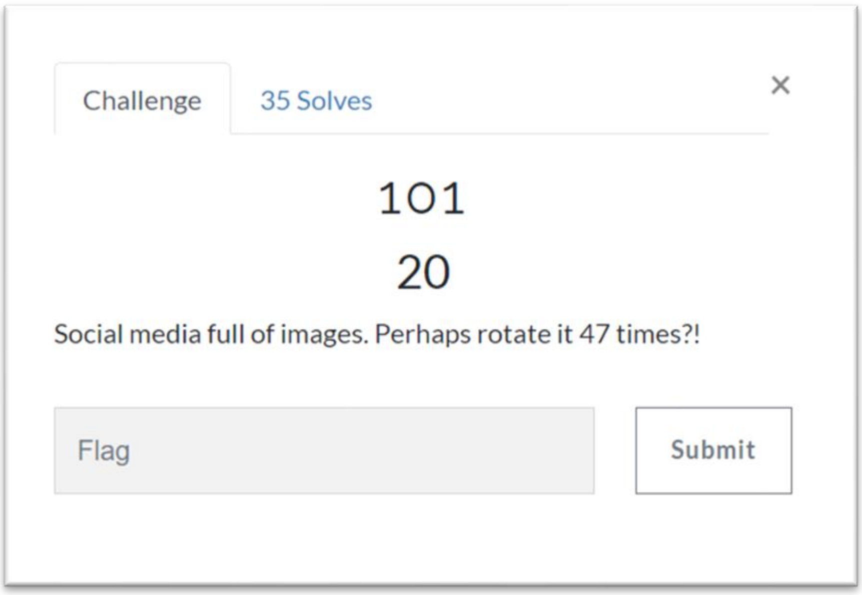
HACK@10 Capture The Flag

Writeups by Pengg0damn

Table of Contents

Category: OSINT

Challenge: 101


Challenge

35 Solves

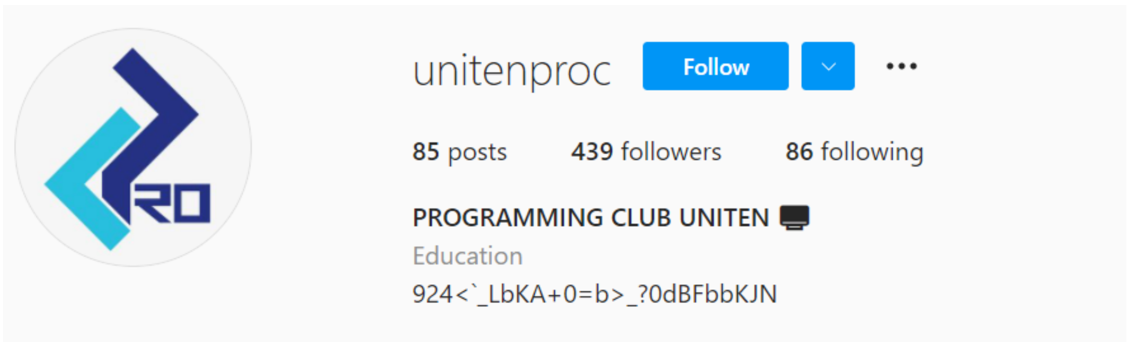
101

20

Social media full of images. Perhaps rotate it 47 times?!

As we know, OSINT (Open-Source Intelligence) challenge required us to go and find the information that can legally be gathered for free and public source about the individual or organization.

Based on the challenge description, it says that social media full of images. So, the flag may be found in Instagram. First, we go through the Uniten and CCI Uniten Instagram but found nothing, then we found that Uniten Programming Club also tagging the competition poster, so we try to look at their Instagram. Then we found the cipher text.


unitenproc

85 posts 439 followers 86 following

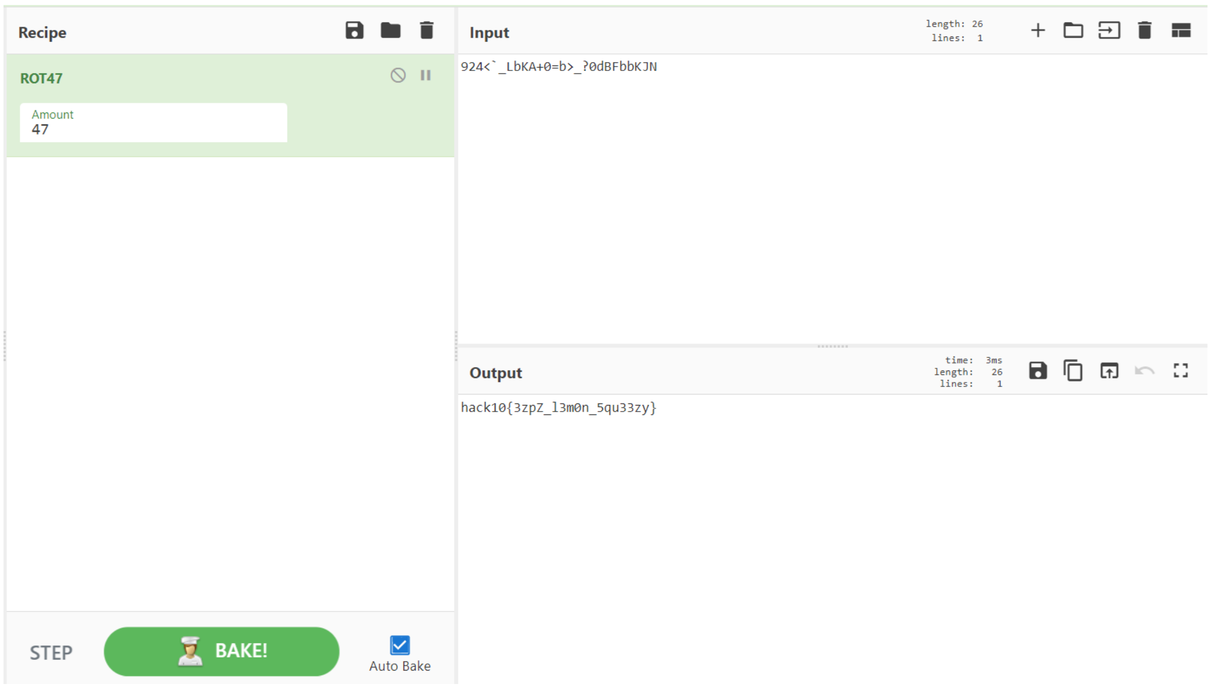
PROGRAMMING CLUB UNITEN

Education

924<`_LbKA+0=b>_?0dBFbbKJN

What we need to do is decrypt the cipher text using ROT 47 method.

We use CyberChef to decode and get the flag.



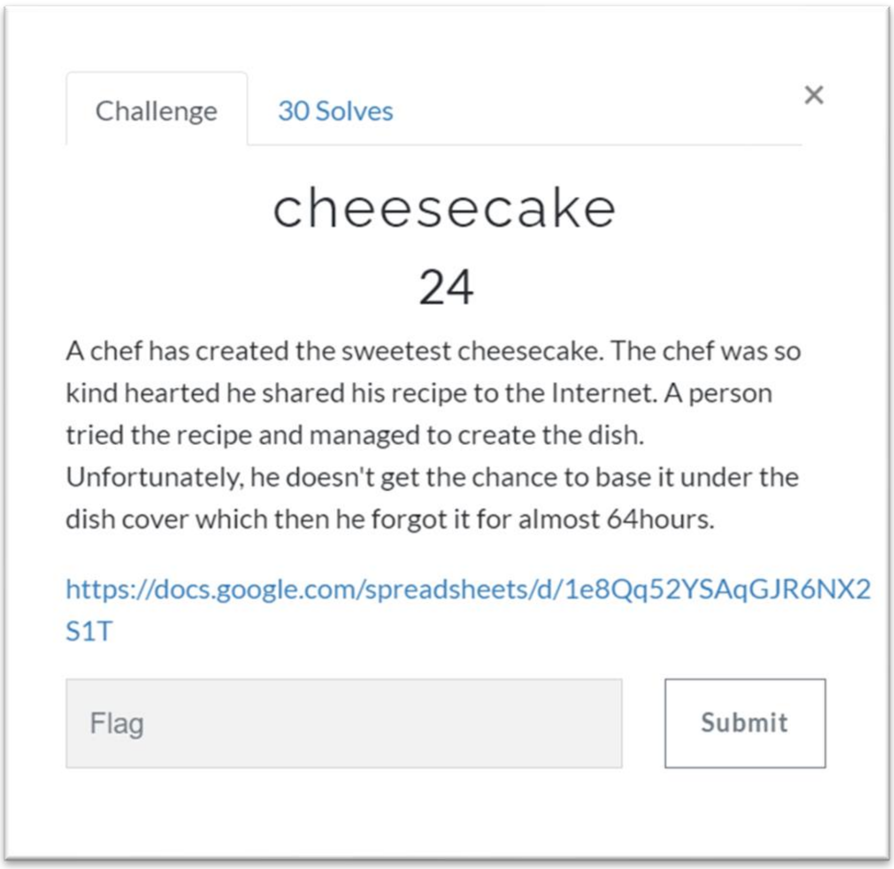
```
924<`_LbKA+0=b>_?0dBFbbKJN
```

```
hack10{3zpZ_l3m0n_5qu33zy}
```

Flag: hack10{3zpZ_l3m0n_5qu33zy}

Category: Misc

Challenge: cheesecake



Challenge | 30 Solves

cheesecake

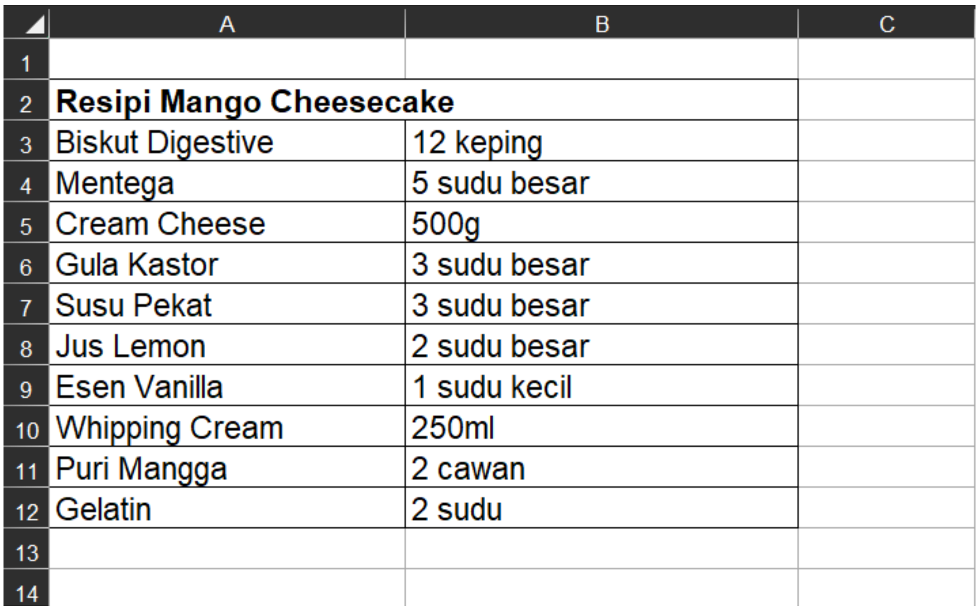
24

A chef has created the sweetest cheesecake. The chef was so kind hearted he shared his recipe to the Internet. A person tried the recipe and managed to create the dish. Unfortunately, he doesn't get the chance to base it under the dish cover which then he forgot it for almost 64hours.

https://docs.google.com/spreadsheets/d/1e8Qq52YSAqGJR6NX2S1T

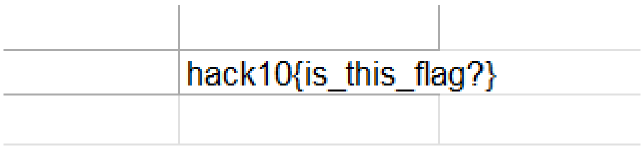
Flag | Submit

This challenge link takes us to Microsoft Excel site that show cheesecake recipe. So, to make our investigation easier, we download the excel file.



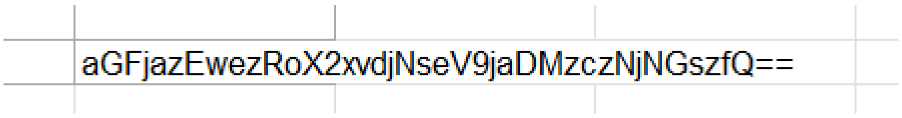
	A	B	C
1			
2	**Resipi Mango Cheesecake**		
3	Biskut Digestive	12 keping	
4	Mentega	5 sudu besar	
5	Cream Cheese	500g	
6	Gula Kastor	3 sudu besar	
7	Susu Pekat	3 sudu besar	
8	Jus Lemon	2 sudu besar	
9	Esen Vanilla	1 sudu kecil	
10	Whipping Cream	250ml	
11	Puri Mangga	2 cawan	
12	Gelatin	2 sudu	
13			
14			

We go through the file but nothing suspicious found. Then we found that there are 2 more sheet that hidden. So, we unhide the sheet and search through it, WALLA, we found something.



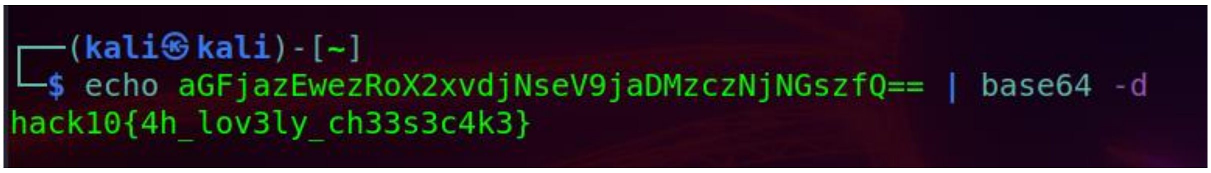
hack10{is_this_flag?}

We thought it was flag, but when we submit, the flag is incorrect. Then we continue searching in sheet 3. At the same row and column, we found the cipher text.



aGFjazEwezRoX2xvdjNseV9jaDMzczNjNGszfQ==

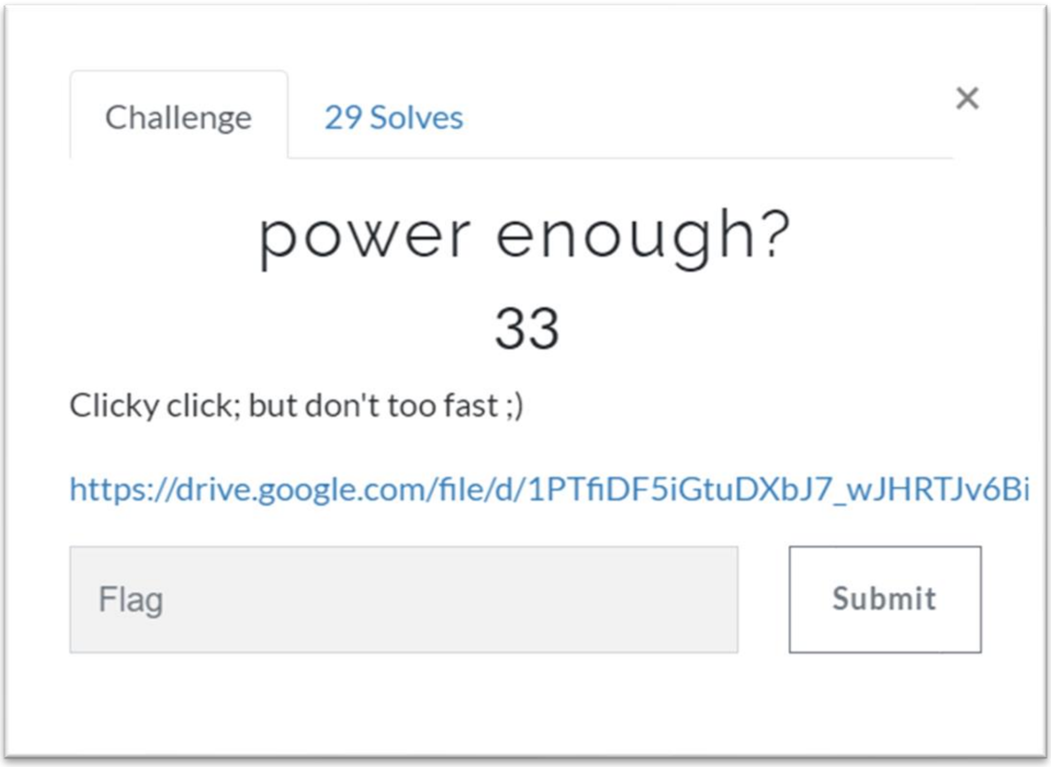
And it may be the flag, so we decrypt it using base64.



```
┌──(kali㉿kali)-[~]
└─$ echo aGFjazEwezRoX2xvdjNseV9jaDMzczNjNGszfQ== | base64 -d
hack10{4h_lov3ly_ch33s3c4k3}
```

Flag: hack10{4h_lov3ly_ch33s3c4k3}

Challenge: power enough?



Challenge

29 Solves

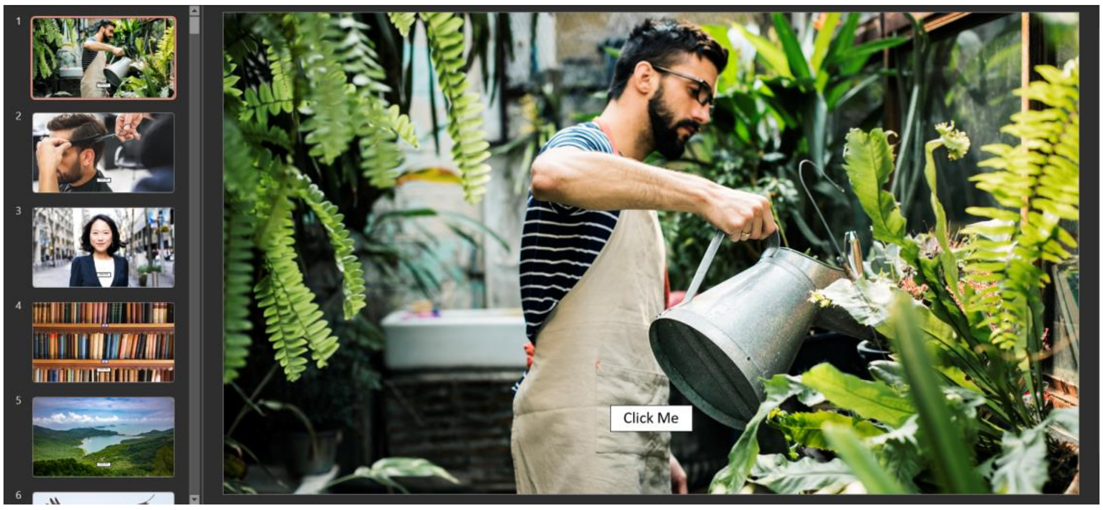
power enough?

33

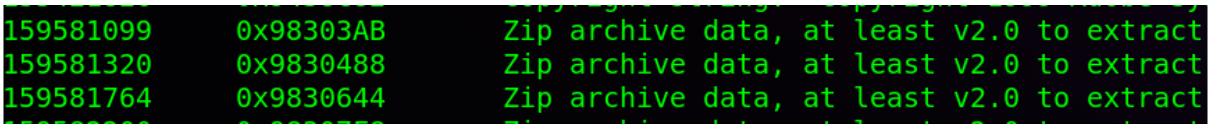
Clicky click; but don't too fast ;)

https://drive.google.com/file/d/1PTfiDF5iGtuDXbJ7_wJHRTJv6Bi

The link given in this challenge give us pptm file, so that we can open using PowerPoint. We try to find the flag in the pptm but it was useless.



So, we run binwalk to check the file state and found interesting thing inside the pptm file.



```
159581099     0x98303AB     Zip archive data, at least v2.0 to extract
159581320     0x9830488     Zip archive data, at least v2.0 to extract
159581764     0x9830644     Zip archive data, at least v2.0 to extract
```

It shows that we can extract something from the pptm, so we change the pptm extension to zip and extract the file. We search through all file and find the flag.

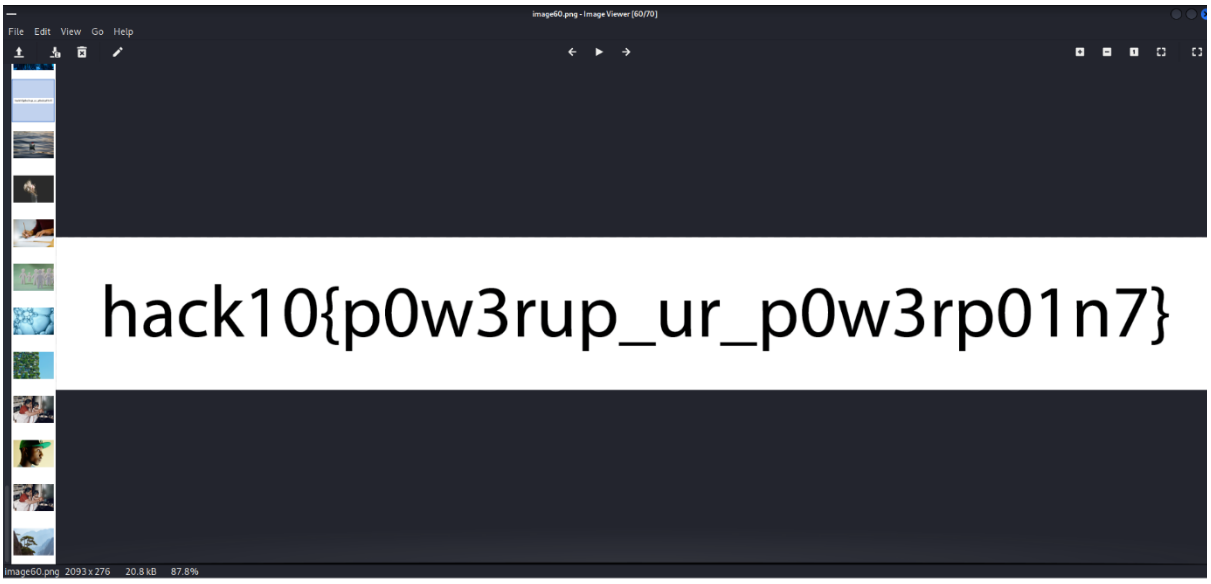
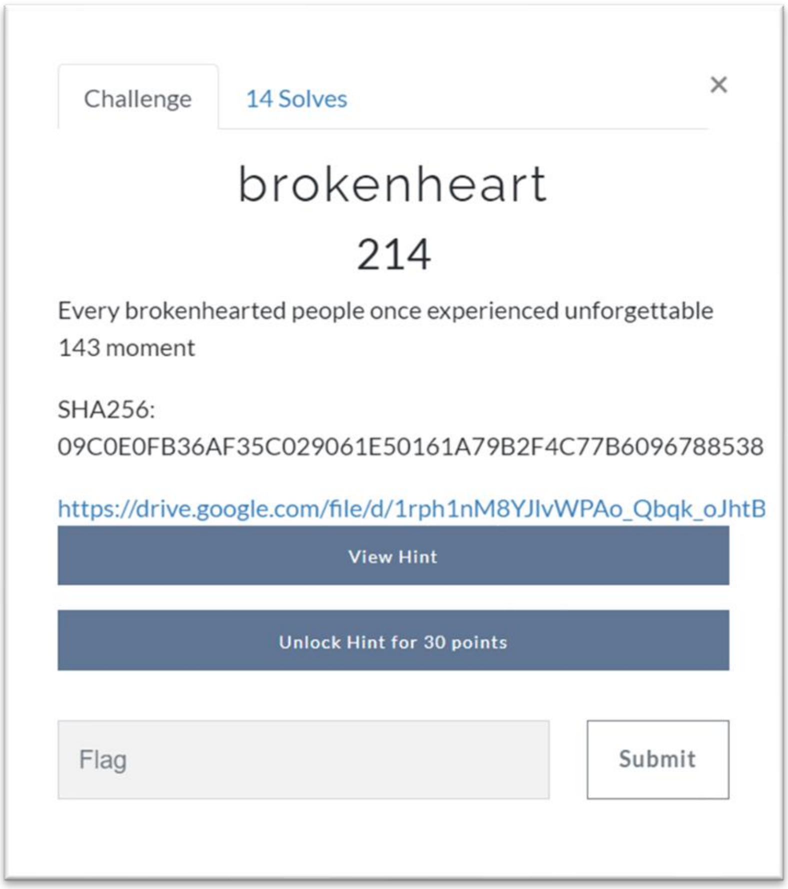


image60.png - Image Viewer [60/70]
File Edit View Go Help
image60.png 2093 x 276 20.8 kB 87.8%

hack10{p0w3rup_ur_p0w3rp01n7}

Flag: hack10{p0w3rup_ur_p0w3rp01n7}

Challenge: brokenheart


Challenge

14 Solves

brokenheart

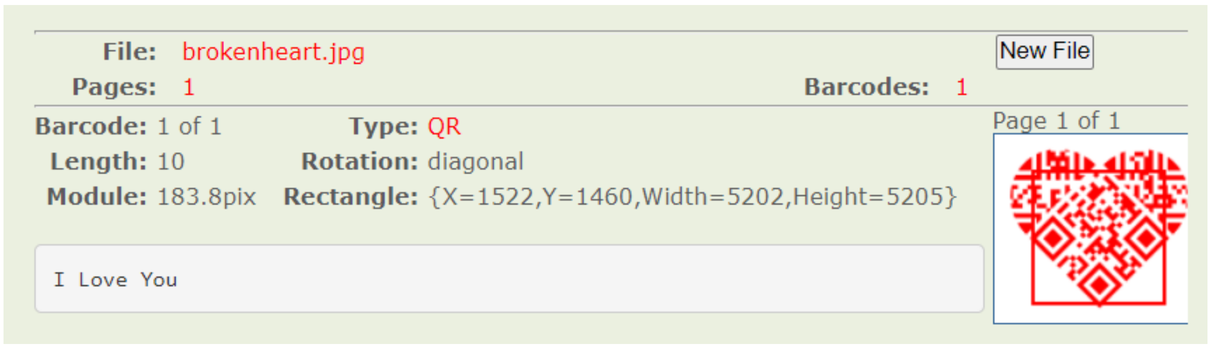
214

Every brokenhearted people once experienced unforgettable 143 moment

SHA256: 09C0E0FB36AF35C029061E50161A79B2F4C77B6096788538

https://drive.google.com/file/d/1rph1nM8YJlvWPAo_Qbqk_oJhtB

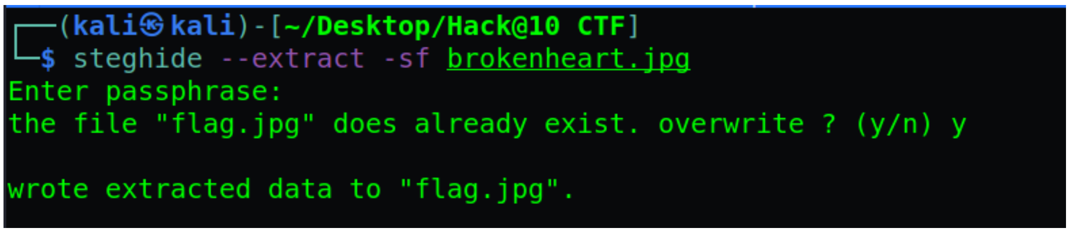
The link gives us an image file of qr code. We try to read the code and it only show I Love You.


File: brokenheart.jpg
Pages: 1 **Barcodes:** 1

Barcode: 1 of 1 **Type:** QR
Length: 10 **Rotation:** diagonal
Module: 183.8pix **Rectangle:** {X=1522,Y=1460,Width=5202,Height=5205}

```
I Love You
```

So as this is the image, we try basic tool like StegSolve. But when we try to extract, it needs a password. My assumption is very brutal as I try the ILoveYou as the password and it extract the flag, Woo-hoo.

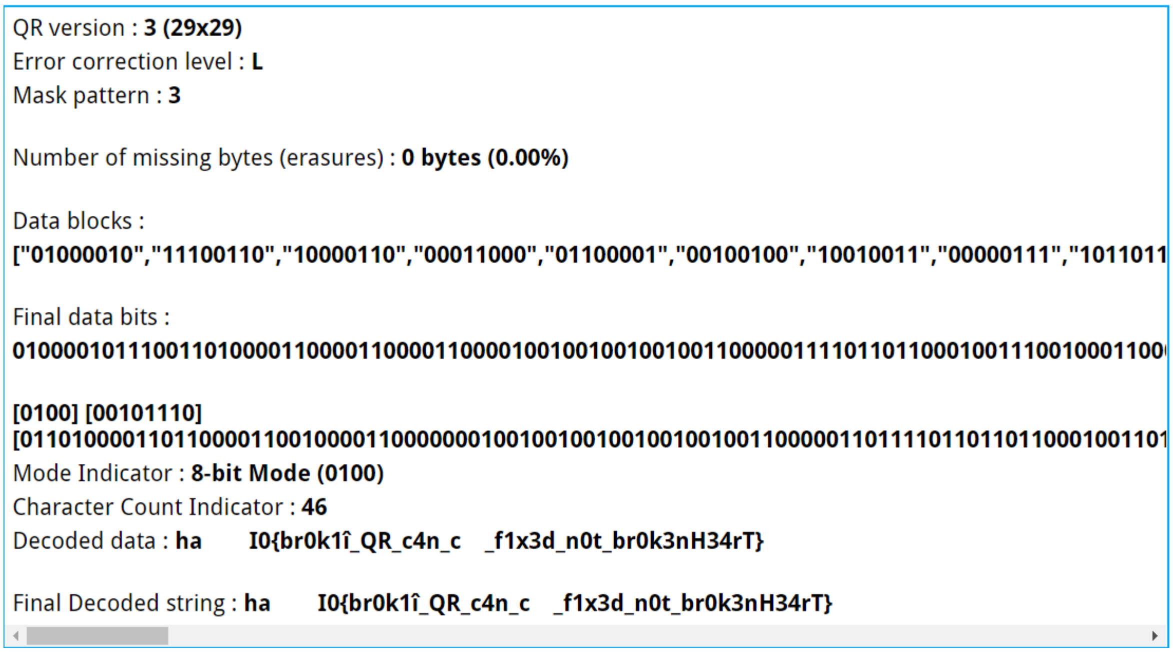


```
┌──(kali㉿kali)-[~/Desktop/Hack@10 CTF]
└─$ steghide --extract -sf brokenheart.jpg
Enter passphrase:
the file "flag.jpg" does already exist. overwrite ? (y/n) y

wrote extracted data to "flag.jpg".
```

Then we got the FLAG. Just kidding, it appears that the flag.jpg is another qr code. ;)

But the problem is the qr code is broken and cannot be read, so we edit the qr so that it can be read. Still, it cannot be read, we edit once again to revert the white and black colour. Finally, we can read the qr code, but it still broken.

So, we try to open the qr using qr forensic tools.



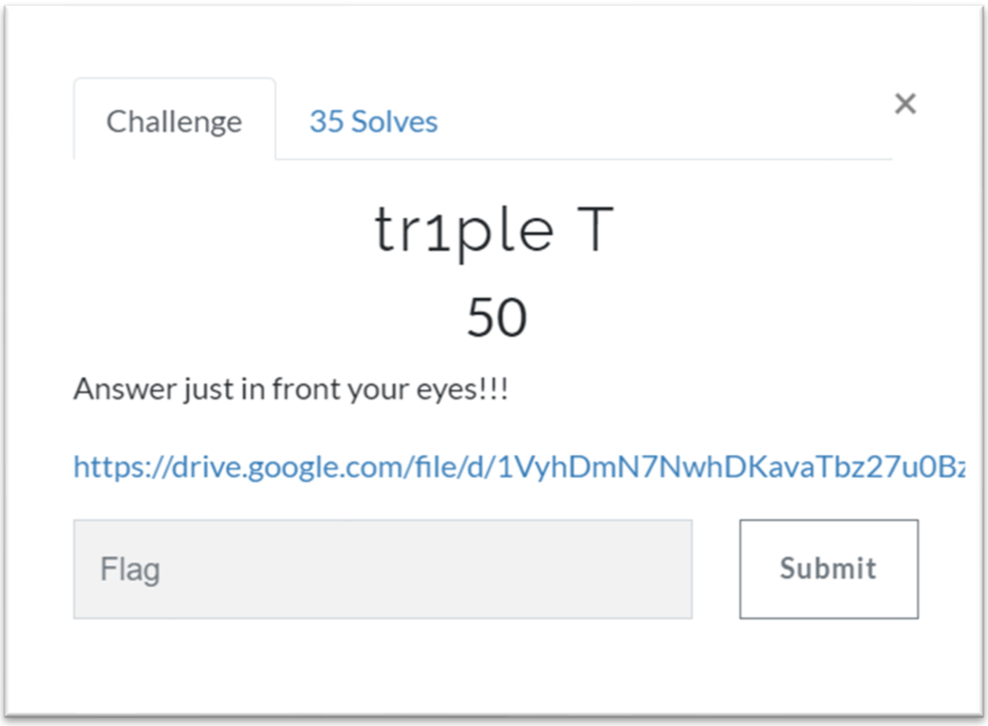
```
QR version : 3 (29x29)
Error correction level : L
Mask pattern : 3

Number of missing bytes (erasures) : 0 bytes (0.00%)

Data blocks :
["01000010","11100110","10000110","00011000","01100001","00100100","10010011","00000111","1011011

Final data bits :
0100001011100110100001100001100001100001001001001001001100000111101101100010011100100011000

[0100] [00101110]
[01101000011011000011001000011000000001001001001001001001001100000110111101101101100010011101
Mode Indicator : 8-bit Mode (0100)
Character Count Indicator : 46
Decoded data : ha      I0{br0k1î_QR_c4n_c   _f1x3d_n0t_br0k3nH34rT}

Final Decoded string : ha      I0{br0k1î_QR_c4n_c   _f1x3d_n0t_br0k3nH34rT}
```

We found the qr information, what we need to do is we need to repair the qr code, but I don't know how to do it. Therefore, I try to craft the flag myself as it was human readable. Then we got the flag. XD

Flag: hack10{br0k3n_QR_c4n_b3_f1x3d_n0t_br0k3nH34rT}

Category: Cryptography

Challenge: tr1ple T



Challenge
35 Solves

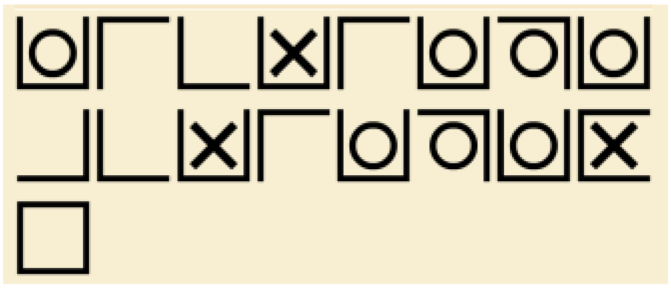
tr1ple T

50

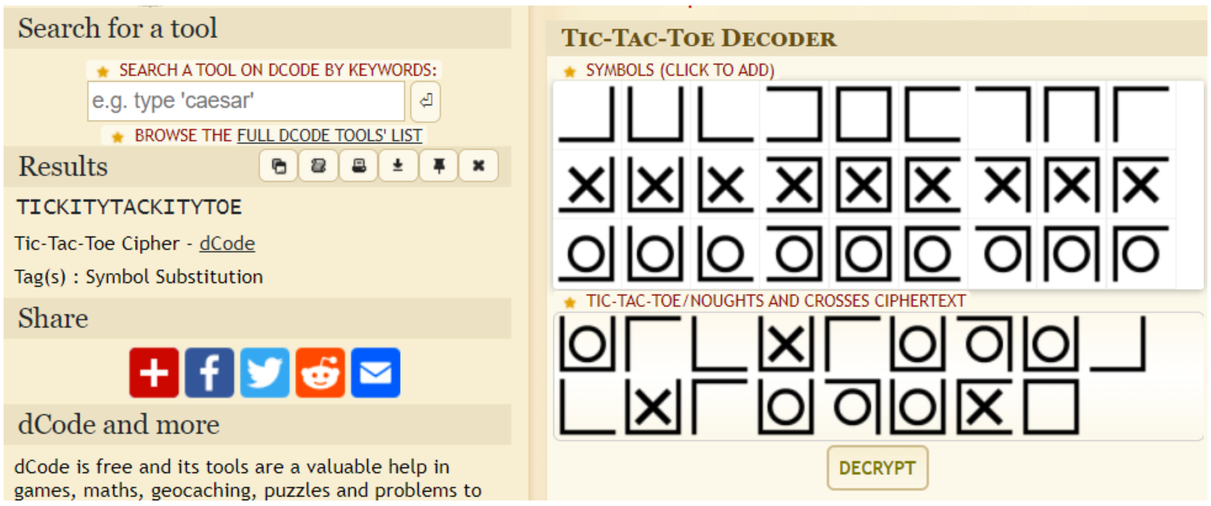
Answer just in front your eyes!!!

https://drive.google.com/file/d/1VyhDmN7NwhDKavaTbz27u0Bz

Cryptography challenge, yeayy….. . So, from the link we got image file like cryptographic text.



From further research we found that this is Tic-Tac-Toe cipher as the name of the challenge is triple T, so we can directly decrypt the cipher.


Search for a tool

SEARCH A TOOL ON DCODE BY KEYWORDS:

e.g. type 'caesar'

BROWSE THE FULL DCODE TOOLS' LIST

Results

TICKITYTACKITYTOE

Tic-Tac-Toe Cipher - dCode

Tag(s) : Symbol Substitution

Share

dCode and more

dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to

TIC-TAC-TOE DECODER

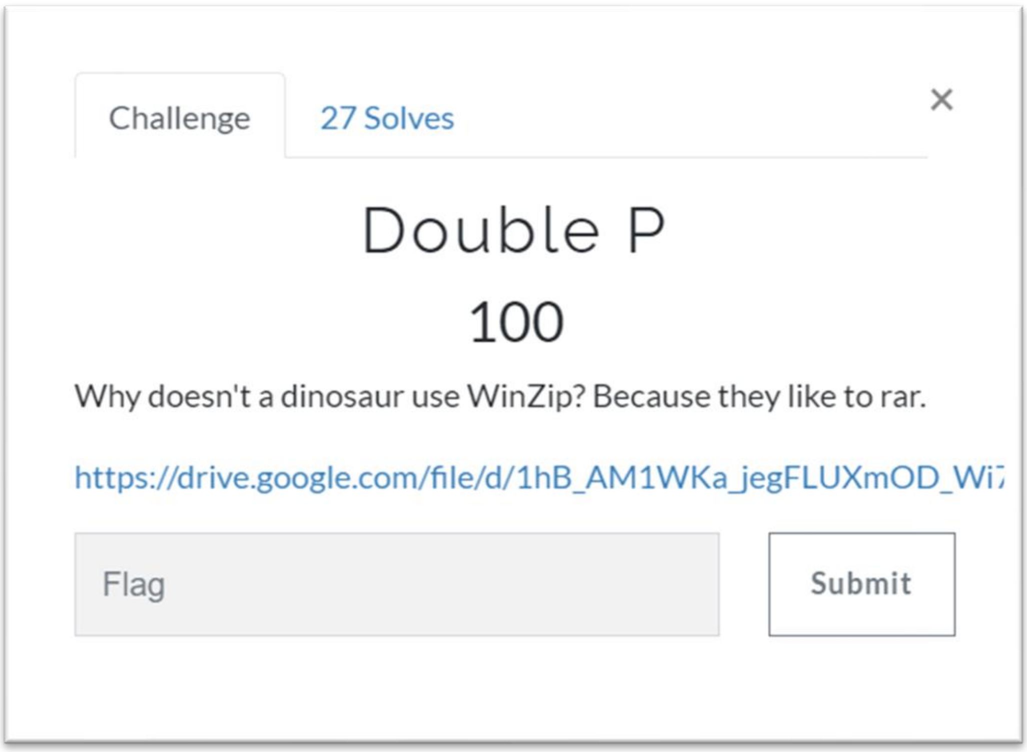
SYMBOLS (CLICK TO ADD)

TIC-TAC-TOE/NOUGHTS AND CROSSES CIPHERTEXT

DECRYPT

Flag: hack10{TICKITYTACKITYTOE}

Challenge: Double P


Challenge
27 Solves

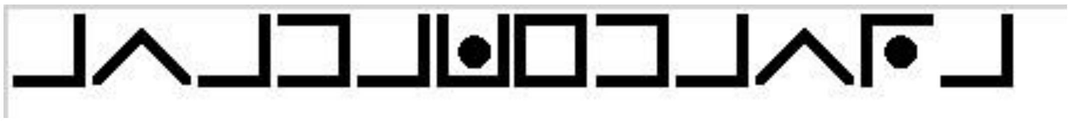
Double P

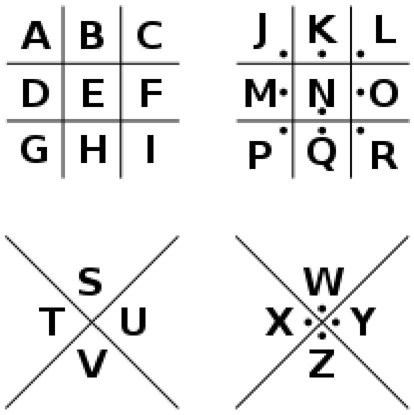
100

Why doesn't a dinosaur use WinZip? Because they like to rar.

https://drive.google.com/file/d/1hB_AM1WKa_jegFLUXmOD_Wi7

Another cryptography challenges. We got another cipher image. After searching a little bit, we found that this is pigpen cipher.

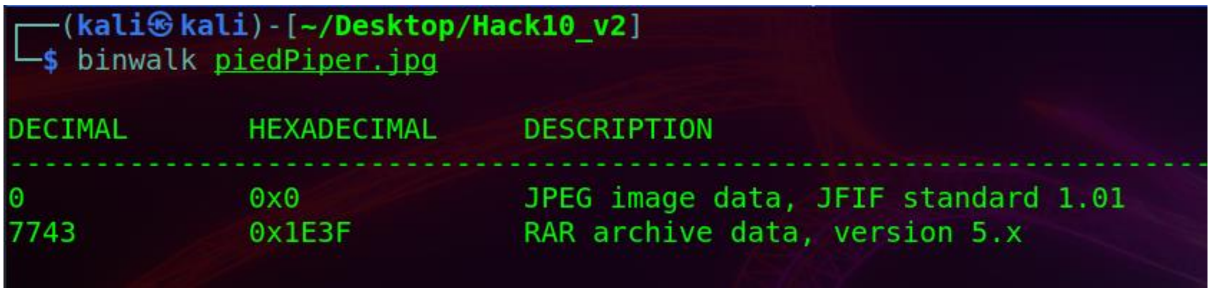




We found the hidden message that we thought it a flag but it not. So, what is this?

Hidden message: avadakedavra

We further our investigation using binwalk and found something fishy inside the image file.



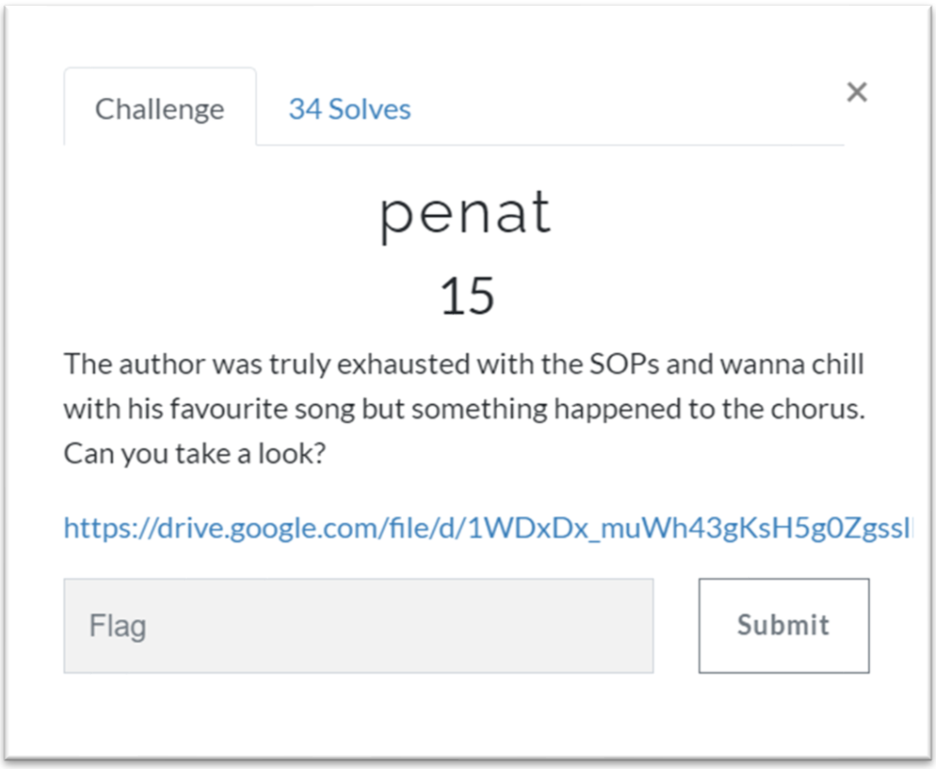
```
┌──(kali㉿kali)-[~/Desktop/Hack10_v2]
└─$ binwalk piedPiper.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
7743          0x1E3F          RAR archive data, version 5.x
```

We found RAR file inside the image, so let’s bring out the RAR file. We try to extract the file, but it requests a password. But wait, how about we use the message that we decrypt earlier “avadakedavra”. And the file extracted then we read the flag.

Flag: hack10{y0u_f0uNd_m3!}

Category: Steganography

Challenge: penat


Challenge

34 Solves

×

penat

15

The author was truly exhausted with the SOPs and wanna chill with his favourite song but something happened to the chorus. Can you take a look?

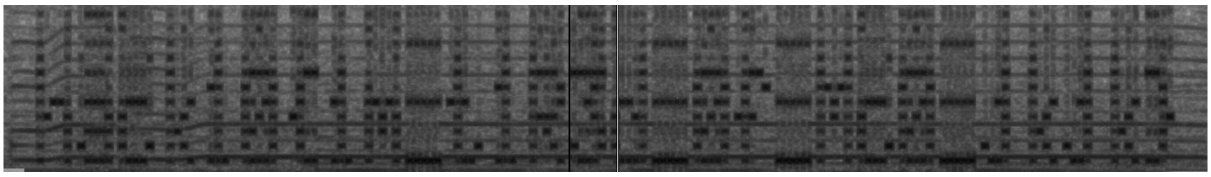
https://drive.google.com/file/d/1WDxDx_muWh43gKsH5g0Zgssl

Flag

Submit

So, we got into steganography challenge, the challenge name is same as our condition now. We got the wav sound file, and we try to open the file. It come out the song that we never hear before but somehow we hear weird sound like alien inside the song. So, the first thought is it may have spectrogram message inside the file.

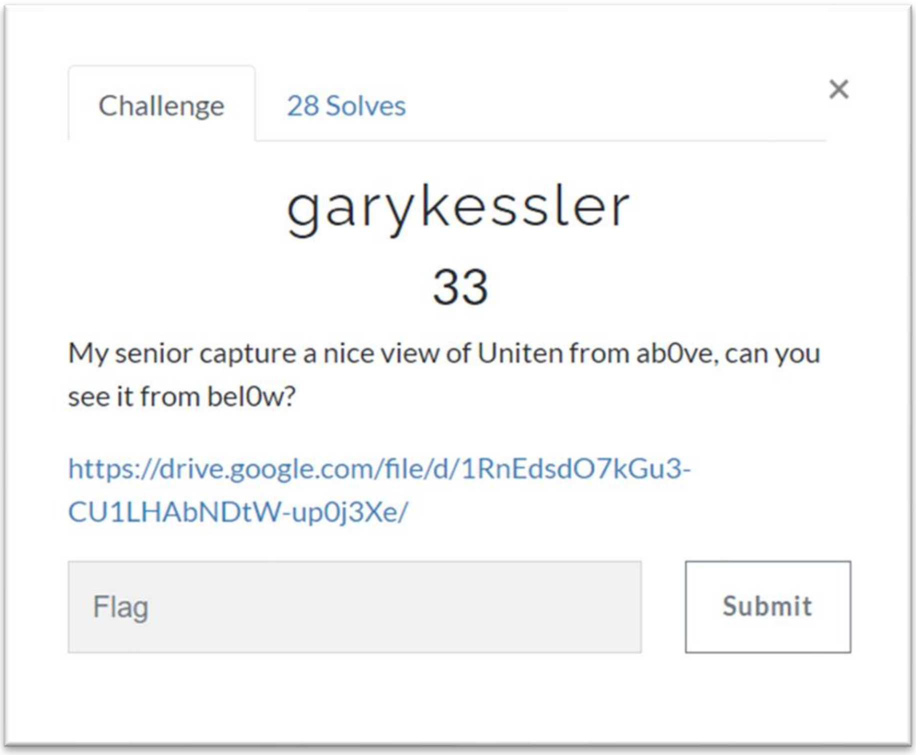
We open the wav file using sonic visualizer and edit the file so that it will show us the sound spectrogram. And we got the flag.



We have hard time when reading the flag, but by using our BLOOD, SWEAT AND TEARS we got the flag.

Flag: hack10{1m_t1R3d_0f_Mc0_jkjk}

Challenge: garykessler


Challenge

28 Solves

garykessler

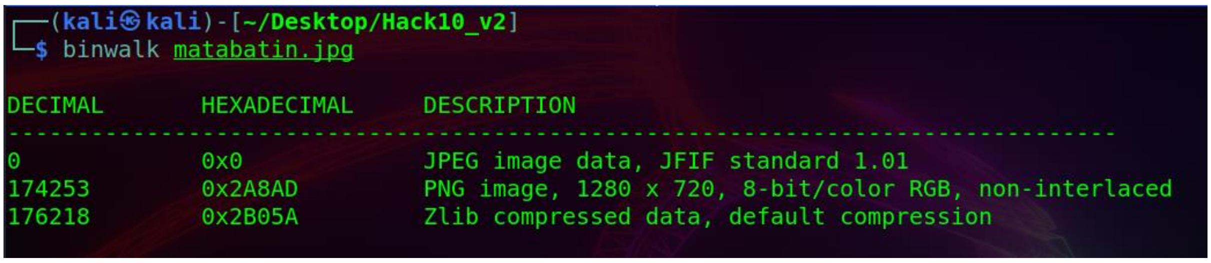
33

My senior capture a nice view of Uniten from ab0ve, can you see it from bel0w?

https://drive.google.com/file/d/1RnEdsdO7kGu3-CU1LHAbNDtW-up0j3Xe/

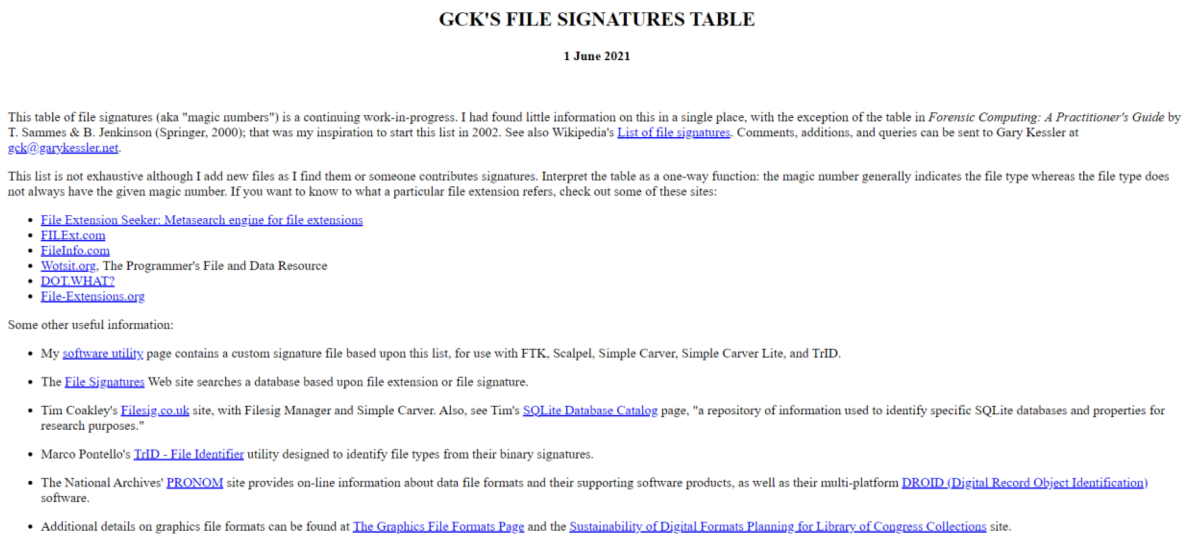
Our next steganography challenge is garykessler. The link provides us with beautiful Uniten photo.

So, as the first step of steganography, we will try with basic steganography tools like StegSolve, Zsteg, StegSeek and Strings. But we found nothing. Next, we use binwalk to check the file binary.



```
┌──(kali㉿kali)-[~/Desktop/Hack10_v2]
└─$ binwalk matabatin.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
174253        0x2A8AD         PNG image, 1280 x 720, 8-bit/color RGB, non-interlaced
176218        0x2B05A         Zlib compressed data, default compression
```

We found something! The binwalk show that it has PNG file inside the photo. But we cannot extract it normally. We make further research and realise something with the challenge name, “garykessler”. We try to search about garykessler and found a page show about hex signature of the file.



GCK'S FILE SIGNATURES TABLE

1 June 2021

This table of file signatures (aka "magic numbers") is a continuing work-in-progress. I had found little information on this in a single place, with the exception of the table in *Forensic Computing: A Practitioner's Guide* by T. Sammes & B. Jenkinson (Springer, 2000); that was my inspiration to start this list in 2002. See also Wikipedia's List of file signatures. Comments, additions, and queries can be sent to Gary Kessler at gck@garykessler.net.

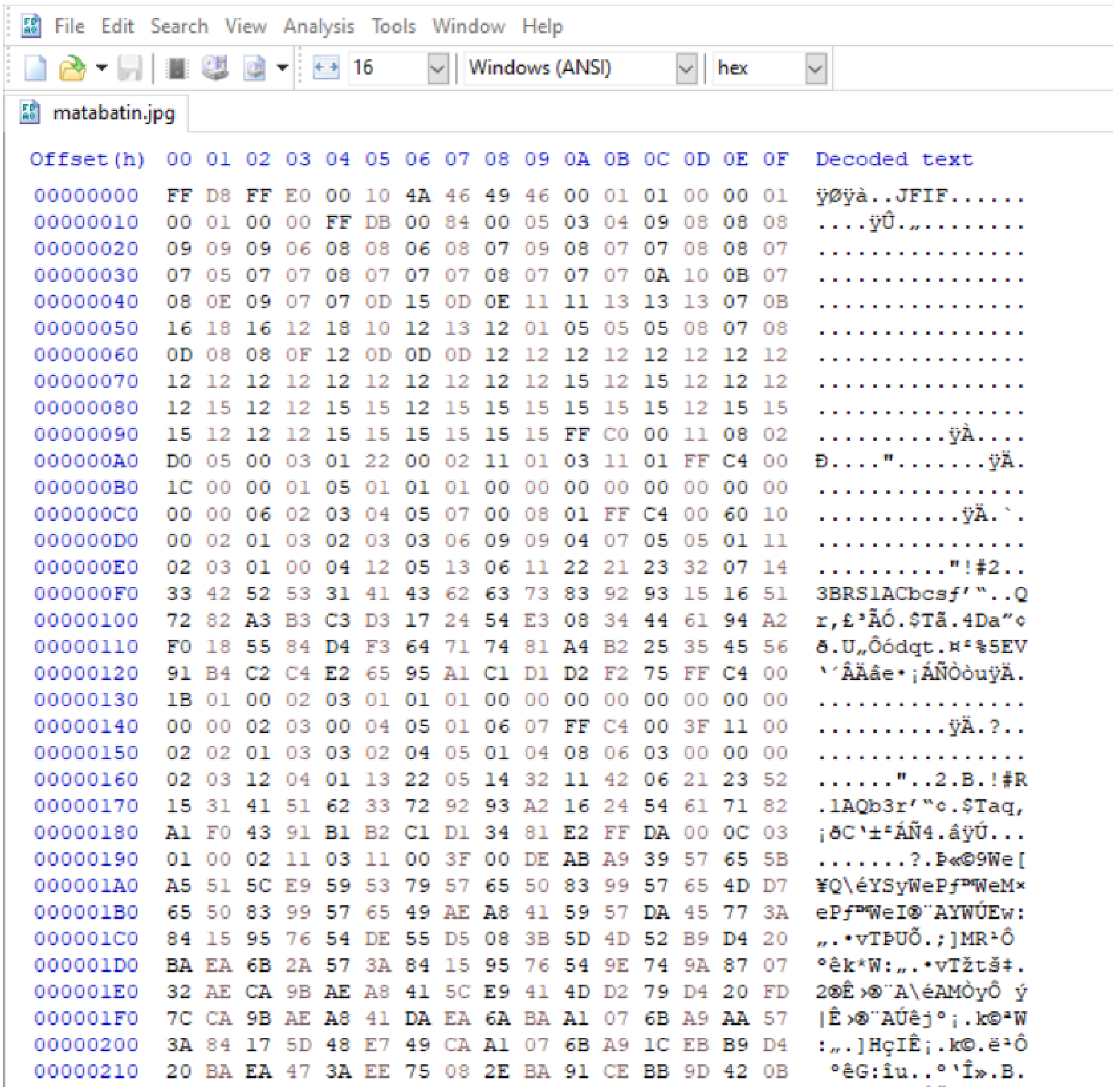
This list is not exhaustive although I add new files as I find them or someone contributes signatures. Interpret the table as a one-way function: the magic number generally indicates the file type whereas the file type does not always have the given magic number. If you want to know to what a particular file extension refers, check out some of these sites:

- File Extension Seeker: Metasearch engine for file extensions
- FILExt.com
- FileInfo.com
- Wotsit.org, The Programmer's File and Data Resource
- DOT.WHAT?
- File-Extensions.org

Some other useful information:

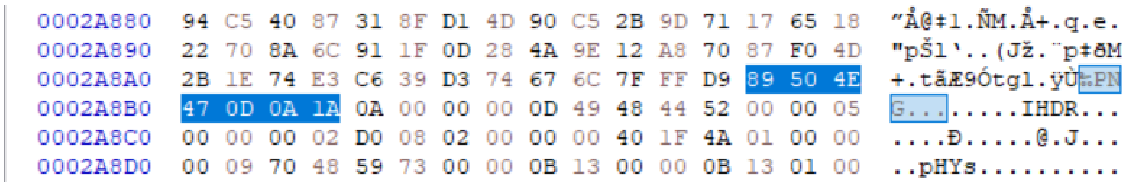
- My software utility page contains a custom signature file based upon this list, for use with FTK, Scalpel, Simple Carver, Simple Carver Lite, and TrID.
- The File Signatures Web site searches a database based upon file extension or file signature.
- Tim Coakley's Filesig.co.uk site, with Filesig Manager and Simple Carver. Also, see Tim's SQLite Database Catalog page, "a repository of information used to identify specific SQLite databases and properties for research purposes."
- Marco Pontello's TrID - File Identifier utility designed to identify file types from their binary signatures.
- The National Archives' PRONOM site provides on-line information about data file formats and their supporting software products, as well as their multi-platform DROID (Digital Record Object Identification) software.
- Additional details on graphics file formats can be found at The Graphics File Formats Page and the Sustainability of Digital Formats Planning for Library of Congress Collections site.

So, we try to open the image using hex editor and guess what we found?



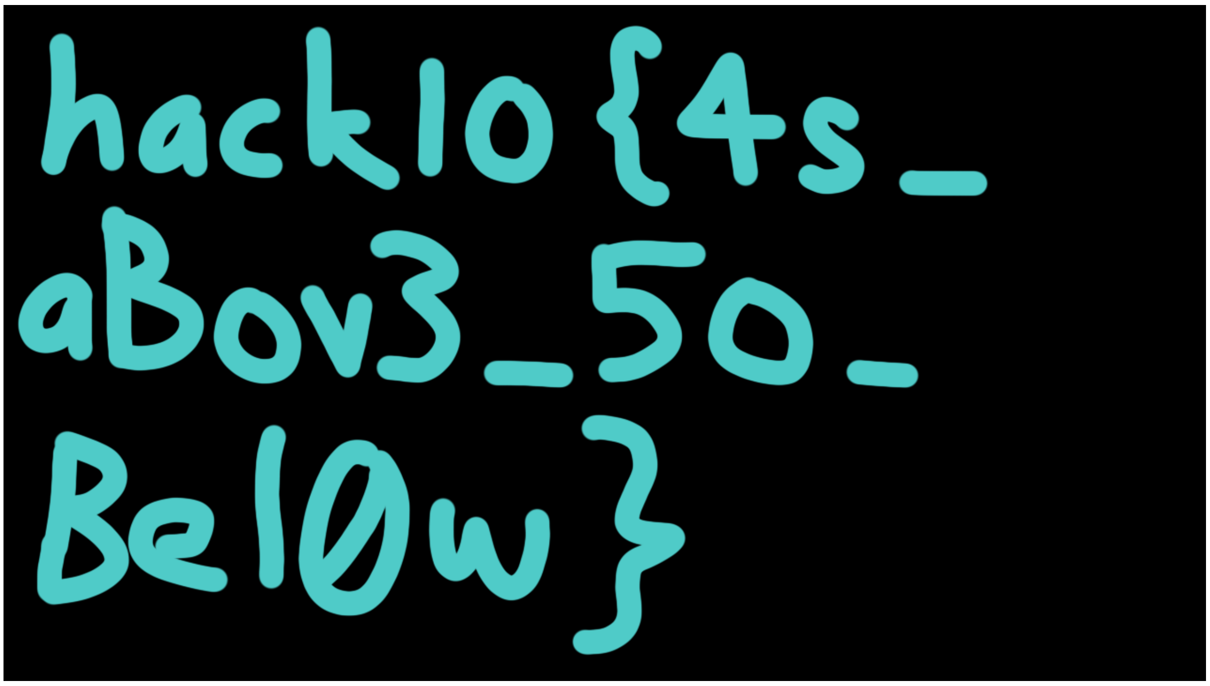
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01  ÿØÿà..JFIF......
00000010  00 01 00 00 FF DB 00 84 00 05 03 04 09 08 08 08  ....ÿÛ.„........
00000020  09 09 09 06 08 08 06 08 07 09 08 07 07 08 08 07  ................
00000030  07 05 07 07 08 07 07 07 08 07 07 07 0A 10 0B 07  ................
00000040  08 0E 09 07 07 0D 15 0D 0E 11 11 13 13 13 07 0B  ................
00000050  16 18 16 12 18 10 12 13 12 01 05 05 05 08 07 08  ................
00000060  0D 08 08 0F 12 0D 0D 0D 12 12 12 12 12 12 12 12  ................
00000070  12 12 12 12 12 12 12 12 12 12 15 12 15 12 12 12  ................
00000080  12 15 12 12 15 15 12 15 15 15 15 15 15 12 15 15  ................
00000090  15 12 12 12 15 15 15 15 15 15 FF C0 00 11 08 02  ..........ÿÀ....
000000A0  D0 05 00 03 01 22 00 02 11 01 03 11 01 FF C4 00  Ð...."......ÿÄ.
000000B0  1C 00 00 01 05 01 01 01 00 00 00 00 00 00 00 00  ................
000000C0  00 00 06 02 03 04 05 07 00 08 01 FF C4 00 60 10  ...........ÿÄ.`.
000000D0  00 02 01 03 02 03 03 06 09 09 04 07 05 05 01 11  ................
000000E0  02 03 01 00 04 12 05 13 06 11 22 21 23 32 07 14  .........."!#2..
000000F0  33 42 52 53 31 41 43 62 63 73 83 92 93 15 16 51  3BRS1ACbcsƒ'"..Q
00000100  72 82 A3 B3 C3 D3 17 24 54 E3 08 34 44 61 94 A2  r‚£³ÃÓ.$Tã.4Da”¢
00000110  F0 18 55 84 D4 F3 64 71 74 81 A4 B2 25 35 45 56  ð.U„Ôódqt.¤²%5EV
00000120  91 B4 C2 C4 E2 65 95 A1 C1 D1 D2 F2 75 FF C4 00  '´ÂÄâe•¡ÁÑÒòuÿÄ.
00000130  1B 01 00 02 03 01 01 01 00 00 00 00 00 00 00 00  ................
00000140  00 00 02 03 00 04 05 01 06 07 FF C4 00 3F 11 00  ..........ÿÄ.?..
00000150  02 02 01 03 03 02 04 05 01 04 08 06 03 00 00 00  ................
00000160  02 03 12 04 01 13 22 05 14 32 11 42 06 21 23 52  ......"..2.B.!#R
00000170  15 31 41 51 62 33 72 92 93 A2 16 24 54 61 71 82  .1AQb3r'"¢.$Taq‚
00000180  A1 F0 43 91 B1 B2 C1 D1 34 81 E2 FF DA 00 0C 03  ¡ðC'±²ÁÑ4.âÿÚ...
00000190  01 00 02 11 03 11 00 3F 00 DE AB A9 39 57 65 5B  .......?.Þ«©9We[
000001A0  A5 51 5C E9 59 53 79 57 65 50 83 99 57 65 4D D7  ¥Q\éYSyWePƒ™WeM×
000001B0  65 50 83 99 57 65 49 AE A8 41 59 57 DA 45 77 3A  ePƒ™WeI®¨AYWÚEw:
000001C0  84 15 95 76 54 DE 55 D5 08 3B 5D 4D 52 B9 D4 20  „.•vTÞUÕ.;]MR¹Ô 
000001D0  BA EA 6B 2A 57 3A 84 15 95 76 54 9E 74 9A 87 07  ºêk*W:„.•vTžtš‡.
000001E0  32 AE CA 9B AE A8 41 5C E9 41 4D D2 79 D4 20 FD  2®Ê›®¨A\éAMÒyÔ ý
000001F0  7C CA 9B AE A8 41 DA EA 6A BA A1 07 6B A9 AA 57  |Ê›®¨AÚêjº¡.k©ªW
00000200  3A 84 17 5D 48 E7 49 CA A1 07 6B A9 1C EB B9 D4  :„.]HçIÊ¡.k©.ë¹Ô
00000210  20 BA EA 47 3A EE 75 08 2E BA 91 CE BB 9D 42 0B   ºêG:îu..º'Î».B.
```

It takes a long time in this investigation, we try to change the file header but it not working. After further observation, we found the PNG file header inside the hex.



```
0002A880  94 C5 40 87 31 8F D1 4D 90 C5 2B 9D 71 17 65 18  ”Å@‡1.ÑM.Å+.q.e.
0002A890  22 70 8A 6C 91 1F 0D 28 4A 9E 12 A8 70 87 F0 4D  "pŠl'..(Jž.¨p‡ðM
0002A8A0  2B 1E 74 E3 C6 39 D3 74 67 6C 7F FF D9 89 50 4E  +.tãÆ9Ótgl.ÿÙ‰PN
0002A8B0  47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 05  G.......IHDR...
0002A8C0  00 00 00 02 D0 08 02 00 00 00 40 1F 4A 01 00 00  ....Ð.....@.J...
0002A8D0  00 09 70 48 59 73 00 00 0B 13 00 00 0B 13 01 00  ..pHYs..........
```

So, what we do is, we remove all the hex before the PNG header and save the file. Guess what? We got the flag, yeayyyy…..

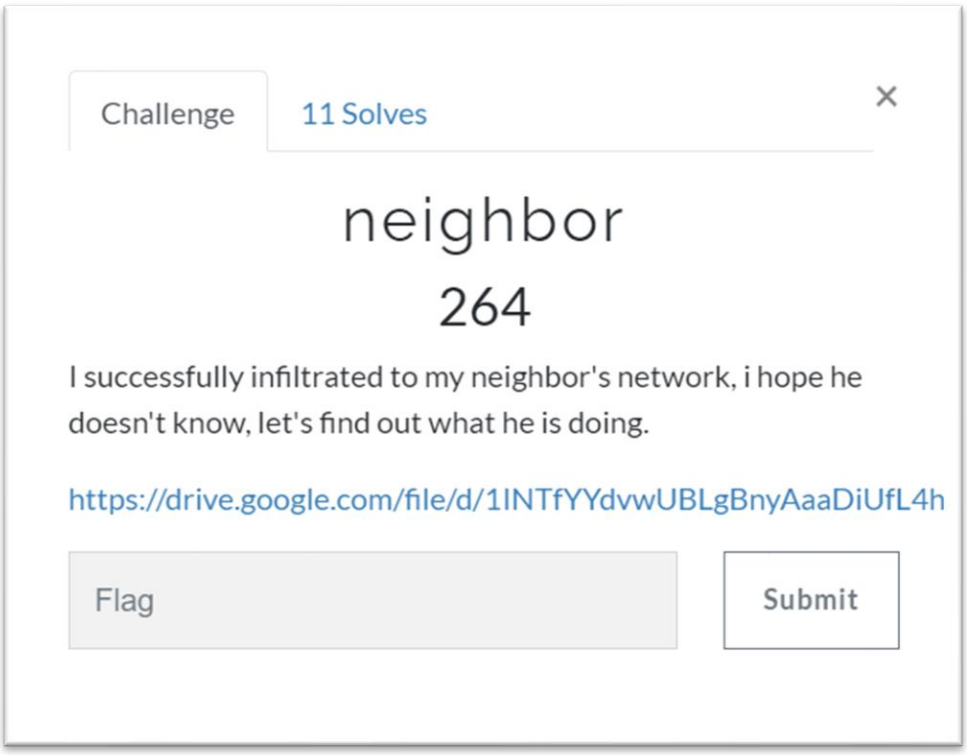


hack10{4s_
aBov3_5o_
Bel0w}

Flag: hack10{4s_aBov3_5o_Bel0w}

Category: Forensics

Challenge: neighbor



Challenge | 11 Solves

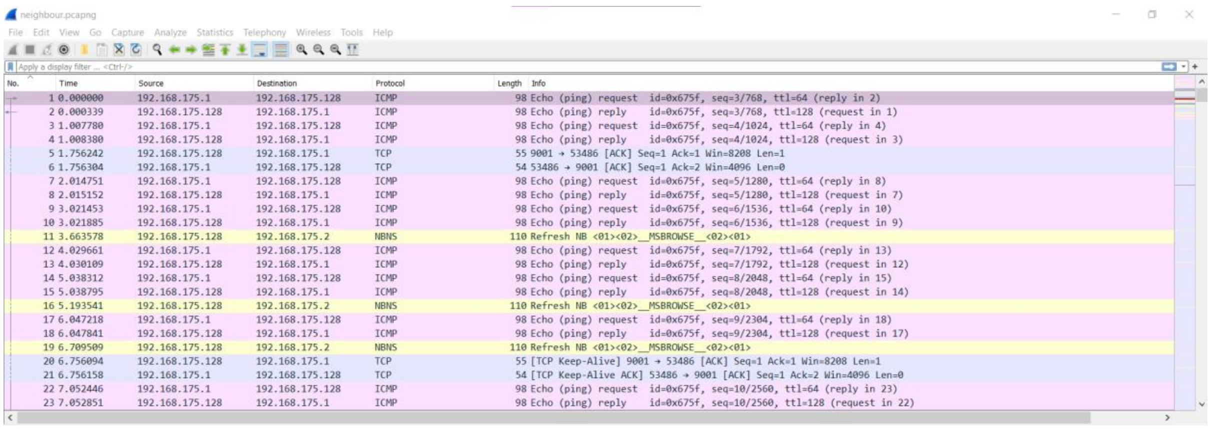
neighbor

264

I successfully infiltrated to my neighbor's network, i hope he doesn't know, let's find out what he is doing.

https://drive.google.com/file/d/1INTfYYdvwUBLgBnyAaaDiUfL4h

This forensics challenge is interesting and very brutal. We almost give up when try to solve it. From the link given, we got the pcapng file, so we try open it using the Wireshark.



neighbour.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time      Source           Destination      Protocol  Length Info
   1 0.000000  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=3/768, ttl=64 (reply in 2)
   2 0.000339  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=3/768, ttl=128 (request in 1)
   3 1.007780  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=4/1024, ttl=64 (reply in 4)
   4 1.008380  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=4/1024, ttl=128 (request in 3)
   5 1.756242  192.168.175.128  192.168.175.1    TCP        55 9001 → 53486 [ACK] Seq=1 Ack=1 Win=8208 Len=1
   6 1.756304  192.168.175.1    192.168.175.128  TCP        54 53486 → 9001 [ACK] Seq=1 Ack=2 Win=4096 Len=0
   7 2.014751  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=5/1280, ttl=64 (reply in 8)
   8 2.015152  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=5/1280, ttl=128 (request in 7)
   9 3.021453  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=6/1536, ttl=64 (reply in 10)
  10 3.021885  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=6/1536, ttl=128 (request in 9)
  11 3.663578  192.168.175.128  192.168.175.2    NBNS      110 Refresh NB <01><02>__MSBROWSE__<02><01>
  12 4.029661  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=7/1792, ttl=64 (reply in 13)
  13 4.030109  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=7/1792, ttl=128 (request in 12)
  14 5.038312  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=8/2048, ttl=64 (reply in 15)
  15 5.038795  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=8/2048, ttl=128 (request in 14)
  16 5.193541  192.168.175.128  192.168.175.2    NBNS      110 Refresh NB <01><02>__MSBROWSE__<02><01>
  17 6.047218  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=9/2304, ttl=64 (reply in 18)
  18 6.047841  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=9/2304, ttl=128 (request in 17)
  19 6.709509  192.168.175.128  192.168.175.2    NBNS      110 Refresh NB <01><02>__MSBROWSE__<02><01>
  20 6.756094  192.168.175.128  192.168.175.1    TCP        55 [TCP Keep-Alive] 9001 → 53486 [ACK] Seq=1 Ack=1 Win=8208 Len=1
  21 6.756158  192.168.175.1    192.168.175.128  TCP        54 [TCP Keep-Alive ACK] 53486 → 9001 [ACK] Seq=1 Ack=2 Win=4096 Len=0
  22 7.052446  192.168.175.1    192.168.175.128  ICMP       98 Echo (ping) request  id=0x675f, seq=10/2560, ttl=64 (reply in 23)
  23 7.052851  192.168.175.128  192.168.175.1    ICMP       98 Echo (ping) reply    id=0x675f, seq=10/2560, ttl=128 (request in 22)
```

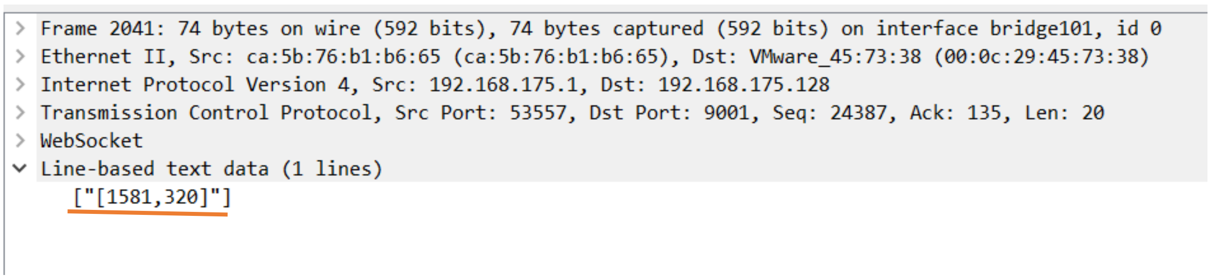
Our observation saw that there are many WebSocket connection inside this pcapng file. We try to read through all the packets but find nothing. We make a further investigation and find one website that we thing may look same as this challenge, because it was the writeup for past ctf so we try to understand the concept.

Source:

https://www.petermstewart.net/otterctf-2018-network-challenges-look-at-me-write-up/

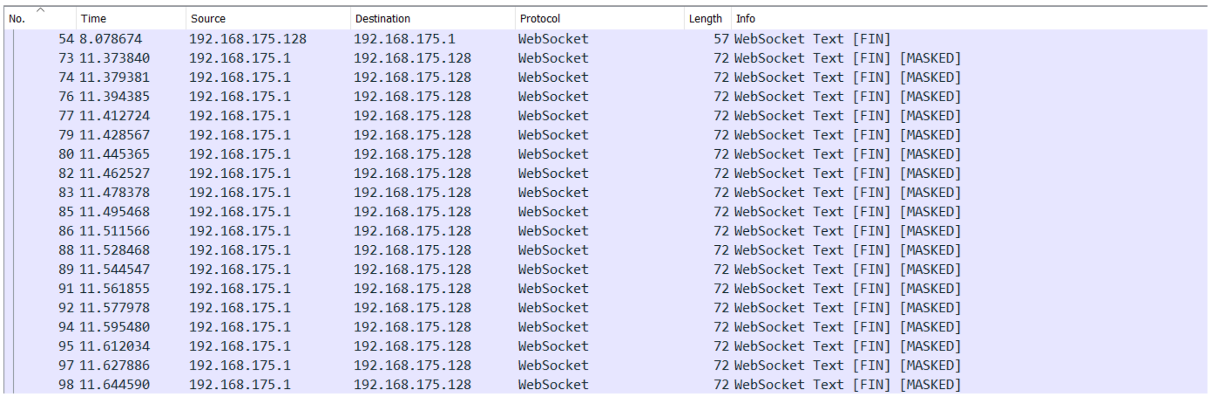
https://www.cyborgsecurity.com/cyborg_labs/cyborg-security-2020-ctf-solutions/

After some reading, we found that inside the WebSocket packet, it contains the data that looks like coordinate, so based on the writeups, we can plot the graph using the coordinate.



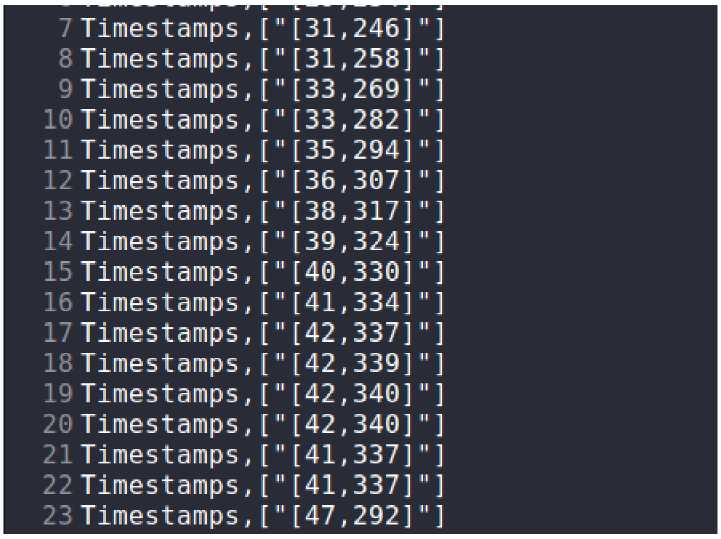
```
> Frame 2041: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface bridge101, id 0
> Ethernet II, Src: ca:5b:76:b1:b6:65 (ca:5b:76:b1:b6:65), Dst: VMware_45:73:38 (00:0c:29:45:73:38)
> Internet Protocol Version 4, Src: 192.168.175.1, Dst: 192.168.175.128
> Transmission Control Protocol, Src Port: 53557, Dst Port: 9001, Seq: 24387, Ack: 135, Len: 20
> WebSocket
v Line-based text data (1 lines)
    ["[1581,320]"]
```

So, we filter the packet and make the output so that only the WebSocket data are available.



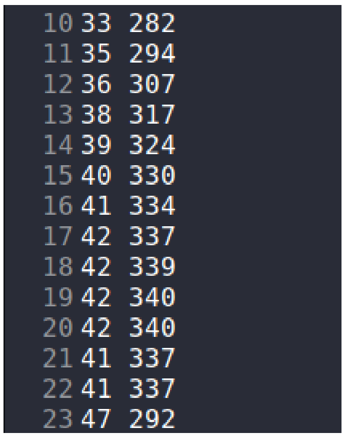
```
No.  Time       Source           Destination      Protocol    Length Info
  54 8.078674   192.168.175.128  192.168.175.1    WebSocket       57 WebSocket Text [FIN]
  73 11.373840  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  74 11.379381  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  76 11.394385  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  77 11.412724  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  79 11.428567  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  80 11.445365  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  82 11.462527  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  83 11.478378  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  85 11.495468  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  86 11.511566  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  88 11.528468  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  89 11.544547  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  91 11.561855  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  92 11.577978  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  94 11.595480  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  95 11.612034  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  97 11.627886  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
  98 11.644590  192.168.175.1    192.168.175.128  WebSocket       72 WebSocket Text [FIN] [MASKED]
```

Then, we use Tshark to get only the coordinate data.



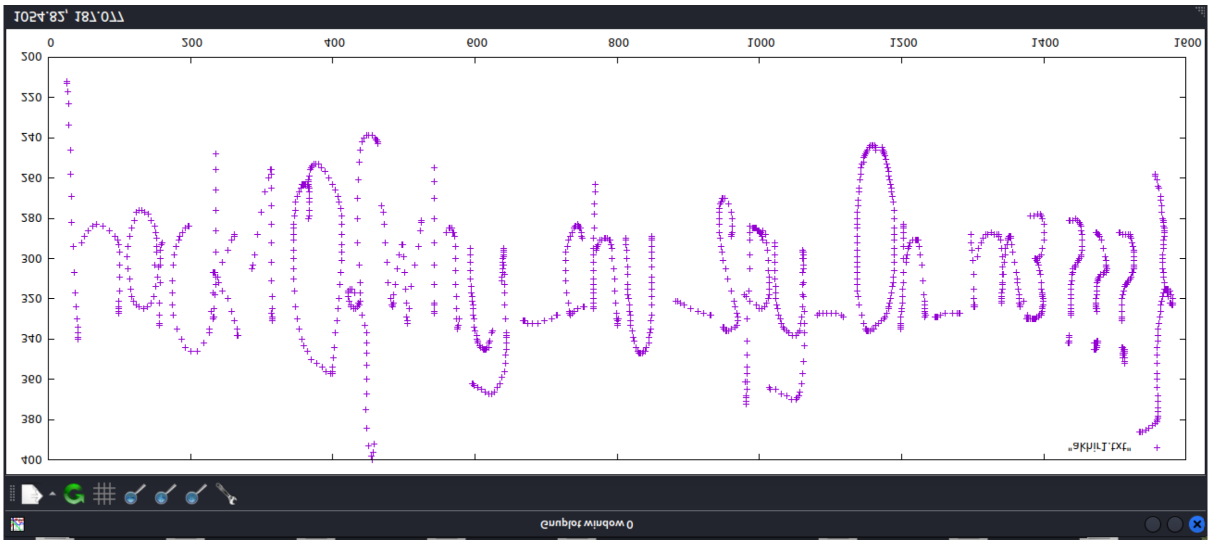
```
Timestamps,["[31,246]"]
Timestamps,["[31,258]"]
Timestamps,["[33,269]"]
Timestamps,["[33,282]"]
Timestamps,["[35,294]"]
Timestamps,["[36,307]"]
Timestamps,["[38,317]"]
Timestamps,["[39,324]"]
Timestamps,["[40,330]"]
Timestamps,["[41,334]"]
Timestamps,["[42,337]"]
Timestamps,["[42,339]"]
Timestamps,["[42,340]"]
Timestamps,["[42,340]"]
Timestamps,["[41,337]"]
Timestamps,["[41,337]"]
Timestamps,["[47,292]"]
```

We got it, but it still cannot plot the graph, so using sed command, we remove the special characters and any other words that not required.



```
33 282
35 294
36 307
38 317
39 324
40 330
41 334
42 337
42 339
42 340
42 340
41 337
41 337
47 292
```

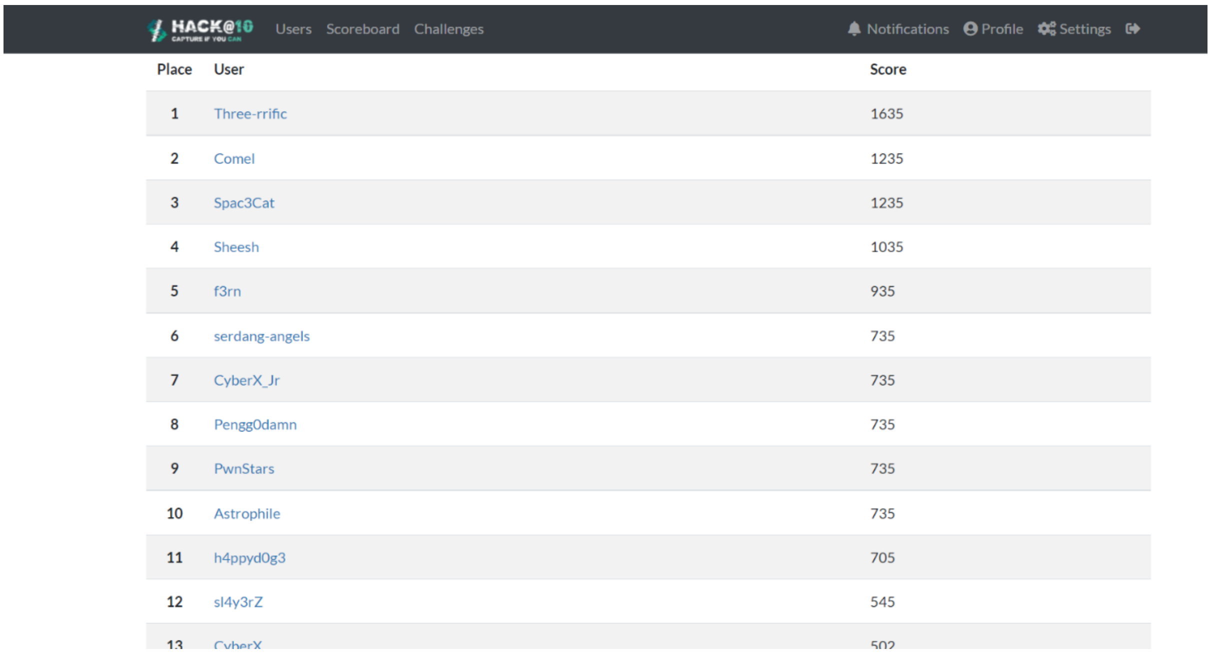
Perfect !! So, we got the coordinate already, so let’s plot the coordinate to build the graph. To make it easy, we use Gnuplot in Linux to automatically plot the graph, and finally, we got our precious flag.



Flag: hack10{why_chu_spy_0n_m3???}

Conclusion

So, that all the flag that our team got, we are new in this capture the flag field and need to learn more about hacking and cyber security to get more experience. This capture the flag event is so interesting and fun because we can learn many new things. We want to thank Uniten because give us these opportunities and knowledge from two days' workshop. Many new things that we got to learn from the workshop and increase our knowledge. We hope that this CTF will be back next year with more fun challenges for beginner like us. We are looking forward to joining more capture the flag event from Uniten.



Place	User	Score
1	Three-rrific	1635
2	Comel	1235
3	Spac3Cat	1235
4	Sheesh	1035
5	f3rn	935
6	serdang-angels	735
7	CyberX_Jr	735
8	Pengg0damn	735
9	PwnStars	735
10	Astrophile	735
11	h4ppyd0g3	705
12	sl4y3rZ	545
13	CyberX	502

We don't stand a chance with the other great and strong team ;)

Thank You. ^_^

CTF Grade: ★★★★★