Write Up CTF-ARA-2023

Team : Haripotar
Asal : SMKN 04 Malang
Anggota :

- **Fikri Muhammad Abdillah (FlaB) - Captain**
- **Firda Gheitsa Sahira (abcfirdaa)**

A. Cryptography

1. One Time Password (?)

Kami mendapatkan link google drive yang berisi,

```
A: 161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b3b3e42
B: 151d616075737f322e2d130b381666547d3d4470054660287f33663d2a2e32

XOR: 415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d
```

Sekilas itu terlihat seperti hex encoding, lalu saya mencoba men decode satu-persatu

```
You, 10 hours ago | 1 author (You)
from pwn import *
A= '161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b3b3e42'
B= '151d616075737f322e2d130b381666547d3d4470054660287f33663d2a2e32'

XOR= '415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d'

quote = binascii.a2b_hex("%s" % (XOR.strip())).decode("ASCII").replace(';', '\n- ')
print(quote)      You, 10 hours ago • add file ara-ctf-2023 …
```

Lalu saat kami mendecode variable hex, flag langsung terlihat di hasil decode tersebut

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\One Time Password\try\decrypt.py"
ARA2023{th3_p_5t4nd5_f0r_p4dzz}

[Done] exited with code=0 in 1.213 seconds
```

2. Secrets Behind a Letter

Kami mendapatakan Attachments yang berisi link google drive yang berisi,

```
p:
125753336941212676905219718556916381441368103311882482367708803389058118834850641048656498349278197256176955544721003413618
961620223116533015328101013442 73
q:
124974834261750724658521679369605262322848918767879810806711627835614115216758091122045736173583897427325462935027095851292
058857260784924171098675123987 47
c:
360629344957317929086395350628331806510228135895355928518025722643282990274064139273468524542176277933151448929420268869808
236222401574057174997879599430405407341221428388984827675412726778370913038246699129635727146561394220118530281335561114050
725265098398467015701334377461027276449823447125718443322802 18

e = 65537
```

Sekilas dapat dilihat jika itu adalah decrypt RSA, lalu kami mendecode dengan program python

```python
def decrypt(p, q, c, e):
    n = p * q
    phi_n = (p - 1) * (q - 1)

    d = pow(e, -1, phi_n)

    m = pow(c, d, n)

    text = ""
    while m > 0:
        text = chr(m % 256) + text
        m //= 256

    return text

p = 125753336941212676905219718556916381441368103311882482367708803389058118834850641048656498349278197256176955544721003413618961620223116533015321
q = 124974834261750724658521679369605262322848918767879810806711627835614115216758091122045736173583897427325462935027095851292058857260784924171091
c = 360629344957317929086395350628331806510228135895355928518025722643282990274064139273468524542176277933151448929420268869808236222401574057174995
e = 65537

m = decrypt(p, q, c, e)

print("decrypt:", m)      You, 2 hours ago • merapikan …
```

Lalu mendapatkan hasil yang berupa flag

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\Secrets Behind a Letter\try\decrypt_rsa.py"
decrypt: ARA2023{1t_turn5_0ut_to_b3_an_rsa}

[Done] exited with code=0 in 0.209 seconds
```

3. L0V32X0R

Kami mendapatkan tulisan text dari soal "001300737173723a70321e3971331e352975351e247574387e3c".

Bisa dilihat, jika itu kemungkinan adalah hex encoding, lalu saya mencoba decode biasa, tetapi kami hanya mendapatkan flag yang tidak jelas, lalu kami melihat bahwa judul soal tersebut berisi hint yaitu X0R, kemungkinan besar hasil decode tadi harus di xor terlebih dahulu. Karena kami tidak mendapatkan key xor maupun program encrypt nya, kami membuat program bruteforce xor lalu mencari format flag "ARA2023{" pada setiap hasil xor tersebut

```python
from pwn import binascii
import string
cipher = binascii.unhexlify('001300737173723a70321e3971331e352975351e247574387e3c').decode()

def combine(a,b):
    ret = []
    for x in a:
        for y in b:
            ret.append(x+y)
    return ret

brute = string.printable
knowed_text = "ARA2023{"
brute_list = ['']
is_breaks = False
loops = 1
while True:
    print("length key: "+str(loops))
    brute_list = combine(brute_list,brute)
    for b in brute_list:
        text = ''      You, 2 hours ago • add solve l0v32x0r …
        for t in cipher:
            for x in b:
                text += chr(ord(t)^ord(x))
        if knowed_text in text:
            is_breaks = True
            print("result: "+text)
            break
    if is_breaks:
        break
    loops += 1
```

Lalu kami menemukan flag dari hasil program diatas

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\L0v32x0r\try\decrypt.py"
length key: 1
result: ARA2023{1s_x0r_th4t_e45y?}

[Done] exited with code=0 in 0.85 seconds
```

4. SH4-32

Kami mendapatkan text, dan Dictionary yang sepertinya berisi text yang belum di hash

9be9f4182c157b8d77f97d3b20f68ed6b8533175831837
c761e759c44f6feeb8

Awalnya kami kira text tersbut harus di decode hex terlebih dahulu, tetapi saat saya lihat hasil hash 'SHA', kami tidak jadi untuk mendecode nya,

```
import hashlib
import sys
from pwn import binascii

def combine(a,b, separator = False):
    ret = []
    for x in a:
        for y in b:
            if separator:
                ret.append(x+"_"+y)
            else:
                ret.append(x+y)
    return ret


d_raw = open("Dictionary.txt").readlines()
DICTIONARY = []
for d in d_raw:
    DICTIONARY.append(d.strip())

cipher = "9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8"
format = "ARA2023{"
list_brute = []
breaks = False
loop = 1
is_separator = True
is_with_format = True
print('with format & separator:')
while True:
    print(loop)
    if is_with_format:
        if list_brute == []:
            list_brute = combine(format,DICTIONARY)
        else:
            list_brute = combine(list_brute,DICTIONARY, is_separator)
    else:
        if list_brute == []:
            list_brute = combine([''],DICTIONARY)
```

```
        else:
            list_brute = combine(list_brute,DICTIONARY, is_separator)

    for t in list_brute:
        if is_with_format:
            t = t+"}"
        temp_hash = []
        temp_hash.append(hashlib.sha256((t).encode()).hexdigest())
        temp_hash.append(hashlib.sha384((t).encode()).hexdigest())
        temp_hash.append(hashlib.sha224((t).encode()).hexdigest())
        temp_hash.append(hashlib.sha512((t).encode()).hexdigest())
        if cipher in temp_hash:
            print("result: "+ t)
            breaks = True
            break
    if breaks:
        break
    if loop >= 2:
        if is_separator == True:
            print("now with out separator")
            is_separator = False
            list_brute = []
            loop == 1
            continue
        else:
            if is_with_format == True:
                print("now with out format & with separator")
                is_with_format = False
                is_separator = True
                list_brute = []
                loop == 1
                continue
            else:
                sys.exit("this not work")
    else:
        loop += 1
```

dari program diatas, karena kami tidak tahu apakah itu di gabung dengan format flag atau tidak, jadi kami brute force kata2nya, dengan length 2 sebagai limit nya lalu mendapatkan hasil

```
PS D:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\SH4-32\try> python .\brute.py
with format & separator:
1
2
now with out separator
2
now with out format & with separator
2
result: 415241323032337b6834736833645f30525f6e4f545f6834736833647d
PS D:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\SH4-32\try>
```

Karena kami rasa itu bukan flag nya, lalu kami mencoba decode hex lalu mendapatkan flag

```
ARA2023{h4sh3d_0R_nOT_h4sh3d}
```

5. babychall

Kami mendapatkan 2 link, satu merupakan link youtube (yang sepertinya hanya pengalih) dan satunya link google drive yang berisi

```
c1=509969731048456631083797511312030854324124901983127146636568236482330384792981928614518342469302081401101736990585279190
201154325867054004673456478065223313964476508476501330132466733908792227191692488624202782563229677187017004587292077931247
581664386414481123144899458632318819823527907651305350040900536 77
c2=267508635447697542205541466679550468324230594820076134825002840126688202849479272407247353088803134399798848563936737592
797410030710740677510369519880070370418141473628138846420542912315960504818663485277171790970486464711281758602468229998786
860793305963427955632147620481352120168266232851008649621582146 1
c3=372306582432525907436085711050273578627909729872088332130179411714487538156548399016995266514337713248268953556712559444
148939479639349790682573103673159357012708043907991216696351530129164022711907226189975003929117377671433165523764958829869
356951469708539142754817174002688326449871579887275755133514419 19

n1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343315900732
228502875706659244844711538497850413046440270578916645981161000807526427004236918404837363404678029443944950655102252423415
631977020625826867728898231382737396728896847618010577420408630133
n2=931056210596864748168902154945548028315189484201609417035227591216197858512706086341303074502275579879768181623319822896
342150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346260620609607392698
287892741372743639700562761394340393158600525564173406969985092 71
n3=659185096507422784949713632908748491812683643160126567693391200040007029452719425330975298849640631093770367158471761962
809438072619868485930004241433202800532790214113942672682553377834949016063196874573515869153146628004346323329889788580859
315868302836948815387590083604866619368842022749733871082147541 01
```

Dapat dilihat, bahwa itu sepertinya ada RSA-CRT (Chineese Reminder Theorem), Lalu saya memakai program python

```
c1=509969731048456631083797511312030854324124901983127146636568236482330384792981928614518342469302081401101736990585279190201154325867054004673456478065223313964476508476501330132466733908792227191692488624202782563229677187017004587292077931247581664386414481123144899458632318819823527907651305350040900536 77
c2=267508635447697542205541466679550468324230594820076134825002840126688202849479272407247353088803134399798848563936737592797410030710740677510369519880070370418141473628138846420542912315960504818663485277171790970486464711281758602468229998786860793305963427955632147620481352120168266232851008649621582146 1
c3=372306582432525907436085711050273578627909729872088332130179411714487538156548399016995266514337713248268953556712559444148939479639349790682573103673159357012708043907991216696351530129164022711907226189975003929117377671433165523764958829869356951469708539142754817174002688326449871579887275755133514419 19

n1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343315900732284150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346260620609607392698287892741372743639700562761394340393158600525564173406969985092 71
n2=931056210596864748168902154945548028315189484201609417035227591216197858512706086341303074502275579879768181623319822896342150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346260620609607392698287892741372743639700562761394340393158600525564173406969985092 71
n3=659185096507422784949713632908748491812683643160126567693391200040007029452719425330975298849640631093770367158471761962809438072619868485930004241433202800532790214113942672682553377834949016063196874573515869153146628004346323329889788580859315868302836948815387590083604866619368842022749733871082147541 01

import functools
from Crypto.Util.number import inverse

def chinese_remainder(n, a):
    sum = 0
    prod = functools.reduce(lambda a, b: a*b, n)
    for n_i, a_i in zip(n, a):
        p = prod // n_i
        sum += a_i * inverse(p, n_i) * p
    return sum % prod
```

```python
def inv_pow(c, e):
    low = -1
    high = c+1
    while low + 1 < high:
        m = (low + high) // 2
        p = pow(m, e)
        if p < c:
            low = m
        else:
            high = m
    m = high
    assert pow(m, e) == c
    return m

N = [n1, n2, n3]
C = [c1, c2, c3]
e = 3
a = chinese_remainder(N, C)
for n, c in zip(N, C):
    assert a % n == c
m = inv_pow(a, e)
print(bytes.fromhex(hex(m)[2:]).decode())
```

Saat kami run program tersebut, ternyata langsung mendapatkan flag dari hasil program tesebut

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Cryptography\babychall\try\decrypt_crt.py"
ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}

[Done] exited with code=0 in 0.566 seconds
```

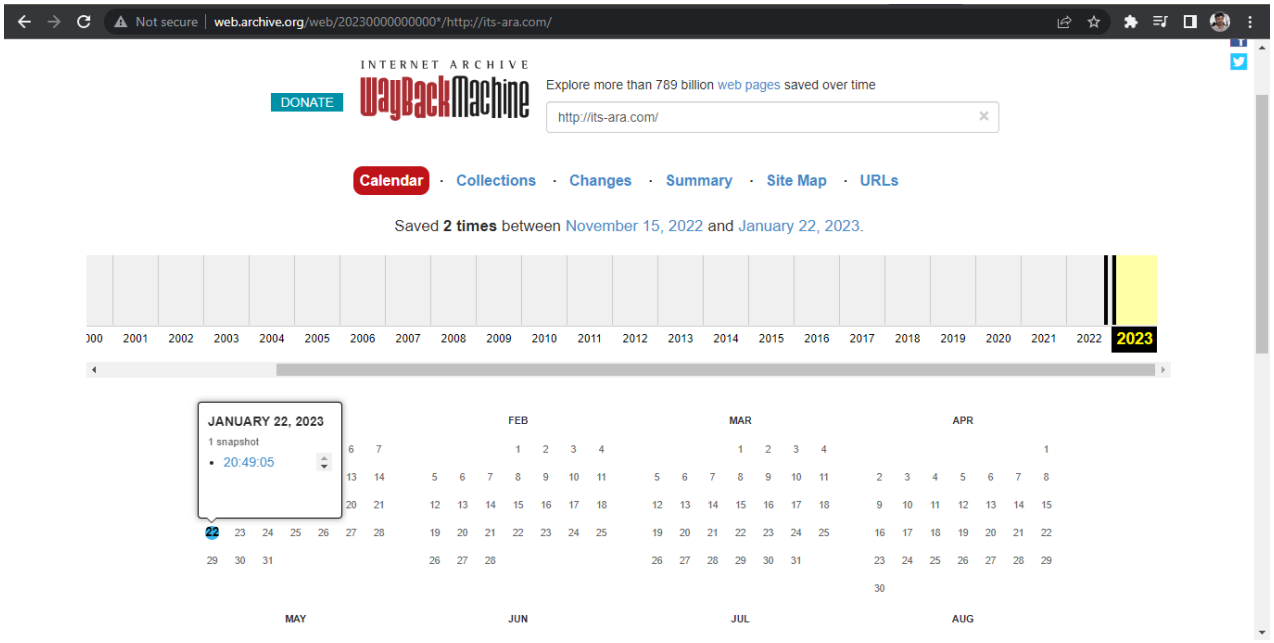
B. OSINT

1. Time Machine

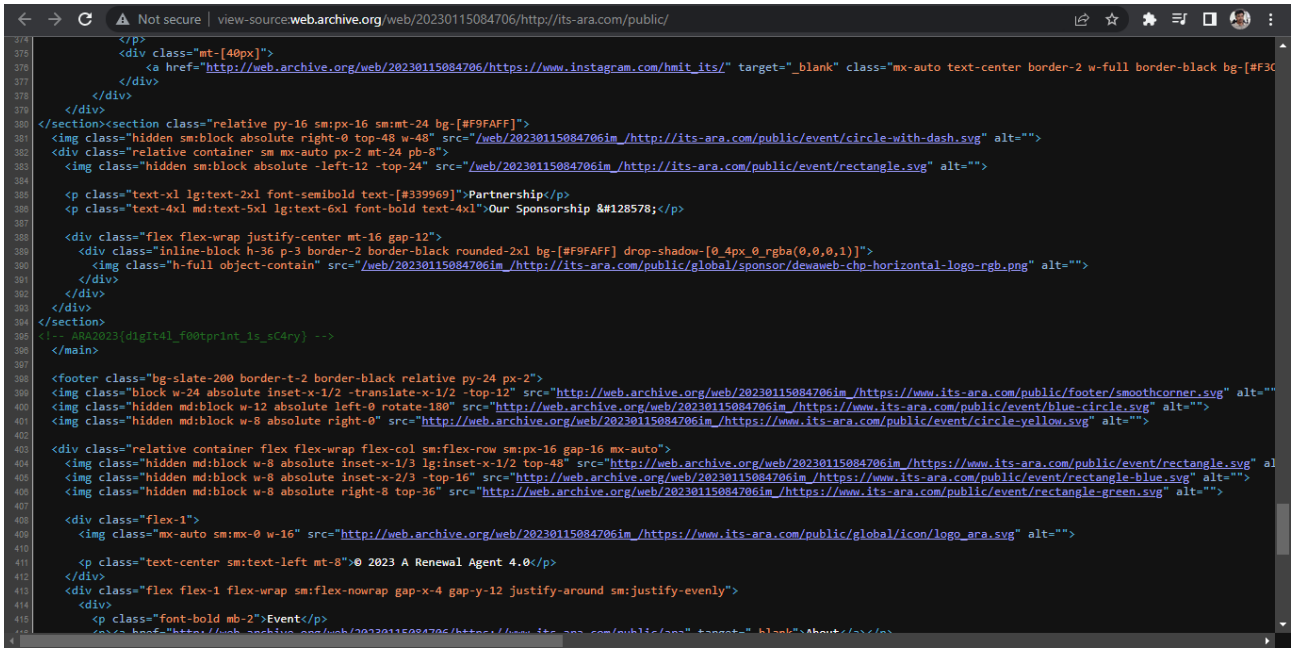
Kami mendapatkan soal

> There was a secret leaked on Official ARA Website. It can only seen on January 22nd 2023. Can you turn back the time?

Saat kami membaca soal tersebut, kami langsung mencoba memakai web.archive.org dan memasukkan link website ARA
Lalu kami menemukan ada archive tanggal 22 januari



Lalu kami membuka source code web tersebut, lalu mendapatkan flag nya



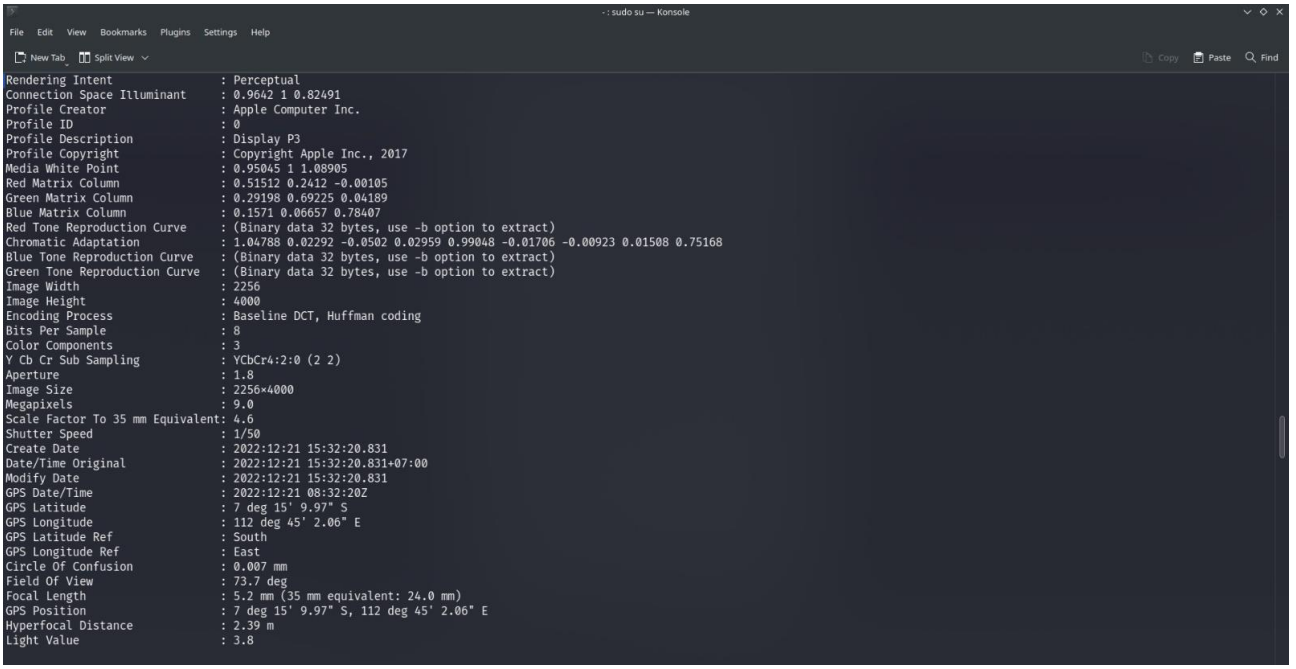
2. Backroom

Kami mendapatkan attachments yang berupa link google drive, yang berisi gambar suatu tempat

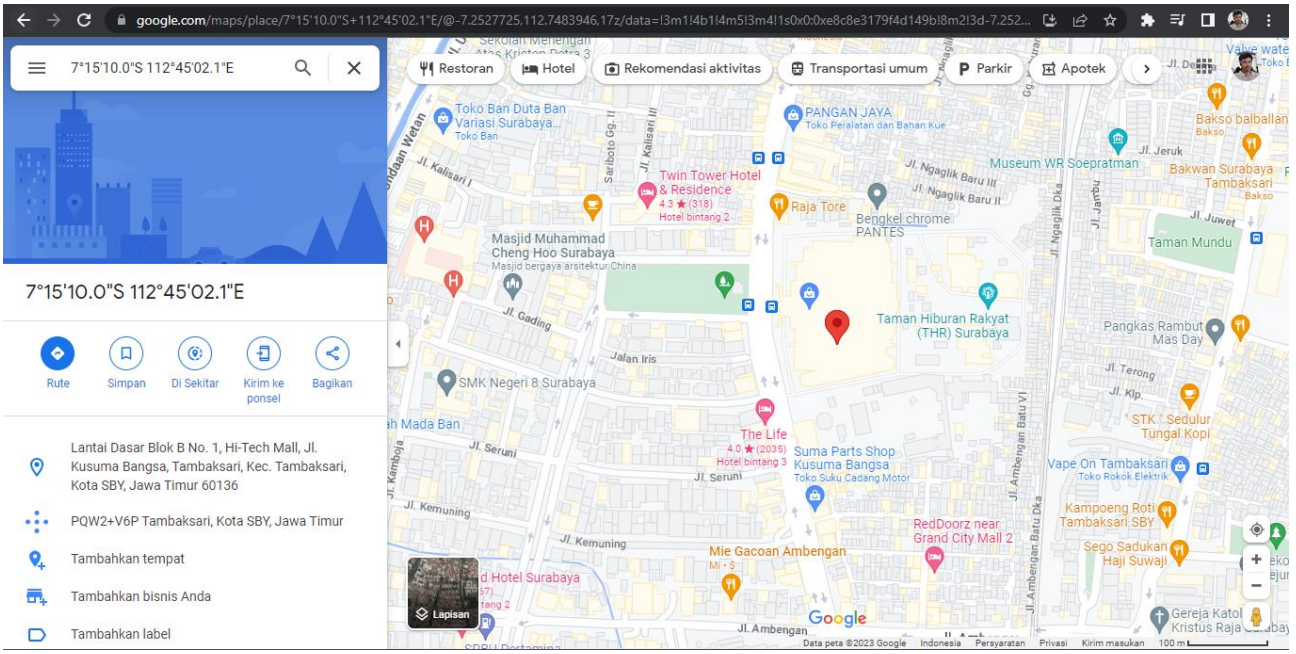
Dari soal ctf ini, kami diberikan hint

> I found a place that give me a backroom vibes. I think I like this place, so I give this place 5 star. Can you find this place?

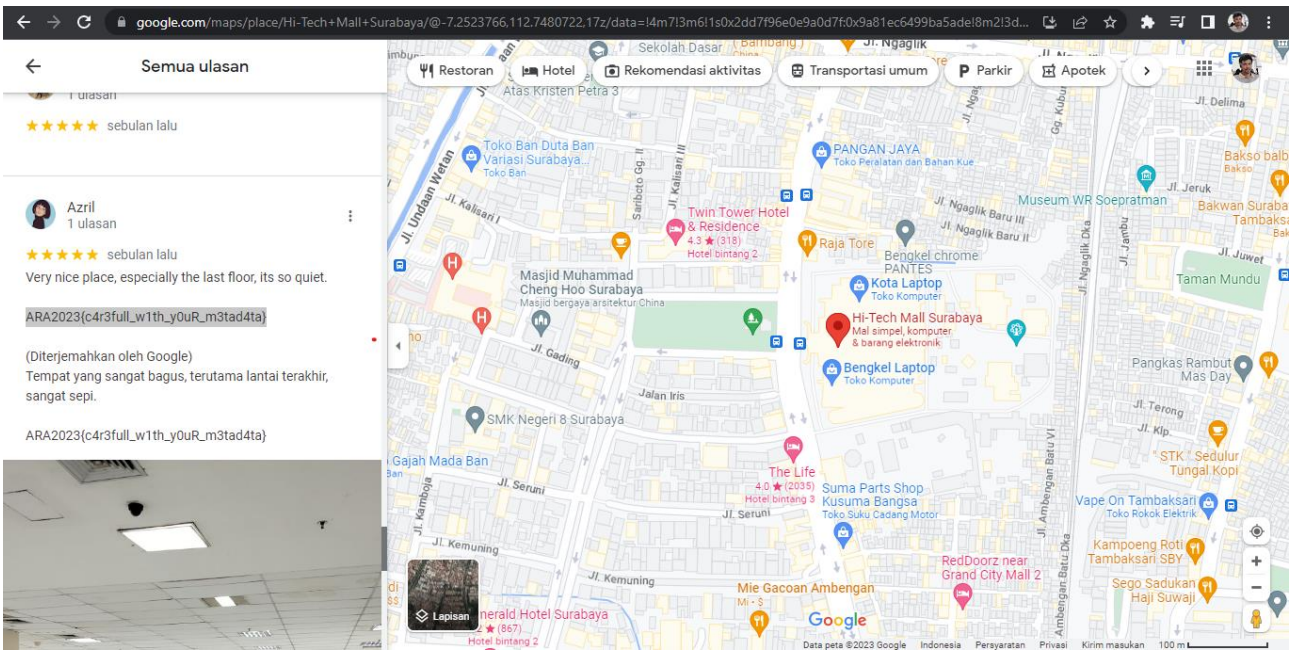
Lalu dari hint tersebut, kami mencari lokasi dari foto tersebut, kami menemukan lokasi dari meta data gambar tersebut menggunakan Exiftool



Lalu kami mencoba mencari lokasi menggunakan google maps, dan mendapatkan lokasinya berada pada ‘Hi-Tech Mall Surabaya’



Lali kami mencari ulasan tersebut di google maps, dan mendapatkan flag



C. Web Exploitation

1. Dewaweb

   Kami diberikan link http:/103.152.242.116:8417/

   Lalu saat kami menelusuri common case nya, terdapat part flag yang di command pada source web tersebut, lalu kami mendapatkan semua part yaitu part 1-4 yang ada di:

   - Part 1 = Source code web – '/'
   - Part 2 = custom js - '/js/custom.js'
   - Part 3 = custom css – '/css/style.css'
   - Part 4 = Header respond web

   yang jika di satukan akan mendapat flag

   ARA2023{s4nt4I_dUlu_g4k_s1h?XD}

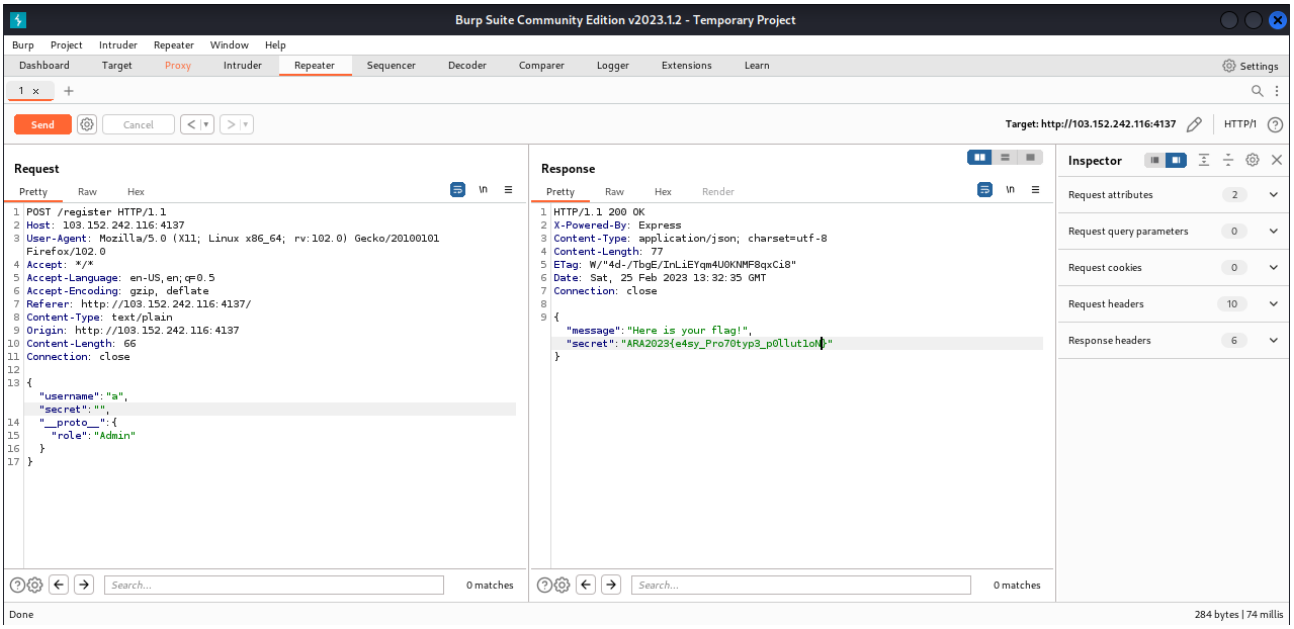
2. Pollution

   Kami diberikan link http://103.152.242.116:4137/ dan juga attachments yang berisi source code web tersebut, di source code nya kami menemukan file secret.js yang kemungkinan besar berisi flag.

   Lalu saat kami analisis program nya, kami menemukan bahwa secret di return jika `newUser.role === "Admin"` yang jika di lihat sekilas, program tidak mungkin bisa akses ke code tersebut karena di line 22 sudah di lakukan pengecekan (if) yang sama dan newUser dengan user berisi variable yang sama.

```
18  app.post('/register', (req, res) => {
19      let user = JSON.parse(req.body);
20
21      // Haha, even you can set your role to Admin, but you don't have the secret!
22      if (user.role == "Admin") {
23          console.log(user.secret);
24          if(user.secret !== secret.value) return res.send({
25              "message": "Wrong secret! no Admin!"
26          });
27          return res.send({
28              "message": "Here is your flag!",
29              secret: secret.value
30          });
31      }
32
33      let newUser = Object.assign(baseUser, user);
34      if(newUser.role === "Admin") {
35          return res.send({
36              "message": "Here is your flag!",
37              secret: secret.value
38          });
39      }
40
41      else return res.send({
42          "message": "No Admin? no flag!"      You, 11 hours ago • add file ara-ctf
43      });
44  })
```

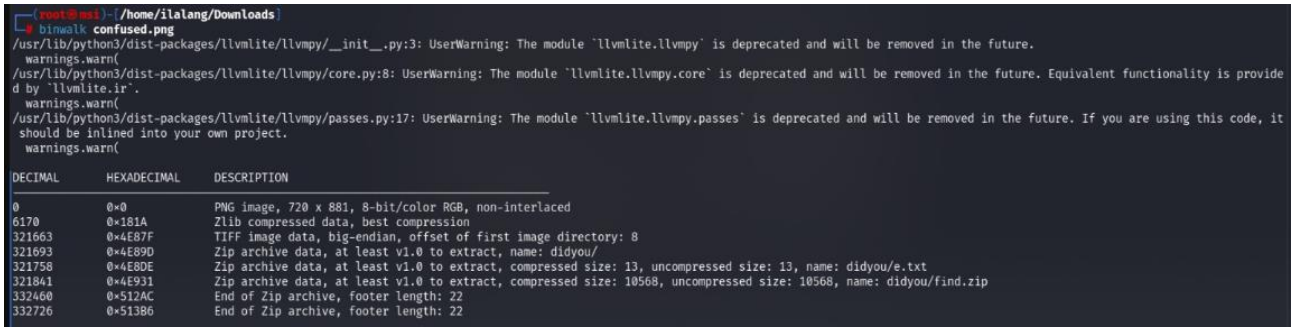
Lalu kami menganalisa nya lebih lanjut, dan kami menemukan sesuatu yang mencurigakan dengan assign ulang di code program tersebut (line 33, `Object.assign(baseUser, user), kami memikirkan kemungkinan adanya exploit di program tersebut, lalu kami mencoba mencari refrensi merupakan exploit pollution `Object.assign()` dan menemukan refrensi https://security.snyk.io/vuln/SNYK-JS-NESTEDOBJECTASSIGN-1065977, lalu kami menemukan bahwa exploit web tersebut adalah Prototype Pollution, lalu kami mencobanya dengan burp dan mendapat flag nya



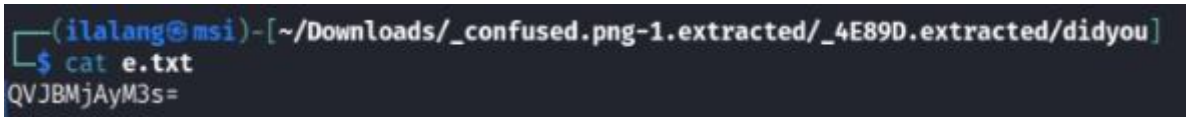
D. Forensic

1. Thinker

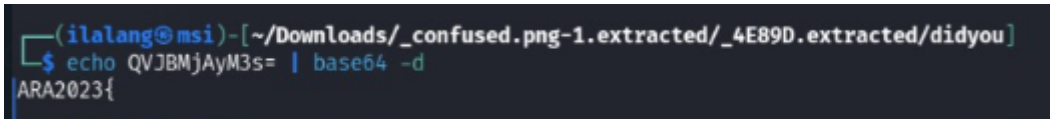
Pertama kami diberikan attachments link google drive yang berisi gambar confused.jpg
Saat di bin walk, kami menemukan file zip



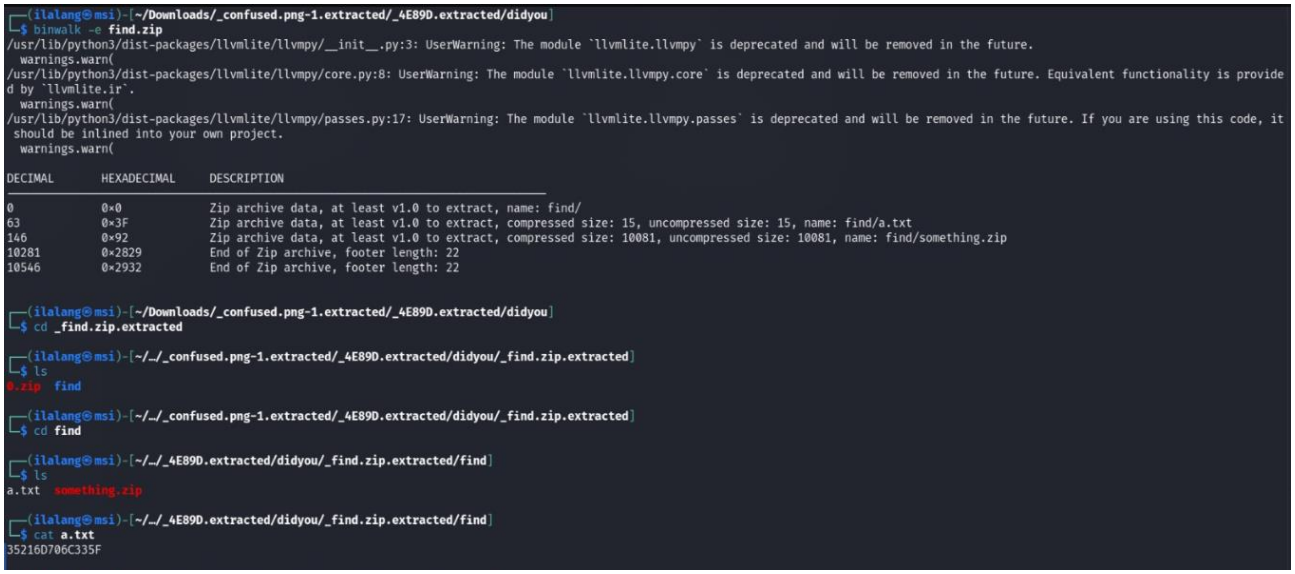
Saat kami menelusuri hasil extract gambar tersebut, kami menemukan file e.txt yang berisi text base64 encoded



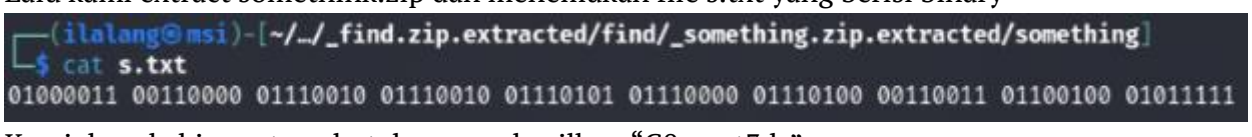
Lalu kami decode base64 lalu mendapatkan hasil yang merupakan awal dari flag tersebut,



Lalu kami extract find.zip dan menemukan file a.txt yang berisi hex encoded



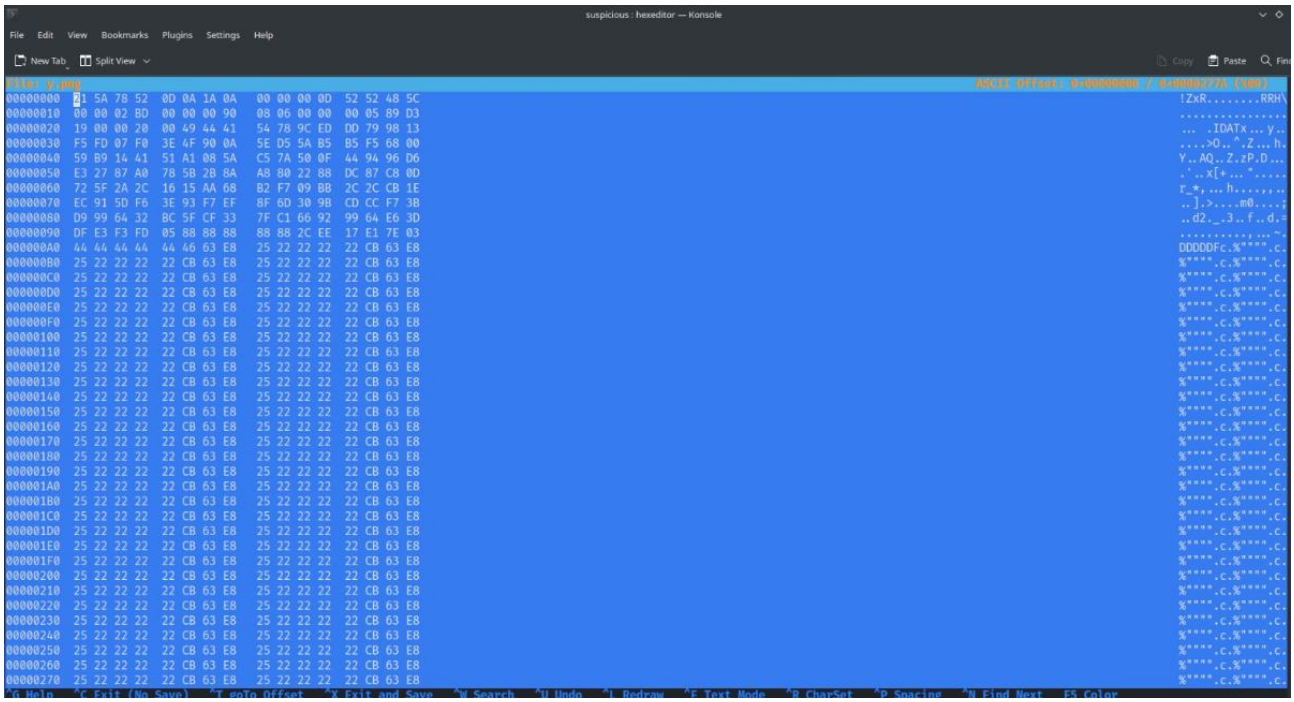
Lalu kami decode hex tersebut dan menhasilkan “5!mpl3_”
Lalu kami extract somethink.zip dan menemukan file s.txt yang berisi binary



Kami decode binary tersebut dan menghasilkan “C0rrupt3d_”
Lalu kami extract suspicious.zip dan menemukan file for y.png

```
┌──(ilalang㉿msi)-[~/../_find.zip.extracted/find/_something.zip.extracted/something]
└─$ binwalk -e suspicious.zip
/usr/lib/python3/dist-packages/llvmlite/llvmpy/__init__.py:3: UserWarning: The module `llvmlite.llvmpy` is deprecated and will be removed in the future.
  warnings.warn(
/usr/lib/python3/dist-packages/llvmlite/llvmpy/core.py:8: UserWarning: The module `llvmlite.llvmpy.core` is deprecated and will be removed in the future. Equivalent functionality is provided by `llvmlite.ir`.
  warnings.warn(
/usr/lib/python3/dist-packages/llvmlite/llvmpy/passes.py:17: UserWarning: The module `llvmlite.llvmpy.passes` is deprecated and will be removed in the future. If you are using this code, it should be inlined into your own project.
  warnings.warn(

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zip archive data, at least v1.0 to extract, name: suspicious/
69            0x45            Zip archive data, at least v2.0 to extract, compressed size: 9204, uncompressed size: 10107, name: suspicious/y.png
9514          0x252A          End of Zip archive, footer length: 22


┌──(ilalang㉿msi)-[~/../_find.zip.extracted/find/_something.zip.extracted/something]
└─$ cd _suspicious.zip.extracted

┌──(ilalang㉿msi)-[~/../find/_something.zip.extracted/something/_suspicious.zip.extracted]
└─$ ls
0.zip  suspicious

┌──(ilalang㉿msi)-[~/../find/_something.zip.extracted/something/_suspicious.zip.extracted]
└─$ cd suspicious

┌──(ilalang㉿msi)-[~/../_something.zip.extracted/something/_suspicious.zip.extracted/suspicious]
└─$ ls
y.png
```

Lalu saat kami membukanya, foto tersebut telah corrupt, lalu kami membukanya dengan hex editor



Bisa dihilihat bahwa header foto tersebut telah corrupt, lalu kami mengganti nya menjadi ‘89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52’

Dan kami dapat membuka foto tersebut yang berisi

Dapat dilihat bahwa itu adala encode code ascii, lalu kami decode code tersebut dan menghasilkan “1m4ge5}”, lalu kami mengumpulkan semua partnya dan menjadi **ARA2023{5!mpl3_C0rrupt3d_1m4ge5}**

E. Misc

1. @B4SH

Pertama-tama kami diberikan hint yaitu

```
"5A495A323032337B346D62793077625F67733066397
3675F677334675F2167355F345F733468733F7D".
```

Sekilas dapat dilihat bahwa text tersebut merupakan hex encoded, lalu kami decode dan mendapatkan hasil

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Misc\@B4SH\try\unhex.py"
ZIZ2023{4mby0wb_gs0f9sg_gs4g_!g5_4_s4hs?}

[Done] exited with code=0 in 1.121 seconds
```

Kami rasa itu bukanlah flag yang benar, lalu kami menganalisisnya lebih lanjut, dan kami menemukan bahwa itu telah di encrypt menggunakan affine cipher, dan kami mendecrypt nya menggunakan python

```python
def mod_inv (num, mod):
    for x in range(0,mod + 1):
        if ((num*x)%mod == 1):
            return x

def brute(cipher, knowed_word = None):
    if not knowed_word == None:
        if len(knowed_word) > len(cipher):
            return 'error length'
    decipher = ""
    for i in range(0,26):
        if (i%2 =/= 0) and (i =/= 13):
            for j in range(0,26):
                inv = mod_inv(i,26)
                decipher_temp = ''
                for c in cipher:
                    v = ord(c)
                    if (v >= 65) and (v <= 90):
                        cip = ((v - 65 - j)*inv + 26)%26 + 65
                    elif (v >= 97) and (v <= 122):
                        cip = ((v - 97 - j)*inv + 26)%26 + 97
                    else:
                        cip = v
                    decipher_temp += chr(cip)
                if not knowed_word == None:
                    if knowed_word in decipher_temp:
                        decipher += decipher_temp+'\n'
                    continue
                decipher += decipher_temp+'\n'
    return decipher

cipher = "ZIZ2023{4mby0wb_gs0f9sg_gs4g_!g5_4_s4hs?}"

print(brute(cipher, 'ARA2023'))
```

Lalu kami mendapatkan hasil

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Misc\@B4SH\try\affine.py"
ARA2023{4nyb0dy_th0u9ht_th4t_!t5_4_h4sh?}


[Done] exited with code=0 in 0.266 seconds
```

2. D0t N D4sh3s

   Pertama-tama kami diberikan Chall File berupa link google drive, yang berisi dot dan dash, yang merupakan code morse

   Lalu kami decode code morse dan mendapatkan hasil binary lalu saya mendecode nya

```
MORSE_CODE = { 'A':'.-', 'B':'-...',
                    'C':'-.-.', 'D':'-..', 'E':'.',
                    'F':'..-.', 'G':'--.', 'H':'....',
                    'I':'..', 'J':'.---', 'K':'-.-',
                    'L':'.-..', 'M':'--', 'N':'-.',
                    'O':'---', 'P':'.--.', 'Q':'--.-',
                    'R':'.-.', 'S':'...', 'T':'-',
                    'U':'..-', 'V':'...-', 'W':'.--',
                    'X':'-..-', 'Y':'-.--', 'Z':'--..',
                    '1':'.----', '2':'..---', '3':'...--',
                    '4':'....-', '5':'.....', '6':'-....',
                    '7':'--...', '8':'---..', '9':'----.',
                    '0':'-----', ', ':'--..--', '.':'.-.-.-',
                    '?':'..--..', '/':'-..-.', '-':'-....-',
                    '(':'-.--.', ')':'-.--.-'}

def decrypt_message(morse):
    morse += ' '
    decipher = ''
    citext = ''
    for letter in morse:
        if letter == '/':
            continue
        if letter != ' ':
            i = 0
            citext += letter
        else:
            i += 1
            if i == 2 :
                decipher += ' '
            else:
                decipher +=
list(MORSE_CODE.keys())[list(MORSE_CODE.values()).index(citext)]
                citext = ''
    return decipher

def binary_to_text(binaryed):
    temp = ""
    text = ""
    for b in binaryed:
        if b == ' ':
            text += chr(int(temp, 2))
            temp = ""
        else :
            temp += b
    if len(temp) == 8:
        text += chr(int(temp, 2))
    return text
```

```
binary = decrypt_message("----- .---- ----- ----- ----- ----- ----- .---- / ----
- .---- ----- .---- ----- ----- .---- ----- / ----- .---- ----- ----- ----- ----
- ----- .---- / ----- ----- .---- .---- ----- ----- .---- ----- / ----- ----- .-
--- .---- ----- ----- ----- ----- / ----- ----- .---- .---- ----- ----- .---- --
--- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- .---- .---- .----
.---- ----- .---- .---- / ----- ----- .---- ----- ----- ----- ----- .---- / ----
- .---- .---- .---- ----- .---- ----- ----- / ----- .---- .---- .---- ----- ----
- .---- .---- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .-
--- ----- .---- ----- .---- ----- / ----- .---- .---- .---- ----- .---- ----- .-
--- / ----- ----- .---- .---- ----- .---- ----- .---- / ----- .---- .---- .----
----- .---- ----- ----- / ----- .---- ----- .---- .---- .---- .---- .---- / ----
- ----- .---- .---- ----- .---- ----- ----- / ----- .---- ----- .---- .---- .---
- .---- .---- / ----- .---- .---- ----- .---- .---- ----- .---- / ----- ----- .-
--- .---- ----- ----- ----- ----- / ----- .---- .---- .---- ----- ----- .---- --
--- / ----- .---- .---- .---- ----- ----- .---- .---- / ----- ----- .---- .----
----- ----- .---- .---- / ----- .---- ----- .---- .---- .---- .---- .---- / ----
- .---- .---- ----- ----- ----- ----- .---- / ----- .---- .---- ----- ----- .---
- .---- ----- / ----- .---- .---- .---- ----- .---- ----- ----- / ----- ----- .-
--- .---- ----- ----- .---- .---- / ----- ----- .---- .---- ----- ----- ----- .-
--- / ----- ----- .---- .---- ----- ----- .---- ----- / ----- .---- ----- .----
.---- .---- .---- .---- / ----- .---- .---- ----- ----- ----- ----- .---- / ----
- ----- .---- .---- ----- ----- ----- .---- / ----- ----- .---- ----- ----- ----
- ----- .---- / ----- .---- .---- .---- .---- .---- ----- .----")
print(binary_to_text(binary))
```

Lalu kami mendapatkan hasil yang merupakan flag

```
[Running] python -u "d:\Programming\Cyber Security\CTF-its-ara 2023\Misc\D0ts N D4sh3s\try\conv.py"
ARA2023{!ts_ju5t_4_m0rs3_aft312_a1!}

[Done] exited with code=0 in 0.249 seconds
```

3. In-sanity check

Pertama-tama kami diberikan attachments yang merupakan link docs.google.com,
Saat kami membukanya, file tersebut dapat dirubah oleh semua orang, termasuk para participant, lalu kami membuka history file tersebut, dan menemukan flag pada file history tanggal 22 Februari