Write Up

Capture The Flag

ARA 2023

Written by ctf masbro team

Nur Muhammad A Y as masnurrm
Elshe Erviana A as matryochska
Dhafin Almas N as fhinnn

Thanks to

HMIT (Himpunan Mahasiswa Teknologi Informasi) ITS periode 2021-2022 as the organizer

Repository write up can be accessed on the link below.
https://github.com/elshiraphine/ctf-writeup/tree/main/2023-ARA

masnurrm feat: dewaweb ... 1 hour ago History

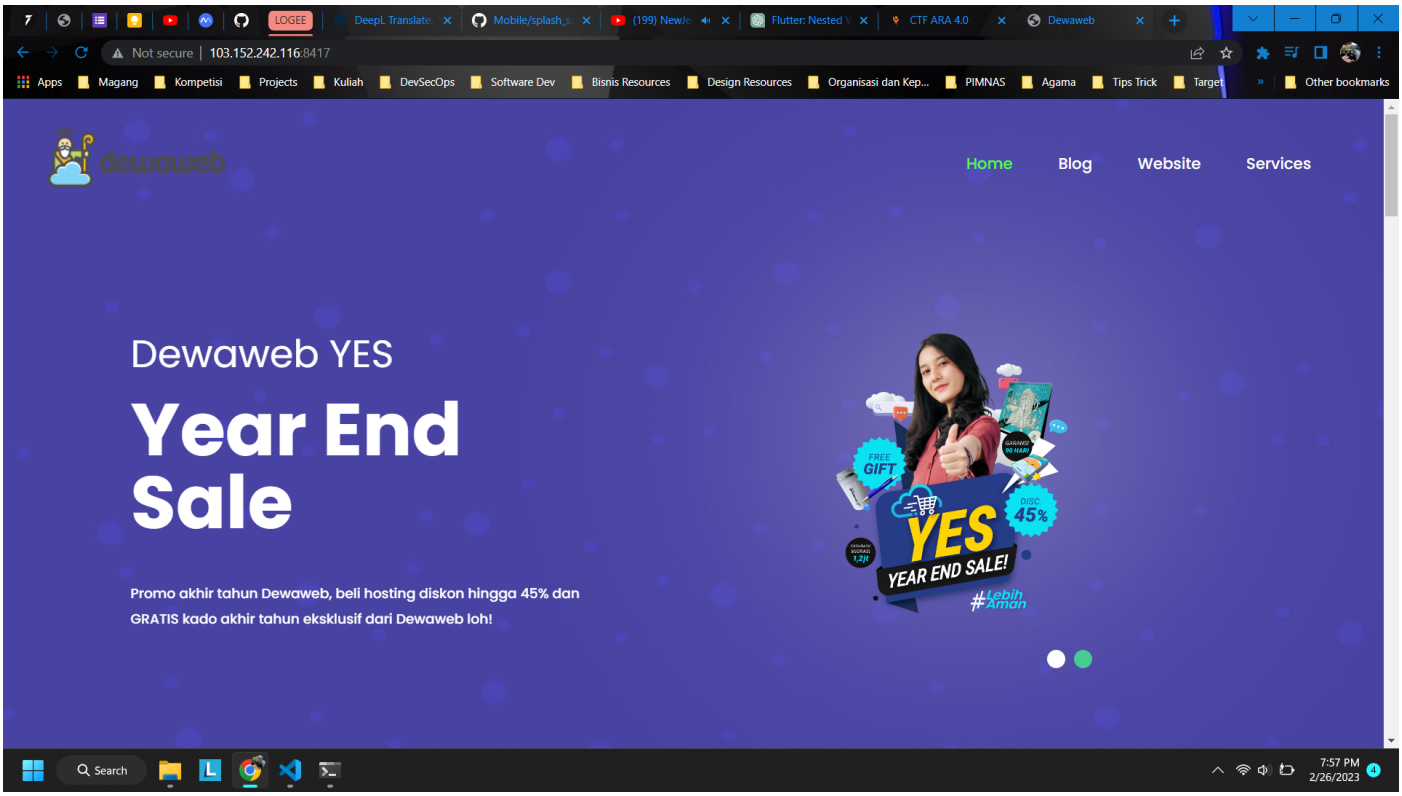
..

assets	1 hour ago
readme.md	1 hour ago

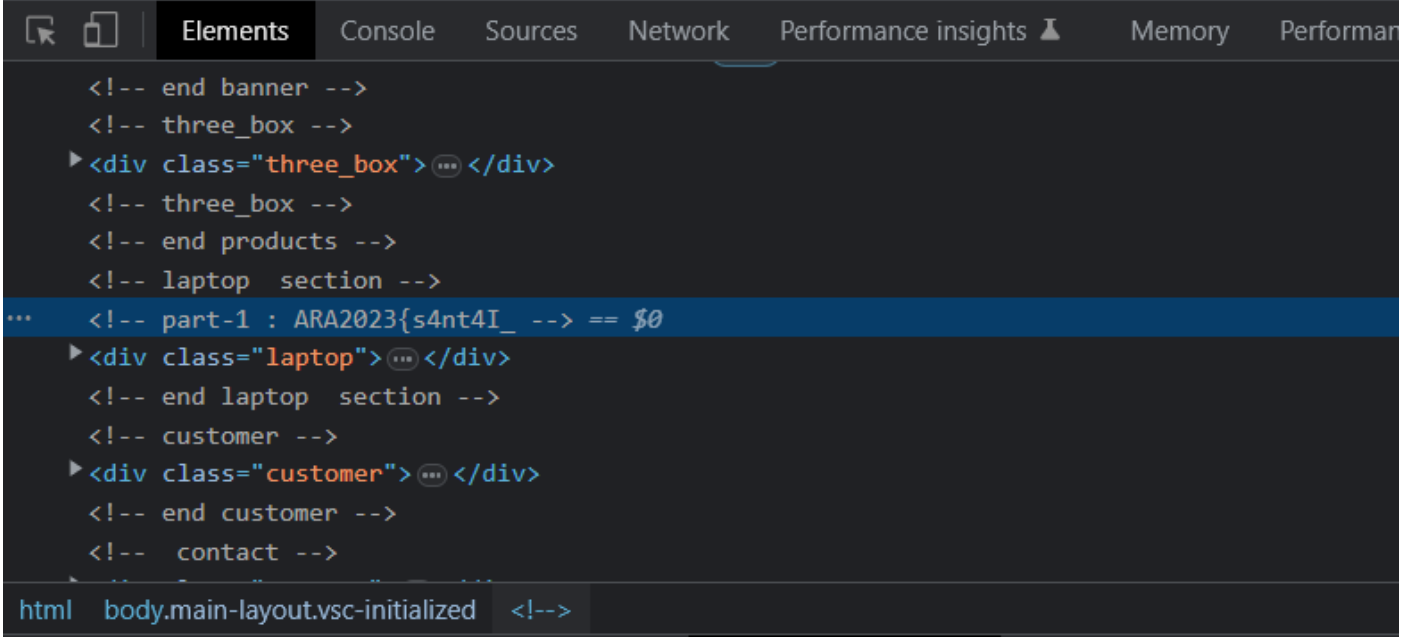
readme.md

Web Exploitation - Dewaweb

In this challenge, we were given an IP Address of a website.

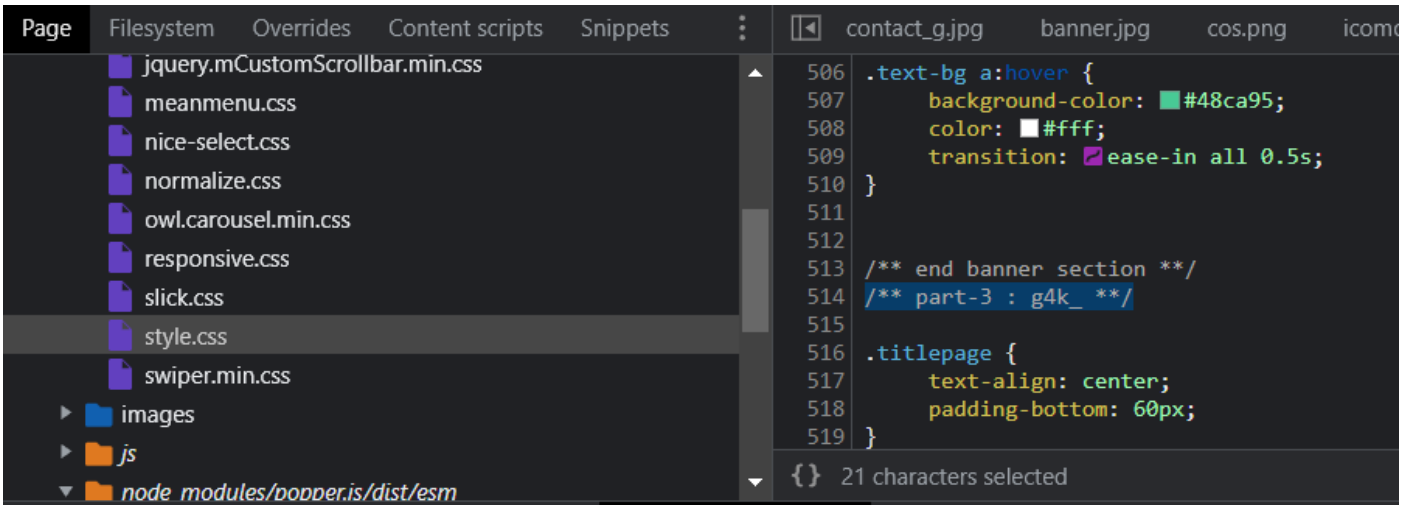


Our approach was checking the sources of the website. First, we checked the HTML source of the given website. It gives us the first part of the flag here.



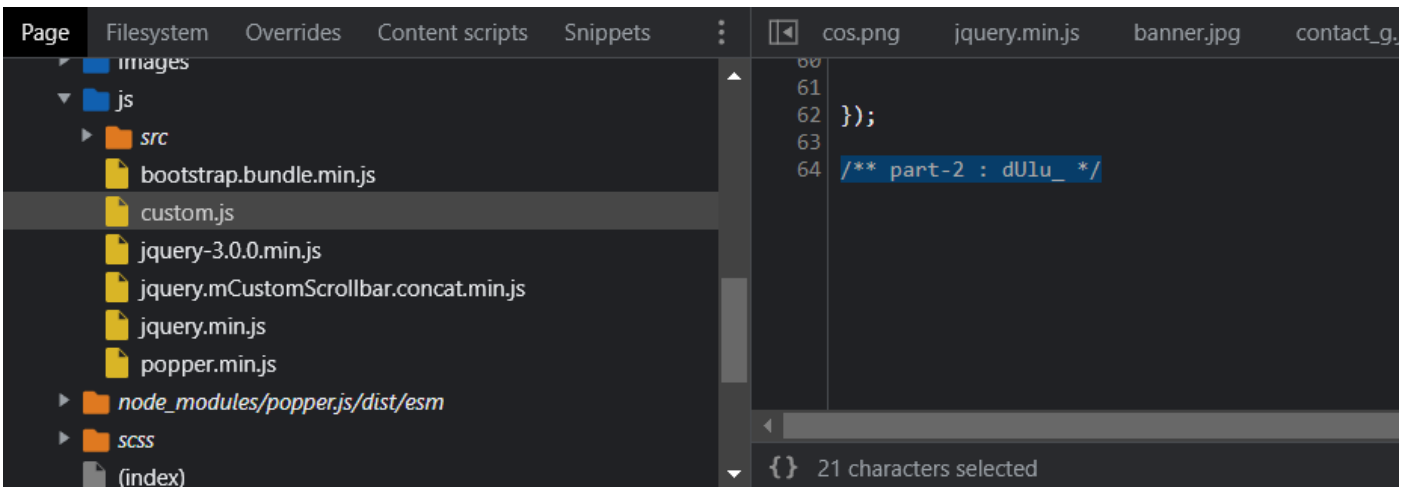

```
part-1 : ARA2023{s4nt4I_
```

Next, we try to check the CSS sources of the website. It gives us the third part of the flag instead of the second part. It located in `/css/style.css`.



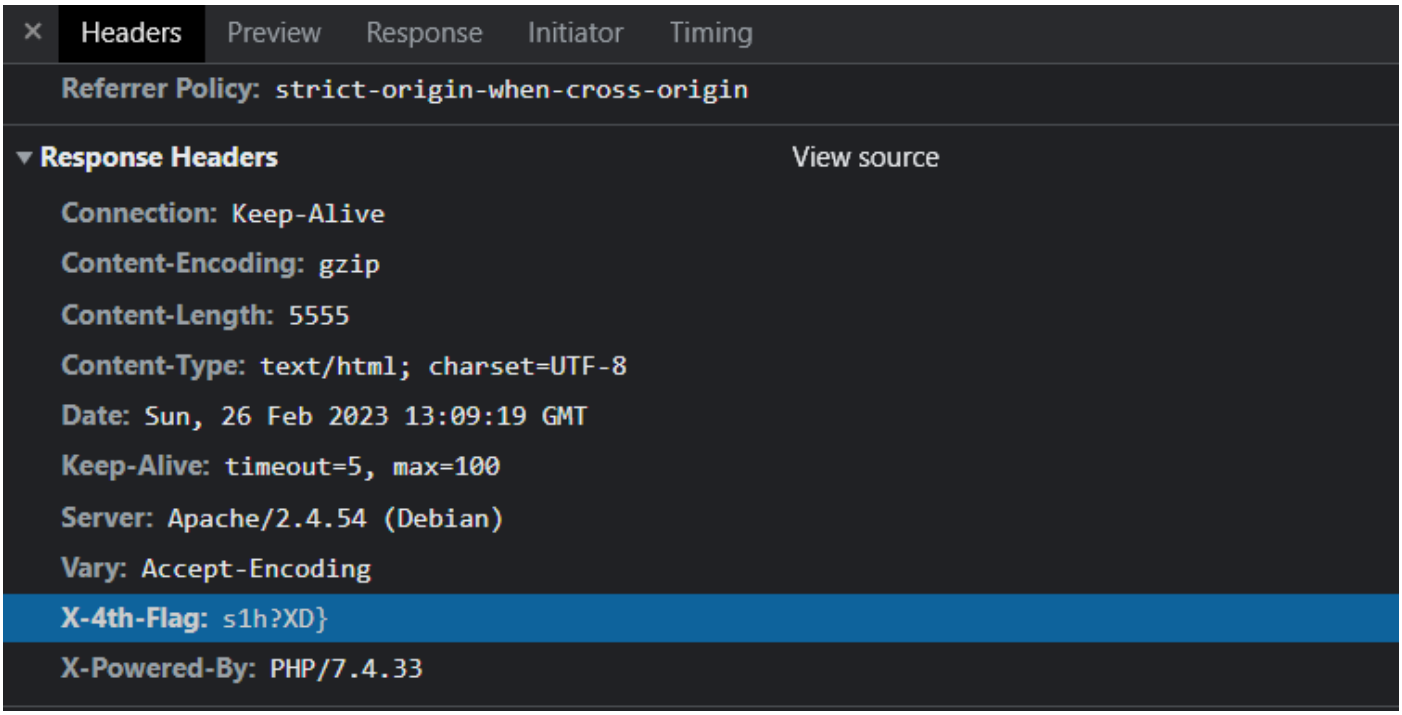

```
part-3 : g4k_
```

Then, we check the JS sources of the website. It gives us the second part of the flag here. It located in `/js/custom.js`.



```
part-2 : dUlu_
```

The last part, we got it in response header.




```
X-4th-Flag: s1h?XD}
```

So, when all of part were arranged, it will be full flag.

```
ARA2023{s4nt4I_dUlu_g4k_s1h?XD}
```

Give feedback

masnurrm feat: Time MAchine ... 1 hour ago History

..

assets	1 hour ago
readme.md	1 hour ago

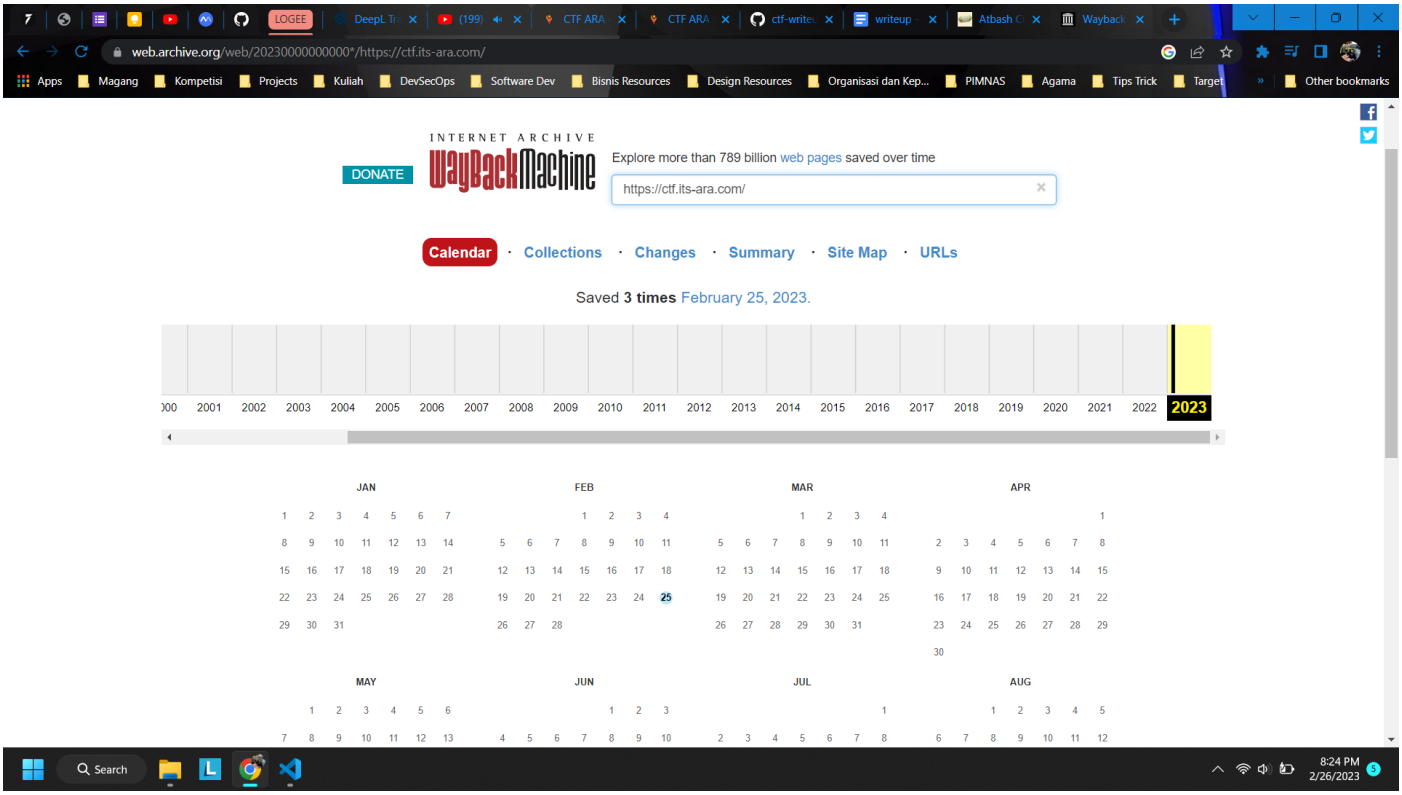
readme.md

OSINT - Time Machine

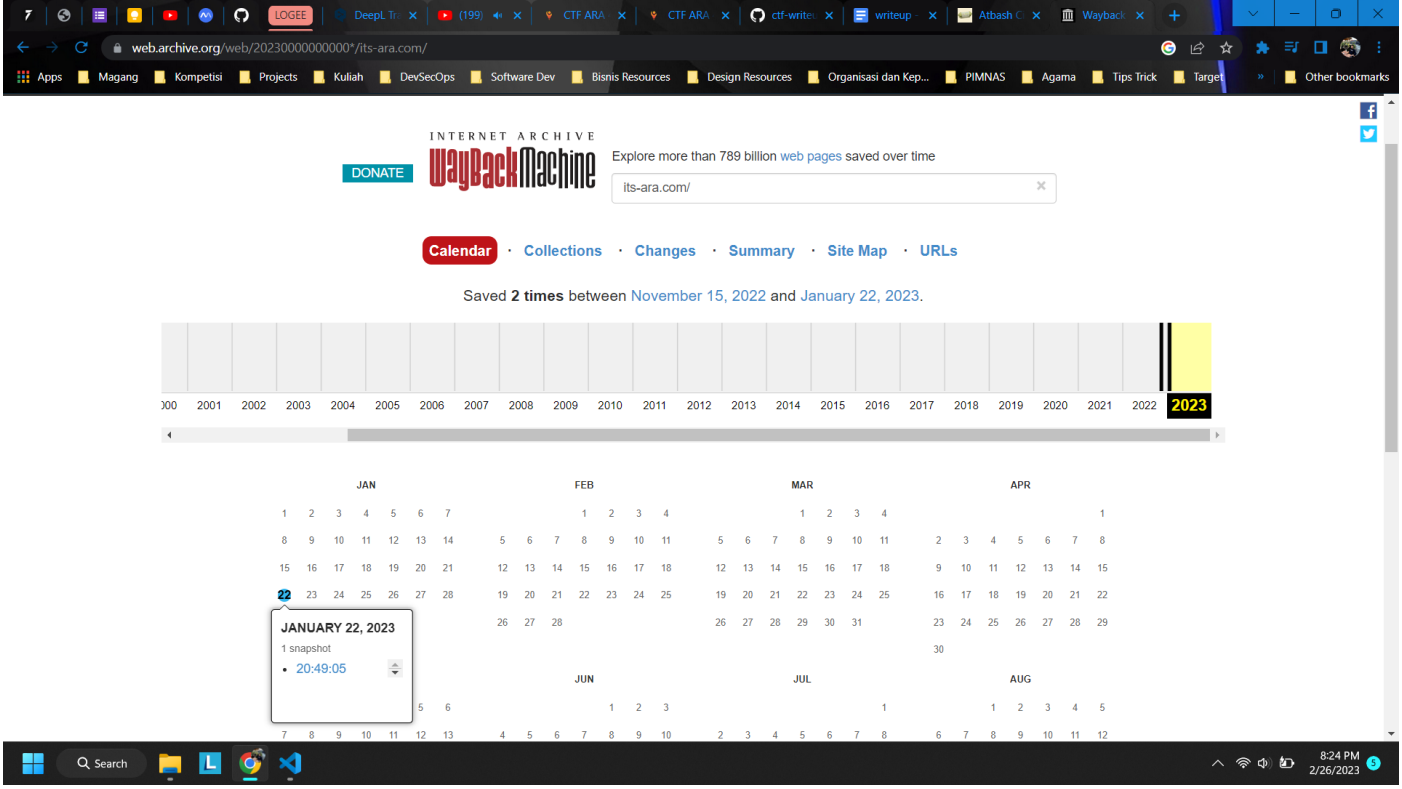
In this challenge, we were given these clue.

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

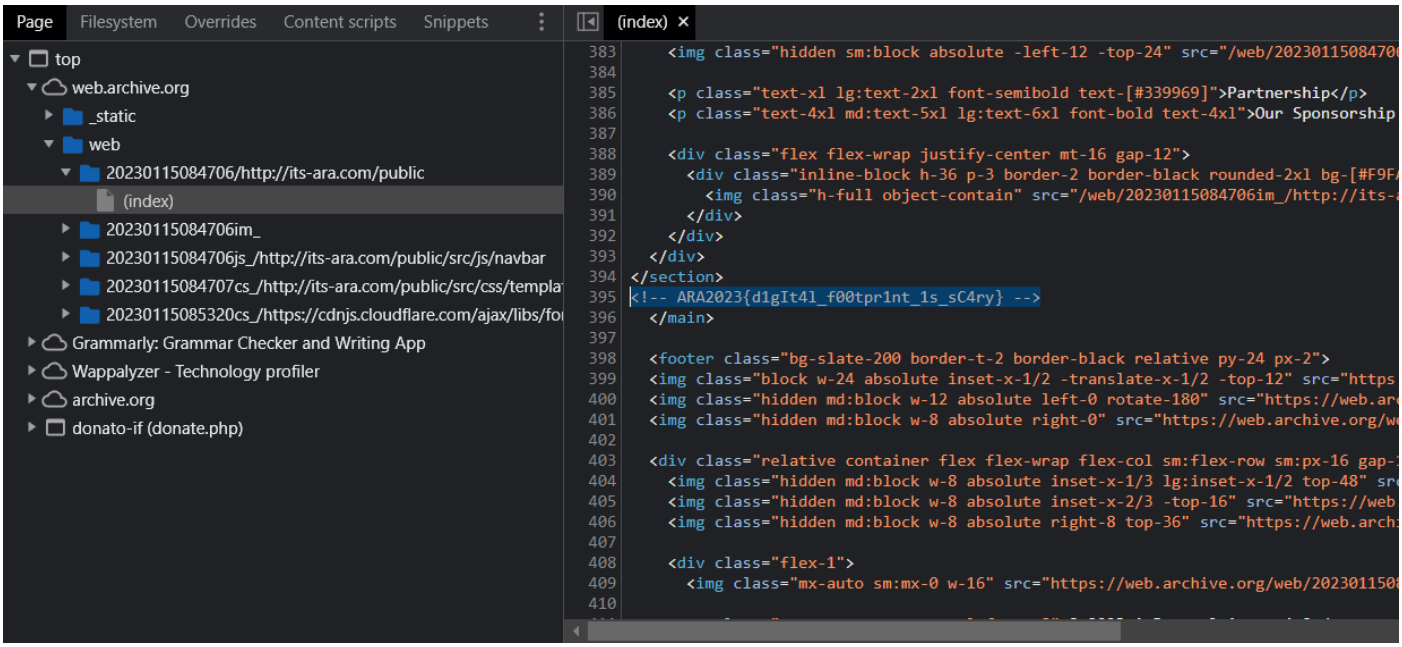
So, we try to see the history or tracking back the website into January 22nd of 2023 using WayBack Machine. First try we got tricked. It doesn't show the result as what we expected.



Then, we realized that the clue were only said 'Official ARA Website', not the 'ARA CTF Website. So, we try to remove the subdomain. We got it.



And we finally got the flag in `index` file.



So, this is the flag.

```
ARA2023{d1gIt4l_f00tpr1nt_1s_sC4ry}
```

Give feedback

masnurrm feat: inshanity check ... 25 minutes ago History

..	
assets	38 minutes ago
readme.md	25 minutes ago

readme.md

OSINT - Backroom

In this challenge, we were given an attachment as follows.

Or can be accessed by this URL.

First time, we try to search it on Google Search by Image. But, we didn't find anything. Then, we try to analyze this image with `exiftool`. Then, we got these information.

```
File Name                       : IMG20221221153220.jpg
Directory                       : .
File Size                       : 2.1 MiB
File Modification Date/Time     : 2023:02:26 20:41:58+07:00
File Access Date/Time           : 2023:02:26 20:45:16+07:00
File Inode Change Date/Time     : 2023:02:26 20:41:58+07:00
File Permissions                : -rwxrwxrwx
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Exif Byte Order                 : Little-endian (Intel, II)
Make                            : realme
Camera Model Name               : realme 9
Orientation                     : Horizontal (normal)
X Resolution                    : 72
Y Resolution                    : 72
Resolution Unit                 : inches
Modify Date                     : 2022:12:21 15:32:20
Y Cb Cr Positioning             : Centered
Interoperability Index          : R98 - DCF basic file (sRGB)
Interoperability Version        : 1.0
Exposure Time                   : 1/50
F Number                        : 1.8
Exposure Program                : Program AE
ISO                             : 1125
Exif Version                    : 0220
Date/Time Original              : 2022:12:21 15:32:20
Create Date                     : 2022:12:21 15:32:20
Offset Time Original            : +07:00
Components Configuration        : Y, Cb, Cr, -
Shutter Speed Value             : 1/50
Aperture Value                  : 1.7
Brightness Value                : 42949671.79
Exposure Compensation           : 0
Max Aperture Value              : 1.7
Metering Mode                   : Center-weighted average
Flash                           : Off, Did not fire
Focal Length                    : 5.2 mm
Maker Note Unknown Text         : (Binary data 147 bytes, use -b option to extract)
Warning                         : Invalid EXIF text encoding for UserComment
User Comment                    : oplus_32
Sub Sec Time                    : 831
Sub Sec Time Original           : 831
Sub Sec Time Digitized          : 831
Flashpix Version                : 0100
Color Space                     : sRGB
Exif Image Width                : 0
Exif Image Height               : 0
Sensing Method                  : Not defined
Scene Type                      : Directly photographed
Exposure Mode                   : Auto
White Balance                   : Auto
Focal Length In 35mm Format     : 24 mm
Scene Capture Type              : Standard
GPS Version ID                  : 2.2.0.0
GPS Time Stamp                  : 08:32:20
GPS Map Datum                   : WGS-84
GPS Date Stamp                  : 2022:12:21
XMP Toolkit                     : Image::ExifTool 10.96
Date/Time Digitized             : 2022:12:21 15:32:20+07:00
Date Created                    : 2022:12:21 15:32:20.831+07:00
Profile CMM Type                : Apple Computer Inc.
Profile Version                 : 4.0.0
Profile Class                   : Display Device Profile
Color Space Data                : RGB
```

```
Profile Connection Space        : XYZ
Profile Date Time               : 2018:06:24 13:22:32
Profile File Signature          : acsp
Primary Platform                : Apple Computer Inc.
CMM Flags                       : Not Embedded, Independent
Device Manufacturer             : Unknown (OPPO)
Device Model                    :
Device Attributes               : Reflective, Glossy, Positive, Color
Rendering Intent                : Perceptual
Connection Space Illuminant     : 0.9642 1 0.82491
Profile Creator                 : Apple Computer Inc.
Profile ID                      : 0
Profile Description             : Display P3
Profile Copyright               : Copyright Apple Inc., 2017
Media White Point               : 0.95045 1 1.08905
Red Matrix Column               : 0.51512 0.2412 -0.00105
Green Matrix Column             : 0.29198 0.69225 0.04189
Blue Matrix Column              : 0.1571 0.06657 0.78407
Red Tone Reproduction Curve     : (Binary data 32 bytes, use -b option to extract)
Chromatic Adaptation            : 1.04788 0.02292 -0.0502 0.02959 0.99048 -0.01706 -0.00923
0.01508 0.75168
Blue Tone Reproduction Curve    : (Binary data 32 bytes, use -b option to extract)
Green Tone Reproduction Curve   : (Binary data 32 bytes, use -b option to extract)
Image Width                     : 2256
Image Height                    : 4000
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Aperture                        : 1.8
Image Size                      : 2256x4000
Megapixels                      : 9.0
Scale Factor To 35 mm Equivalent: 4.6
Shutter Speed                   : 1/50
Create Date                     : 2022:12:21 15:32:20.831
Date/Time Original              : 2022:12:21 15:32:20.831+07:00
Modify Date                     : 2022:12:21 15:32:20.831
GPS Date/Time                   : 2022:12:21 08:32:20Z
GPS Latitude                    : 7 deg 15' 9.97" S
GPS Longitude                   : 112 deg 45' 2.06" E
GPS Latitude Ref                : South
GPS Longitude Ref               : East
Circle Of Confusion             : 0.007 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
```

It feels so clueless until we read the description of the challenge again.

> 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?

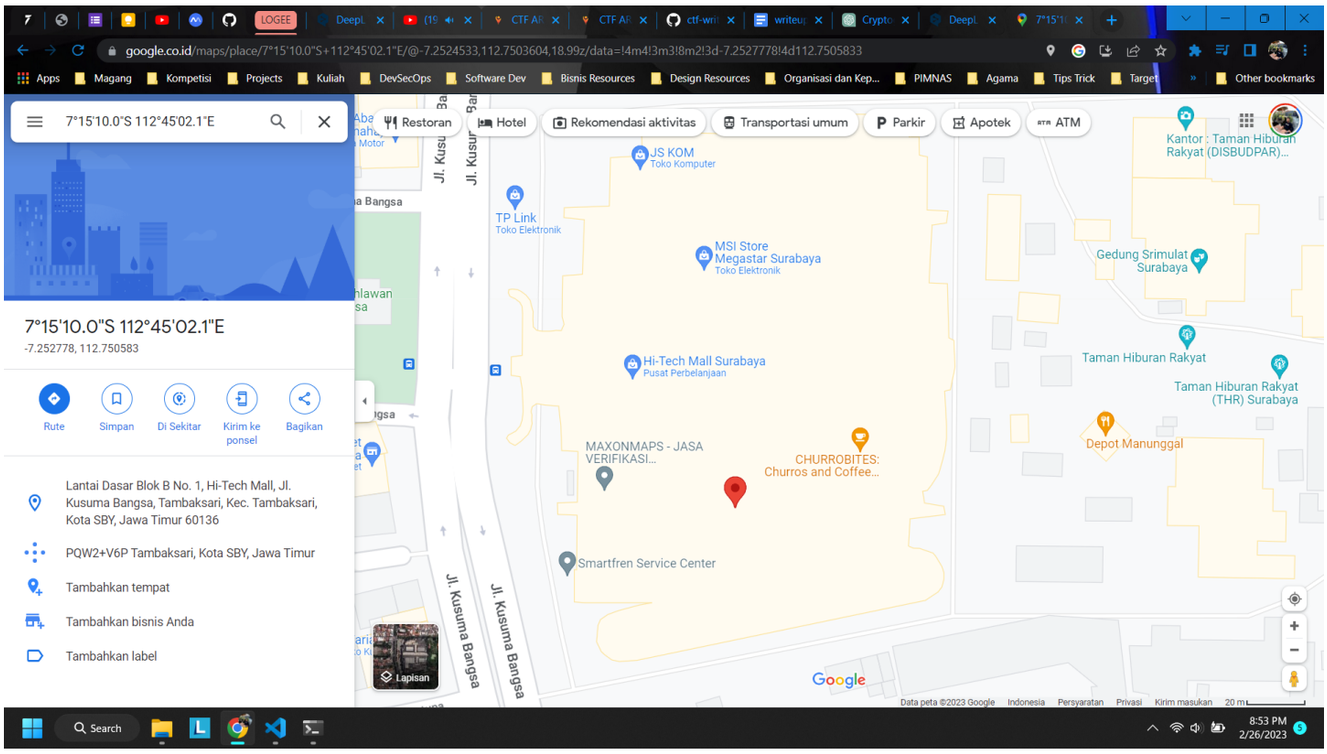
it seemed to highlighted the word 'place'. So, based on the data file given by the `exiftool`. We try to 'find' the place with the GPS latitude and longitude data.

```
GPS Latitude                    : 7 deg 15' 9.97" S
GPS Longitude                   : 112 deg 45' 2.06" E
```

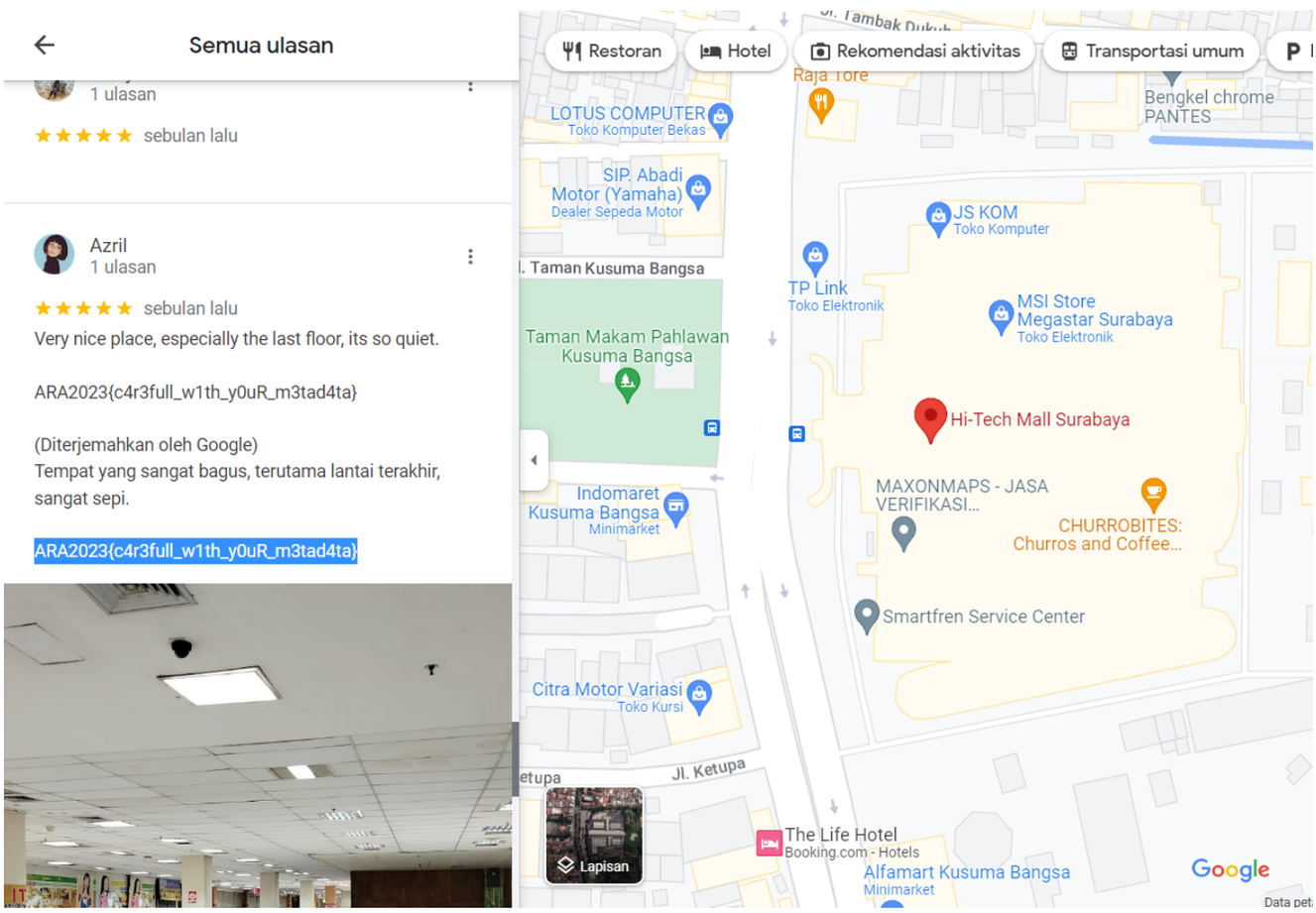
Use the Google Maps, so, we need to convert it into this form.

```
7°15'9.97"S
112°45'2.06"E
```

Open the Google Maps, input the latitude and longitude data into search bar with `longitude latitude` format.



When we see, there is several place that pinned here. As the description challenge said, "so I give this place 5 star". Try to search one by one from these places the 5 stars review (sort by latest and highest rating). Then, we got it in Hi-Tech Mall Surabaya as follows.



So, this is the flag.

```
ARA2023{c4r3full_w1th_y0uR_m3tad4ta}
```

Give feedback

masnurrm feat: inshanity check ... 26 minutes ago History

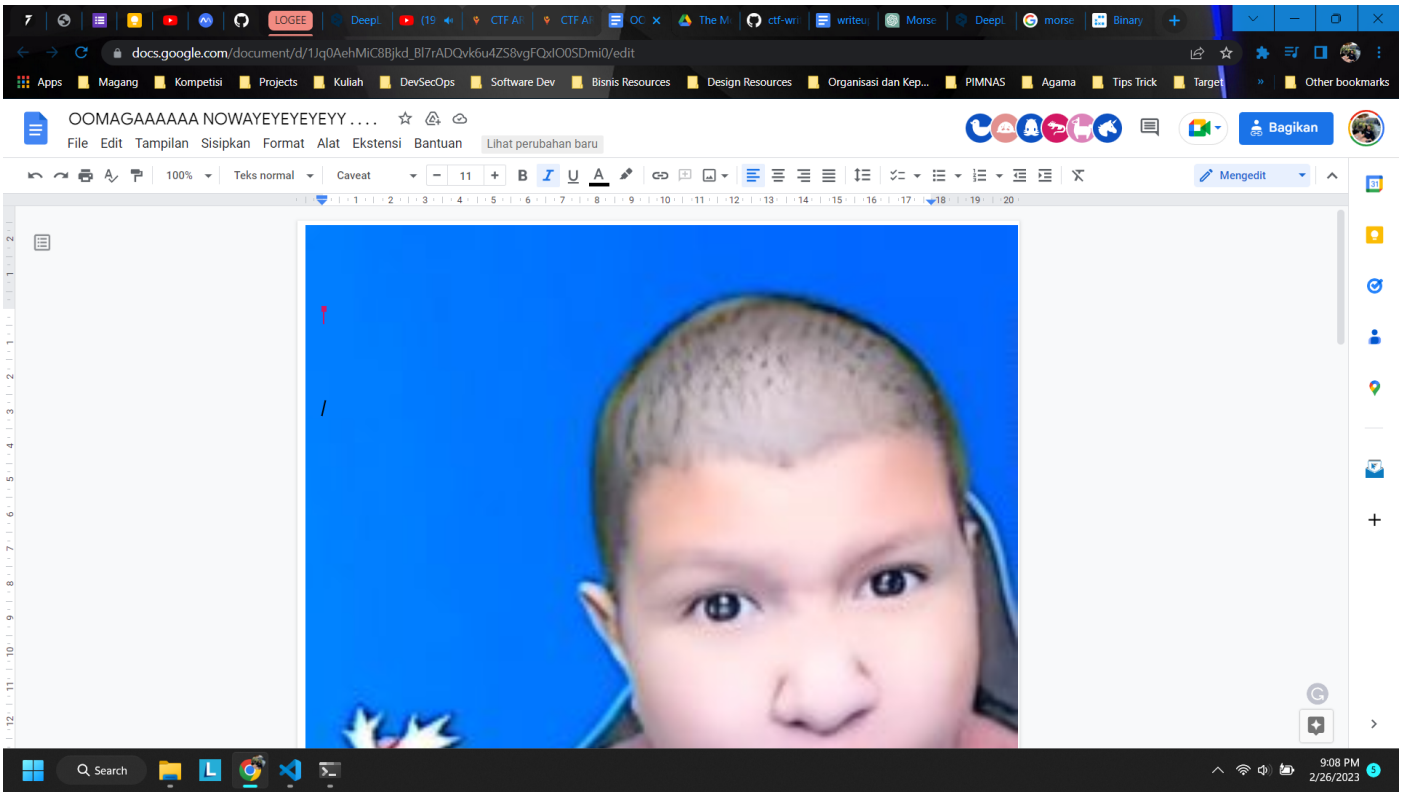
..

assets	26 minutes ago
readme.md	26 minutes ago

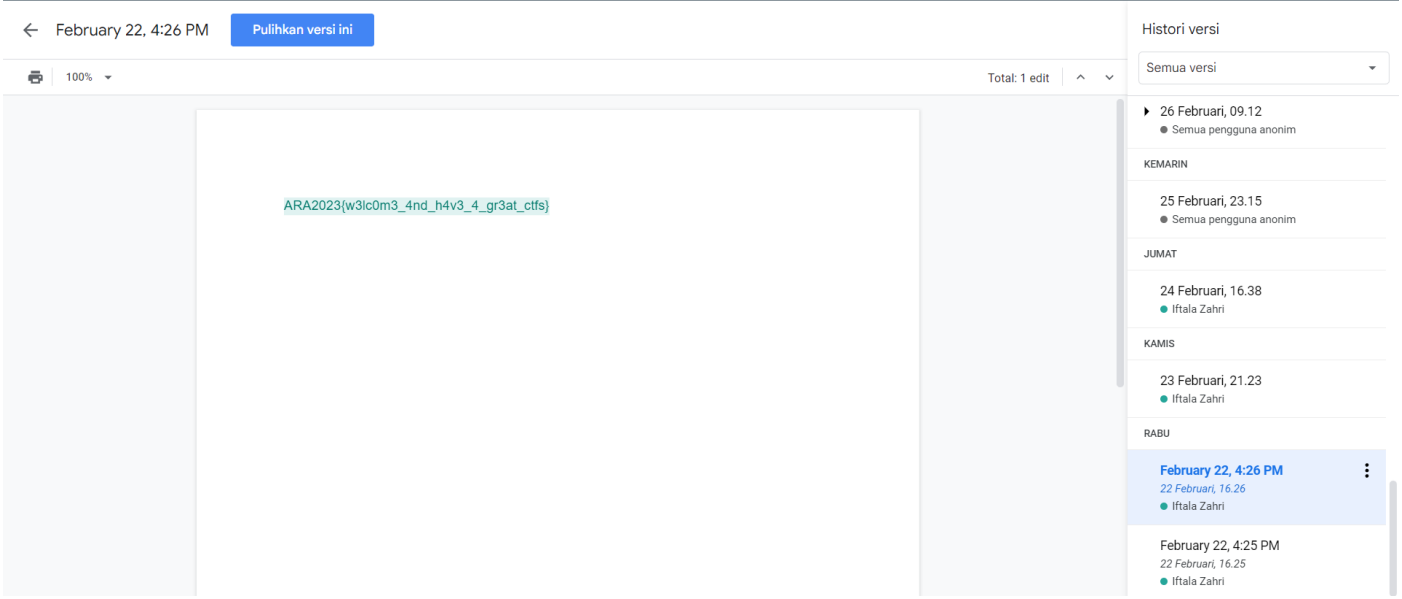
readme.md

Misc - in-shanity check

In this challenge, we were given a Google Docs URL here.



Nothing to do here. But remember the description that said "Even the flag for sanity check is gone?". What can we do when something currently lost or removed? Yes, we search for its history. So, check the history of this docs, and check for the oldest version of the docs. We got the flag in February 22, 4:26 PM version.



So, this is the flag.

```
ARA2023{w3lc0m3_4nd_h4v3_4_gr3at_ctfs}
```

Give feedback

masnurrm feat: dots n dashes ... 32 minutes ago History

..

readme.md 32 minutes ago

readme.md

Misc - D0ts N D4sh3s

In this challenge, we were given an attachment with morse code in it as follows.

```
----- .---- ----- ----- ----- ----- ----- .---- / ----- .---- ----- .---- ----- ----- .----
----- / ----- .---- ----- ----- ----- ----- ----- .---- / ----- ----- .---- .---- ----- ---
-- .---- ----- / ----- ----- .---- .---- ----- ----- ----- ----- / ----- ----- .---- .----
----- ----- .---- ----- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- .---- .--
-- .---- .---- ----- .---- .---- / ----- ----- .---- ----- ----- ----- ----- .---- / -----
.---- .---- .---- ----- .---- ----- ----- / ----- .---- .---- .---- ----- ----- .---- .----
/ ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- .---- ----- .--
-- ----- / ----- .---- .---- .---- ----- .---- ----- .---- / ----- ----- .---- .---- -----
.---- ----- .---- / ----- .---- .---- .---- ----- .---- ----- ----- / ----- .---- ----- .--
-- .---- .---- .---- .---- / ----- ----- .---- .---- ----- .---- ----- ----- / ----- .----
----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- .---- .---- ----- .---- / ---
-- ----- .---- .---- ----- ----- ----- ----- / ----- .---- .---- .---- ----- ----- .---- --
--- / ----- .---- .---- .---- ----- ----- .---- .---- / ----- ----- .---- .---- ----- -----
.---- .---- / ----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- ---
-- ----- ----- .---- / ----- .---- .---- ----- ----- .---- .---- ----- / ----- .---- .----
.---- ----- .---- ----- ----- / ----- ----- .---- .---- ----- ----- .---- .---- / ----- ---
-- .---- .---- ----- ----- ----- .---- / ----- ----- .---- .---- ----- ----- .---- ----- /
----- .---- ----- .---- .---- .---- .---- .---- / ----- .---- .---- ----- ----- ----- -----
.---- / ----- ----- .---- .---- ----- ----- ----- .---- / ----- ----- .---- ----- ----- ---
-- ----- .---- / ----- .---- .---- .---- .---- .---- ----- .----
```

We try to convert this morse code. Thanks to morsedecoder.com. We got the binary code as follows.

```
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
```

Using the binary decoder from cryptii.com, we got the flag.

So, this is the flag.

```
ARA2023{!ts_ju5t_4_m0rs3_aft312_a1!}
```

Give feedback

masnurrm fix: @bash ... 1 hour ago History

..

assets	1 hour ago
readme.md	1 hour ago

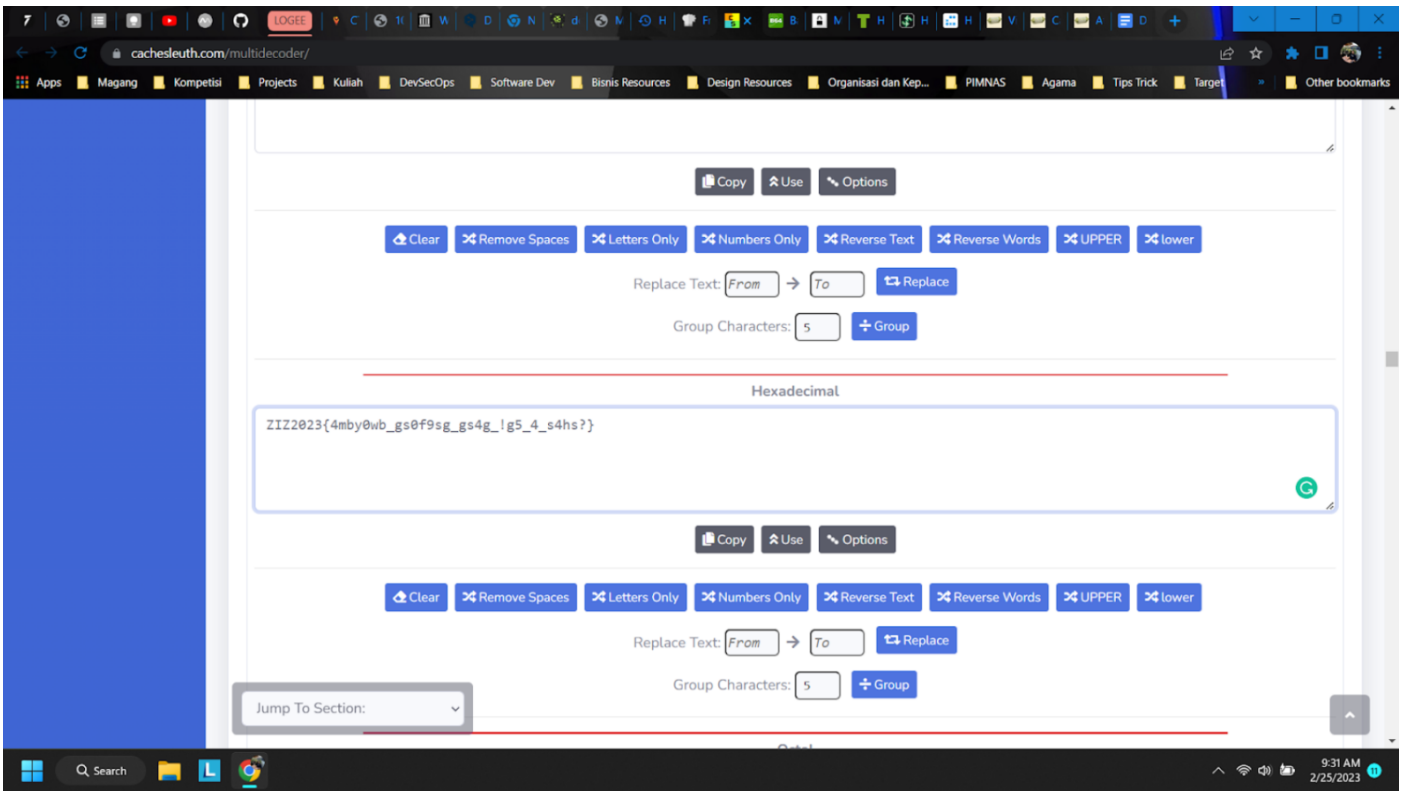
readme.md

Misc - @Bash

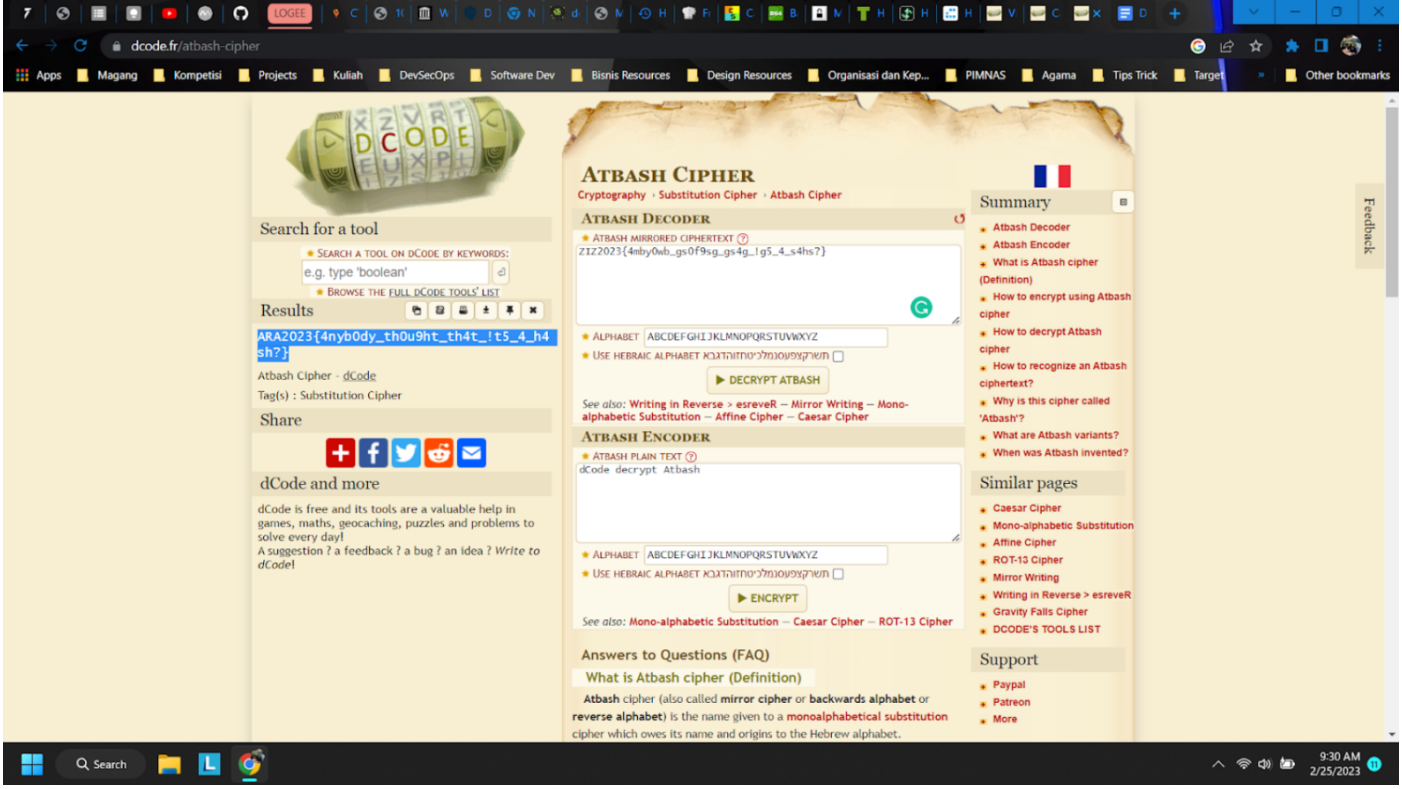
In this challenge, we were given a string as follows.

```
5A495A323032337B346D62793077625F677330663973675F677334675F2167355F345F733468733F7D
```

So, we use a online multi decoder from cachesleuth.com. We got some suspicious results here in hexadecimal.



Then, we try some decoder to decode the results. After trying some methods, we got it using AtBash chipher decoder. Thanks to dcode.fr.



So, this is the flag.

```
ARA2023{4nyb0dy_th0u9ht_th4t_!t5_4_h4sh?}
```

Give feedback

fhinnn feat:Thinker ... 6 minutes ago History

..

assets	6 minutes ago
README.md	6 minutes ago

README.md

Forensic - Thinker

Kita diberikan gambar berformat PNG. lalu saya cek di meta data nya dengan membukanya melalui notepad dan menemukan beberpa keanehan.

```
□       ‡Pz:®□MÒ□ŸÐ¦:æ□□µx@,□±3Ê?^xß:□□úÔçi•¾ëØèe‡âñ□D□Ð□˜Ûµ¥□‘nªÄ□□□□#“í\Žúùà□@Z<¬{D–ø□9MËŽ®ê
×¦□ww
6x□□□´,£ý□¶þ3□sb□FÿüÕØ□‘□ðùkœzâÿÇ}û□ÕDUâ¡□}ª)□Ðt5
□×ªÛ¾ý?PK□□□□
   4^GV
□      □ íA   suspicious/UT□ □$□âcux
□□è□  □è□  PK□□□□□  □ ‘\GVßS¦²ô#  {' □ □       ¤□E   suspicious/y.pngUT□ □□~âcux
□□è□  □è□  PK□□   □ □ § ƒ$   PK□□□□
   Ž^GV
□      □ íA   something/UT□ □Ì□âcux
□□è□  □è□  PK□□□□□  □ ³HFVÒúí})  Z  □ □    □  ¤□D   something/s.txtUT□ □2       ácux
□□è□  □è□  PK□□□□
   p^GVÓ‡‡,@% @% □ □       ¤□¶  something/suspicious.zipUT□ □“□âcux
□□è□  □è□  PK□□   □ □ □□  H&   PK□□□□
   ê^GV         □ □      □ íA   find/UT□ □w,âcux
□□è□  □è□  PK□□□□
    HFV/Cðw□   □
□    □  ¤□?  find/a.txtUT□ □à ácux
□□è□  □è□  PK□□□□
   ²^GV»ü×,a' a' □ □       ¤□’  find/something.zipUT□ □□,âcux
□□è□  □è□  PK□□   □ □ ó  ?(   PK□□□□
        _GV          □      □ íA   didyou/UT□ □²,âcux
□□è□  □è□  PK□□□□
   =GFV• V˜


□    □  ¤□A   didyou/e.txtUT□ □f ácux
□□è□  □è□  PK□□□□
   ÷^GV□D,,H)  H)  □ □       ¤□”  didyou/find.zipUT□ □’,âcux
□□è□  □è□  PK□□   □ □ ô  %*
```

ditemukan ada beberapa file didalam. jadi untuk lebih ditailnya saya cek menggunakan `binwalk` untuk melihat isi dari PNG tersebut.

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 720 x 881, 8-bit/color RGB, non-interlaced
6170          0x181A          Zlib compressed data, best compression
321663        0x4E87F         TIFF image data, big-endian, offset of first image directory: 8
321693        0x4E89D         Zip archive data, at least v1.0 to extract, name: didyou/
321758        0x4E8DE         Zip archive data, at least v1.0 to extract, compressed size: 13, uncompressed size: 13, name: didyou/e.
txt
321841        0x4E931         Zip archive data, at least v1.0 to extract, compressed size: 10568, uncompressed size: 10568, name: did
you/find.zip
332460        0x512AC         End of Zip archive, footer length: 22
332726        0x513B6         End of Zip archive, footer length: 22
```

ternyata benar dan ditemukan beberapa file ada dialam PNG tersebut dan di kompres dengan ZIP, lalu selanjutnya saya lakukan extract agar bisa mengakses hidden file tersebut. Hasil dari extract dari file PNG tersebut sebagai berikut.

didyou	07/02/2023 23:56	File folder	
4E89D.zip	26/02/2023 20:13	Compressed (zipped)...	11 KB
181A	26/02/2023 20:13	File	0 KB
181A.zlib	26/02/2023 20:13	ZLIB File	319 KB

didalam direktori `didyou` ditemukan 2 file yaitu `e.txt` dan `find.zip`

e.txt	06/02/2023 20:57	Text Document	1 KB
find.zip	07/02/2023 23:55	Compressed (zipped)...	11 KB

didalam `e.txt` ditemukan kode yang sepertinya kode `base 64` dan coba saya translate. dan benar ternyata itu adalah potongan flag nya.

```
QVJBMjAyM3s=
```

Output

time: 59ms
length: 17471
lines: 652

Recipe (click to load)	Result snippet	Properties
From_Quoted_Printable()	QVJBMjAyM3s	Valid UTF8 Entropy: 3.28
From_Base64('A-Za-z0-9+/=',true,false)	ARA2023{	Valid UTF8 Entropy: 2.50
	QVJBMjAyM3s=.	Matching ops: From Base64, From Quoted

lalu selanjutnya saya coba membuka file `find.zip` nya dan ditemukan file `a.txt` dan `something.zip`.

a.txt	Text Document	1 KB	No
something.zip	Compressed (zipped) Folder	10 KB	No

didalam `a.txt` ditemukan kode juga dan saya mencoba translate. Dan ternyata itu adalah potongan flag yang menggunakan kode hex.

```
35216D706C335F
```

Output

time: 31ms
length: 11629
lines: 436

Recipe (click to load)	Result snippet	Properties
From_Hex('None')	5!mpl3_	Valid UTF8 Entropy: 2.81
	35216D706C335F	Matching ops: From Hex, From Hexdump Valid UTF8 Entropy: 3.18

lalu selanjutnya saya mencoba buka file `something.zip` dan menemukan `s.txt` dan `suspicious.zip`.

s.txt	Text Document	1 KB
suspicious.zip	Compressed (zipped) Folder	10 KB

didalam `s.txt` ditemukan kode biner dan saya translate ternyata itu potongan dari flag juga.

```
01000011 00110000 01110010 01110010 01110101 01110000 01110100 00110011 01100100
01011111
```

Output

time: 2ms
length: 10
lines: 1

```
C0rrupt3d_
```

lalu selanjutnya saya buka file `suspicious.zip` nya dan menemukan PNG. namun PNG tersebut tidak dapat dibuka, lalu coba saya lihat menggunakan hex editor dan membandingkannya dengan PNG yang normal ada sebuah keanehan yaitu header file nya berbeda.

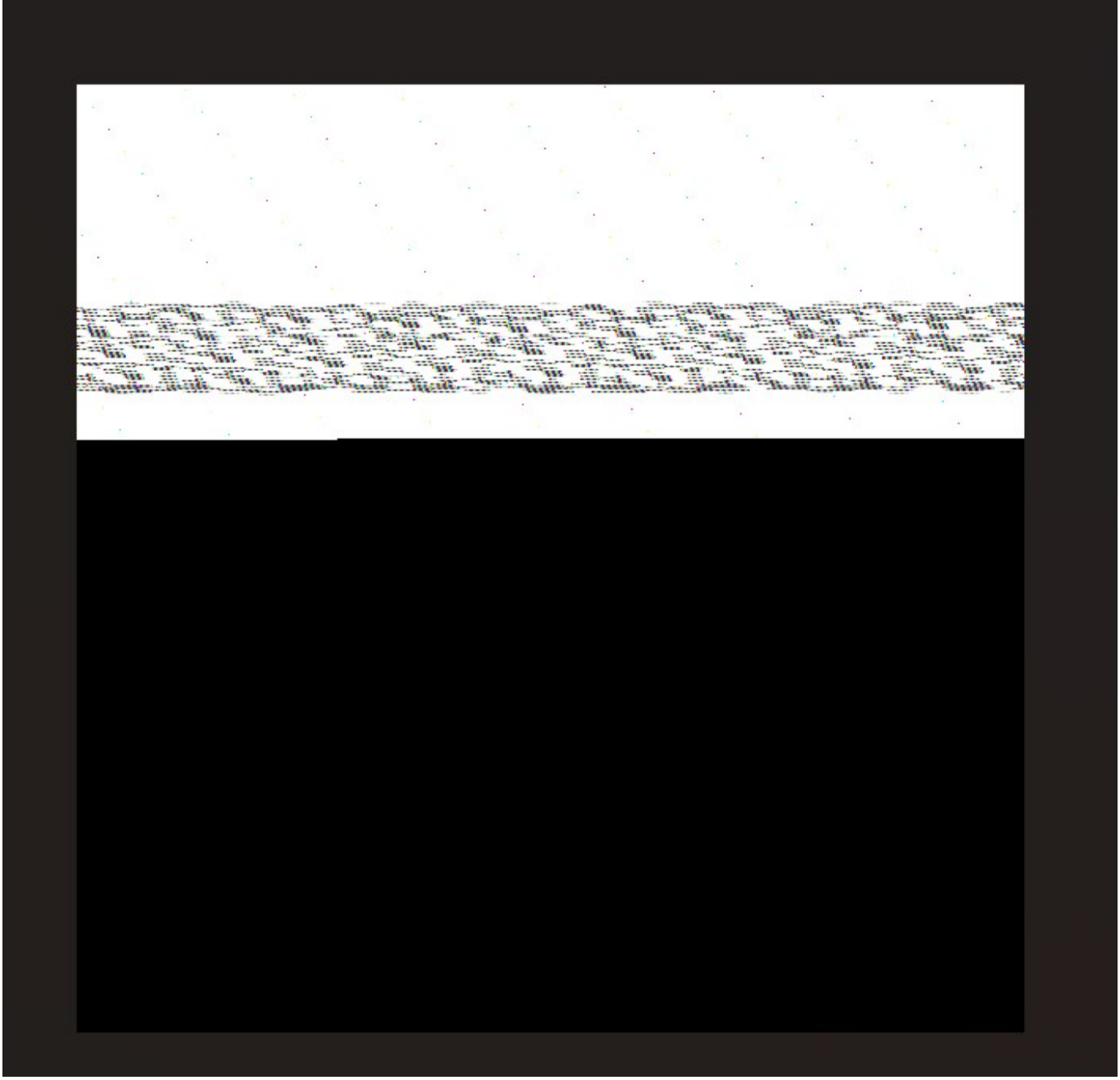
```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52    ‰PNG........IHDR
0010h: 00 00 1F 40 00 00 11 94 08 02 00 00 00 50 03 1F    ...@...”.....P..
0020h: C7 00 00 00 09 70 48 59 73 00 00 2E 23 00 00 2E    Ç....pHYs...#...
0030h: 23 01 78 A5 3F 76 00 00 20 00 49 44 41 54 78 9C    #.x¥?v.. .IDATxœ
0040h: EC DC 41 01 C0 20 10 C0 B0 63 FE 7D 20 08 03 48    ìÜA.À .À°cþ} ..H
0050h: D9 83 CA 48 24 54 40 D7 3D 7B 00 00 00 00 00 00    ÙƒÊH$T@×={......
0060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0070h: 00 98 F9 34 00 00 00 00 00 00 00 00 00 00 00 00    .˜ù4............
0080h: 00 00 00 00 00 00 00 00 00 00 00 78 0C DC 01 00    ...........x.Ü..
0090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
00A0h: 00 00 00 00 00 62 E0 0E 00 00 00 00 00 00 00 00    .....bà.........
00B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 10 03    ................
00C0h: 77 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    w...............
```

y.png

Edit As: Hex | Run Script | Run Template: PNGTemplate.bt

```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    0123456789ABCDEF
0000h: 21 5A 78 52 0D 0A 1A 0A 00 00 00 0D 52 52 48 5C    !ZxR........RRH\
0010h: 00 00 02 BD 00 00 00 90 08 06 00 00 00 05 89 D3    ...½.........‰Ó
0020h: 19 00 00 20 00 49 44 41 54 78 9C ED DD 79 98 13    ... .IDATxœíÝy˜.
0030h: F5 FD 07 F0 3E 4F 90 0A 5E D5 5A B5 B5 F5 68 00    õý.ð>O..^ÕZµµõh.
0040h: 59 B9 14 41 51 A1 08 5A C5 7A 50 0F 44 94 96 D6    Y¹.AQ¡.ZÅzP.D”–Ö
0050h: E3 27 87 A0 78 5B 2B 8A A8 80 22 88 DC 87 C8 0D    ã'‡ x[+Š¨€"ˆÜ‡È.
0060h: 72 5F 2A 2C 16 15 AA 68 B2 F7 09 BB 2C 2C CB 1E    r_*,..ªh²÷.»,,Ë.
0070h: EC 91 5D F6 3E 93 F7 EF 8F 6D 30 9B CD CC F7 3B    ì‘]ö>“÷ï.m0›ÍÌ÷;
0080h: D9 99 64 32 BC 5F CF 33 7F C1 66 92 99 64 E6 3D    Ù™d2¼_Ï3.Áf’™dæ=
0090h: DF E3 F3 FD 05 88 88 88 88 88 2C EE 17 E1 7E 03    ßãóý.ˆˆˆˆˆ,î.á~.
00A0h: 44 44 44 44 44 46 63 E8 25 22 22 22 22 CB 63 E8    DDDDDFcè%""""Ëcè
00B0h: 25 22 22 22 22 CB 63 E8 25 22 22 22 22 CB 63 E8    %""""Ëcè%""""Ëcè
00C0h: 25 22 22 22 22 CB 63 E8 25 22 22 22 22 CB 63 E8    %""""Ëcè%""""Ëcè
```

Tetapi IDAT nya ada dan IEND nya pun ada sepertinya hanya ada kesalahan di header nya aja jadi saya ganti headernya dengan punya header PNG lain agar gambar bisa dibuka. Ternyata berhasil tetapi gambar masih tidak memberikan clue.



Selanjutnya saya manipulasi untuk width nya karena sepertinya kurang lebar sehingga tak bisa terbaca. jadi saya tambah widhtnya. Tetapi ternyata masih buram dan sulit terbaca maka saya ganti colornya dari truecolor menjadi alphatruecolor

dari gambar tersebut didapatkan deretan angka desimal lalu saya translate. dan akhirnya mendapatkan potongan akhir dari flag nya.

```
49 109 52 103 101 53 125
```

Output

time: 0ms
length: 7
lines: 1

```
1m4ge5}
```

Akhirnya kita telah dapatkan semua flagnya dan tinggal di gabung saja menjadi flag yang utuh

`ARA2023{5!mpl3_C0rrupt3d_1m4ge5}`

Give feedback

masnurrm feat: sh432 ... 7 minutes ago History

..	
assets	7 minutes ago
readme.md	7 minutes ago
sha256.py	7 minutes ago

readme.md

Cryptography - SH4-32

In this challenge, we were given a hash value as follows.

```
9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8
```

Using the hash analyzer from tunnelsup.com, we know that this is a SHA2-256 type of hash value.

Hash:	9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8
Salt:	Not Found
Hash type:	SHA2-256
Bit length:	256
Character length:	64
Character type:	hexidecimal

Also, we were given an attachment named Dictionary.txt that contains many lines of random strings.

```
123456
123456789
111111
password
qwerty
```

```
abc123
12345678
password1
...
...
...
```

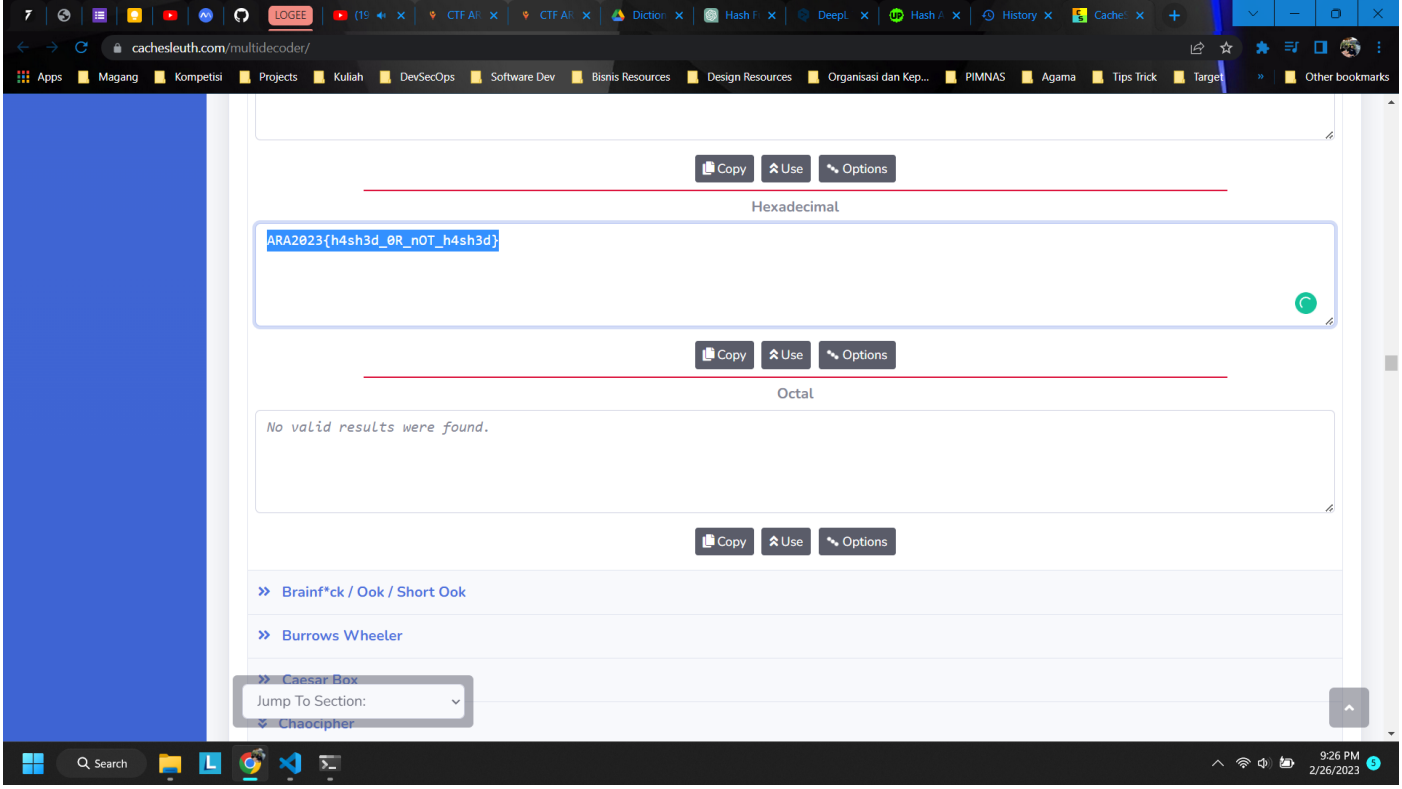
As we know, the SHA2-256 type of hash can't be decrypted. So we try to encrypt the all of strings in the `Dictionary.txt` to find which string that have the encrypted string of SHA2-256 that same as the hash provided in the challenge. We use this script.

```python
import hashlib

with open('Dictionary.txt', 'r') as f:

    for line in f:
        line = line.strip()

        sha256_hash = hashlib.sha256(line.encode()).hexdigest()
        expected = "9be9f4182c157b8d77f97d3b20f68ed6b8533175831837c761e759c44f6feeb8"

        if sha256_hash == expected:
            print(f"{line} : {sha256_hash}")
```

We found this string.

```
415241323032337b6834736833645f30525f6e4f545f6834736833647d
```

Try to convert it into hexadecimal, we got the flag. Thanks to cachesleuth.



So, this is the flag.

```
ARA2023{h4sh3d_0R_nOT_h4sh3d}
```

Give feedback

fhinnn feat:Thinker ... 5 minutes ago History

..

assets	5 minutes ago
README.md	5 minutes ago

README.md

Cryptography - L0v32x0r

Kita diberikan kode tanpa ada clue tambahan lagi
`"001300737173723a70321e3971331e352975351e247574387e3c"`

tapi melihat judul dari challange nya saya langsung melakukan XOR dan ternyata tidak menemukan flag. lalu saya lakukan bruteforce XOR dan akhirnya menemukan flagnya.

Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type 'boolean'

BROWSE THE FULL DCODE TOOLS' LIST

Results

Bruteforce attempt. Only relevant results are displayed.

[Hexadecimal key | Plain text]

ASCII output limited to printable characters (control chars and non-ASCII characters replaced by ◆)

↑↓	↑↓
40	@S@3132z0r^y1s^ui5u^d54x>|
41	ARA2023{1s_x0r_th4t_e45y?}
45	EVE6467◆5w[|4v[pl0p[a01};y
48	H[H;9;:r8zVq9{V}a=}Vl=<p6t
50	PCP#!#"j bNi!cNey%eNt%$h.l

ARA2023{1s_x0r_th4t_e45y?}

Give feedback

masnurrm feat: OTP ... 1 hour ago History

..

readme.md 1 hour ago

readme.md

Cryptography - One Time Password (?)

In this challenge, we were given `txt` file with some line of string hexadecimal written in it.

```
A: 161a1812647a765b37207a1c3b1a7b54773c2b660c46643a1a50662b3b3e42
B: 151d616075737f322e2d130b381666547d3d4470054660287f33663d2a2e32

XOR: 415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d
```

We try to approach it by applying the XOR operation between the given hexadecimal A and B. But, first we need to convert A and B into binary. Thanks to binaryhackconverter.com.

```
A:
0001011000011000110000010001011001000111010011010111011001011011001101110010000001110100011000
B:
0001010100011101011000010111000001110010110101010111111001100101111010111110100001101100010000
```

Then, try to apply the XOR operation between A and B so it will give us the result here.

```
0000001100000101111001000110011000101000011111101101110110111011011111101000110100010100111001
```

Convert the XOR result into binary.

```
XOR:
0100000101010010001100000011001000110111001100100011001101110111001111001111000111010000110100
```

Then, apply the XOR operation between A and B result and the XOR string. It will give us this result.

```
415241323032337b7468335f705f3574346e64355f6630725f7034647a7a7d
```

Lastly, convert it into ASCII characters.

So, this is the flag.

```
ARA2023{th3_p_5t4nd5_f0r_p4dzz}
```

Give feedback

elshiraphine fix: fix typo ... now History

..

assets	12 minutes ago
README.md	now

README.md

Cryptography - Secrets Behind a Letter

In this challenge, we were given a Letter attached below:

```
p:
12575333694121267690521971855691638144136810331188248236770880338905811883485064104865649834927
q:
12497483426175072465852167936960526232284891876787981080671162783561411521675809112204573617358
c:
36062934495731792908639535062833180651022813589535592851802572264328299027406413927346852454217

e = 65537
```

After doing some research, I can confirmed that this challenge is about RSA so I do searching with a powerful keyword and it brought me to this stack-overflow discussion.

Based on that discussion, the top answer mentioned a tools named `RsaCtfTool` in GitHub so I clone the repository and use it to solve this challenge.

Based of their documentation, I used a command `--uncipher` , here is the result:

```
matryochska@Matryoshka: ~

matryochska@Matryoshka:~/RsaCtfTool$ ./RsaCtfTool.py -p 125753336941212676905219718556916381441368
10331188248236770880338905811883485064104865649834927819725617695554472100341361896162022311653301
53281010134427 3 -q 124974834261750724658521679369605262322848918767879810806711627835614115216758
09112204573617358389742732546293502709585129205885726078492417109867512398747 -e 65537 --unciph
er 36062934495731792908639535062833180651022813589535592851802572264328299027406413927346852454217
62779331514489294202688698082362224015740571749978795994304054073412214283889848276754127267783709
13038246699129635727146561394220118530281335561114050725265098398467015701334377461027276449823447
1257184433228021 8
private argument is not set, the private key will not be displayed, even if recovered.

Results for /tmp/tmpvxkhpa8e:

Unciphered data :
HEX : 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00415241323032337b31745f7475726e355f3075745f746f5f62335f616e5f7273617d
INT (big endian) : 193630480523654981469233003289852791996689967037452812291796253508752554841783
5389
INT (little endian) : 88045299140098467544215712458392650202024734820128819762384848347067153644
0970353011760386396035737097671601974041867796142282033752977521881084054411116948205624855765367
1974128796608636758990960278311592289135088705881794270772398389683361713476279323000266217757578
5510731556218362616519212213525454782464
utf-8 : ARA2023{1t_turn5_0ut_to_b3_an_rsa}
utf-16 : 剁㈱㈰笳琱瑟牽咬た瑵瑟影嶇慟彩獲絡
STR : b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00ARA2
023{1t_turn5_0ut_to_b3_an_rsa}'
matryochska@Matryoshka:~/RsaCtfTool$ cd ..
```

So based on this unciphered data, I found that the flag was:

```
ARA2023{it_turn5_0ut_to_b3_an_rsa}
```

Give feedback

elshiraphine / ctf-writeup Private

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main ctf-writeup / 2023-ARA / Cryptography-Help /

elshiraphine style: add line break ... 2 minutes ago History

..

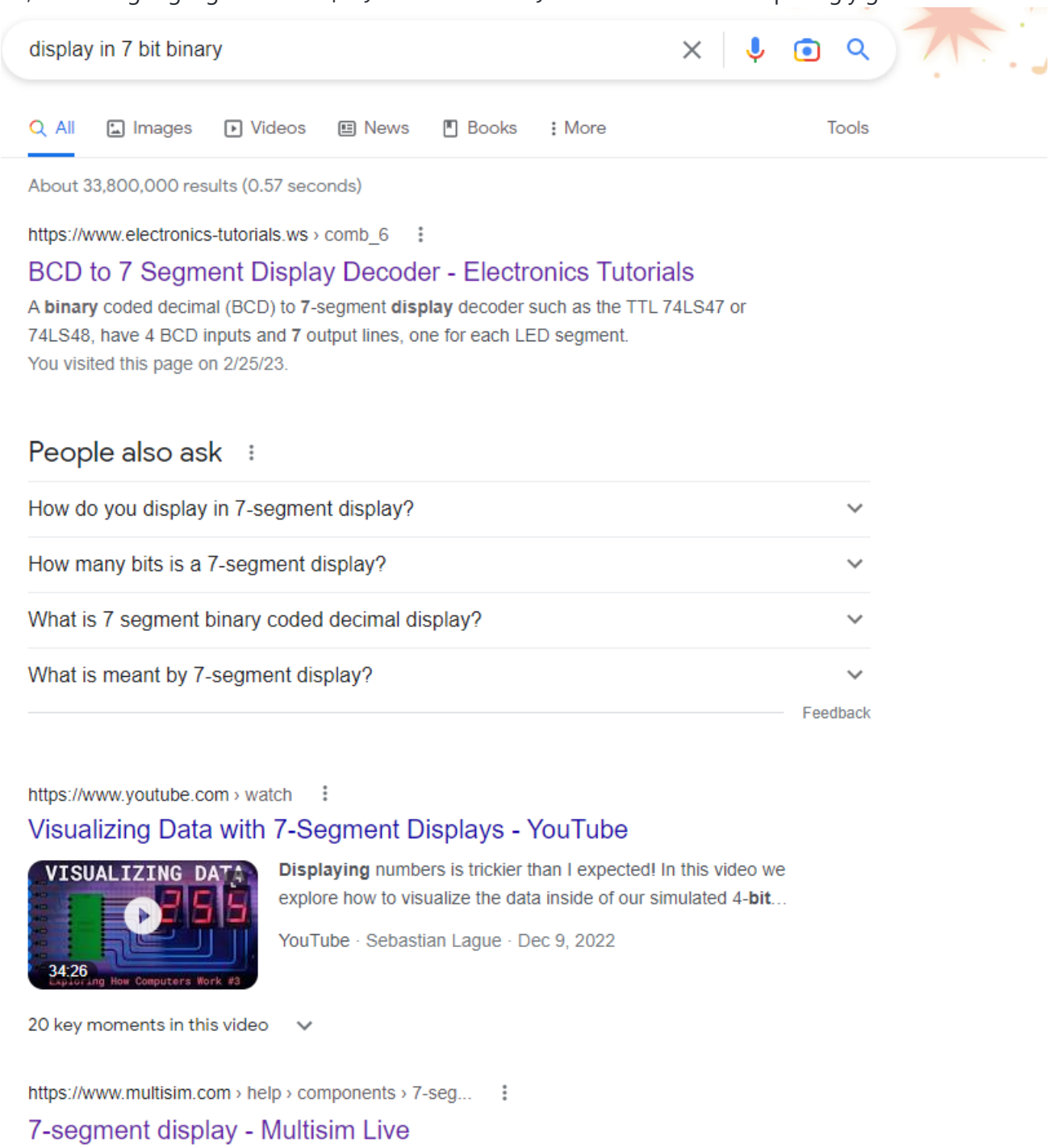
assets	7 minutes ago
README.md	2 minutes ago

README.md

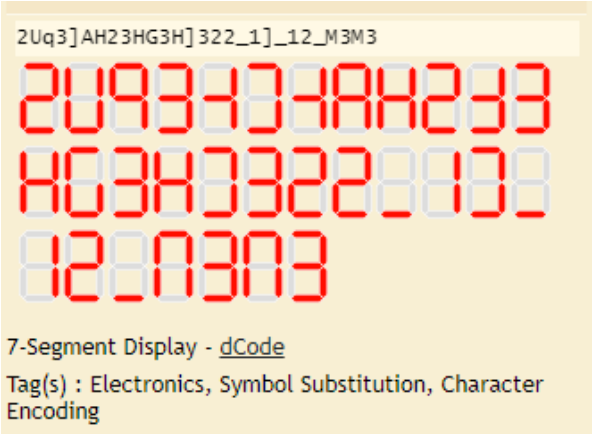
Cryptography - Help

In this challenge we were given 32 number of 7-bit binary number. I was confused because there is no clue about this challenge. But after and hours, I found something interesting from the challenge's statement which is `display in the office` .

So, I started googling about `display in 7 bit binary` and the result was suprisingly good.

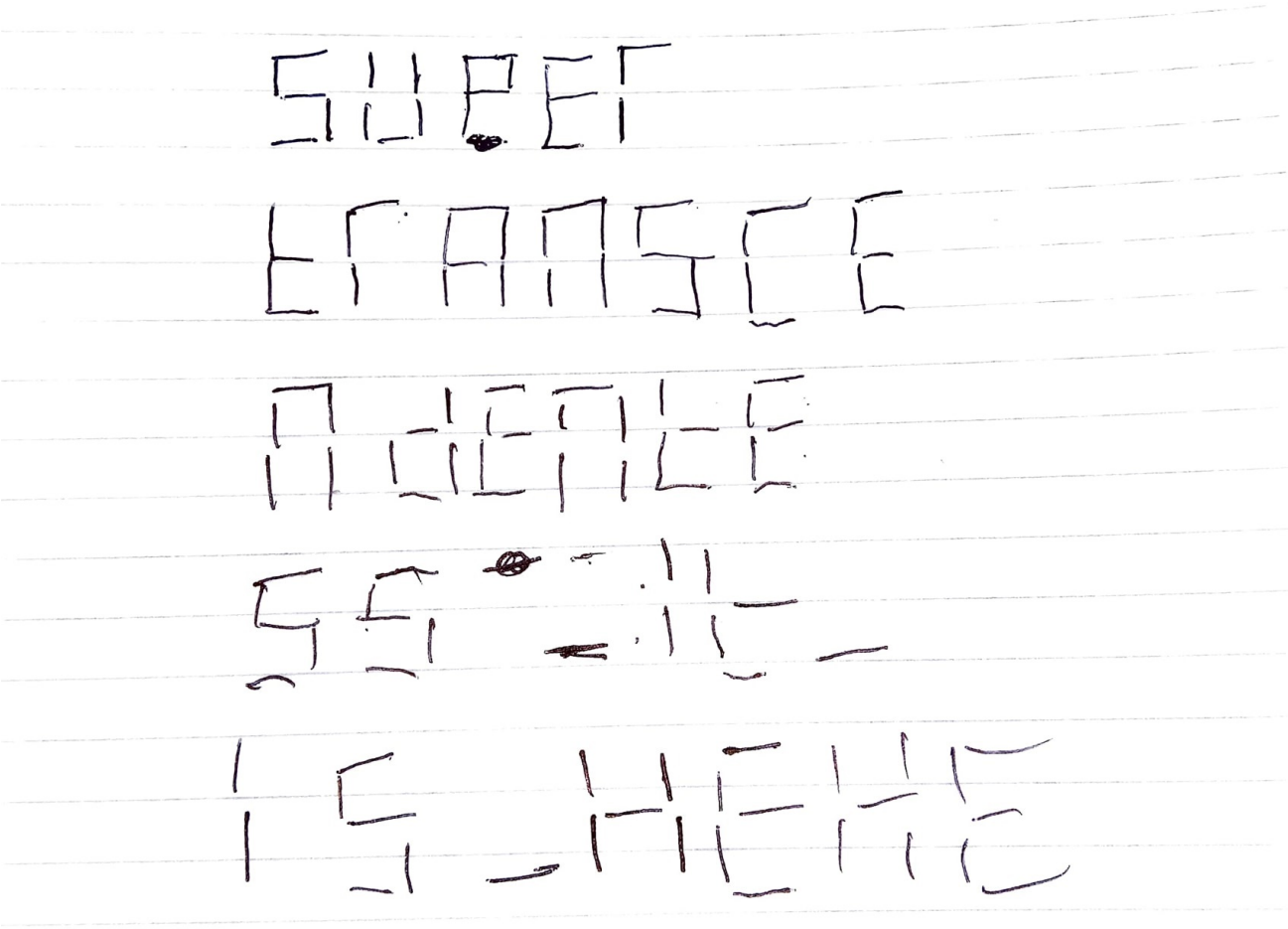


I tried to find a tools to help me mapping this binary number into seven segment display and brought me to this site. The result is not really good because there is some inconsistency between the number and there are some symbols in the result.



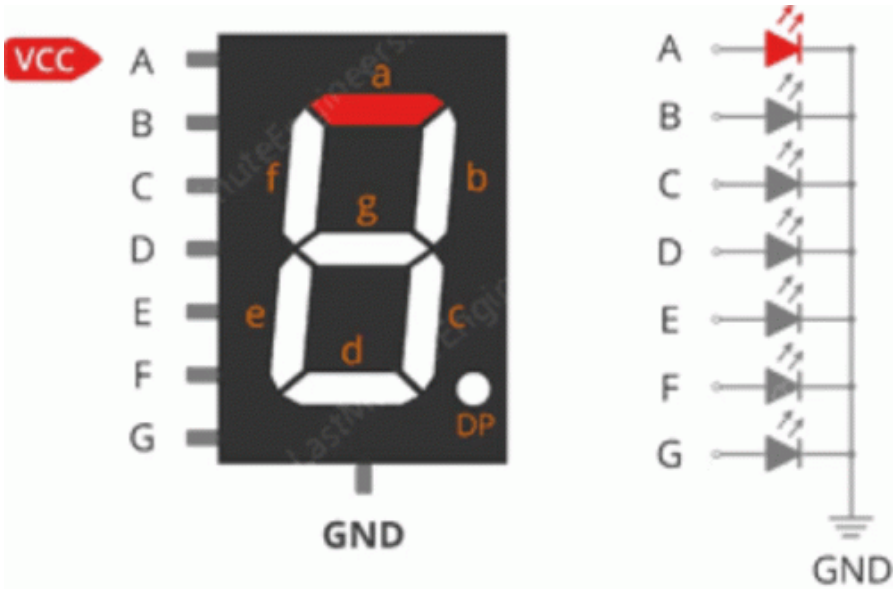
I tried to find another possibilities such as lower-case display in seven segment but confusing because

it is not common.
After quite tired of searching, I tried manually by writing in my book and it is mindblowing as attached below:



SO THE FLAG SHOULD BE

```
ARA2023{supertranscendentess_it_is_hehe}
```



Give feedback

elshiraphine / ctf-writeup Private

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main ctf-writeup / 2023-ARA / Cryptography-babychall /

elshiraphine feat: add Cryptography-babychall write up ... 7 minutes ago History

..

assets	7 minutes ago
README.md	7 minutes ago

README.md

Cryptography - babychall

Babychall challenge is similar to the previous crypto challenge (Secret Behind a Letter) but in this challenge we were not given any information about the key (the `e` value).
We were only given pairs of number:

```
c1=5099697310484566310837975113120308543241249019831271466365682364823303847929819286145183424 6
c2=2675086354476975422055414666795504683242305948200761348250028401266882028494792724072473530 8
c3=3723065824325259074360857110502735786279097298720883321301794117144875381565483990169952665 1

n1=1054811272672182606121568710177576945501427358240871501067504035798774950592304130461813013 5
n2=9310562105968647481689021549455480283151894842016094170352275912161978585127060863413030745 0
n3=6591850965074227849497136329087484918126836431601265676933912000400070294527194253309752988 4
```

So I expected that the key maybe a common key (it should be `e = 3`). To solve this challenge, I used `Chinese Remainder Theory` and found a tools on the internet to solve the Cipher.
And here is the result:

```
Cipher 1:
509969731048456631083797511312030854324124901983127146636568236482330384792981928614518342469302081401101736990585
279190201154325867054004673456478065223313964476508476501330132466733908792227191692488624202782563229677187017004
587292077931247581664386414481123144899458632318819823527907651305350040900536 77,
N1=105481127267218260612156871017757694550142735824087150106750403579877495059230413046181301355871045357138033343
315900732228502875706659244844711538497850413046440270578916645981161000807526427004236918404837363404678029443944
950655102252423415631977020625826867728898231382737396728896847618010577420408630133
Cipher 2:
267508635447697542205541466679550468324230594820076134825002840126688202849479272407247353088803134399798848563936
737592797410030710740677510369519880070370418141473628138846420542912315960504818663485277171790970486464711281758
602468229998786860793305963427955632147620481352120168266232851008649621582146 1,
N2=931056210596864748168902154945548028315189484201609417035227591216197858512706086341303074502275579879768181623
319822896342150371840758647872236812189826020928067578885335871269740910771902427974613189072807590756125774755346
260620609607392698287892741372743639700562761394340393158600525564173406969985092 71
Cipher 3:
372306582432525907436085711050273578627909729872088332130179411714487538156548399016995266514337713248268953556712
559444148939479639349790682573103673159357012708043907991216696351530129164022711907226189975003929117377671433165
523764958829869356951469708539142754817174002688326449871579887275755133514419 19,
N3=659185096507422784949713632908748491812683643160126567693391200040007029452719425330975298849640631093770367158
471761962809438072619868485930004241433202800532790214113942672682553377834949016063196874573515869153146628004346
323329889788580859315868302836948815387590083604866619368842022749733871082147541 01

We can solved M^e with CRT to get
637909221477481890662522977099331448284105784763620732835962595873375029586685193236455464837158506606016969207496
223884942331128512235900054398208957386777517602361390695821101109238744866151486639836319791064679261525747275641
357616469932840585678731249928939472789771140052991172872048660806711660779736719190414518843981239106571327906520
359916864344079307917860102382782982197166977639563020324950 9

If we assume e=3, we take the third root to get:
185461154986387874587859826356816875376614446074216167090696171866021207609015690993071745707423516614034205731355
54815354157770318706 9
Next we convert this integer to bytes, and display as a string.

Decipher: b'ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}'
```

Or you can access on this link.

So based on the result, I found that the flag was:

```
ARA2023{s00000_much_c1ph3r_but_5m4ll_e_5t1ll_d0_th3_j0b}
```

Give feedback

elshiraphine / ctf-writeup Private

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main ctf-writeup / 2023-ARA / OSINT-Hey-detective-can-you-help-me /

elshiraphine style: fix line break, add assets folder, and fix typo ... 2 hours ago History

..

assets	2 hours ago
README.md	2 hours ago

README.md

OSINT - Hey detective, can you help me

In this challenge we were given some informations to find a cosplayer from China who like to post their photos on Facebook and Instagram. There were several instructions given, first I tried to find cosplayer who had collaborated with Sakura (also from china).

On instagram I found a cosplayer account named Sakura with the username sakura.gun. Here is the profile sakura.gun on instagram and looked for accounts that had collaborated with her.

1. First, I found a cosplayer named rakuko but it didn't match the next clue which is **studying in top university in China** because rakuko herself was studying abroad in America.
2. Second, I found another collaboration with account named skylaryuuu, but I thought it was not her that I tried to search because skylaryuuu currently move to Canada.
3. Third, I found an ordinary photo (not doing cosplay) with an account named Yanzikenko. I am interested with this account so I tried to find her page in Facebook.

It turned out that what I did was right because I found several matching photos with the clues given in the challenge. Here is her facebook page.

Based on her photo in Facebook, I tried to follow the instructions as below:

1. Social Media ID
   Based on the Flag format, it should be an Instagram ID so I use this Find Instagram User ID Tools.
   And the value was `44793134117`
2. Her university but in abbreviation.
   So, based on this photo, she graduated from Beijing Normal University. So the second flag pattern should be `BNU` .
3. Mascot name where she was in the doll shop.
   There is a post that should include her photo with a mascot in the doll store She took a picture with a mascot named `Molly` from PopMart.

4. Date and Time when she posted a photo in the bookstore.
   Previously, an instruction mentioned that she was photographed sitting in a bookstore. So, there are two different post which have a photo that it is could be taken in the bookstore.
   First post and the second post.
   So it could be `3Juni2019-10:25` or `14Februari2019-14:59` .
5. The last part, which is the most challenging part is to find redacted flag. There are so many photo which have Kenko and Sakura collaborated. So after an hours read, I found a powerful comments attached below:

AND FINALLY AFTER TRYING SEVERAL POSSIBILITIES, I FOUND THAT THE FLAG WAS:

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

and this is the result:

Awards

Solves

Challenge	Category	Value	Time
Hey detective, can you help me	OSINT	481	February 25th, 2:29:32 PM

Give feedback

elshiraphine / ctf-writeup Private

Code Issues Pull requests Actions Projects Wiki Security Insights Se

main ctf-writeup / 2023-ARA / Misc-Truth /

elshiraphine style: add line break ... 1 minute ago History

..

assets	6 minutes ago
README.md	1 minute ago
isUpper.py	6 minutes ago

README.md

Misc - Truth

We were given a locked PDF File but there is no information about the password. So I search using `brute force PDF Password` keyword on google and found this website.
Next, I followed the instruction to clone, make and configure the tools on my Kali Linux WSL. Then, I copied the PDF file to `run` directory.

```
┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$ ls . | grep Truth.pdf
Truth.pdf

┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$
```

The next step is use this command to generate PDF hash file:

```
pdf2john.pl Truth.pdf > pdf.hash
```

The tools generated a PDF hash file below:

```
┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$ ./pdf2john.pl Truth.pdf > pdf.hash

┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$ cat pdf.hash
Truth.pdf:$pdf$4*4*128*-1060*1*16*077e10eba516a741a6285385b42f5b27*32*df507156115f50098c3d8c6fdb1d66220000000000000000000
00000000000000*32*7a46addd4179a8ab90812ae8876369522d5facc72245be4f28b3559473767d57
```

To did brute forcing password using this tools I use:

```
./john pdf.hash
```

And the tools will brute forcing the password for some moments using their wordlist. After the process was finished, we can show the password using

```
./john --show pdf.hash
```

command and here is the result:

```
┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$ ./john pdf.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PDF [MD5 SHA2 RC4/AES 32/64])
No password hashes left to crack (see FAQ)

┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$ ./john --show pdf.hash
Truth.pdf:subarukun

1 password hash cracked, 0 left

┌──(matryochska㉿Matryoshka)-[~/JohnTheRipper/run]
└─$
```

Based on the result, the password was `subarukun`. After the password was succesfully leaked, I opened the file and read the instructions. Based on the instructions we should erase the title then find uppercase letter, so i used this python code:

```
result = ''.join(c for c in text if c.isupper())
```

or you can run this code.
Based on that code, the result was:

```
PS D:\Workspace\write-up\2023-ARA\Misc-Truth> python .\isUpper.py
SOUNDSLIKEFANDAGO
PS D:\Workspace\write-up\2023-ARA\Misc-Truth>
```

Then, followed the challenge's format, the flag should be:

```
ARA2023{SOUNDS_LIKE_FANDAGO}
```

Give feedback