**Write Up CTF-ARA-2023**

**Team : Haripotar**

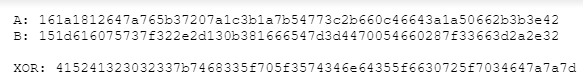
**Asal : SMKN 04 Malang**

**Anggota :**

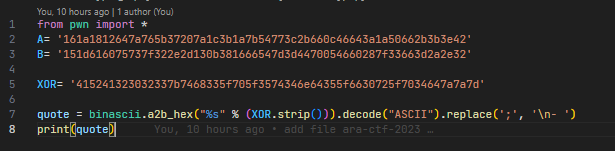
* **Fikri Muhammad Abdillah (FlaB) - Captain**
* **Firda Gheitsa Sahira (abcfirdaa)**

1. **Cryptography**
2. One Time Password (?)

Kami mendapatkan link google drive yang berisi,



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

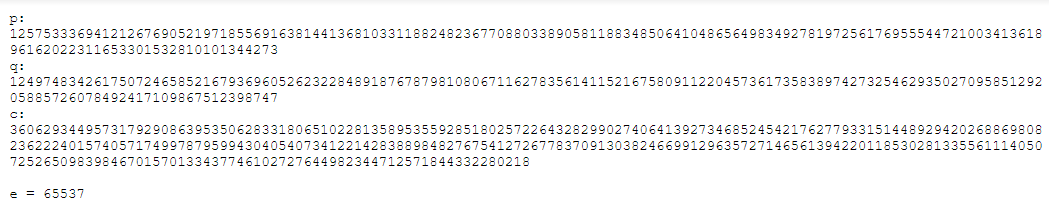


Lalu saat kami mendecode variable hex, flag langsung terlihat di hasil decode tersebut

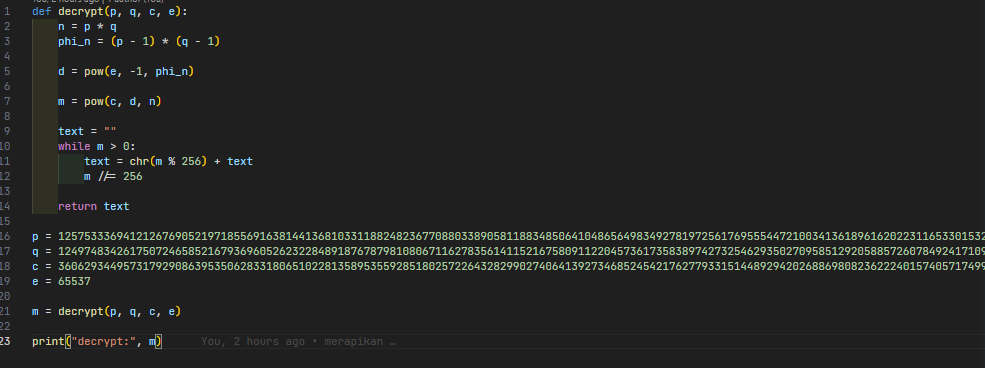


1. Secrets Behind a Letter

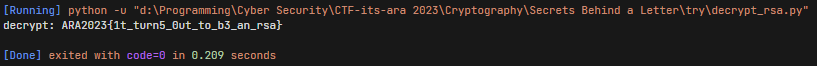
Kami mendapatakan Attachments yang berisi link google drive yang berisi,



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



Lalu mendapatkan hasil yang berupa flag

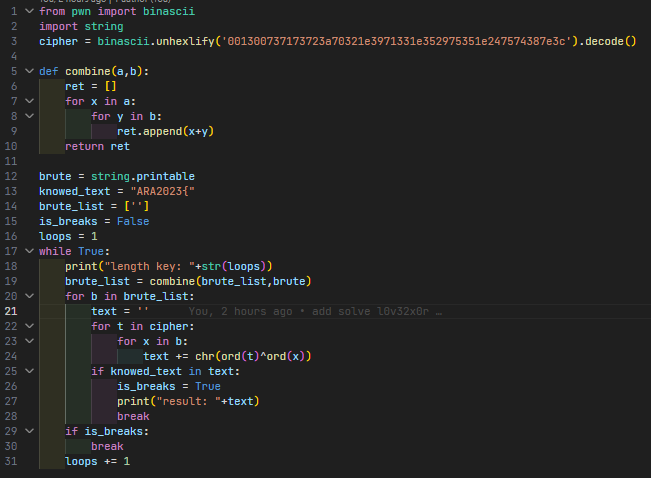


1. L0V32X0R

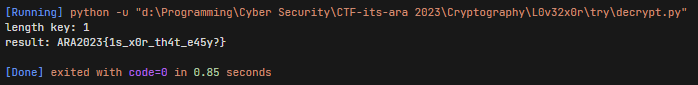
Kami mendapatkan tulisan text dari soal



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



Lalu kami menemukan flag dari hasil program diatas



1. SH4-32

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



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

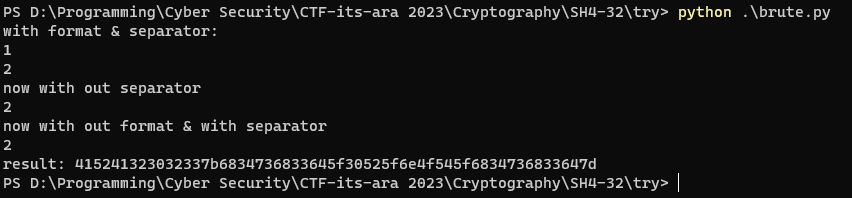
            else:

                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



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



1. babychall

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



Dapat dilihat, bahwa itu sepertinya ada RSA-CRT (Chineese Reminder Theorem),

Lalu saya memakai program python

c1=50996973104845663108379751131203085432412490198312714663656823648233038479298192861451834246930208140110173699058527919020115432586705400467345647806522331396447650847650133013246673390879222719169248862420278256322967718701700458729207793124758166438641448112314489945863231881982352790765130535004090053677

c2=2675086354476975422055414666795504683242305948200761348250028401266882028494792724072473530888031343997988485639367375927974100307107406775103695198800703704181414736281388464205429123159605048186634852771717909704864647112817586024682299987868607933059634279556321476204813521201682662328510086496215821461

c3=37230658243252590743608571105027357862790972987208833213017941171448753815654839901699526651433771324826895355671255944414893947963934979068257310367315935701270804390799121669635153012916402271190722618997500392911737767143316552376495882986935695146970853914275481717400268832644987157988727575513351441919

n1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343315900732228502875706659244844711538497850413046440270578916645981161000807526427004236918404837363404678029443944950655102252423415631977020625826867728898231382737396728896847618010577420408630133

n2=93105621059686474816890215494554802831518948420160941703522759121619785851270608634130307450227557987976818162331982289634215037184075864787223681218982602092806757888533587126974091077190242797461318907280759075612577475534626062060960739269828789274137274363970056276139434039315860052556417340696998509271

n3=65918509650742278494971363290874849181268364316012656769339120004000702945271942533097529884964063109377036715847176196280943807261986848593000424143320280053279021411394267268255337783494901606319687457351586915314662800434632332988978858085931586830283694881538759008360486661936884202274973387108214754101

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

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)

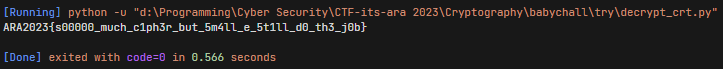
for n, c in zip(N, C):

    assert a % n == c

m = inv\_pow(a, e)

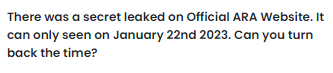
print(bytes.fromhex(hex(m)[2:]).decode())

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



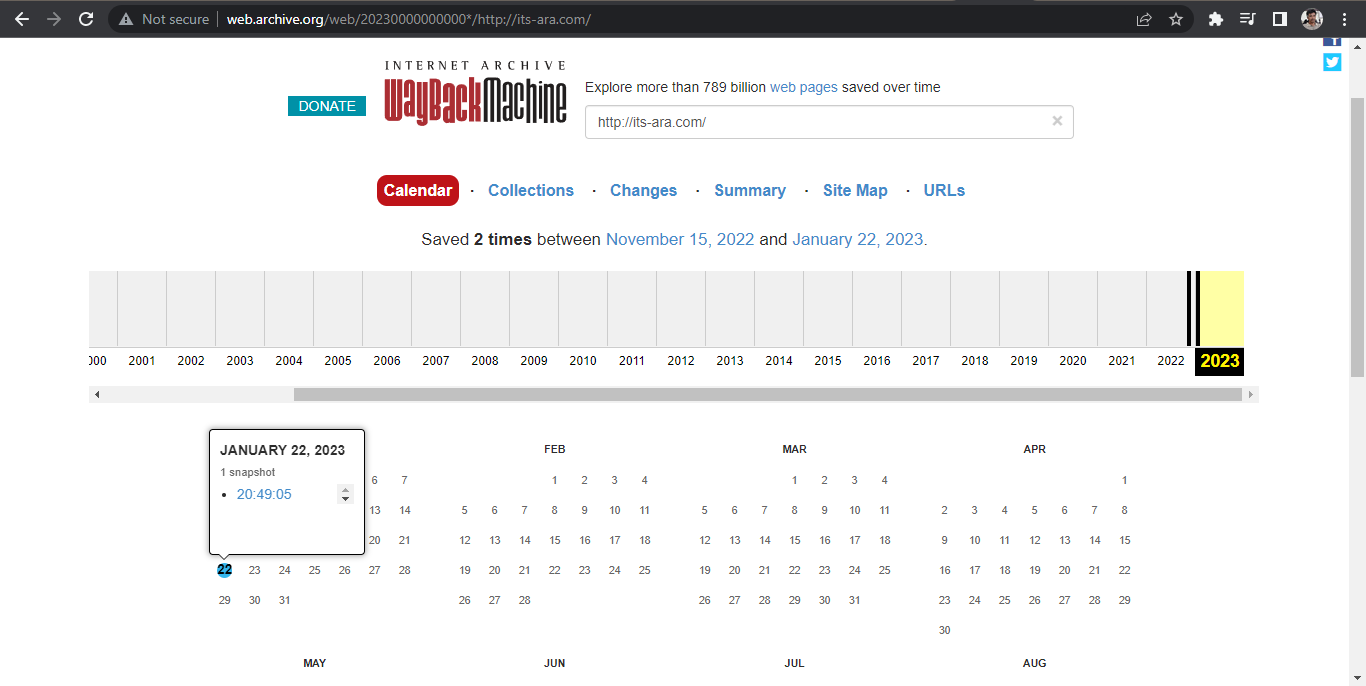
1. **OSINT**
2. Time Machine

Kami mendapatkan soal

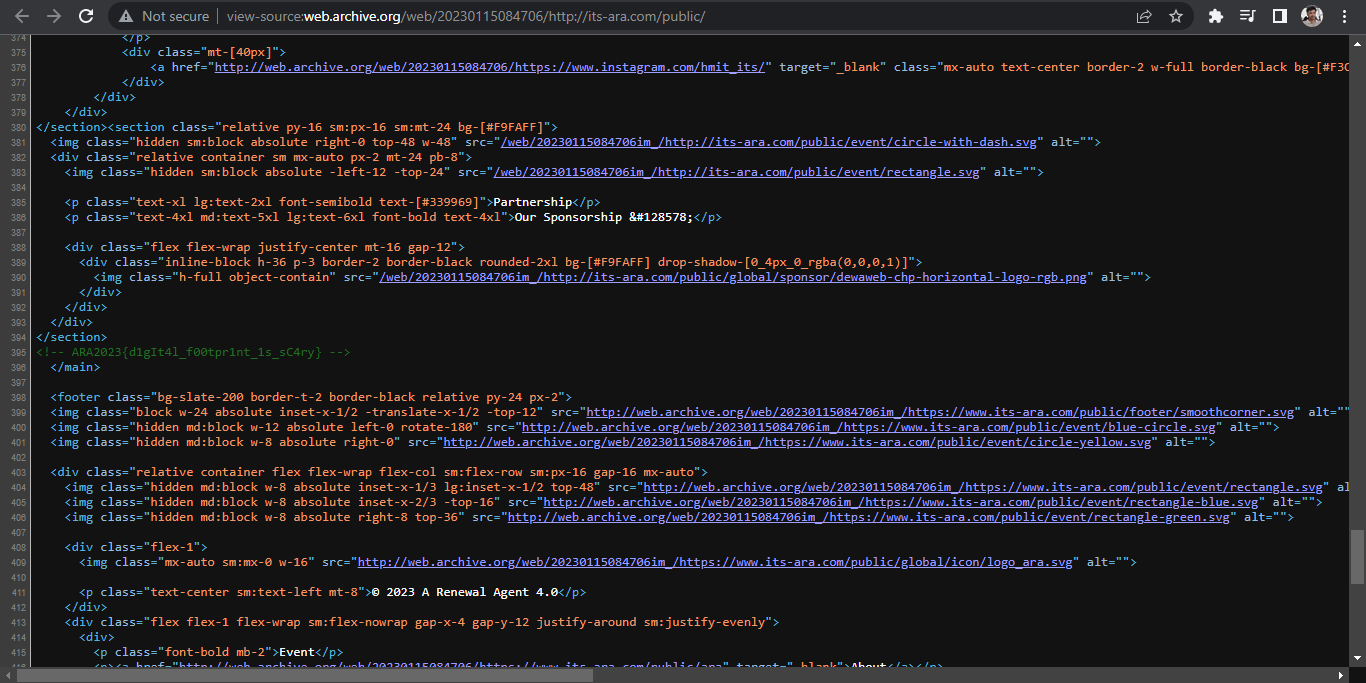


Saat kami membaca soal tersebut, kami langsung mencoba memakai <web.archive.org> dan memasukkan link website ARA

Lalu kami menemukan ada archive tanggal 22 januari



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



1. Backroom

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

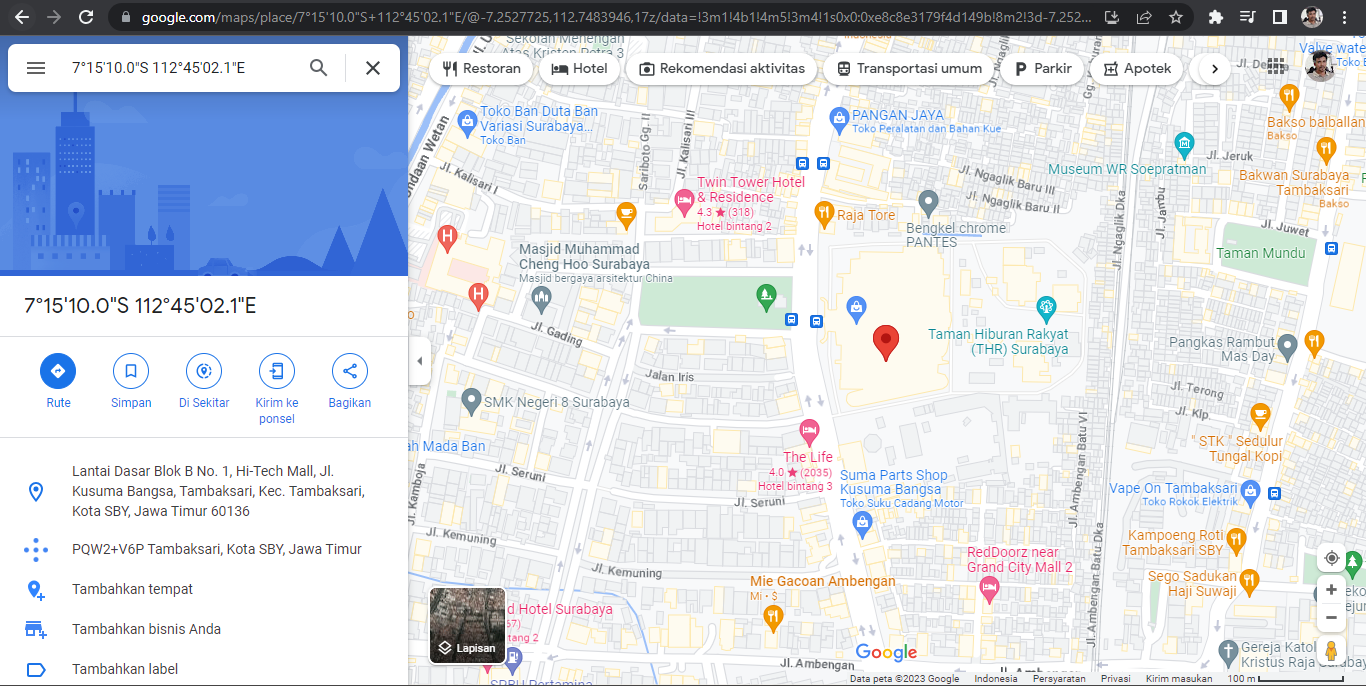
Dari soal ctf ini, kami diberikan hint



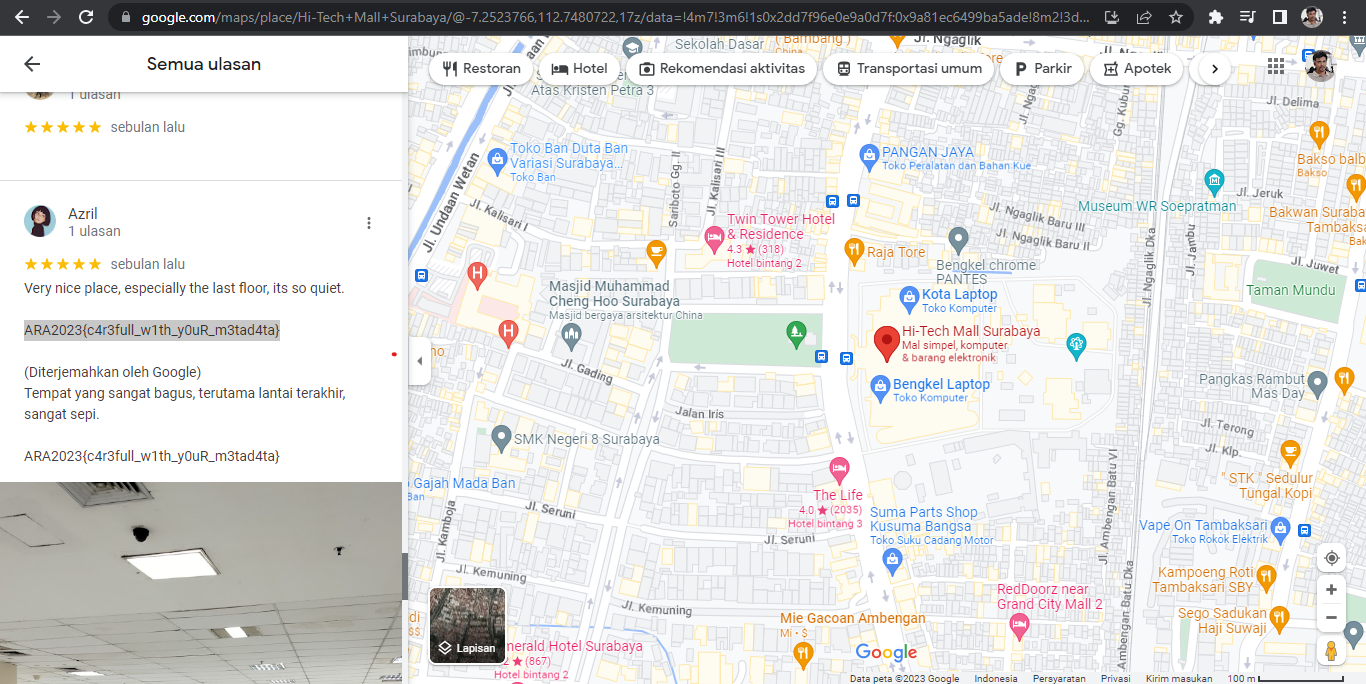
Lalu dari hint tersebut, kami mencari lokasi dari foto tersebut, kami menemukan lokasi dari meta data gambar tersebut menggunakan Exiftool



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



1. **Web Exploitation**
2. Dewaweb

Kami diberikan link [http:/103.152.242.116:8417/](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

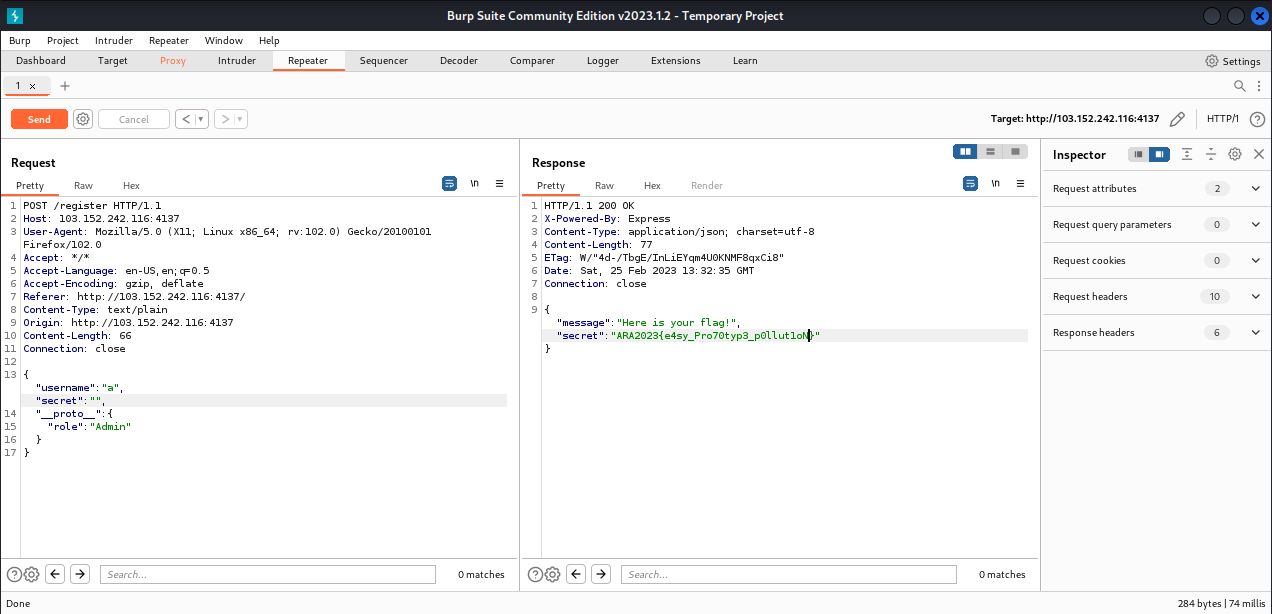
**ARA2023{s4nt4I\_dUlu\_g4k\_s1h?XD}**

1. 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.

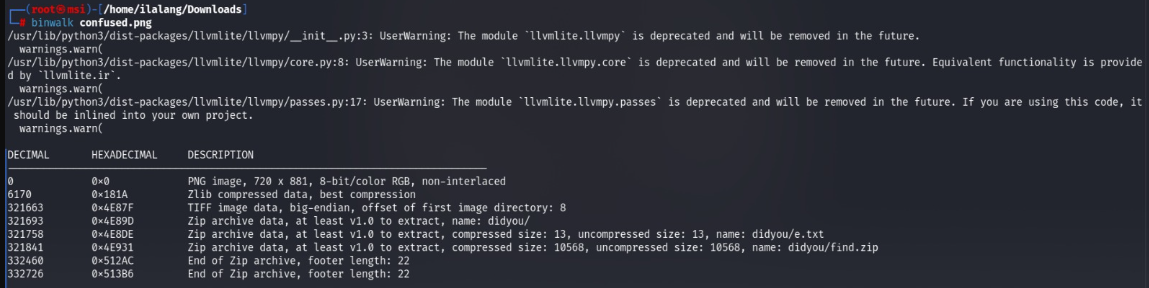
 Lalu kami menganalisa nya lebih lanjut, dan kami menemukan sesuatu yang mencurigakan dengan assign ulang di code program tersebut (line 33, `Object.assign(baseUser, user), kami memikirkan kemungkinan adanya exploit di program tersebut, lalu kami mencoba mencari refrensi merupakan exploit pollution `Object.assign()` 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



1. **Forensic**
2. Thinker

Pertama kami diberikan attachments link google drive yang berisi gambar confused.jpg

Saat di bin walk, kami menemukan file zip



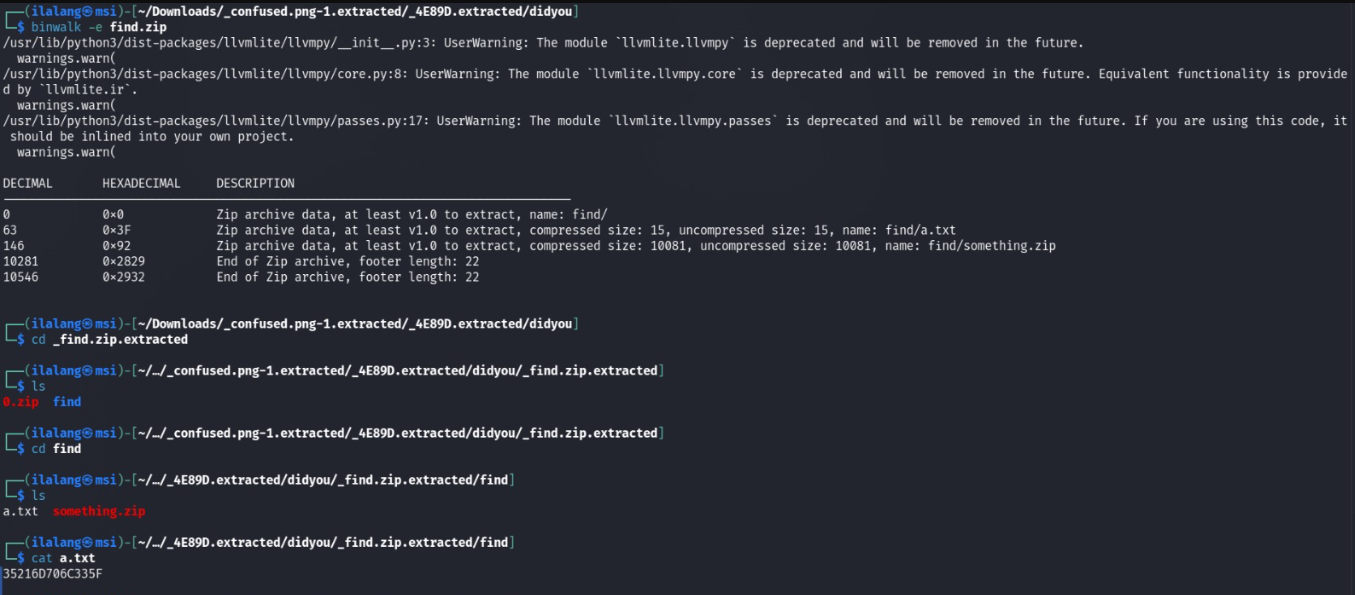
Saat kami menelusuri hasil extract gambar tersebut, kami menemukan file e.txt yang berisi text base64 encoded



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



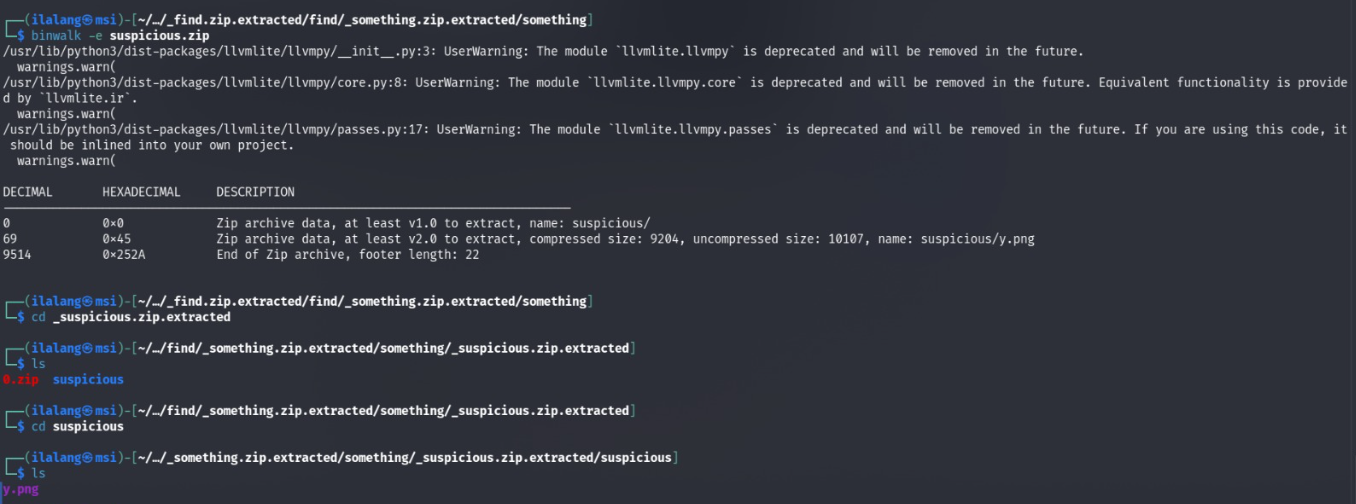
Lalu kami decode hex tersebut dan menhasilkan “5!mpl3\_”

Lalu kami extract somethink.zip dan menemukan file s.txt yang berisi binary

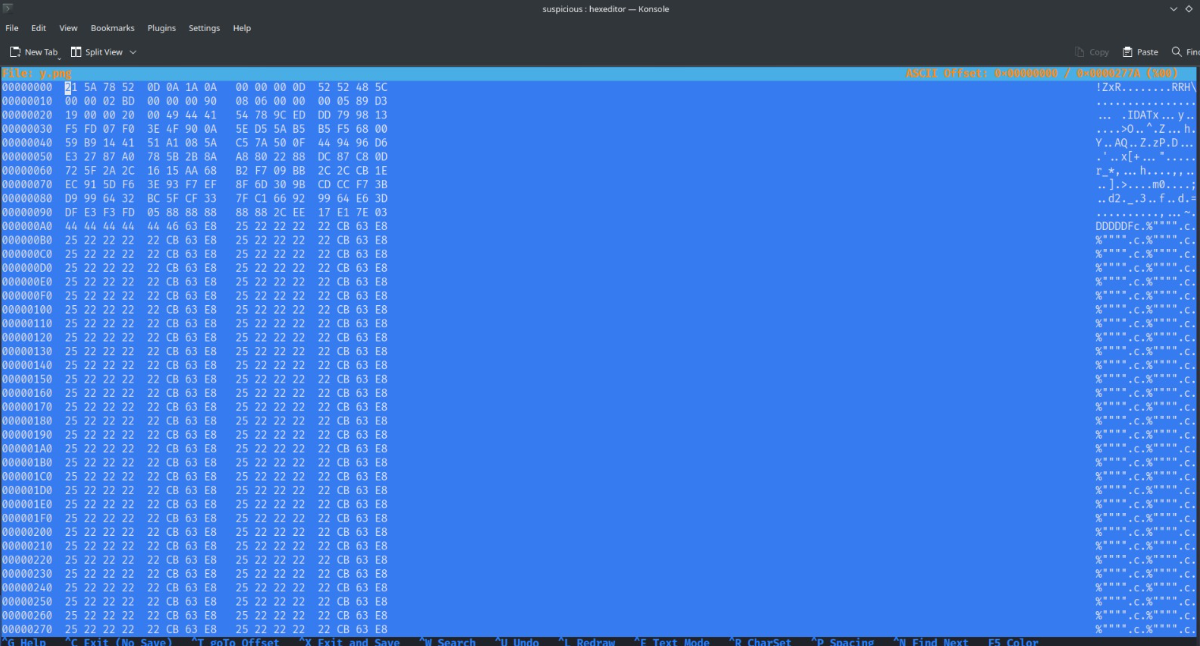


Kami decode binary tersebut dan menghasilkan “C0rrupt3d\_”

Lalu kami extract suspicious.zip dan menemukan file for y.png



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

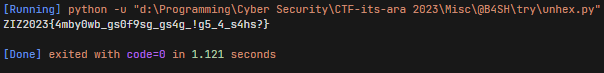
**ARA2023{5!mpl3\_C0rrupt3d\_1m4ge5}**

1. **Misc**
2. @B4SH

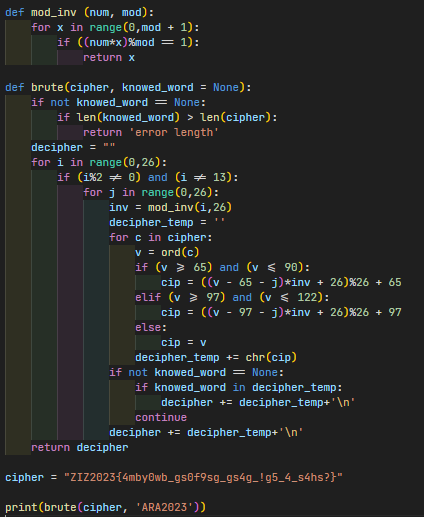
Pertama-tama kami diberikan hint yaitu



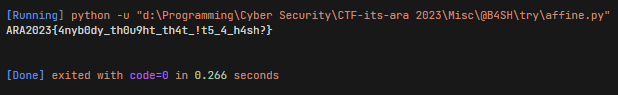
Sekilas dapat dilihat bahwa text tersebut merupakan hex encoded, lalu kami decode dan mendapatkan hasil



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



Lalu kami mendapatkan hasil



1. 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:

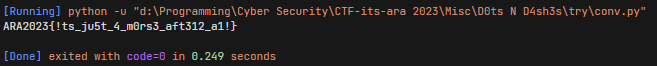
        text += chr(int(temp, 2))

    return text

binary = decrypt\_message("----- .---- ----- ----- ----- ----- ----- .---- / ----- .---- ----- .---- ----- ----- .---- ----- / ----- .---- ----- ----- ----- ----- ----- .---- / ----- ----- .---- .---- ----- ----- .---- ----- / ----- ----- .---- .---- ----- ----- ----- ----- / ----- ----- .---- .---- ----- ----- .---- ----- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- .---- .---- .---- .---- ----- .---- .---- / ----- ----- .---- ----- ----- ----- ----- .---- / ----- .---- .---- .---- ----- .---- ----- ----- / ----- .---- .---- .---- ----- ----- .---- .---- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- .---- ----- .---- ----- / ----- .---- .---- .---- ----- .---- ----- .---- / ----- ----- .---- .---- ----- .---- ----- .---- / ----- .---- .---- .---- ----- .---- ----- ----- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- ----- .---- .---- ----- .---- ----- ----- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- .---- .---- ----- .---- / ----- ----- .---- .---- ----- ----- ----- ----- / ----- .---- .---- .---- ----- ----- .---- ----- / ----- .---- .---- .---- ----- ----- .---- .---- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- ----- ----- ----- .---- / ----- .---- .---- ----- ----- .---- .---- ----- / ----- .---- .---- .---- ----- .---- ----- ----- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- ----- .---- .---- ----- ----- ----- .---- / ----- ----- .---- .---- ----- ----- .---- ----- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- ----- ----- ----- .---- / ----- ----- .---- .---- ----- ----- ----- .---- / ----- ----- .---- ----- ----- ----- ----- .---- / ----- .---- .---- .---- .---- .---- ----- .----")

print(binary\_to\_text(binary))

Lalu kami mendapatkan hasil yang merupakan flag



1. In-sanity check

Pertama-tama kami diberikan attachments yang merupakan link [docs.google.com](https://www.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

