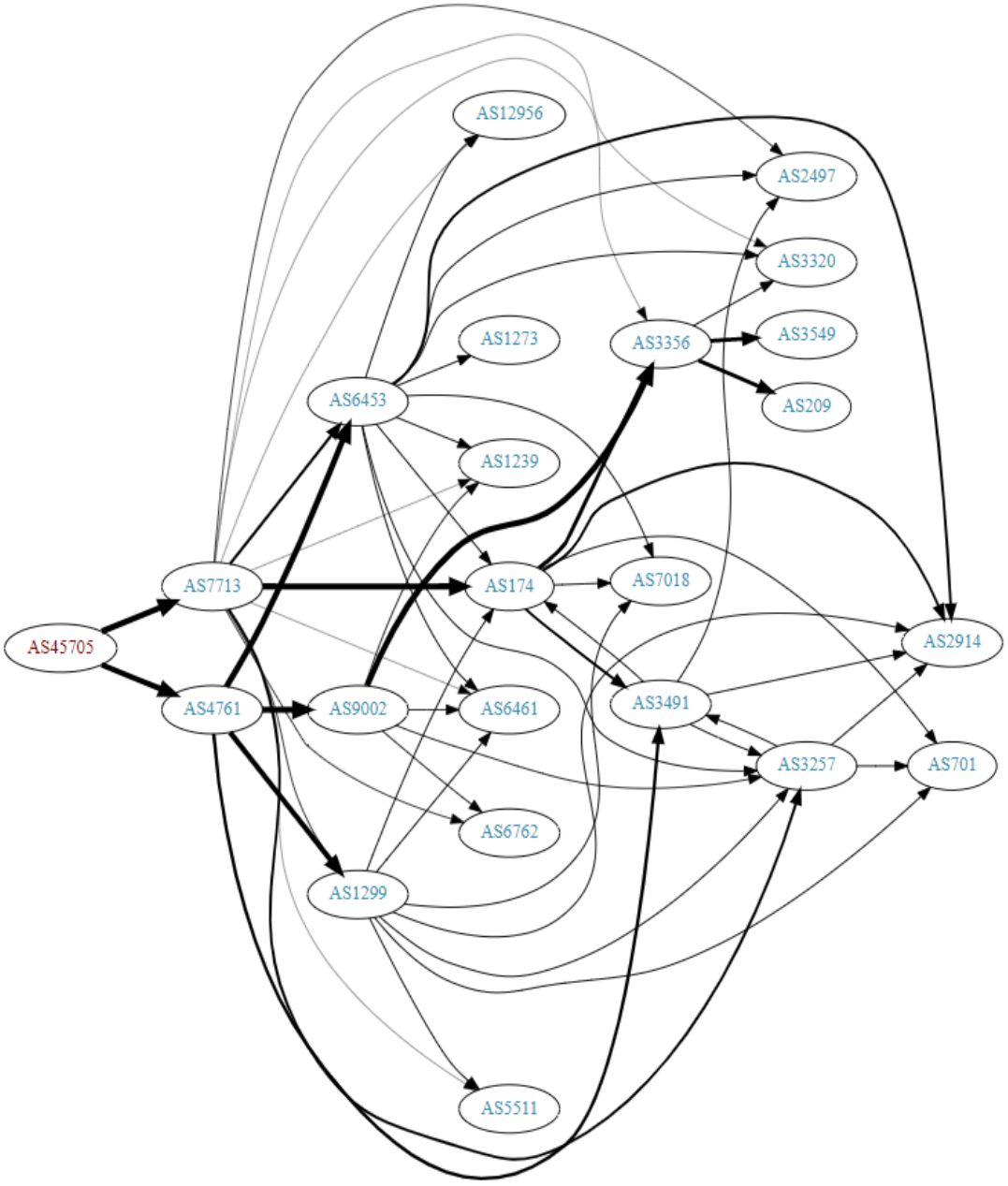
Osint

UGM IP Transit Provider

Langkah Penyelesaian:
Kita diminta untuk mencari ASN dari main ISP Universitas Gadjah Mada

Saya menemukan sebuah website yang menunjukkan ASN berbentuk graph

https://bgpview.io/asn/45705#graph



Di graph ini kita bisa melihat bahwa AS45705 adalah ASN UGM, dan saya sempat terkecoh karena dibilang bahwa Flag Format : FindITCTF{ASxxx} dikarenakan x nya ada 3, saya mengira bahwa hanya perlu mencari ASN yang 3 digit, ternyata yang benar adalah diantara 2 cabang setelah AS45705 (main dan backup) , ternyata AS7713 jawabannya

https://bgpview.io/asn/7713

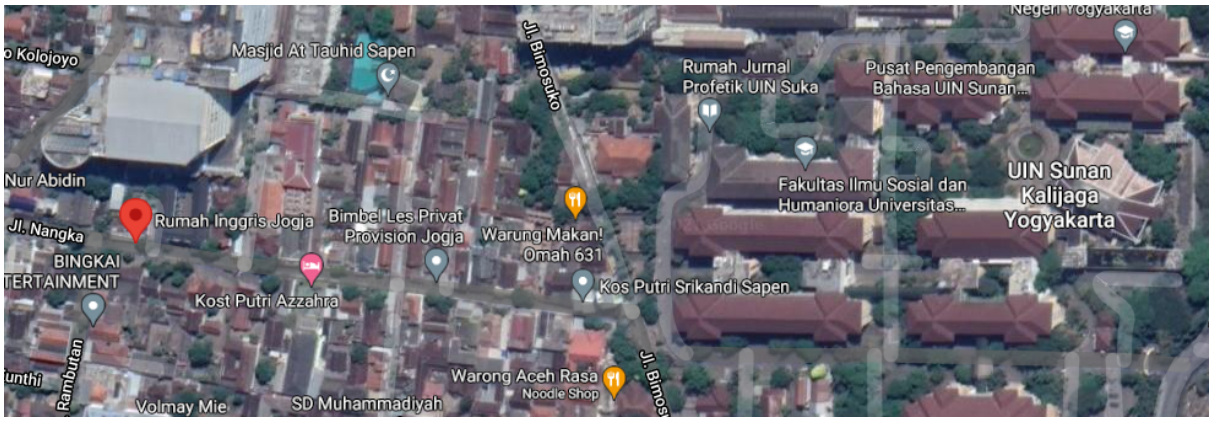
Flag: FindITCTF{AS7713}

Sedia Kos

Langkah Penyelesaian:
Diberikan sebuah gambar jalan sempit dan kita disuruh cari nama jalannya, dengan clue fokus banner rumah inggris jogja.

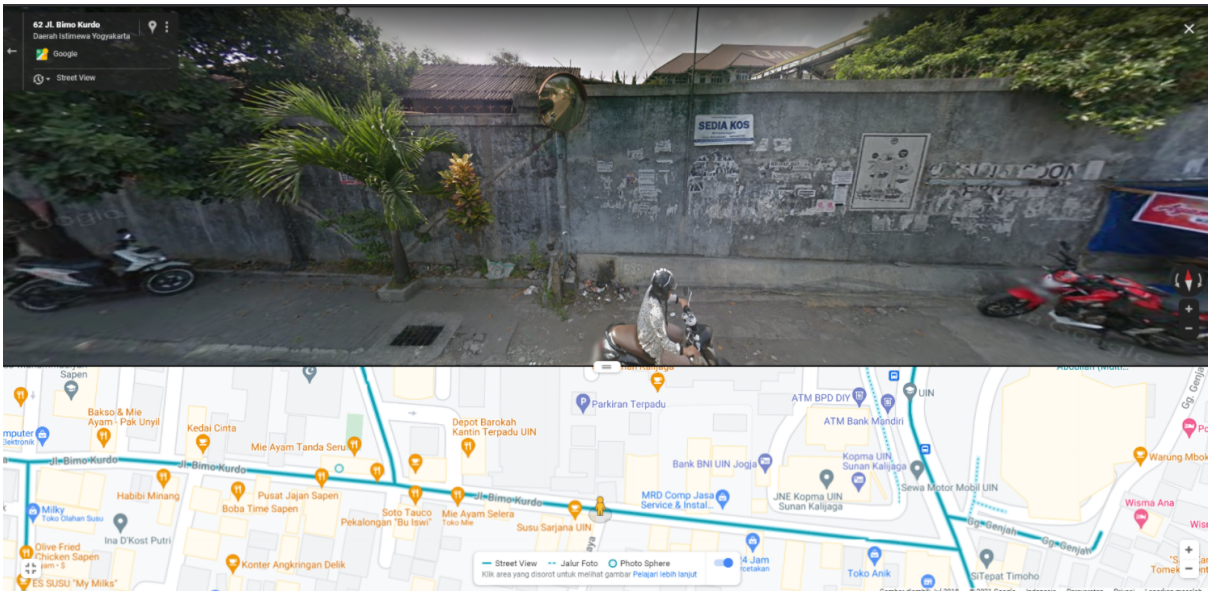
Di kanan atas terlihat sebuah bangunan besar dengan huruf UI- diatasnya

Saya mencoba mencari di sekitar rumah inggris jogja dan menemukan universitas dengan awalan UIN



Menggunakan google street view saya jalan-jalan online wkwk

Ditemukan tempat gambar chall.png diambil



Flag: FindITCTF{Bimo_Kurdo}

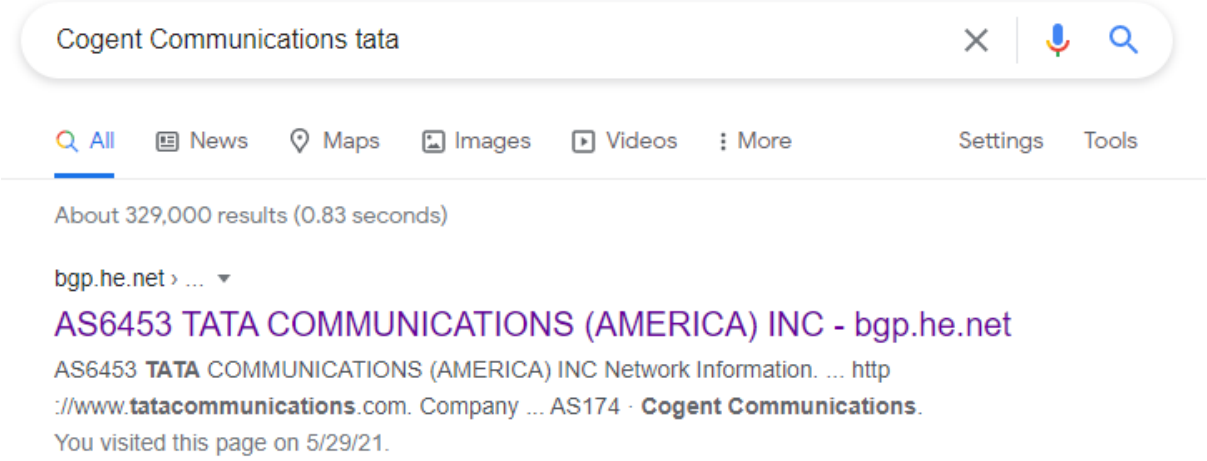
Bogon Address

Langkah Penyelesaian:
Challenge ini meminta flag berupa address range dari IP bogon ISP yang dimaksud.

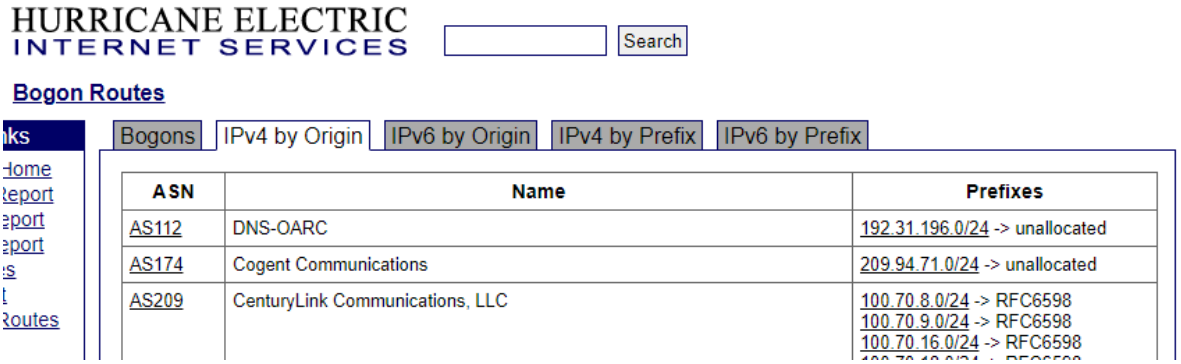
Disini saya mulai merasakan manfaat dari overthinking di challenge osint pertama saat mencari ASN, saya sempat memasukkan ASN ini

https://bgpview.io/asn/174

yang tertuju ke Congent Communications, sedikit mengulik ke arah situ bisa didapatkan



https://bgp.he.net/report/bogons#_bogonsv4asn



Flag: FindITCTF{209.94.71.0/24}

Mega Chad

Langkah Penyelesaian:
Diberikan beberapa hashtag, saya berpikir ini basically challenge mencari flag di postingan media sosial.

Mencoba mencari #megachad tapi tidak beruntung, reverse image search juga tidak beruntung. Jadi saya ke instagrap page mas gigachad.

https://www.instagram.com/berlin.1969/

Di bagian tagged nya ditemukan gambar yang tercantum di challenge oleh helltime.hunter

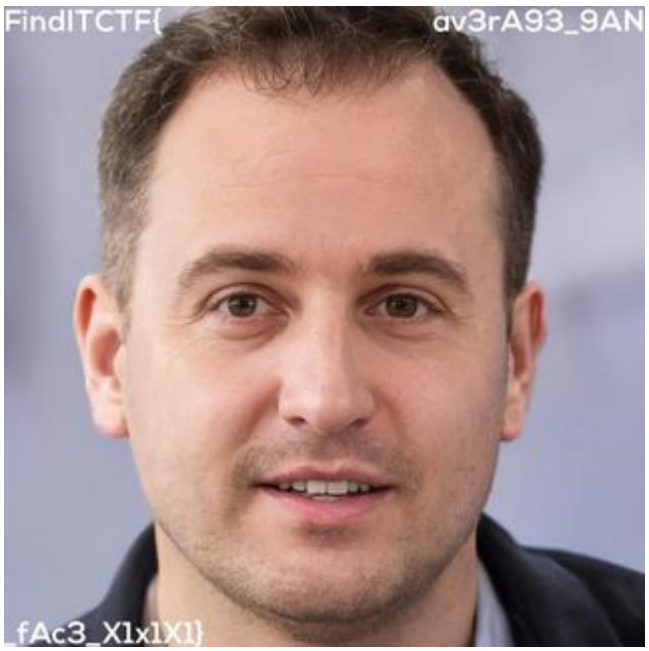
Ada comment di gambar tersebut yang di post 1 minggu yang lalu oleh strabarjo

https://www.instagram.com/strabarjo/

Dia mempost story juga

Kinda sus, jadi saya beneran buka profile picturenya di

https://instagram.fcgk29-1.fna.fbcdn.net/v/t51.2885-19/s320x320/186302327_135220755313248_3433657006464569314_n.jpg?tp=1&_nc_ht=instagram.fcgk29-1.fna.fbcdn.net&_nc_cat=100&_nc_ohc=UaJXJC8IKwMAX_Ddkqu&edm=ABfd0MgBAAAA&ccb=7-4&oh=6398a513b83ef882fa6b1894cb4b6788&oe=60BB0181&_nc_sid=7bff83



Flag: FindITCTF{av3rA93_9AN_fAc3_X1x1X1}

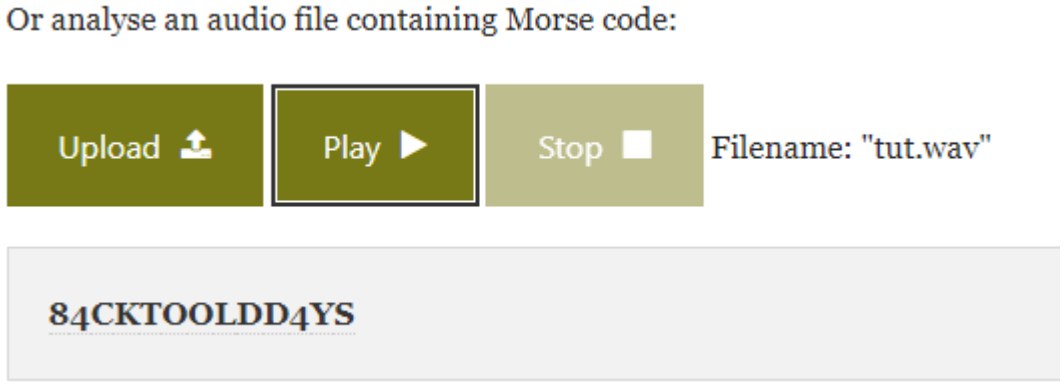
Forensic

Isyarat

Langkah Penyelesaian:
Diberikan file bernama tut.wav

Terdengar seperti morse code, jadi saya decode dengan online tools

https://morsecode.world/international/decoder/audio-decoder-adaptive.html



Flag: FindITCTF{84CKTOOLDD4YS}

Snow

Langkah Penyelesaian:
Diberikan sebuah file txt.txt, berdasarkan judul challenge dan categorynya saya berpikir ini whitespace steganography.

Terbukti dengan adanya whitespace setelah readable characters di txt.txt

Sesuai dengan nama challengenya saya menggunakan tools bernama snow

http://www.darkside.com.au/snow/

```
PS D:\Desktop> .\SNOW.EXE -C .\txt.txt out.txt
PS D:\Desktop> type .\out.txt
https://drive.google.com/file/d/1QGBmDYCZ0byYfslzMl90-pIJJMabda7a/view?usp=sharing
PS D:\Desktop>
```

https://drive.google.com/file/d/1QGBmDYCZ0byYfslzMl90-pIJJMabda7a/view?usp=sharing

Didapatkan link gdrive yang berisi gambar monyet

Jika di binwalk akan ditemukan flag.txt

```
root@kali:~/Documents/findit# binwalk -e xixi2.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
16830         0x41BE          Zip archive data, at least v2.0 to extract, compressed size: 47, uncompressed size: 45, name: flag.txt
17005         0x426D          End of Zip archive, footer length: 22

root@kali:~/Documents/findit# cd _xixi2.png.extracted/
root@kali:~/Documents/findit/_xixi2.png.extracted# ls
41BE.zip  flag.txt
root@kali:~/Documents/findit/_xixi2.png.extracted# cat flag.txt
FindITCTF{573g4No_JU57 _n33D_4_L177l3_3fFOr7}root@kali:~/Documents/findit/_xixi2.png.extracted#
```

Flag: FindITCTF{573g4No_JU57 _n33D_4_L177l3_3fFOr7}

Chemistry Problem

Langkah Penyelesaian:
Diberikan sebuah gambar berupa foto kertas latihan soal kimia yang membuat saya flashback jaman SMA terkutuk.

Saat analisa menggunakan aperisolve terlihat hasil zsteg ada entry wbStego encrypted string, berarti ada hidden data didalamnya.

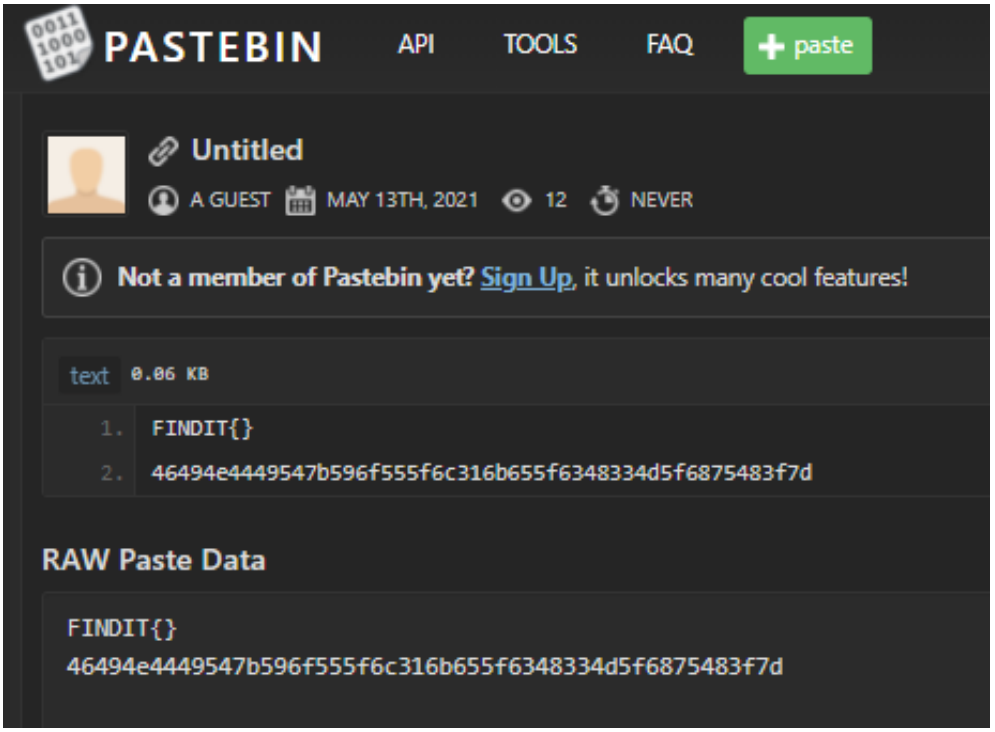
Tapi saat pembuatan writeup aperisolve sedang gangguan jadi forever loading

Penulis menggunakan jsteg

```
root@kali:~/Documents/findit# jsteg reveal IMG_20190613_074643.jpg | base64 -d
https://pastebin.com/cNpYQLcw Passkey: 43nwGkX09Broot@kali:~/Documents/findit#
```

https://pastebin.com/cNpYQLcw Passkey: 43nwGkX09B

Mengikuti link pastebin dan memasukkan passkey, bisa mendapatkan semacam text



Step terakhir melakukan decode hex to ASCII dari 46494e4449547b596f555f6c316b655f6348334d5f6875483f7d dan didapatkan flag

Flag: FINDIT{YoU_l1ke_cH3M_huH?}

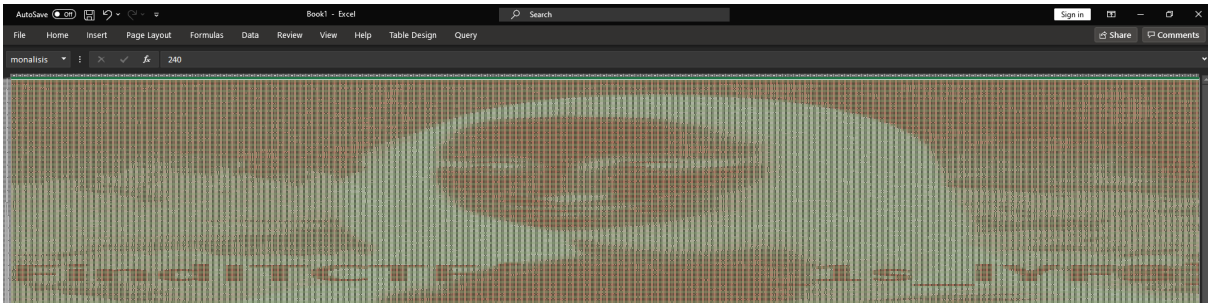
Misc

Monalisis

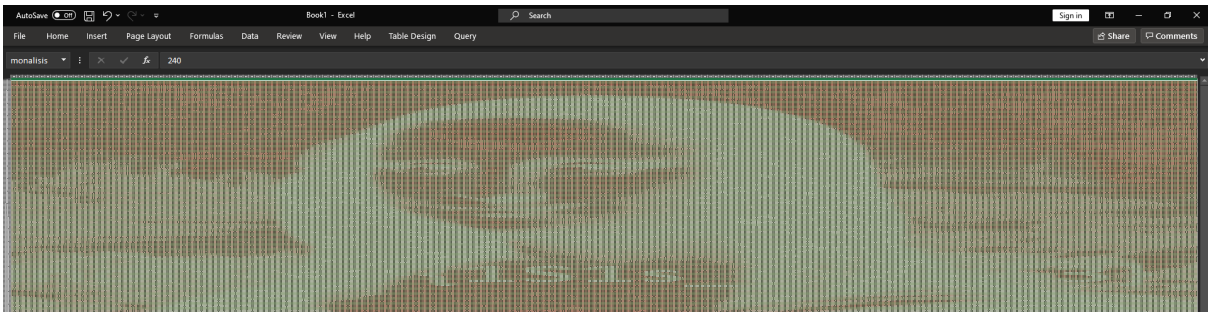
Langkah Penyelesaian:
Diberikan sebuah file CSV yang isinya seperti RGB value, tapi karena saya buka di excel saya jadi teringat 1 video youtube.

https://www.youtube.com/watch?v=UBX2QQHlQ_I&ab_channel=Stand-upMaths

Di menit 2:10 , ditunjukkan cara membuat semacam color gradient menggunakan conditional formatting, dan value yang terlihat di layar lumayan serupa. Jadi saya coba saja



Didapatkan partial flag, jika di slide ke kanan sheet maka terlihat gradient berbeda yang lebih menunjukkan partial flag yang tertutup tadi.



Flag: FindITCTF{1S1s_1s_lYF3}