TSN3251
CTF
Assignment
Writeup

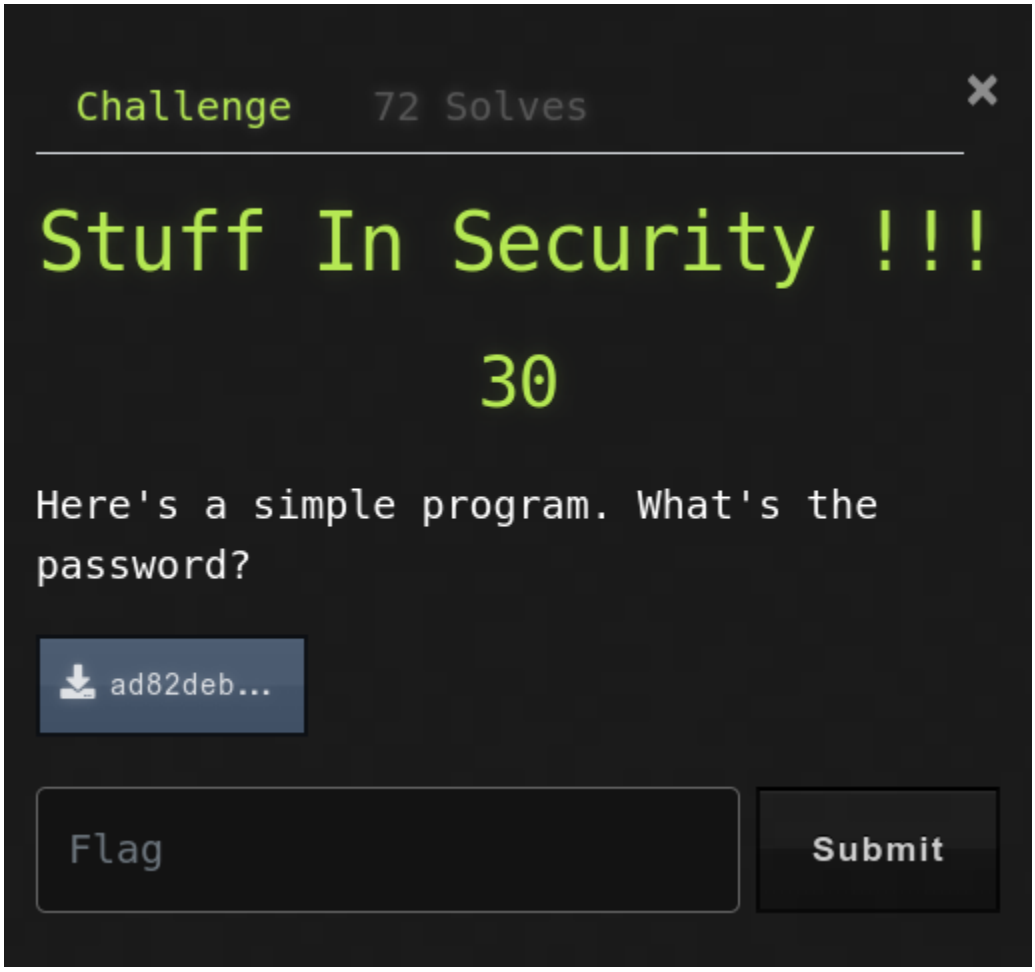
Group Name: Sun_Microsystems

Members:

Student ID	Name	Role
1161100699	Muhamad Afif Bin Kamarul Imran	Leader
1191302540	Tan Shupei	Member
1191302473	Khairul Hanie Hazierah Binti Mohd Azmi	Member
1191302166	Cheok Yi Xuan	Member

Buffer Overflows

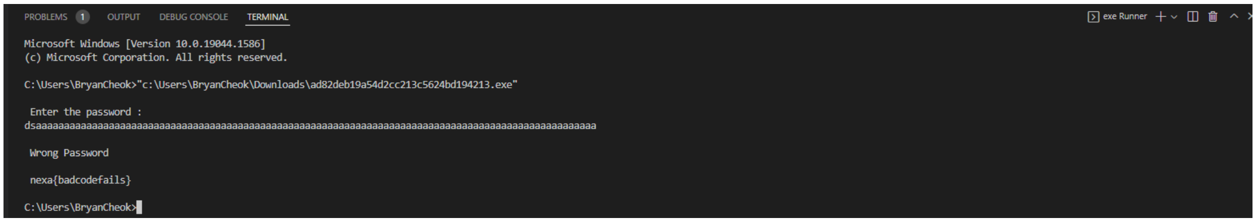
Stuff In Security !!!



Involved members: Cheok Yi Xuan

Thought process/methodologies:

- Debug the .exe file with Visual Studio Code
- Input a random string when prompted to input a password
- No result
- Input a long string when prompted to input a password
- Obtain the flag



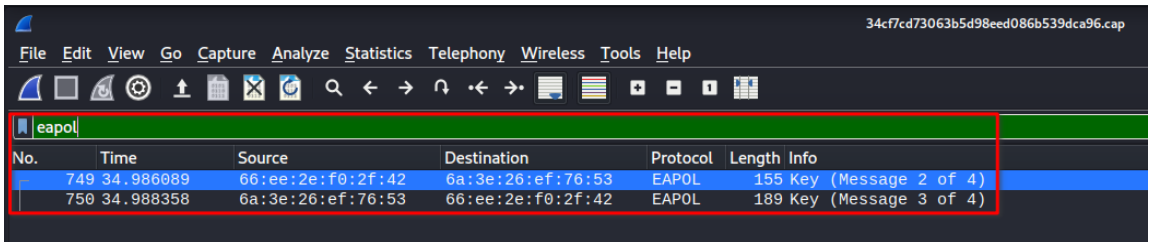
- Flag Revealed: ***nexa{badcodefails}***

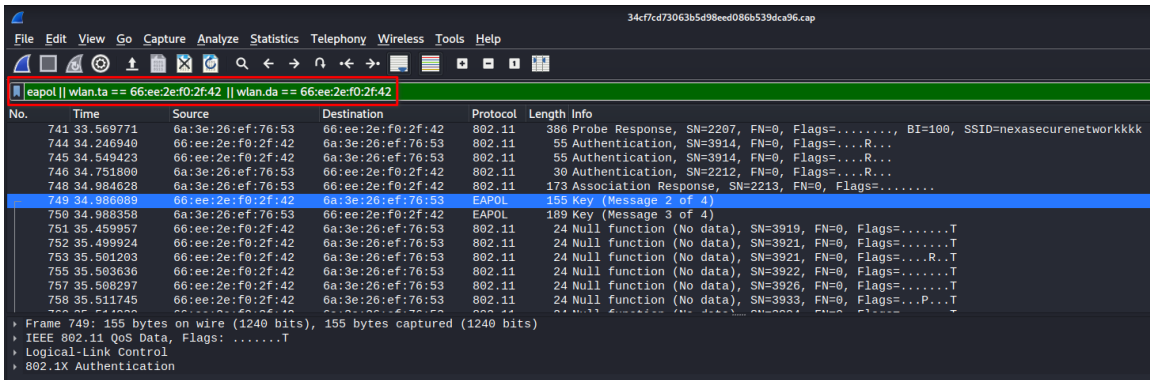
Cracking

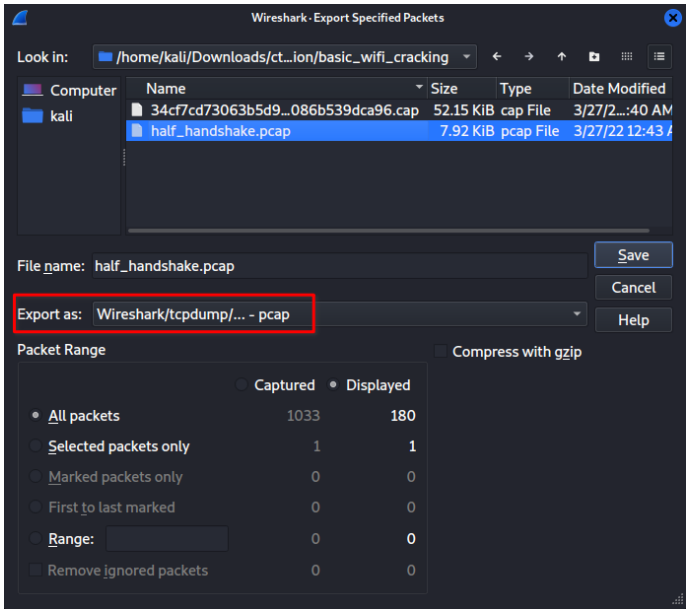
Basic WiFi Cracking

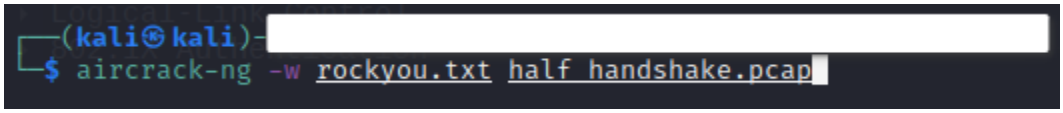
Members involved: Afif

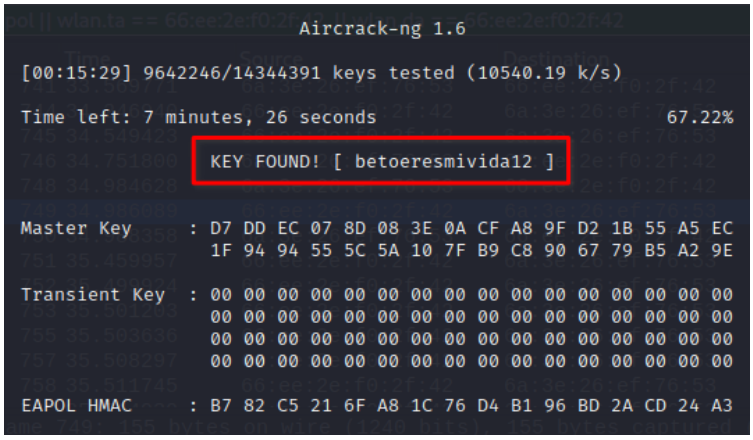
Thought process/methodologies:

- Import packet file on *WireShark*
- Identify common Wi-Fi attack: *half-handshake* as the 4-way handshake is not complete (must be 4 out of 4)
- Search *eapol* flag to list its packets
- 

- Extract source address
- Append *wlan* source and destination addresses through piping with *eapol*
- 

- Export results as a new packet file
- 

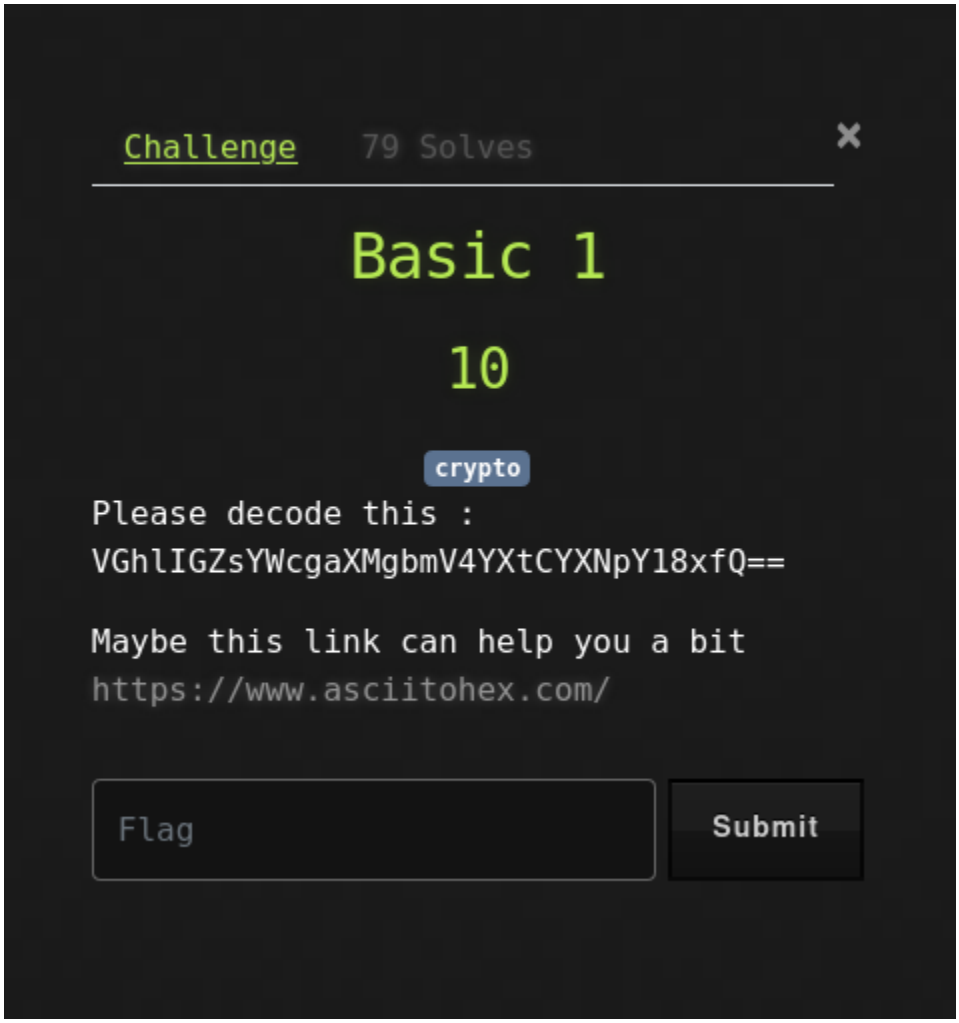
- Execute *aircrack-ng* by comparing the packet file with a *rockyou* breach wordlist
- 

- Wi-Fi password found
- 

- Flag revealed: ***nexa{betoeresmivida12}***

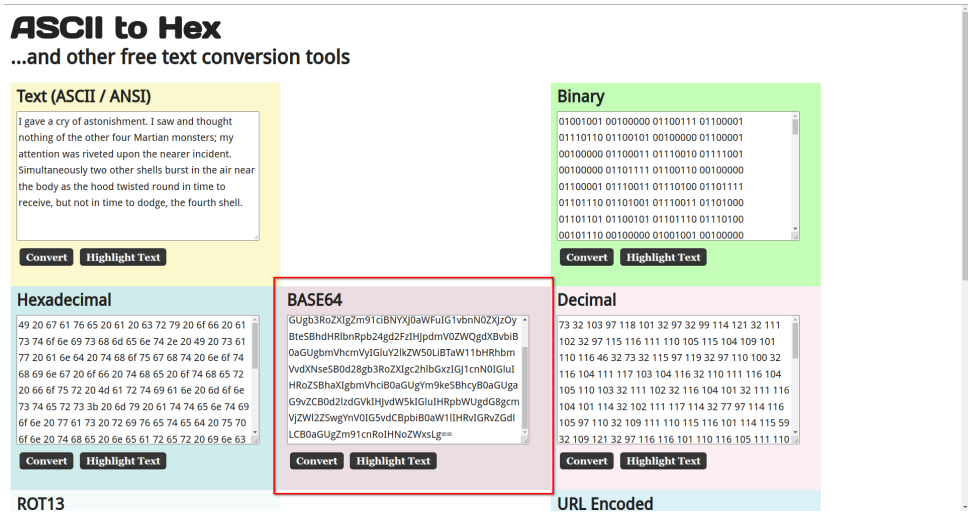
Cryptography

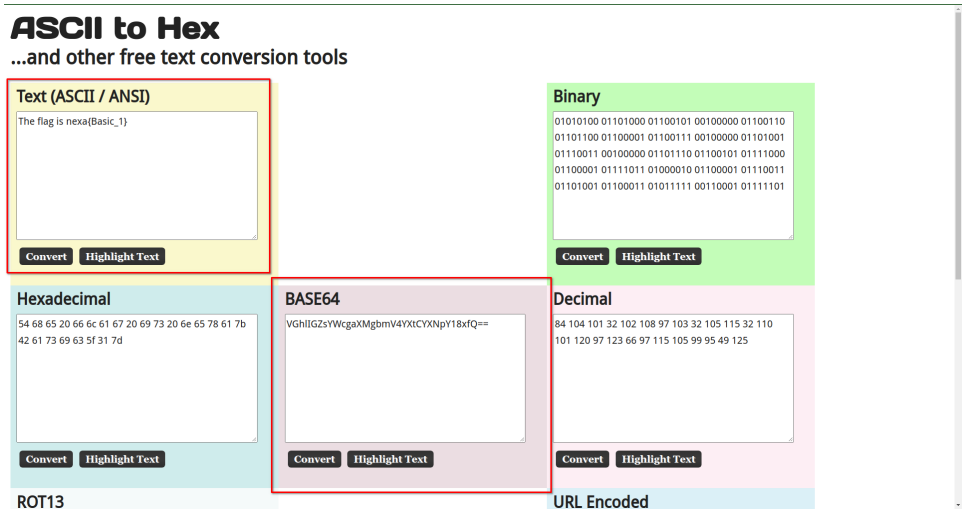
Basic 1



Members involved: Afif

Thought process/methodologies:

- Identify which string it belongs to with different conversion methods
- String given is identified as BASE64
- Input the string into BASE64 converter
- 

- 

- Flag revealed after conversion: ***nexa{Basic_1}***

Basic 2

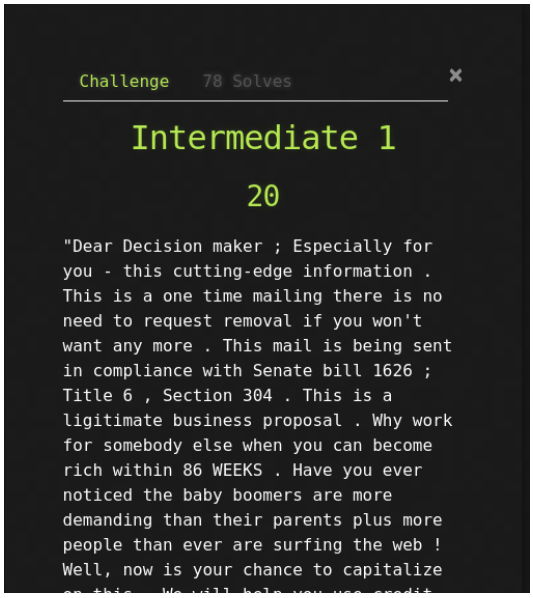
Members involved: Afif

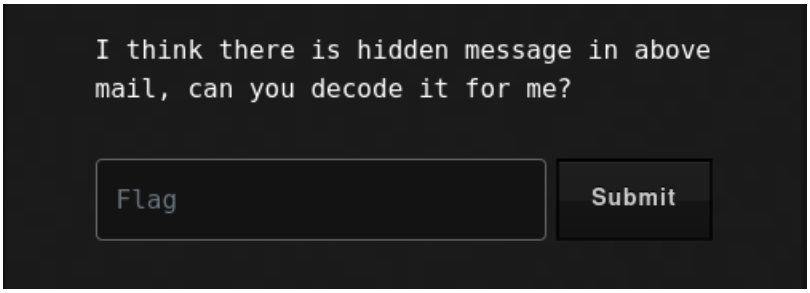
Thought process/methodologies:

- Given input string is identified as morse code
- Convert the string into plaintext

-
- Flag revealed: ***nexa{MORSE_CODE_IS_NICE}***

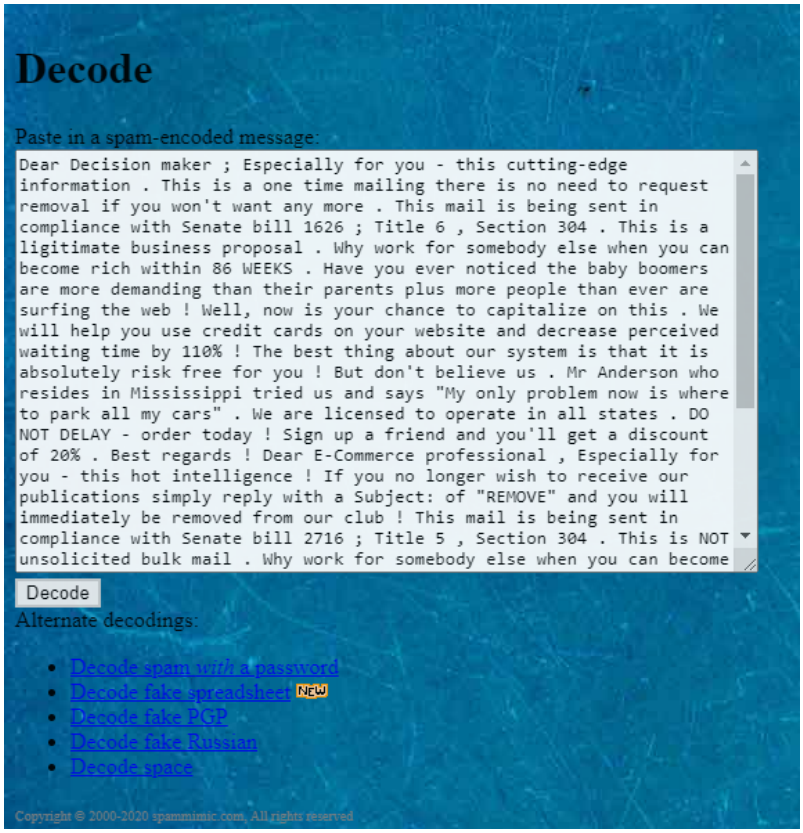
Intermediate 1

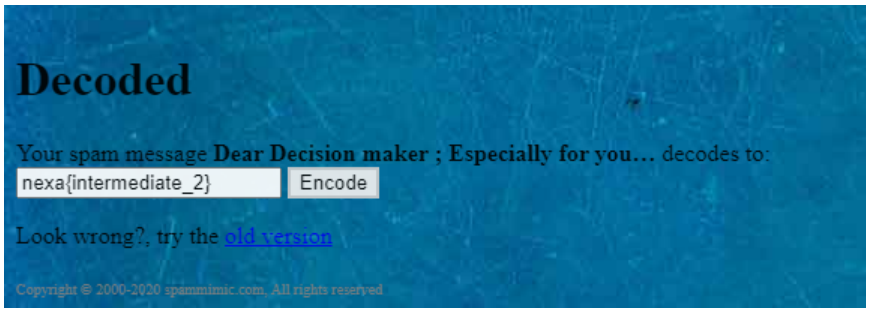




Members involved: Cheok Yi Xuan

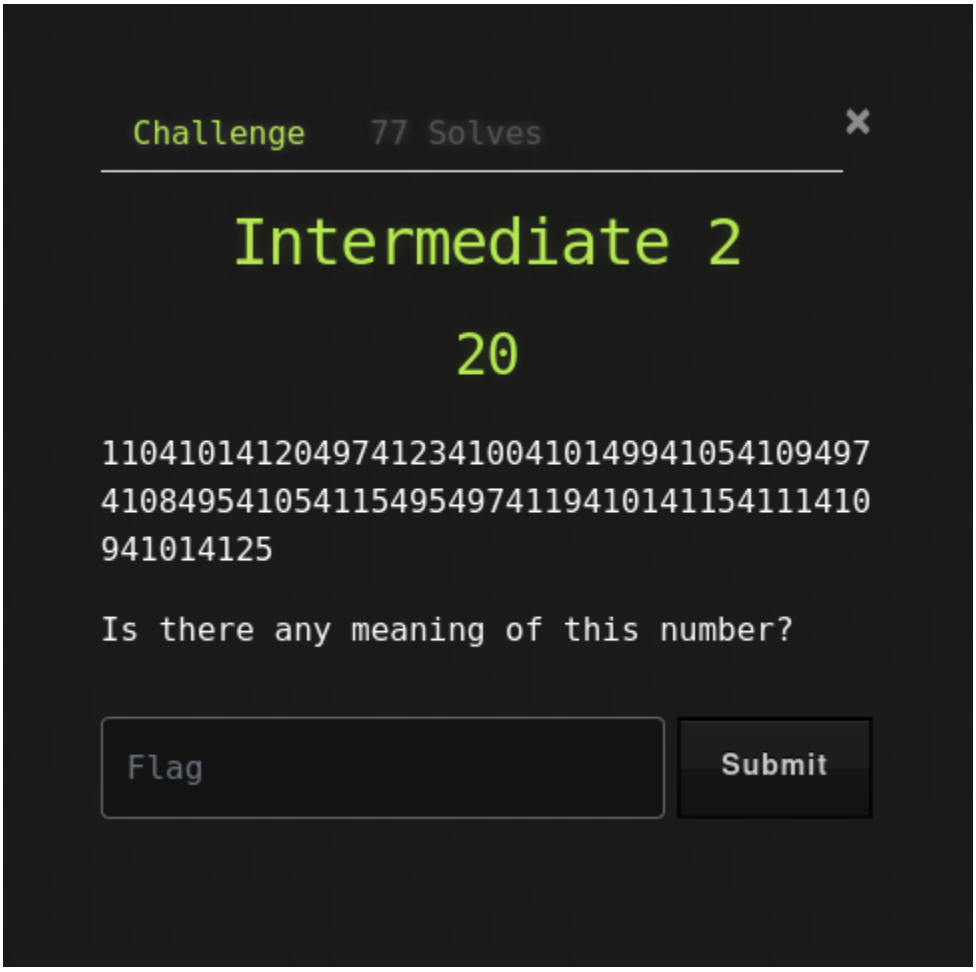
Thought process/methodologies:

- Google the text provided
- Decode the text at http://www.spammimic.com/decode.shtml
- 

- 

- Flag revealed: ***nexa{intermediate_2}***

Intermediate 2



Members involved: Afif

Thought process/methodologies:

- Converting decimal to plaintext
- String given is identified as a decimal or base-10
- Convert the given input into plaintext

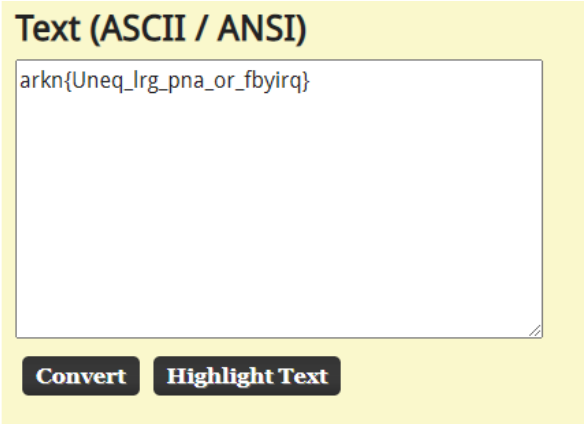
- Flag revealed: ***nexa{decimal_is_awesome}***

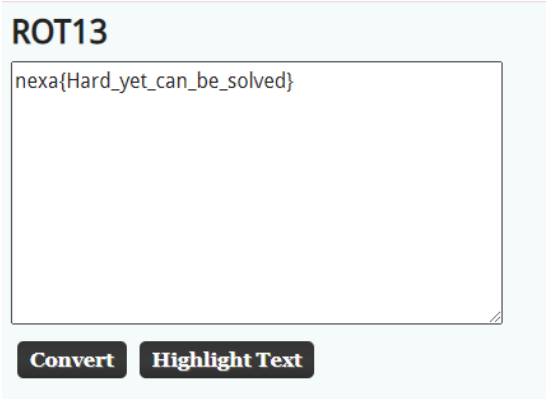
Intermediate 3

Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

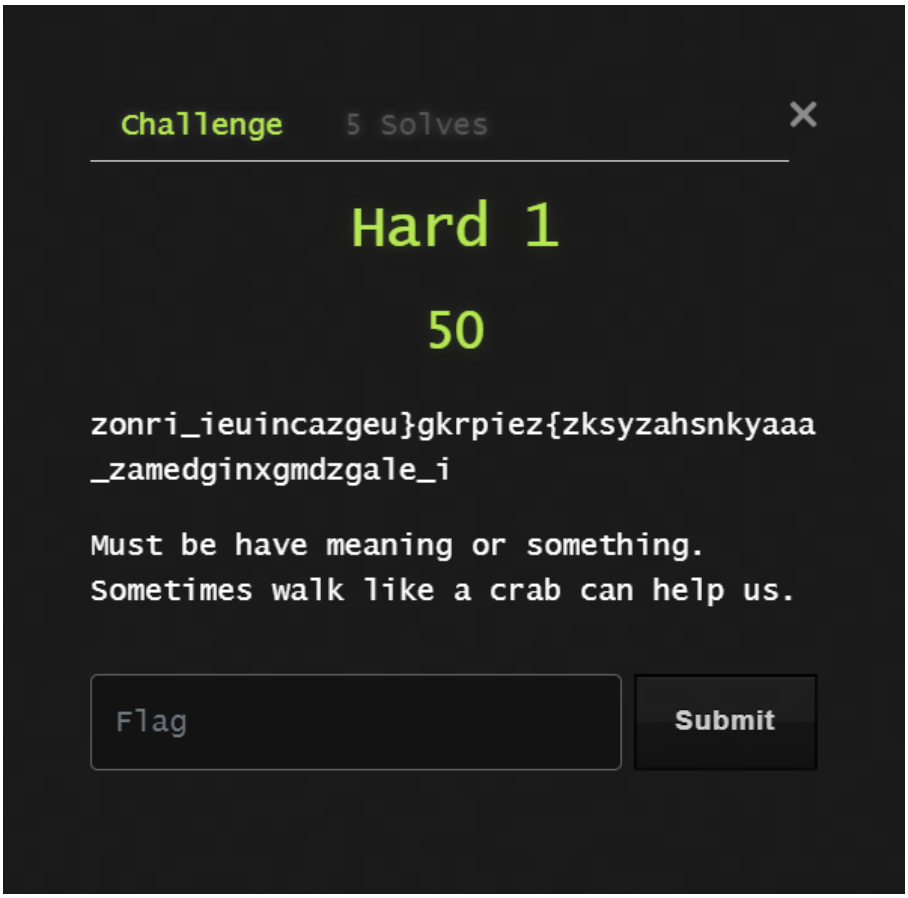
- The question states that base 64 is not enough.
- Firstly, the message given is decoded through the Base64 converter. It will decode the message, however, the message is not decoded thoroughly yet.





- Using ROT13, the flag is revealed to be ***nexa{Hard_yet_can_be_solved}***

Hard 1

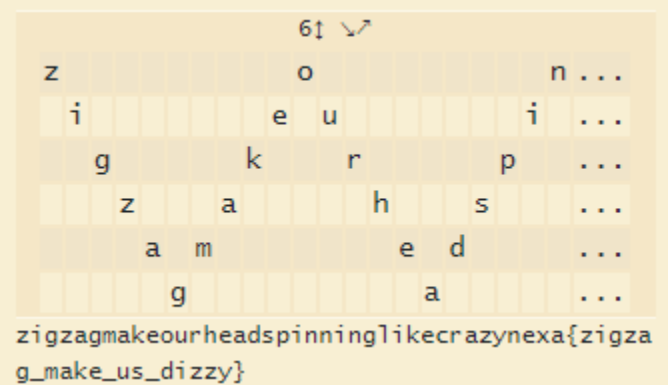


Members involved: Tan Shupei, Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

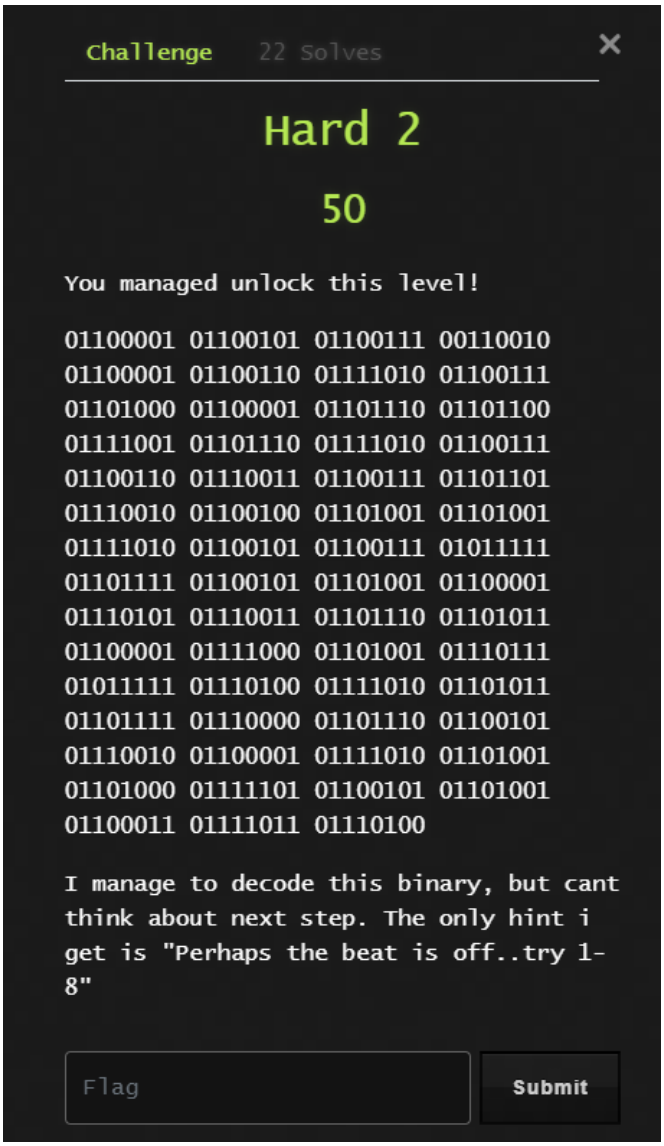
- Tan Shupei:
    - As it is mentioned in the question that “walking like a crab can help us”, two-square cipher (horizontal two-square) is used.
    - “Must be have meaning or something. “ and “Sometimes walk like a crab can help us. “ were used as the secret keywords in the two-square cipher decryption attempt.
    - However, the decryption did not take place successfully.

- Khairul Hanie Hazierah Binti Mohd Azmi:
    - The ciphertext is analyzed using the hint given, which is “Sometimes walk like a crab can help us”
    - The hint indicates that the ciphertext letters are arranged diagonally/zigzag, which hints at the rail fence cipher.



    - Using k = 6, the flag is revealed to be **nexa{zigzag_make_us_dizzy}**

Hard 2



Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

- The binary code is analyzed and converted into text, which is shown to be aeg2afzghanlynzgfsgmrdiizeg_oeiausnkaxiw_tzkopnerazih}eic{t

```
fset}zigzagmakeourheadspinninglikecrazy2nexa
{zigzag_with_of
```

- Using k = 6, the plaintext is starting to make sense, however, the arrangement of the letters is off.

```
6↕ ↘↗ (+6)
a           e...
 z g      h ...
  g  m   r  ...
 i   a  u   ...
z     k o    ...
       e     ...
zigzagmakeourheadspinninglikecrazy2nexa{zigz
ag_with_offset}
```

- Using the hint given, “Perhaps the beat is off..try 1-8”, with the combination of offset = 6 and k = 6, the flag is revealed to be **nexa{zigzag_with_offset}**

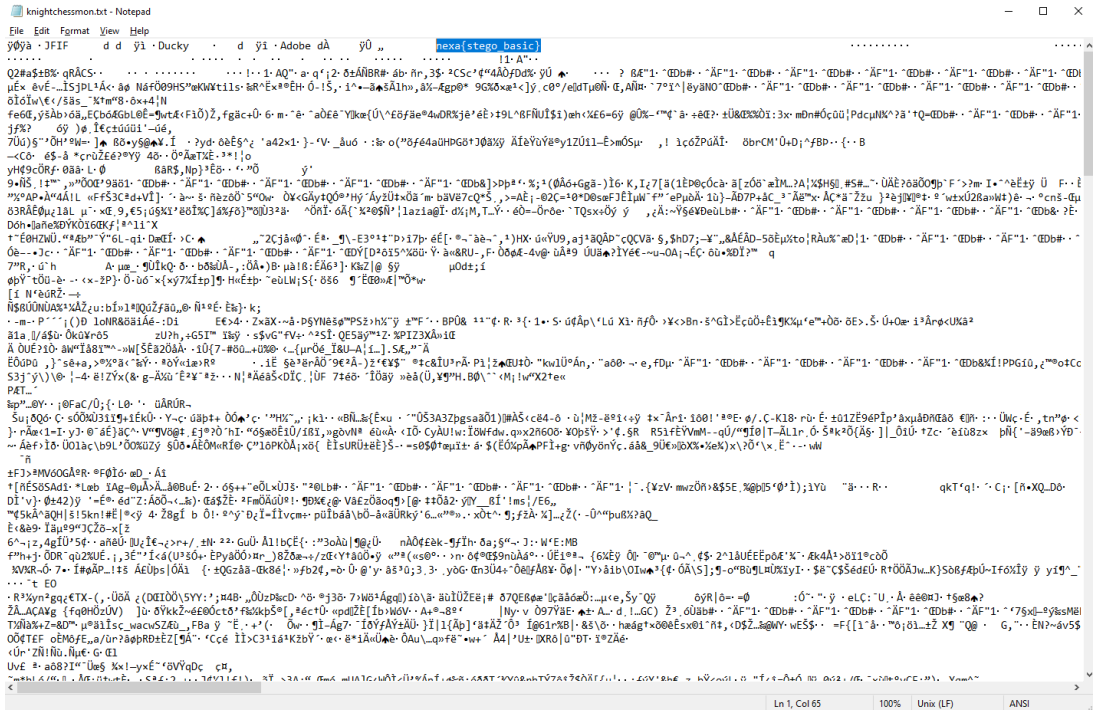
Forensic

Basic 3

Members involved: Cheok Yi Xuan

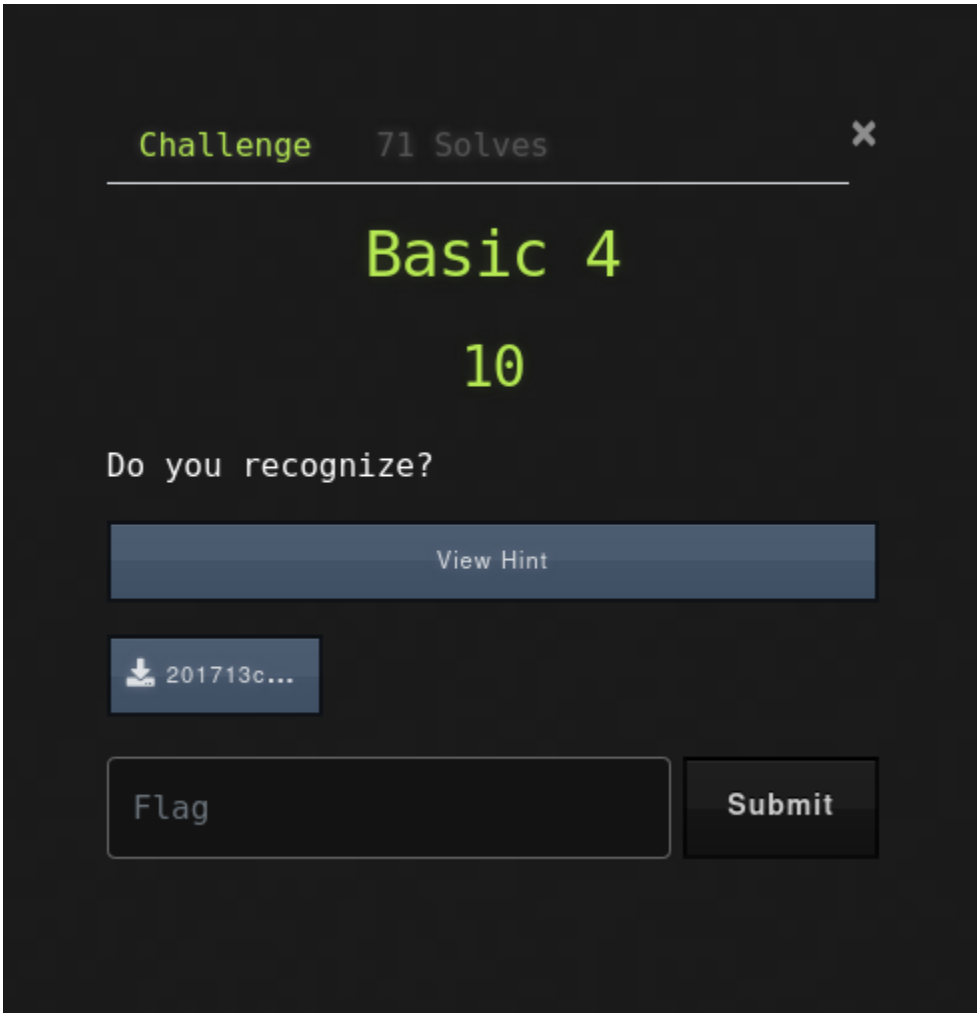
Thought process/methodologies:

- Open the file as text file



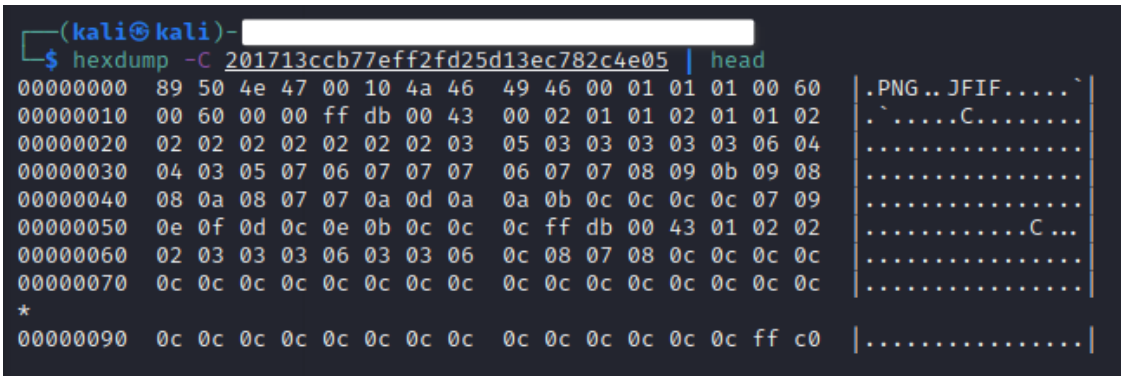
- Flag revealed: ***nexa{stego_basic}***

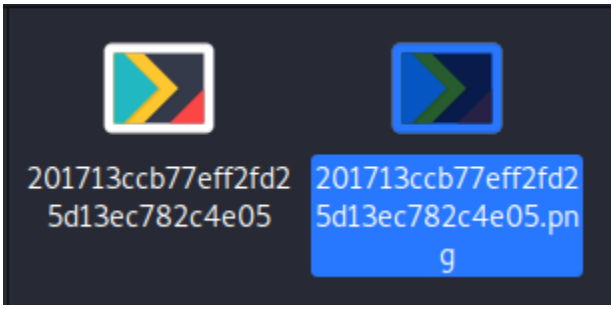
Basic 4

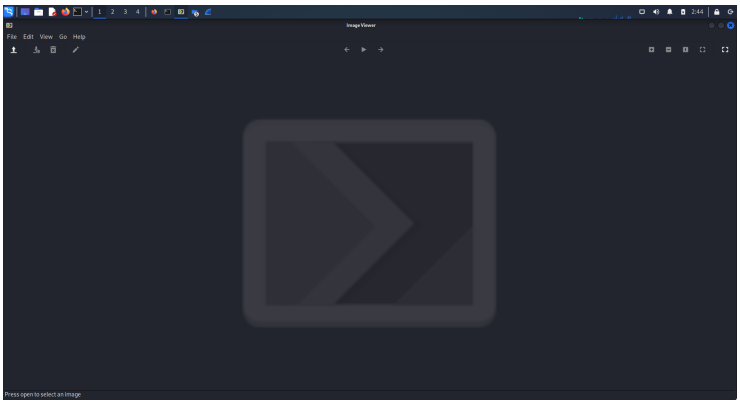


Members involved: Afif, Tan Shupei

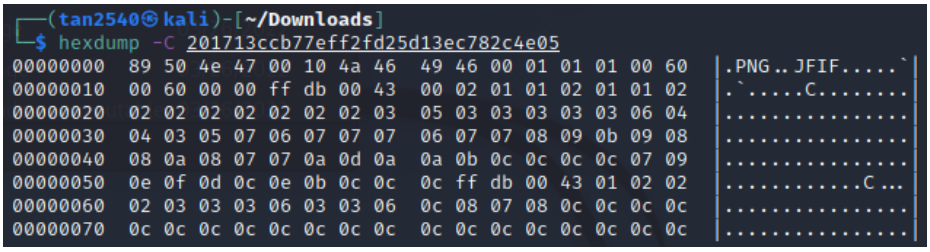
Thought process/methodologies:

- Afif
  - Identify file type by dumping its hex content in hex+ASCII format only from the beginning of the file
  - 

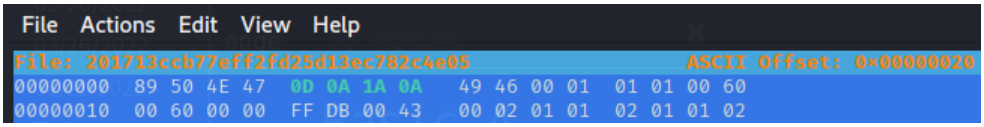
  - File type appears to be in *PNG*
  - Rename the file by adding the *PNG* extension behind it
  - 

  - Upon opening the file, no image is shown and appears to be corrupted
  - 
  - Unable to proceed from here…

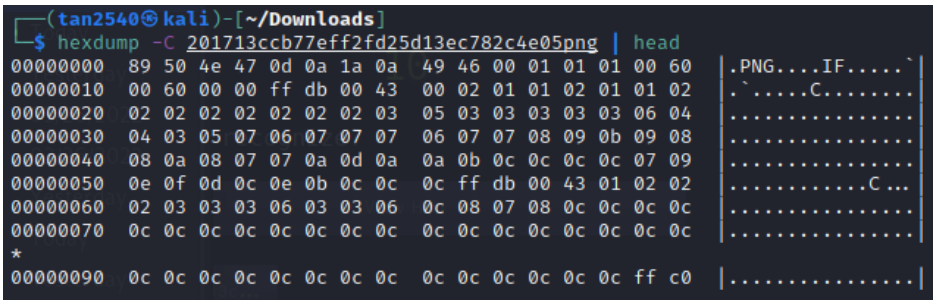
- Tan Shupei
    - Hexdump was used to check the hex content of the file first to see if there is any corruption in the file header.



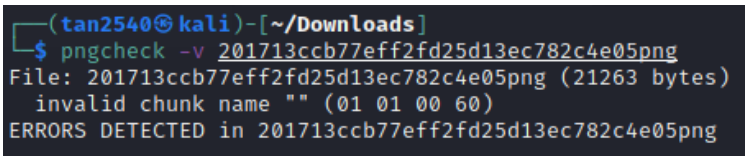
    - By comparing the file signature of a PNG file and the one in the screenshot above, there might be some corruption in the file header as the actual file signature for PNG file should be **89 50 4E 47 0D 0A 1A 0A**, thus hexeditor was used to modify the header.



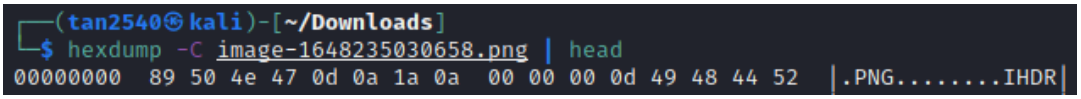
    - However, the modification was not successful as no image was shown while attempting to open the modified file.



    After running through a pngcheck, it was shown that the modified file has an invalid chunk name.



    - After comparing the modified file's header with a valid PNG file's header, another idea came in mind whereby the original file might be a JPEG file rather than a PNG file. Therefore another attempt was made, which is to modify the file header into a JPEG file header.

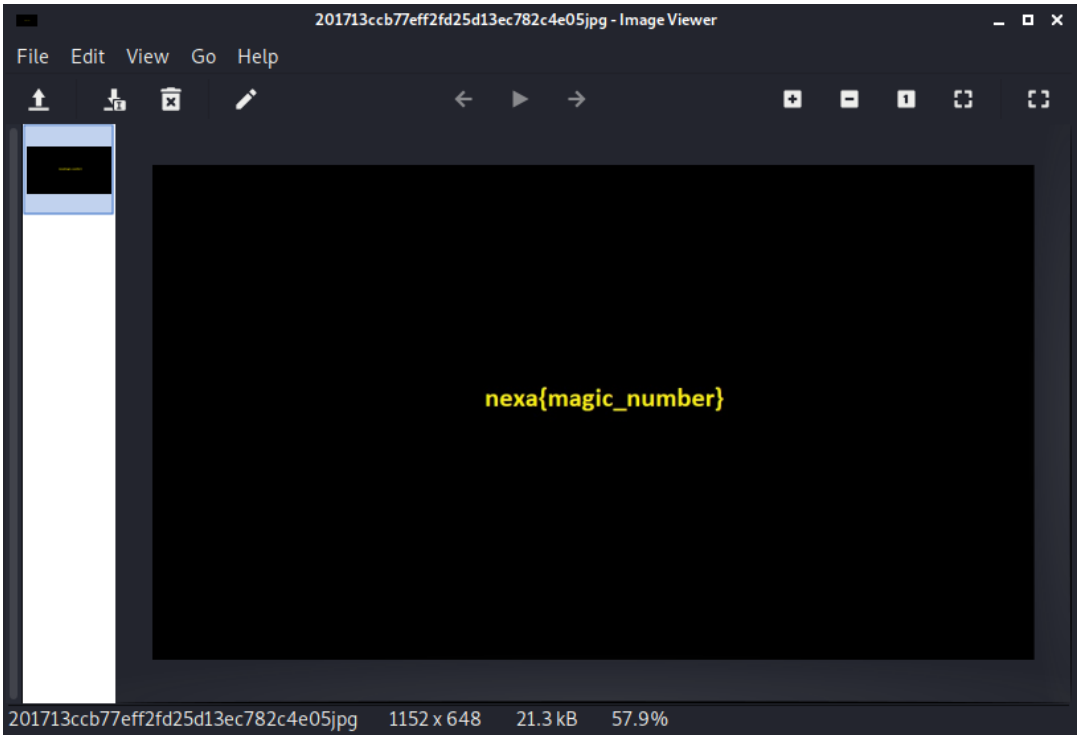


    - As the file signature of a JPEG file is **FF D8 FF E0 00 10 4A 46 49 46 00 01**, it matches more of the original file's hex values. Therefore, hexeditor was

used again to modify the header.

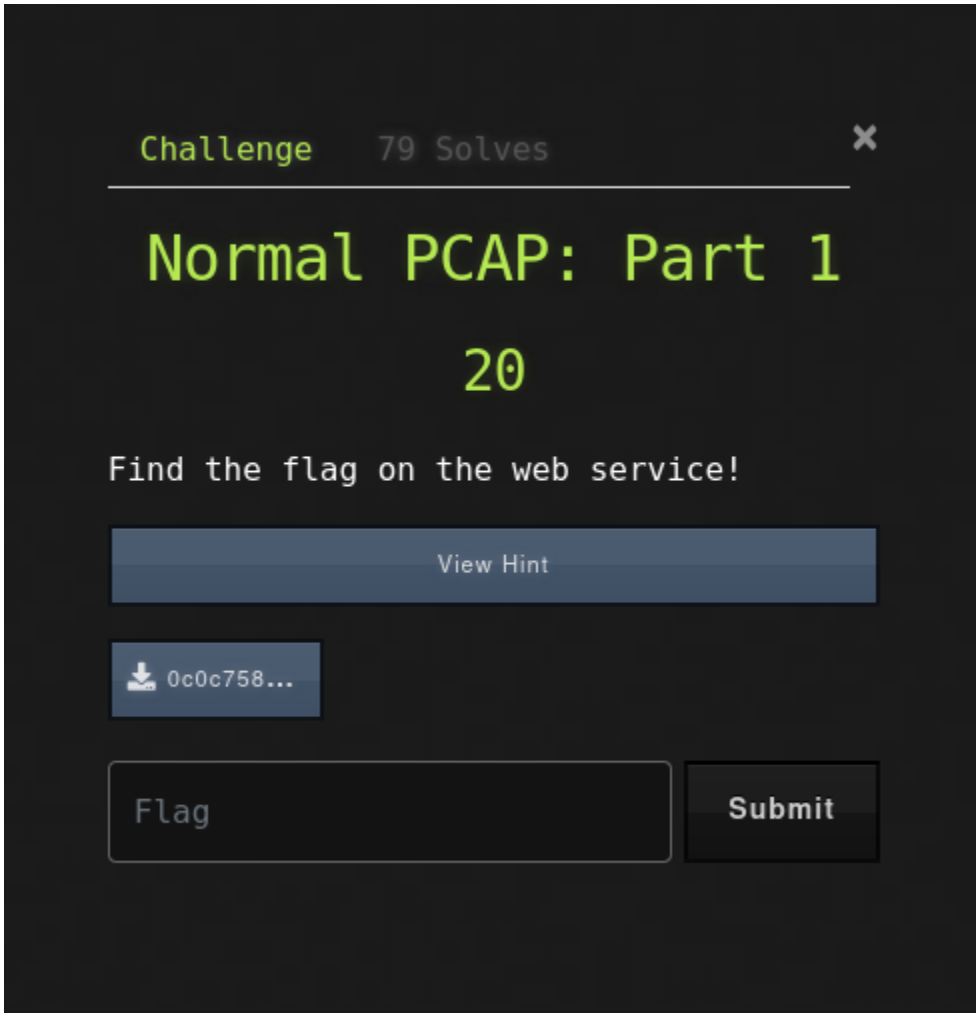
```
┌──(tan2540㉿kali)-[~/Downloads]
└─$ hexdump -C 201713ccb77eff2fd25d13ec782c4e05jpg | head
00000000  ff d8 ff e0 00 10 4a 46  49 46 00 01 01 01 00 60  |......JFIF.....`|
00000010  00 60 00 00 ff db 00 43  00 02 01 01 02 01 01 02  |.`.....C........|
00000020  02 02 02 02 02 02 02 03  05 03 03 03 03 03 06 04  |................|
00000030  04 03 05 07 06 07 07 07  06 07 07 08 09 0b 09 08  |................|
00000040  08 0a 08 07 07 0a 0d 0a  0a 0b 0c 0c 0c 0c 07 09  |................|
00000050  0e 0f 0d 0c 0e 0b 0c 0c  0c ff db 00 43 01 02 02  |............C...|
00000060  02 03 03 03 06 03 03 06  0c 08 07 08 0c 0c 0c 0c  |................|
00000070  0c 0c 0c 0c 0c 0c 0c 0c  0c 0c 0c 0c 0c 0c 0c 0c  |................|
*
00000090  0c 0c 0c 0c 0c 0c 0c 0c  0c 0c 0c 0c 0c 0c ff c0  |................|
```

- This time, the attempt was successful and the image was returned in a JPEG file as shown below.



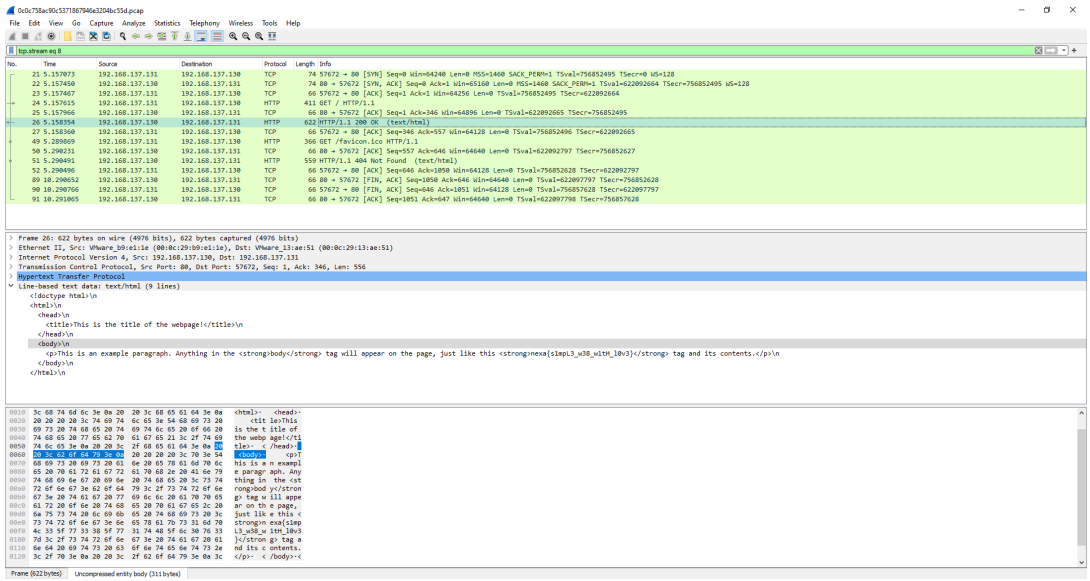
- Flag revealed: ***nexa{magic_number}***
- Unfortunately, the flag was found after the first CTF challenge while going through the questions again and reattempting them therefore we did not manage to get the marks for this question during CTF Challenge 1.

Normal PCAP: Part 1



Members involved: Cheok Yi Xuan

Thought process/methodologies:

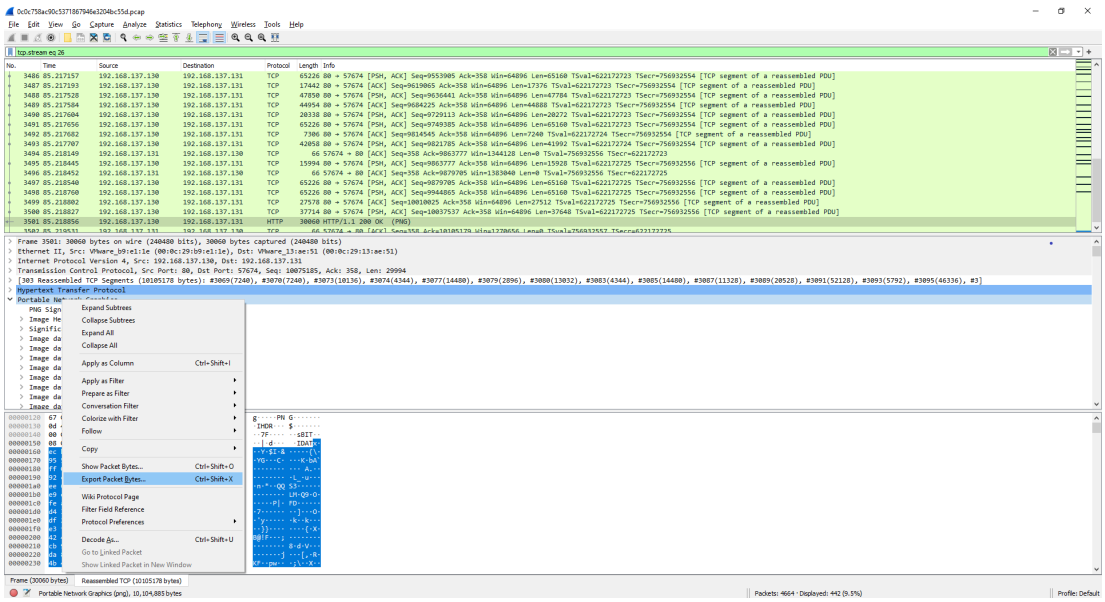
- View the packet with Wireshark
- Find the packet with longer length
- 

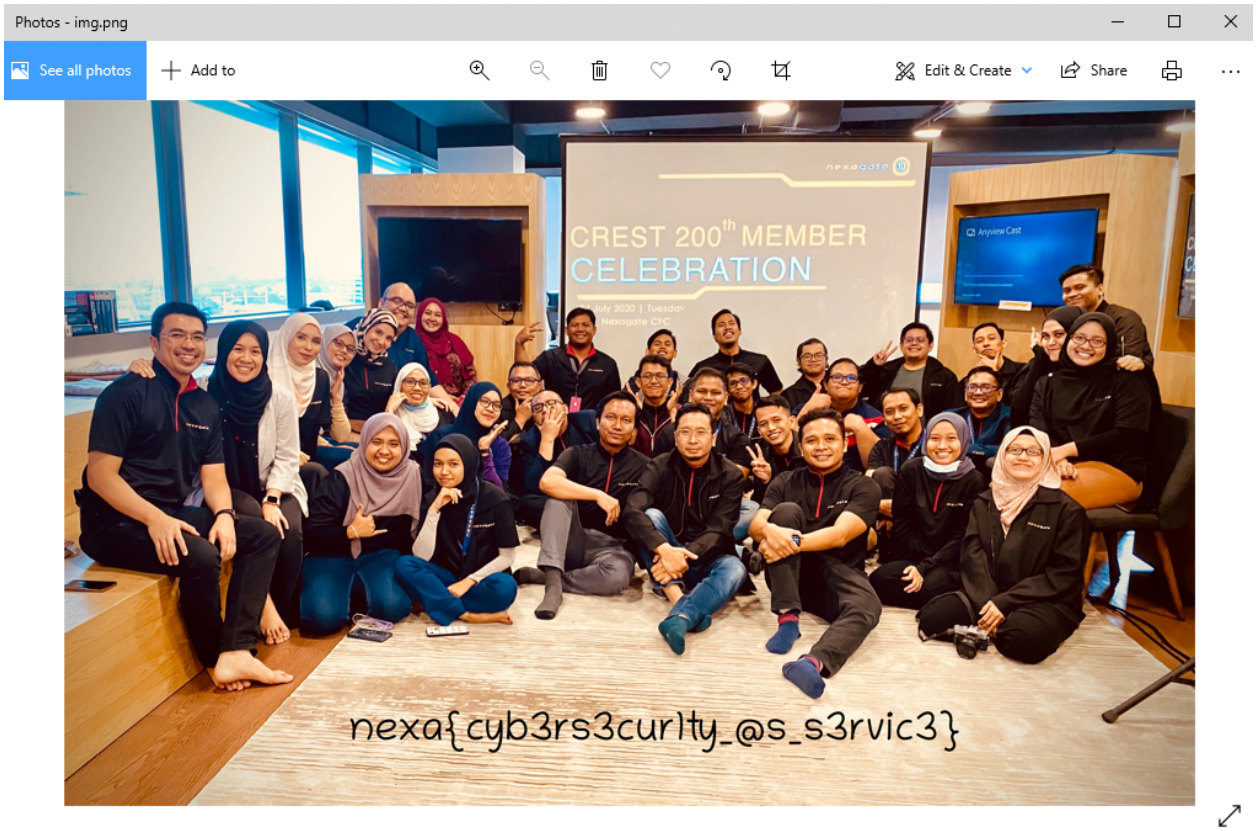
- Flag: ***nexa{s1mpL3_w38_w1tH_l0v3}***

Normal PCAP: Part 2

Members involved: Cheok Yi Xuan

Thought process/methodologies:

- View the file with Wireshark
- Find the PNG attachment
- Build the packet into a file
- 

- 

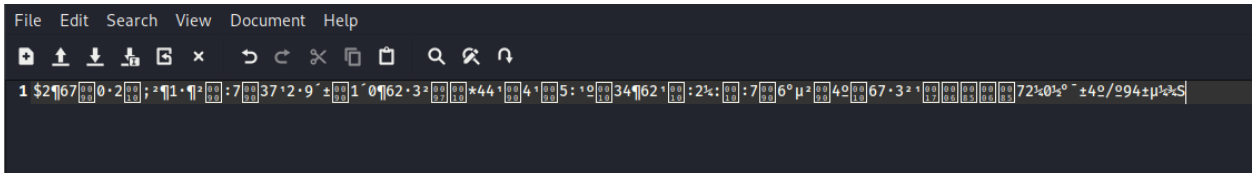
- Flag: ***nexa{cyb3rs3cur1ty_@s_s3rvic3}***

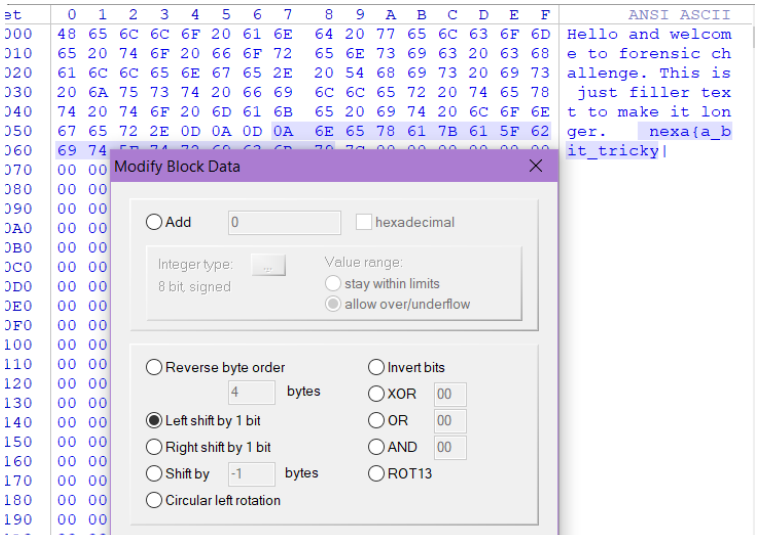
Shift Your Focus

Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

- The question given is about finding the flag in the text document.
- The given text file is opened and analyzed.





- By using a hex editor and shifting the bits to the left once, the flag is revealed to be ***nexa{a_bit_tricky}***

Normal PCAP: Part 3

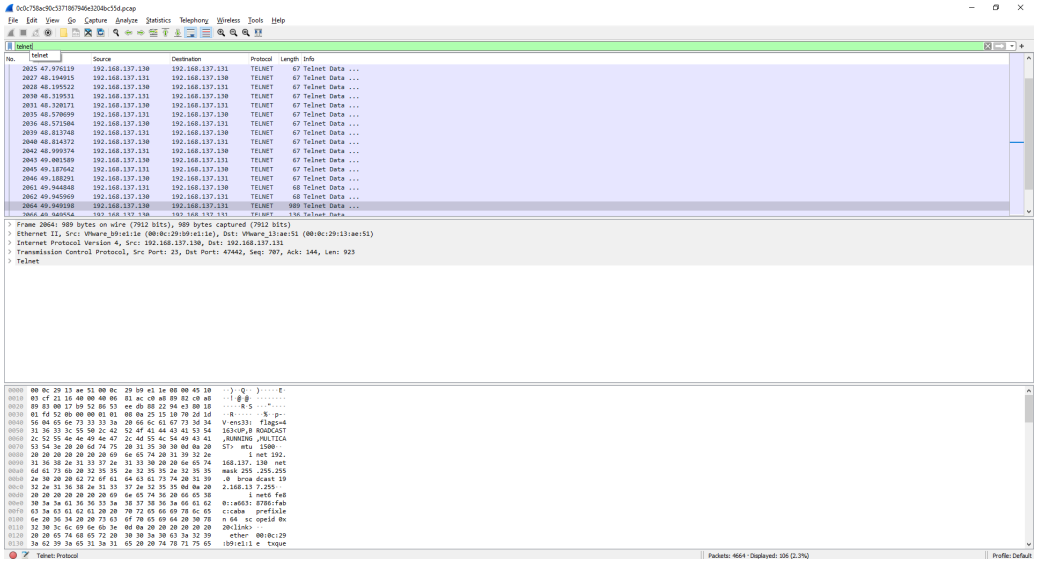
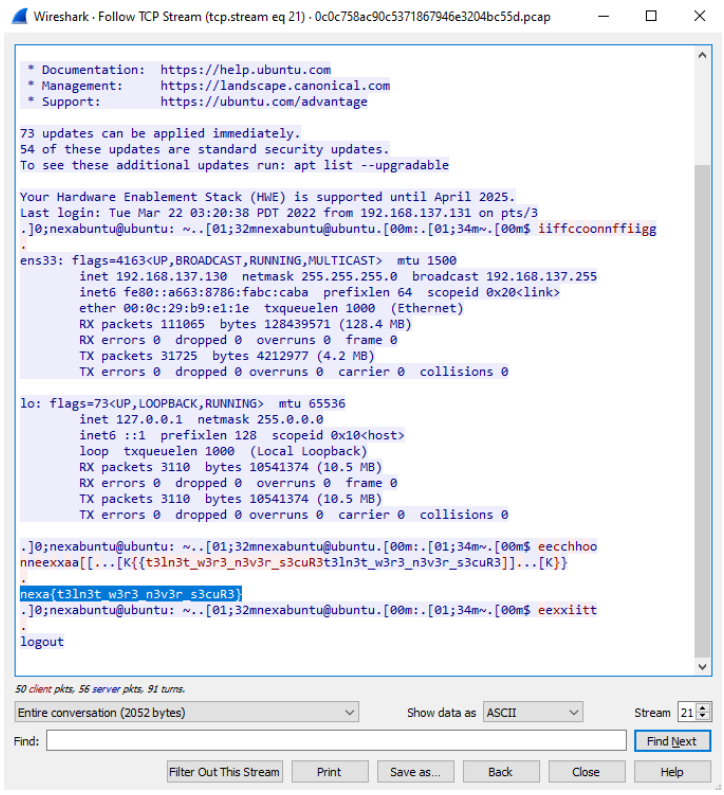
Members involved:

Thought process/methodologies:

Normal PCAP: Part 4

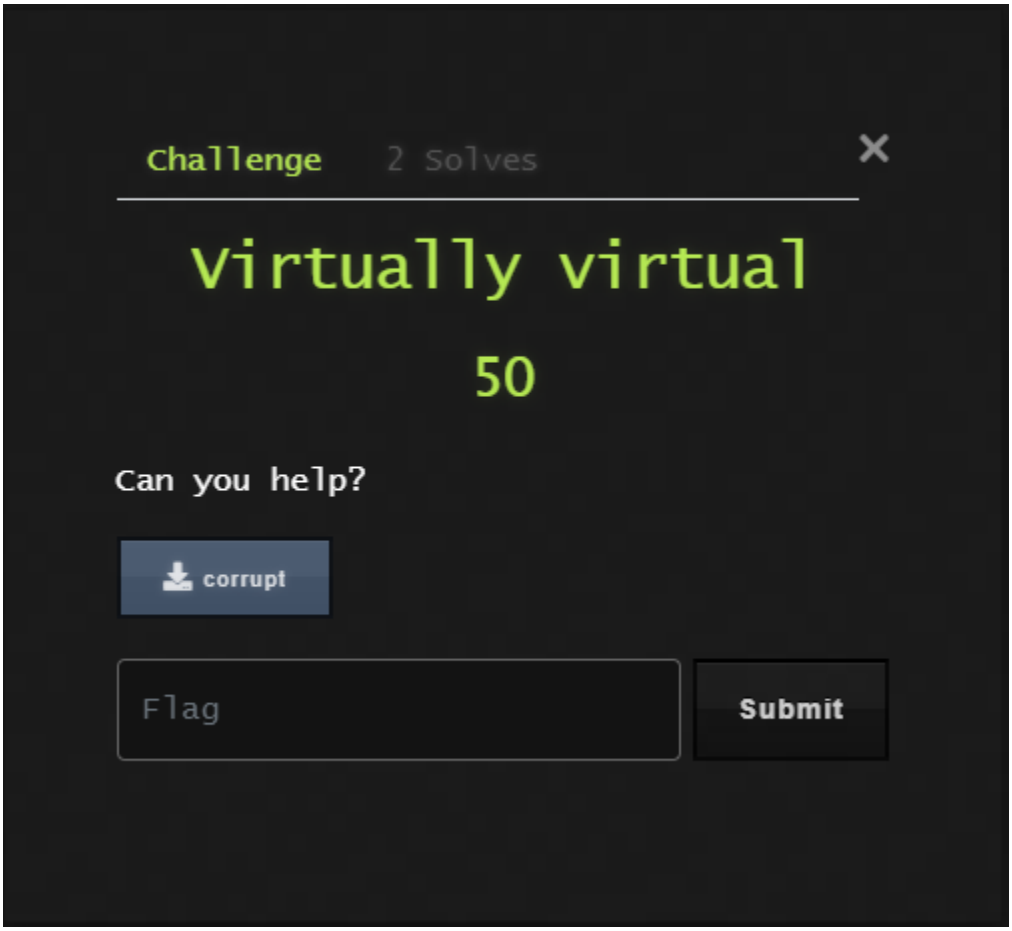
Members involved: Cheok Yi Xuan

Thought process/methodologies:

- Open the file with Wireshark
- Sort by telnet
- 
- 

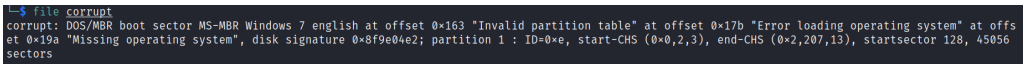
- Flag: ***nexa{t3ln3t_w3r3_n3v3r_s3cuR3}***

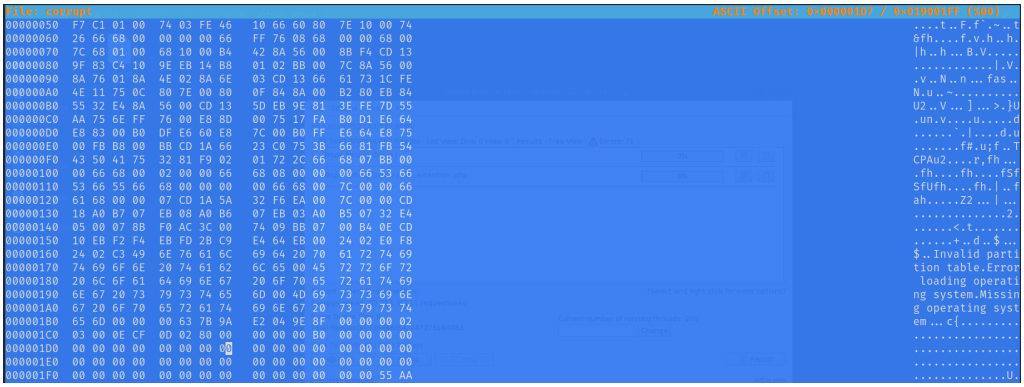
Virtually Virtual



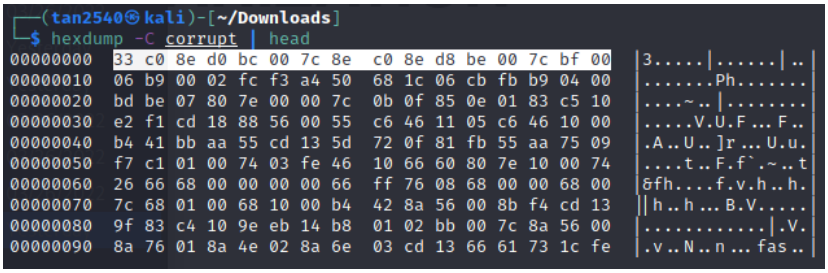
Members involved: Afif, Tan Shupei, Cheok Yi Xuan

Thought process/methodologies:

- Afif:
  - Download a given file
  - Identify the file type with *file* on a terminal
  - 

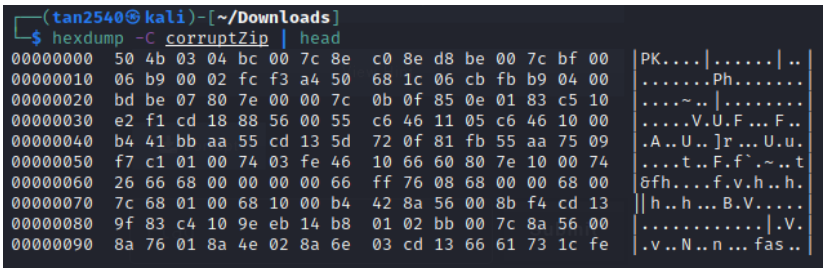
  - File seems to be an *MBR* partition
  - Perform *MBR* partition analysis by using *hexeditor* included in Kali Linux
  - 
#	Flag	Type	Starting LBA address	Size
1	0x00	0x0e	0x80	0xB000
2	0x00	0x00	0x00	0x00
3	0x00	0x00	0x00	0x00
4	0x00	0x00	0x00	0x00

    partition #1:
    - 0x00: non bootable
    - 0x0e: FAT-16 LBA
    - 0x80: 128
    - 0xB000: 45,056
      - 45,056 * 512 = 366,957,580 bytes
      - bytes to gigabytes: roughly 0.337GB
  - Partition number one is a FAT-16 type, seems to contain roughly 337MB of data
  - Unable to proceed from here…

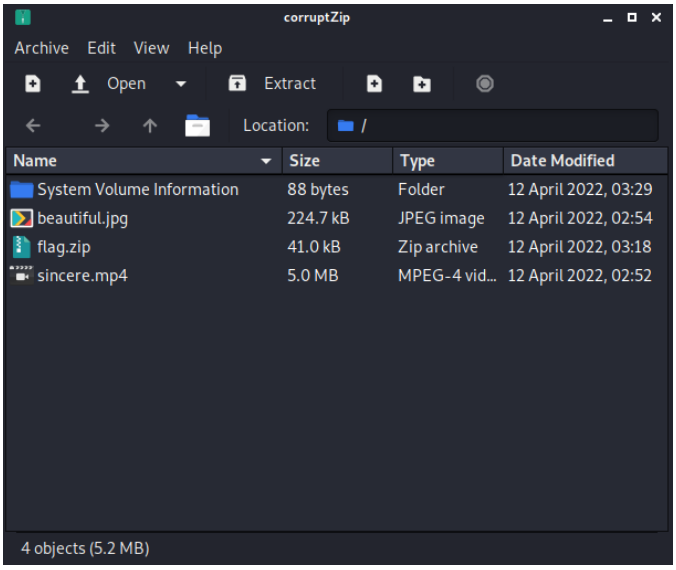
- Tan Shupei
  - As the file type is unknown, hexdump was used to check the file header.
  - 

    However, after going through the list of file signatures, nothing similar was found.

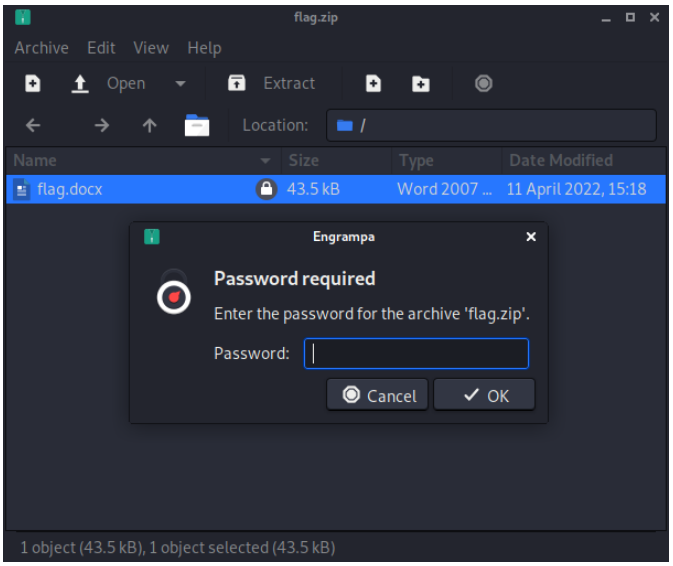
- From the hint given, the file could be a .zip file therefore an attempt was made to modify the file header to a .zip file's signature as shown below.



- Fortunately the approach was correct and 4 files were returned.



- However, the password to extract this file was nowhere to be found and could not be cracked successfully even using John The Ripper or by observing the strings of the files. Words like "beautiful", "sincere", "flag" were used as well but none of them were the password to extract the file.



- Therefore the extraction was unable to take place and the flag could not be revealed any further.
- An attempt to check the file headers of the two files named beautiful.jpg and sincere.mp4 was further made and something strange was spotted.

```
┌──(tan2540㉿kali)-[~/Downloads/corruptUnzip]
└─$ hexdump -C beautiful.jpg | head
00000000  89 50 4e 47 0d 0a 1a 0a  00 00 00 0d 49 48 44 52  |.PNG........IHDR|
00000010  00 00 03 56 00 00 01 b5  08 02 00 00 00 75 71 f2  |...V.........uq.|
00000020  39 00 00 00 01 73 52 47  42 00 ae ce 1c e9 00 00  |9....sRGB.......|
00000030  00 04 67 41 4d 41 00 00  b1 8f 0b fc 61 05 00 00  |..gAMA......a...|
00000040  00 09 70 48 59 73 00 00  0e c3 00 00 0e c3 01 c7  |..pHYs..........|
00000050  6f a8 64 00 00 ff a5 49  44 41 54 78 5e ec fd 87  |o.d....IDATx^...|
00000060  76 24 4b 9a df 09 9a 74  1d 0a 2a d5 15 55 d5 dd  |v$K....t..*..U..|
00000070  64 0f b9 bb 67 cf 90 fb  00 9c 57 59 79 96 b3 0f  |d...g.....WYy...|
00000080  b1 dc 87 db 73 76 76 76  48 0e d9 dd 55 ec 12 b7  |....svvvH...U...|
00000090  ee cd 9b 02 2a 84 6b 53  fb 7d e6 01 64 24 12 22  |....*.kS.}..d$."|
```

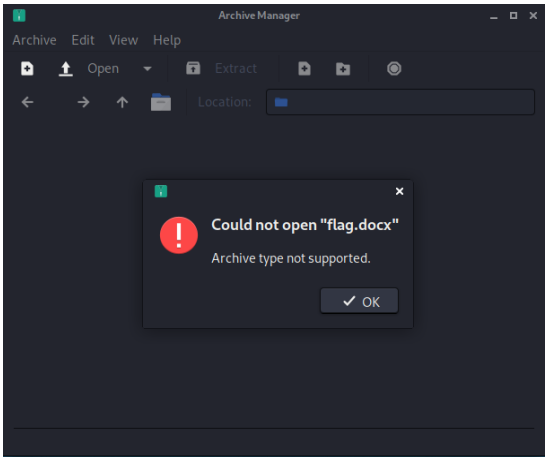
  The original file type was in JPEG however the file header shows the other, which is type PNG. However no differences were spotted while converting the file to a PNG file or modifying the file to a JPEG file.
- This time, John The Ripper was used along with the rockyou.txt wordlist and **lavidaesbella** seems to be the password.

```
┌──(tan2540㉿kali)-[~/Downloads/corruptUnzip]
└─$ zip2john flag.zip > password.txt
ver 2.0 efh 9901 flag.zip/flag.docx PKZIP Encr: cmplen=40831, decmplen=43499, crc=C6B64A22

┌──(tan2540㉿kali)-[~/Downloads/corruptUnzip]
└─$ john --show password.txt
flag.zip/flag.docx:lavidaesbella:flag.docx:flag.zip:flag.zip
```

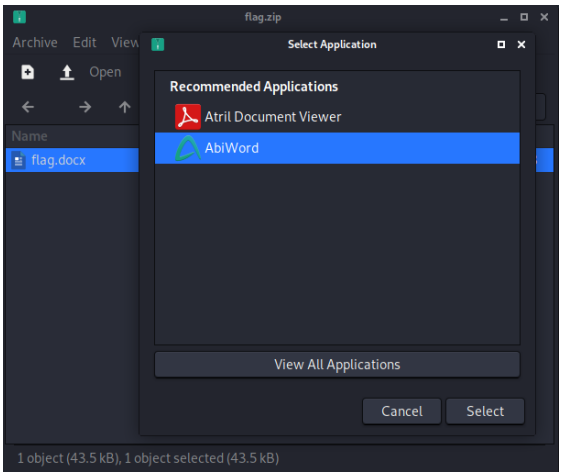
```
┌──(tan2540㉿kali)-[~/Downloads/corruptUnzip]
└─$ john password.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 256/256 AVX2 8x])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
lavidaesbella    (flag.zip/flag.docx)
1g 0:00:00:00 DONE (2022-04-13 23:55) 2.380g/s 19504p/s 19504c/s 19504C/s newzealand..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

- However the extraction still did not take place successfully as the system says that they do not support the .docx file type.

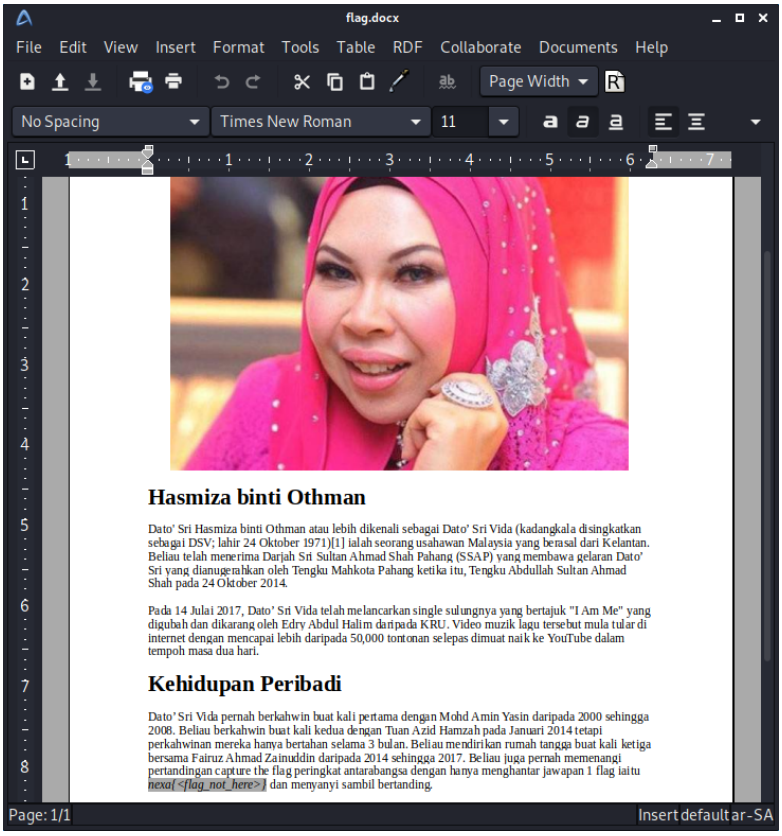
```
┌──(tan2540㉿kali)-[~/Downloads/corruptUnzip]
└─$ unzip -P lavidaesbella /home/tan2540/Downloads/corruptUnzip/flag.zip -d /home/tan2540/Downloads/co
rruptUnzip
Archive:  /home/tan2540/Downloads/corruptUnzip/flag.zip
   skipping: flag.docx               unsupported compression method 99
```



- After multiple attempts in trying to extract the .docx file (e.g. uploading the .zip file to google drive and download then extracting it in Windows etc.), another approach was made to open the file with a free and open-source word processor called Abiword.

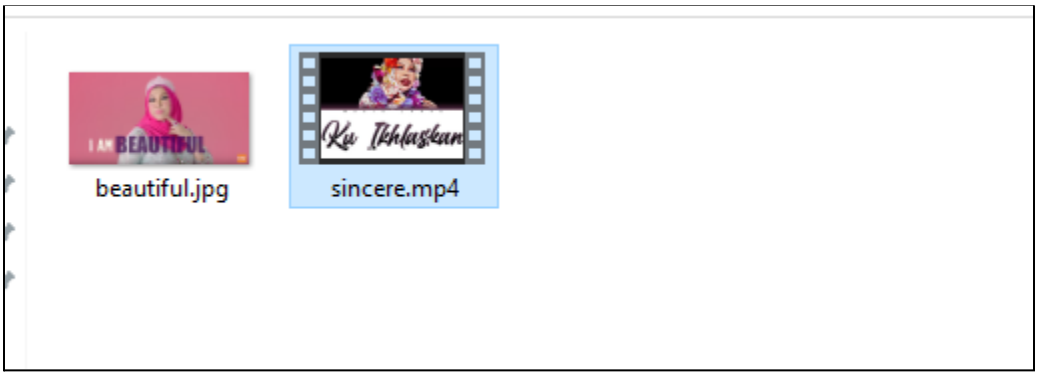


- Finally, the flag.docx file can be viewed and the flag is noticed in the second paragraph.



Flag revealed: ***nexa{<flag_not_here>}***

- Unfortunately, the flag was found after the second CTF challenge while taking more time to crack the password therefore we did not manage to get the marks for this question during CTF Challenge 2.

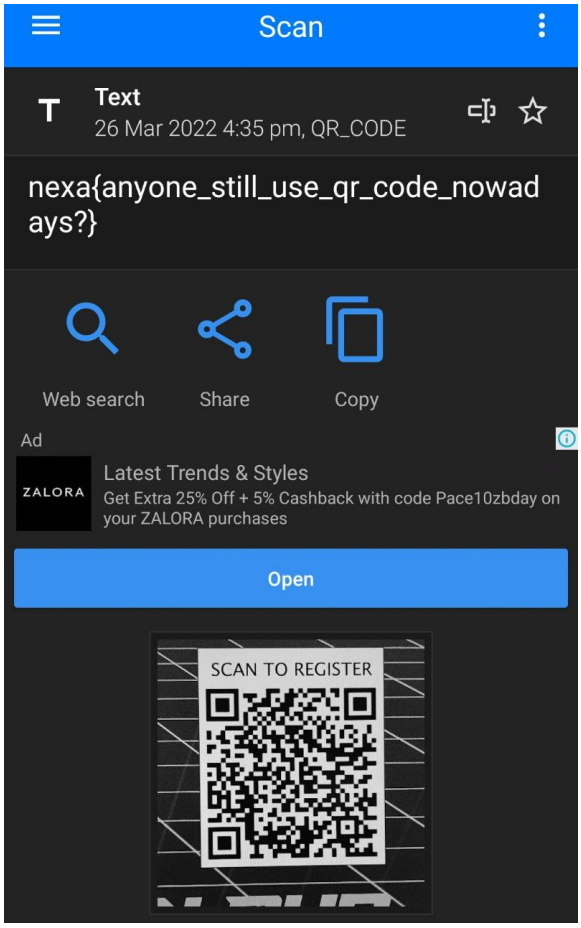
- Cheok Yi Xuan:
    - Change the file extension to .zip
    - Extract the zip file to reveal a .jpg and .mp4 file
    - 

Misc

Challenge Poster

Members involved: Afif

Thought process/methodologies:

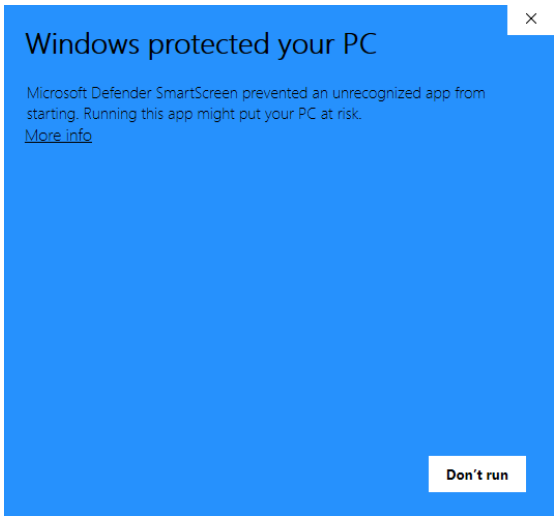
- Open image file
- Scan the QR code
- 

- Flag: ***nexa{anyone_still_use_qr_code_nowadays?}***

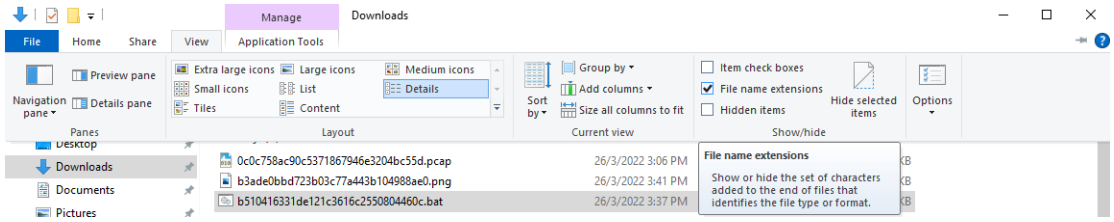
Click for Surprise!!!!

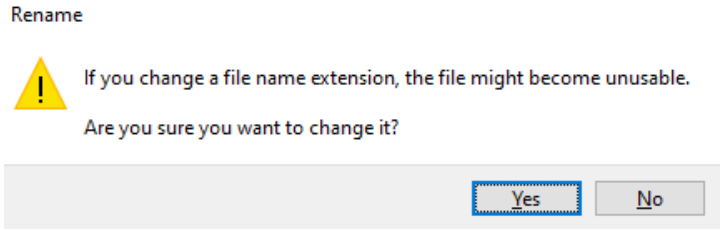
Members involved: Cheok Yi Xuan

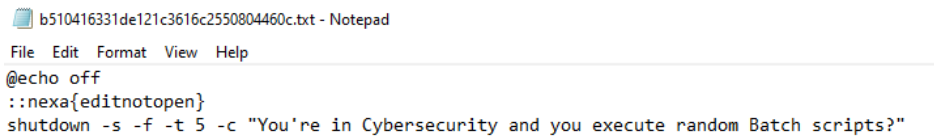
Thought process/methodologies:

- Run the bat file
- 

- Shuts down the computer
- 

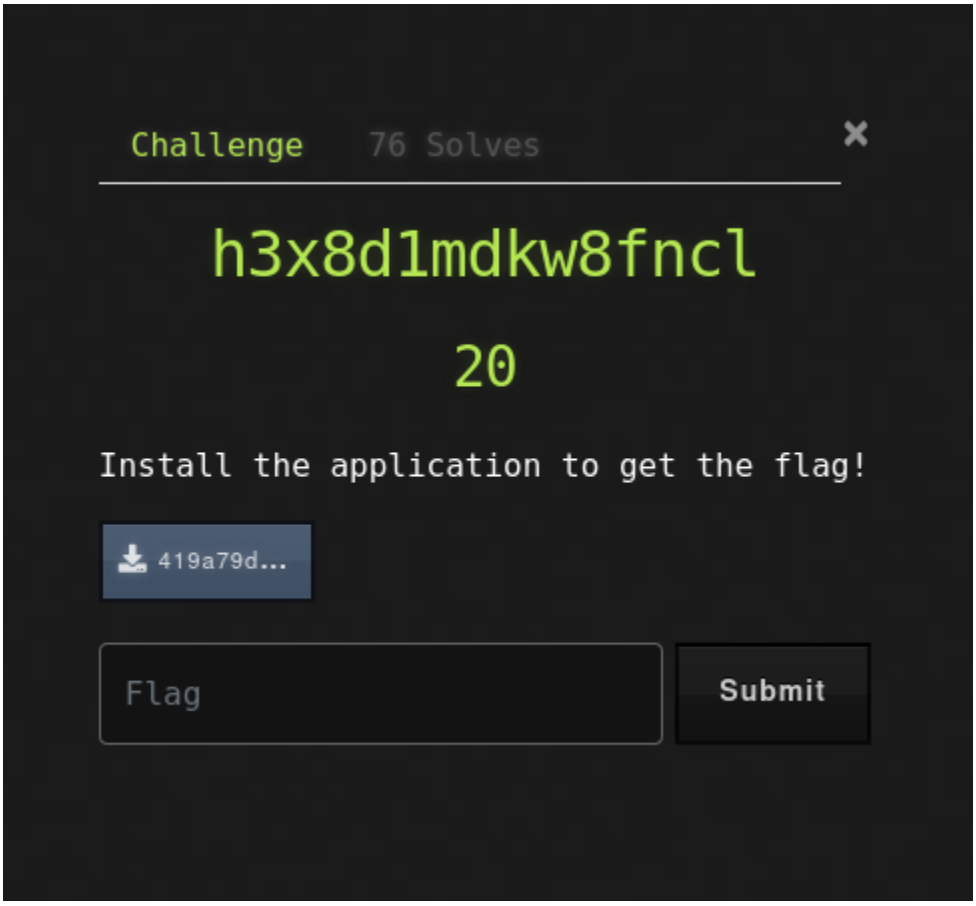
- Change the extension to .txt and open
- 

- 

- 

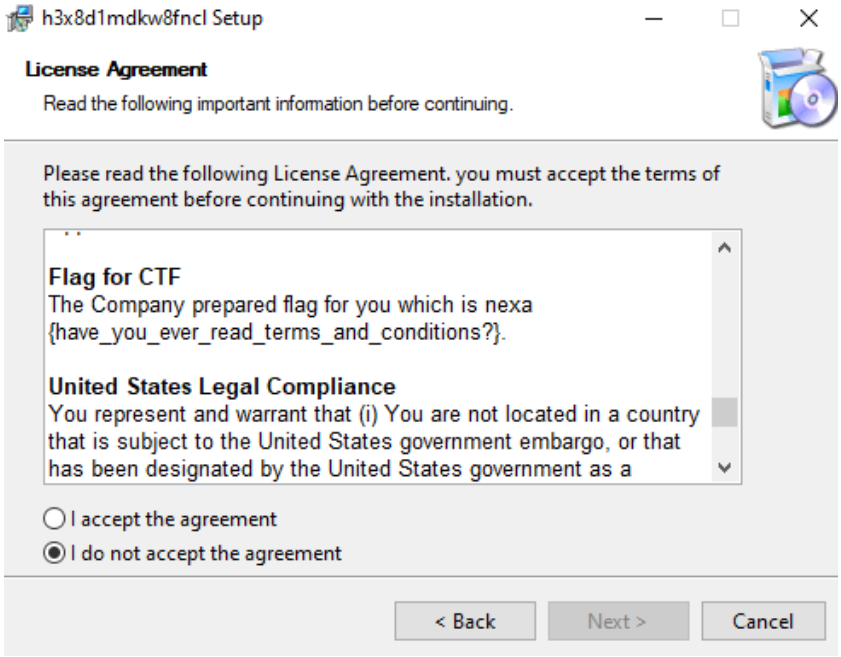
- Flag: ***nexa{editnotopen}***

H3x8d1mdkw8fncl



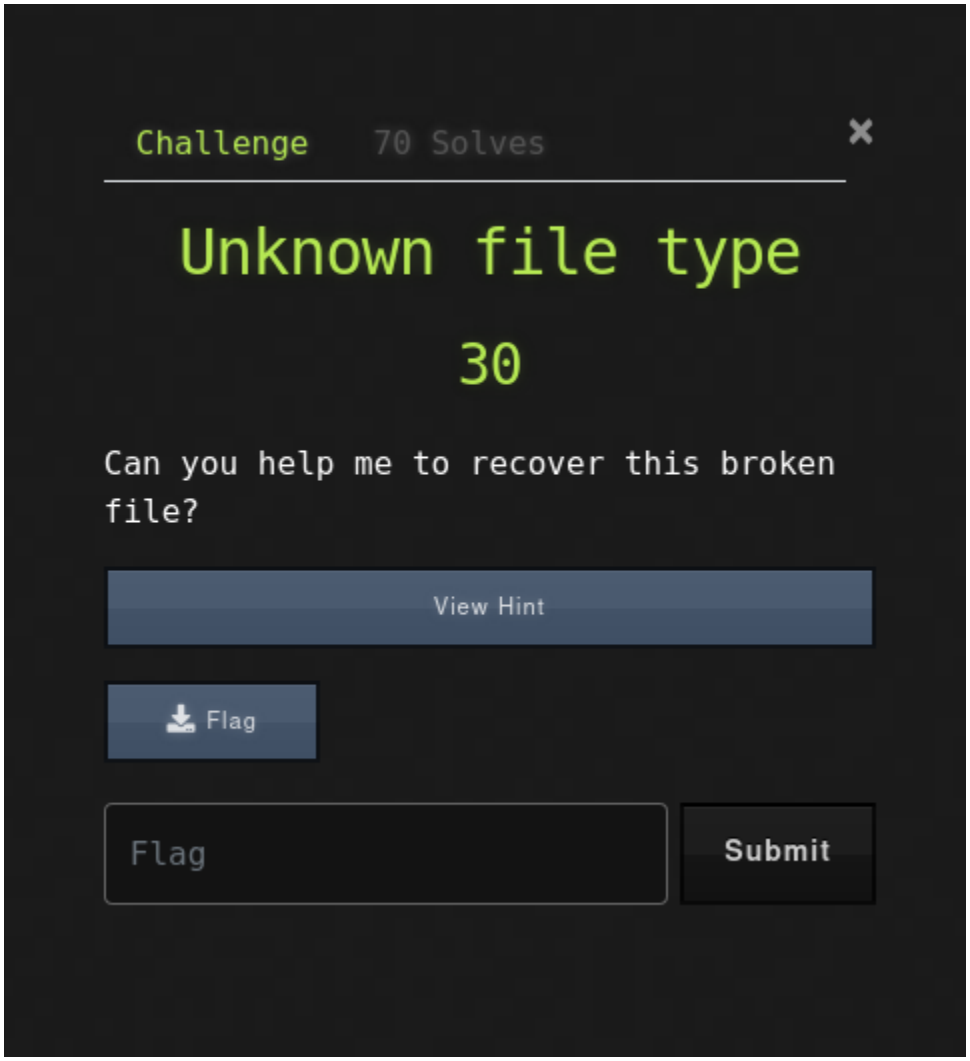
Members involved: Cheok Yi Xuan

Thought process/methodologies:

- Install the application
- Find the link to the youtube video
- Reinstall the application
- Look around the license agreement
- 

- Flag: ***nexa{have_you_ever_read_terms_and_conditions?}***

Unknown File Type



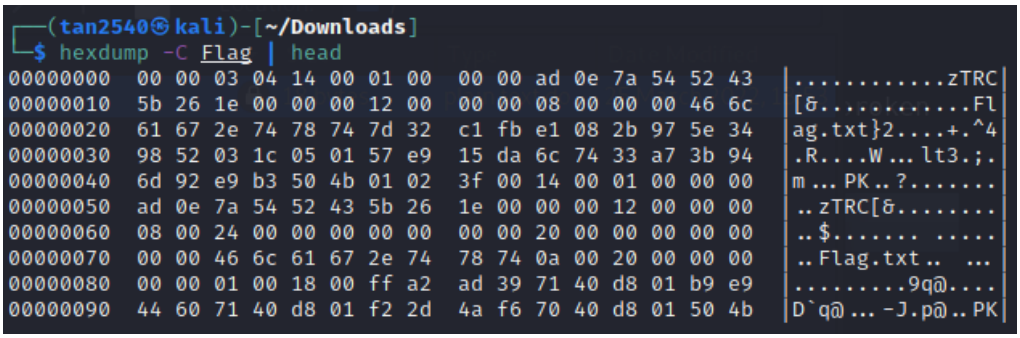
Members involved: Afif, Tan Shupei

Thought process/methodologies:

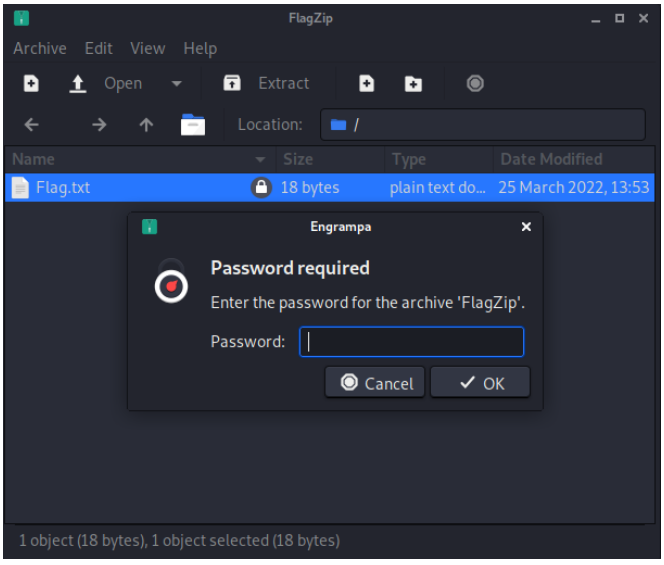
- Afif
  - Identify file extension type by running *file* on the downloaded *Flag* file
  - 

  - Add *.zip* extension behind the file name
  - File returns as an archived file; can be opened by an archiving software
  - Archived file contains a text file and seems to be protected by a password
  - Password not identified yet
  - Unable to solve the challenge…

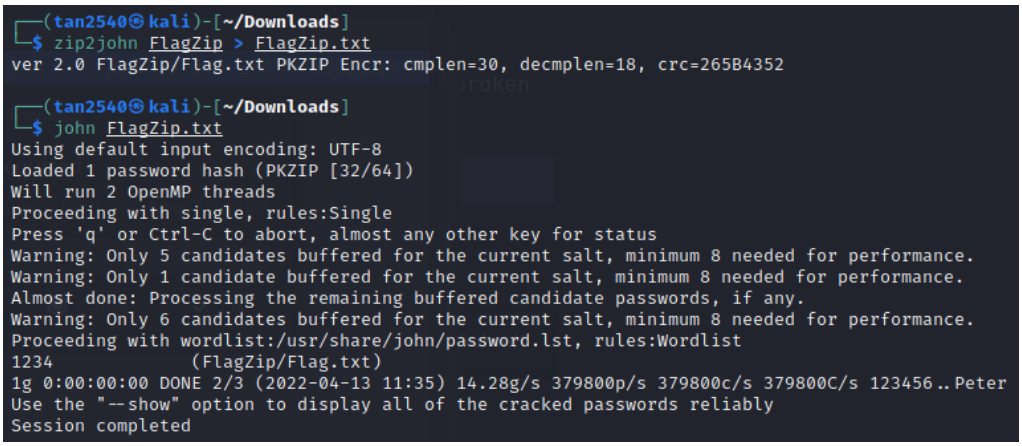
- Tan Shupei
  - Hexdump was used to check the hex content of the file first to identify the file type and to check if there is any corruption at the file header.



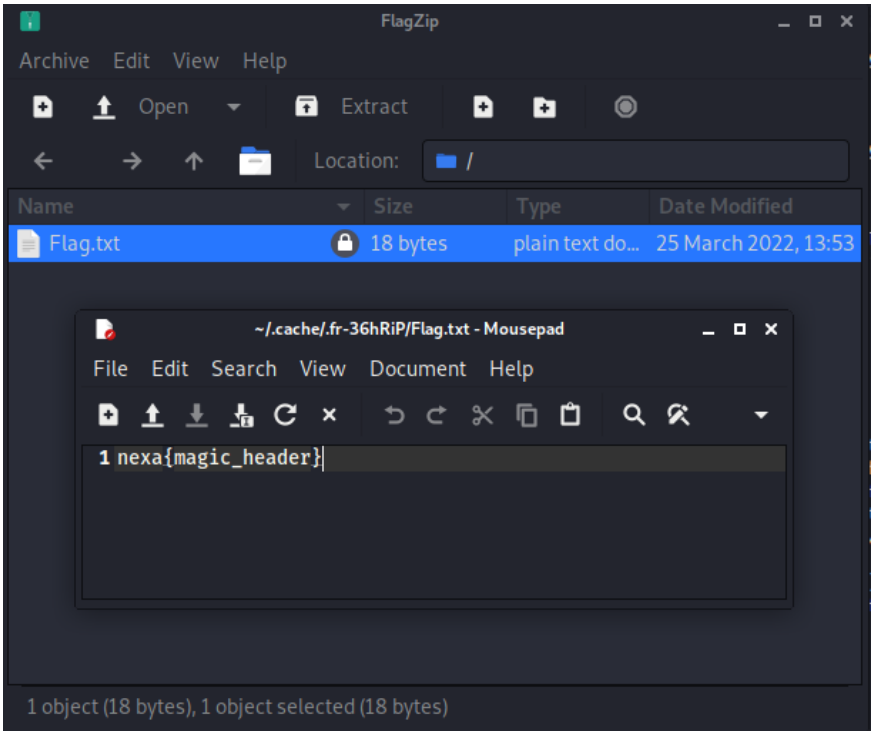
  Flag.txt was noticed on the 8th line therefore the file type is suspected to be a .zip file.
  - After using hexeditor to modify the file header, a .zip file was returned with a Flag.txt file as predicted earlier but password is required to extract the text file.



- John The Ripper was used to crack the password of the .zip file and **1234** seems to be the password to extract the .zip file.



- **1234** was indeed the password and the .zip file extraction took place successfully.

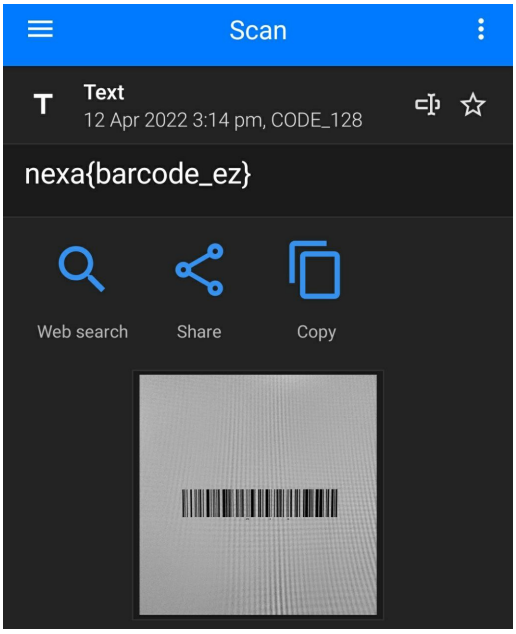


- Flag revealed: ***nexa{magic_header}***
- Unfortunately, same as “Basic 4”, the flag was found after the first CTF challenge while going through the questions again and reattempting them therefore we did not manage to get the marks for this question during CTF Challenge 1.

Barcode I

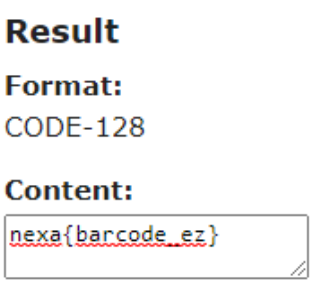
Members involved: Afif, Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

- Afif:
    - The file given is an image file with a PNG extension
    - Open the file
    - 
    - Use a phone application to scan the barcode
    - 

    - Flag is shown as: ***nexa{barcode_ez}***

- Khairul Hanie Hazierah Binti Mohd Azmi:
    - The picture in PNG format was open and inspected.



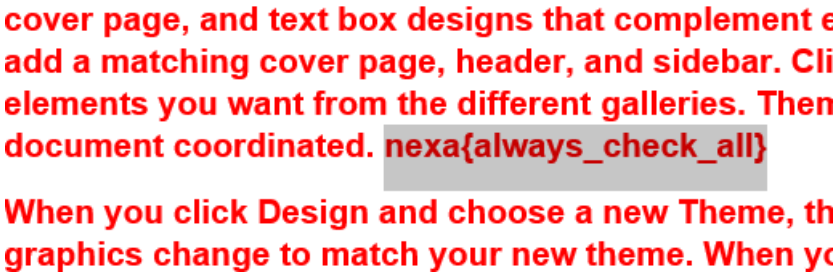
    - Using a barcode scanner online, the flag is revealed to be ***nexa{barcode_ez}***

Document 1

Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

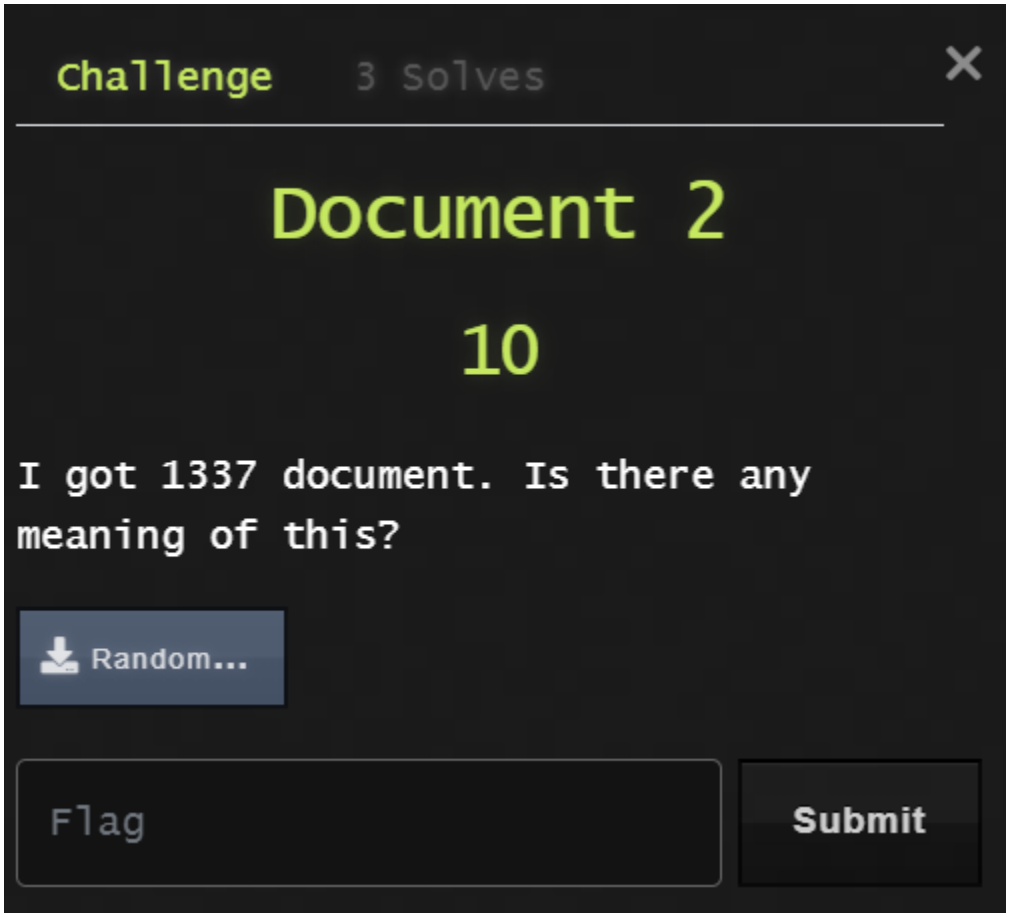
Thought process/methodologies:

- The file given was open and inspected.

- After changing the font color to red, the flag is revealed to be ***nexa{always_check_all}***

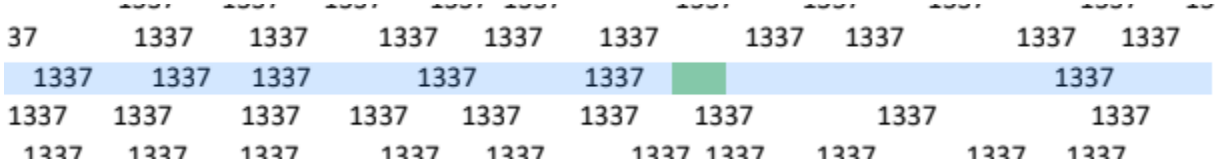
Document 2



Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

Thought process/methodologies:

- The file given was open and inspected.



- Using the control F command, the flag location was known but it cannot be seen.

nt1337from1337the1337different1337galleries.1337nexa{document_can_be_an
oying}1337Themes

- The invisible flag then was copied and pasted on google docs and the flag is revealed to be ***nexa{document_can_be_annoying}***

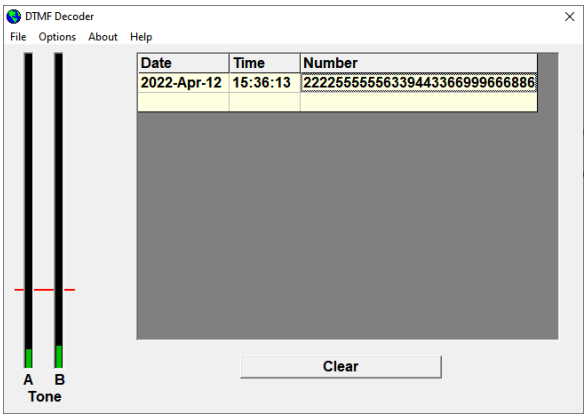
Call From Anonymous !!!

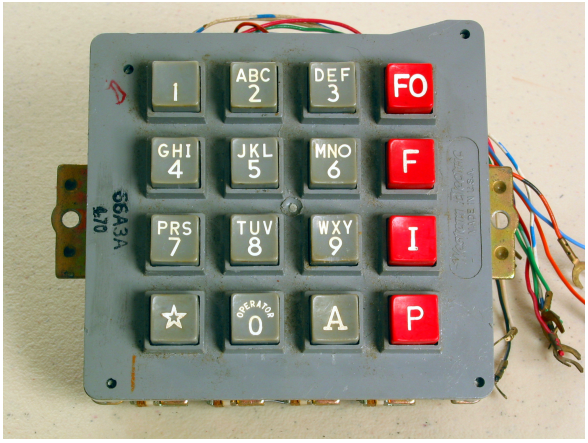
Members involved: Afif, Tan Shupei

Thought process/methodologies:

- Afif:
  - File given is an MP3 audio file
  - Open file
  - File title gives a clue as to what the demonstration is about
  - 

  - The demonstration indicates DTMF
  - Download and install a DTMF decoder software. Link here
  - 

  - Run software, play audio and begin analyzing the given numbers decoded
  - 

  - Audio analysis can be inconsistent due to window time of tone being played each time. A good spot for gap time between each tone is from 100ms to 250ms
  - DTMF recorded: ***22225555556339443366999666886633333633***
  - Find an image referencing a DTMF keypad
  - 

  - Translate DTMF to letters

    - DTMF translated: ***ALMEWHENYOUNEME***
    - Unable to decrypt the message…

- Tan Shupei
    - Based on the writeup provided by Afif, a different approach was done to translate the recorded DTMF which is to group 3 same numbers together if it occurs more than 3 times as shown below:
      222-2-555-555-6-33-9-44-33-66-999-666-88-66-33-33-3-6-33
    - Therefore, the DTMF is translated as C-A-L-L-M-E-W-H-E-N-Y-O-U-N-E-E-D-M-E
    - Flag revealed: ***nexa{callmewhenyouneedme}***

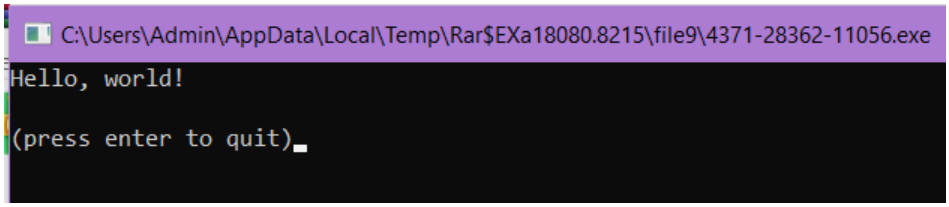
MD5 Collisions

Members involved: Khairul Hanie Hazierah Binti Mohd Azmi

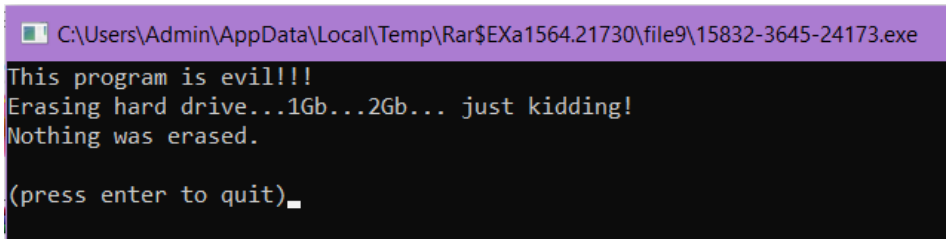
Thought process/methodologies:

- The file given was open and inspected.

13046-26279-51...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2
13312-21101-19...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2
13824-27822-21...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2
15832-3645-241...	6,144	2,422	Application	30/5/2018 7:16 ...	A89052AE
15849-30384-34...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2
15999-17275-48...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2
17580-12781-24...	6,144	2,400	Application	30/5/2018 7:16 ...	745475B2



- After running all the .exe files, most of the files contain the same data which is “Hello, world!”



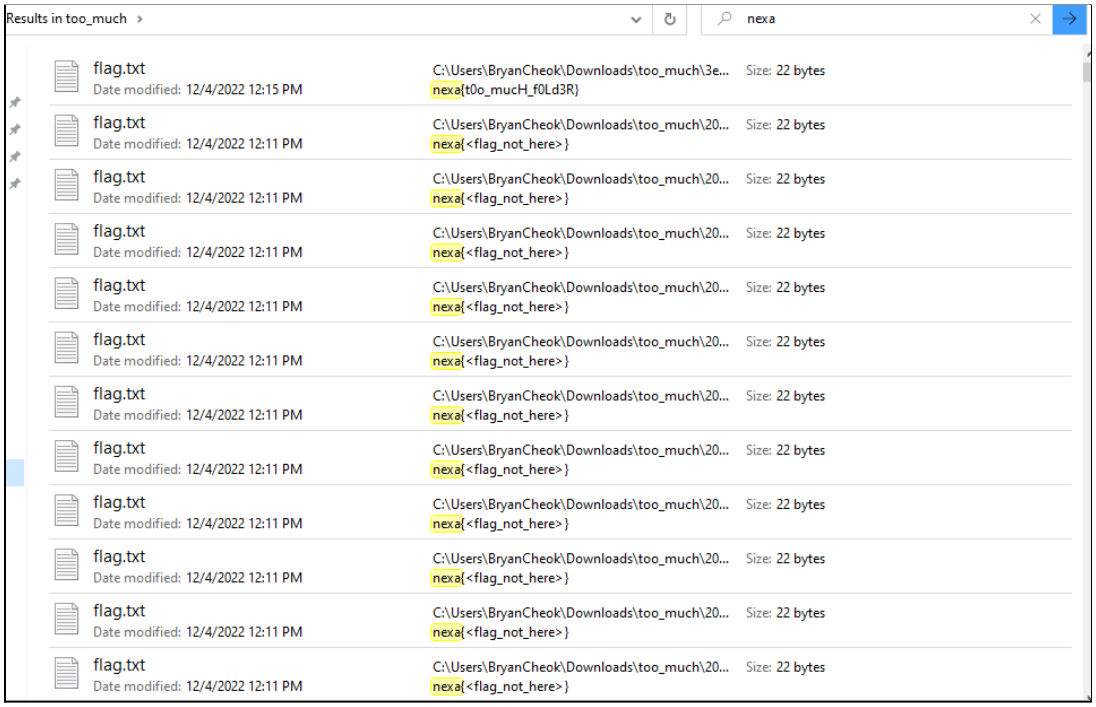
- File “15832-3645-24173.exe” contains different data that performs the malicious tasks.
- Therefore, the flag is revealed to be ***nexa{15832-3645-24173.exe}***

Too Much?

Members involved: Cheok Yi Xuan

Thought process/methodologies:

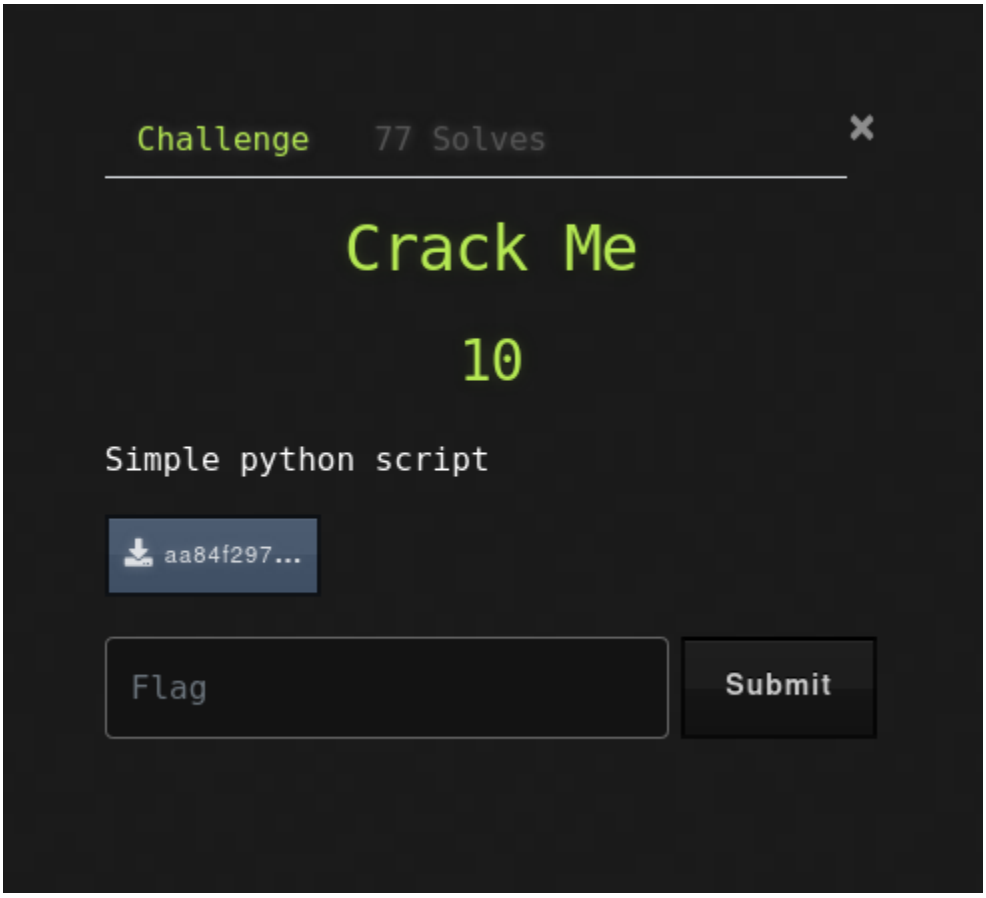
- Search for the word “nexa” in the directory



- Obtain the flag
- Flag revealed: ***nexa{t0o_mucH_f0Ld3R}***

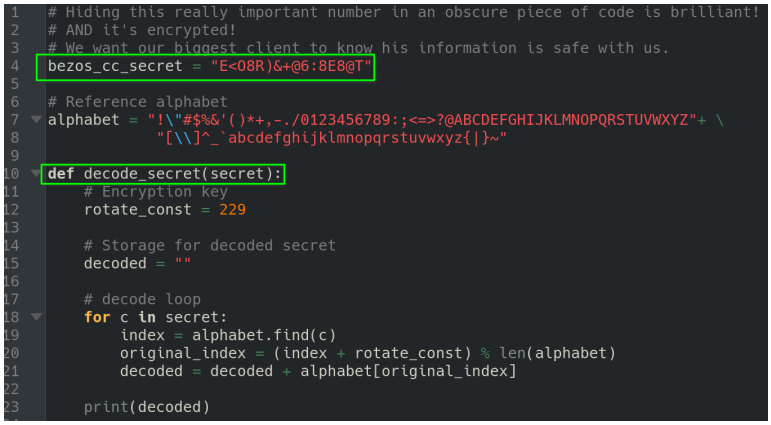
Reverse Engineering

Crack Me

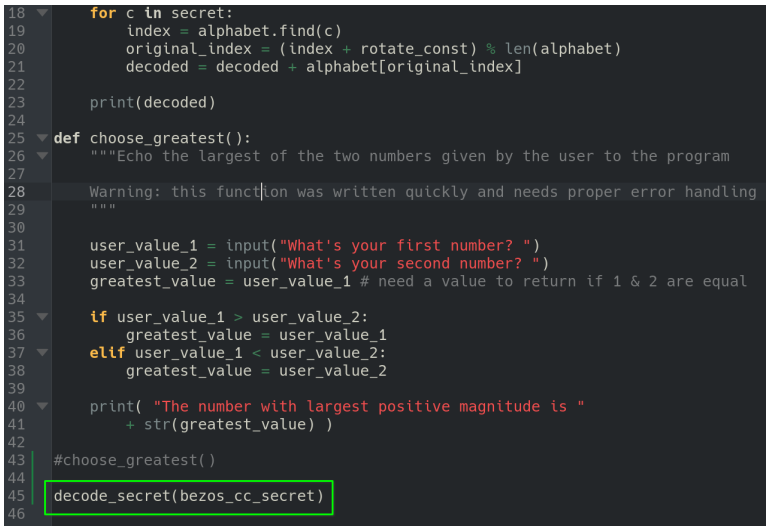


Members involved: Afif

Thought process/methodologies:

- Open Python file with a text editor
- Hint given is a variable which happens to be a secret key is called *bezos_cc_secret*
- 

```
# Hiding this really important number in an obscure piece of code is brilliant!
# AND it's encrypted!
# We want our biggest client to know his information is safe with us.
bezos_cc_secret = "E<O8R)&+@6:8E8@T"

# Reference alphabet
alphabet = "!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ"+ \
           "[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~"

def decode_secret(secret):
    # Encryption key
    rotate_const = 229

    # Storage for decoded secret
    decoded = ""

    # decode loop
    for c in secret:
        index = alphabet.find(c)
        original_index = (index + rotate_const) % len(alphabet)
        decoded = decoded + alphabet[original_index]

    print(decoded)
```

- Call function *decode_secret(secret)*, and pass the *bezos_cc_secret* variable by value
- 

```
    for c in secret:
        index = alphabet.find(c)
        original_index = (index + rotate_const) % len(alphabet)
        decoded = decoded + alphabet[original_index]

    print(decoded)

def choose_greatest():
    """Echo the largest of the two numbers given by the user to the program

    Warning: this function was written quickly and needs proper error handling
    """

    user_value_1 = input("What's your first number? ")
    user_value_2 = input("What's your second number? ")
    greatest_value = user_value_1 # need a value to return if 1 & 2 are equal

    if user_value_1 > user_value_2:
        greatest_value = user_value_1
    elif user_value_1 < user_value_2:
        greatest_value = user_value_2

    print( "The number with largest positive magnitude is "
        + str(greatest_value) )

#choose_greatest()

decode_secret(bezos_cc_secret)
```

- Run the Python file
- 

```
$ python aa84f29754b12220ea353db9c2867fa9.py
nexa{ROTi_canai}
```

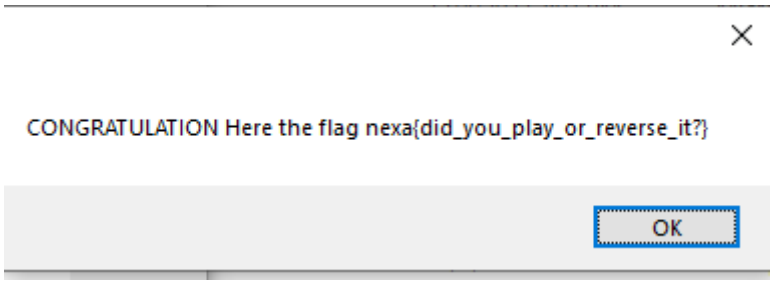
- Flag revealed: ***nexa{ROTi_canai}***

Lets Play

Members involved: Cheok Yi Xuan

Thought process/methodologies:

- Run the MAZE.exe file
- Beat the 2 maze games by searching for the traversal path from the finish line
- Obtain the flag



Flag: ***nexa{did_you_play_or_reverse_it?}***

Steganography

Basic 5

Members involved: Tan Shupei

Thought process/methodologies:

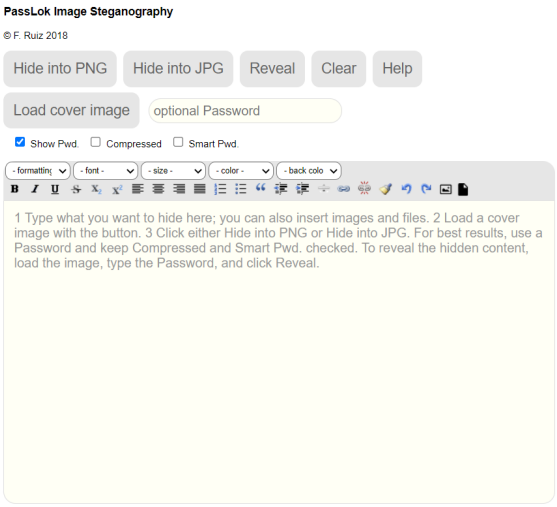
- As the file type of this image file is PNG, steghide cannot be used to extract the flag.

```
┌──(tan2540㉿kali)-[~/Downloads]
└─$ steghide info image-1648235030658.png

steghide: the file format of the file "image-1648235030658.png" is not supported.
```

- An attempt was made using hexdump to check if there is any hidden message at the IEND chunk but nothing strange was spotted.

```
┌──(tan2540㉿kali)-[~/Downloads]
└─$ hexdump -C image-1648235030658.png | tail
00094a20  1b 77 a1 1a 44 34 16 7a  b3 53 84 48 be f6 66 62  |.w..D4.z.S.H..fb|
00094a30  e3 13 aa d0 72 86 bb d1  15 ce 05 04 95 b7 6c 68  |....r.........lh|
00094a40  8a 5d 9a 5a 9a 2a 7b 36  d3 13 67 53 fd bc 7e ee  |.].Z.*{6..gS..~.|
00094a50  f0 a2 2c a8 64 12 23 df  ad 6d 2e 1c 49 ba 4c 50  |..,.d.#..m..I.LP|
00094a60  0c 20 a0 7e ae a2 59 55  a5 21 f7 5d 7c 88 38 16  |. .~..YU.!.]|.8.|
00094a70  01 5e d4 c2 99 6a a0 6b  38 80 2b 74 c2 59 01 e0  |.^...j.k8.+t.Y..|
00094a80  74 ef b9 83 03 90 36 ef  0a 55 66 d3 fe 1f 24 00  |t.....6..Uf...$.|
00094a90  b5 36 98 57 e4 46 00 00  00 00 49 45 4e 44 ae 42  |.6.W.F....IEND.B|
00094aa0  60 82                                             |`.|
00094aa2
```

- Another attempt was made using StegOnline which is used to manipulate the color of the image but still the flag or anything strange was spotted from the results.
- A browser extension named “PassLok Image Steganography” was also used as suggested in the hint given however a password is needed. Words like “flag”, “MMU”, “Basic 5”, “1234” were used but none of them seems to be the correct password.



Intermediate 4

Members involved: Tan Shupei

Thought process/methodologies:

- Since the file type is JPEG instead of PNG, steghide can be used.

```
┌──(tan2540㉿kali)-[~]
└─$ steghide info /home/tan2540/Downloads/RookChessmon.jpeg
"RookChessmon.jpeg":
  format: jpeg
  capacity: 50.6 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "flag.txt":
    size: 19.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

  admin was used as a passphrase as it is bolded in the question.

- Steghide was further used to reveal the flag extracted from the RookChessmon.jpeg file.

```
┌──(tan2540㉿kali)-[~]
└─$ steghide extract -sf /home/tan2540/Downloads/RookChessmon.jpeg
Enter passphrase:
wrote extracted data to "flag.txt".
```

```
┌──(tan2540㉿kali)-[~]
└─$ cat flag.txt
arkn{fgrtb_vf_sha}
```

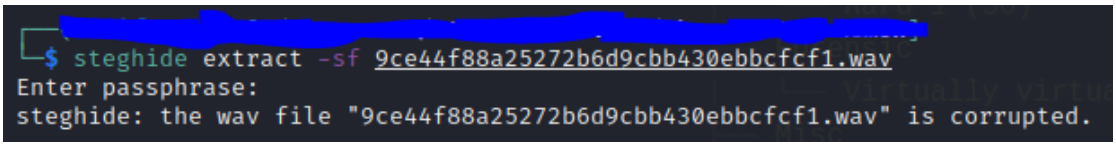
- ROT13 is used to reveal the flag extracted.

- Flag revealed: ***nexa{stego_is_fun}***

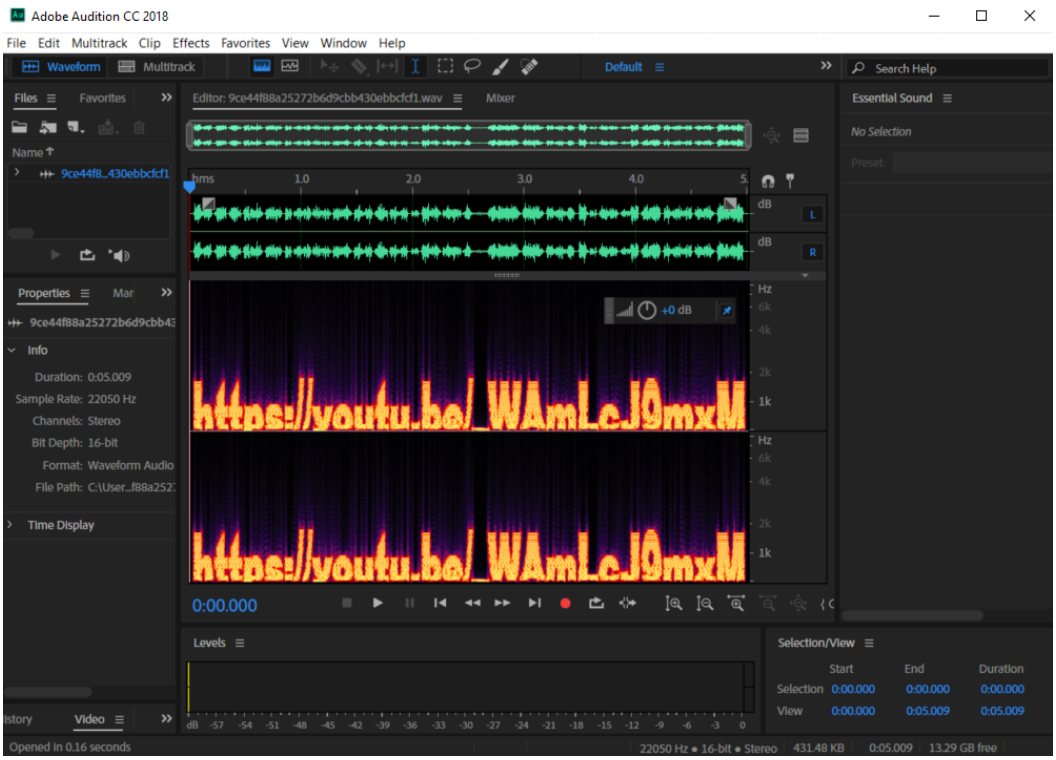
Who’s That Pokémon?

Members involved: Tan Shupei, Afif

Thought process/methodologies:

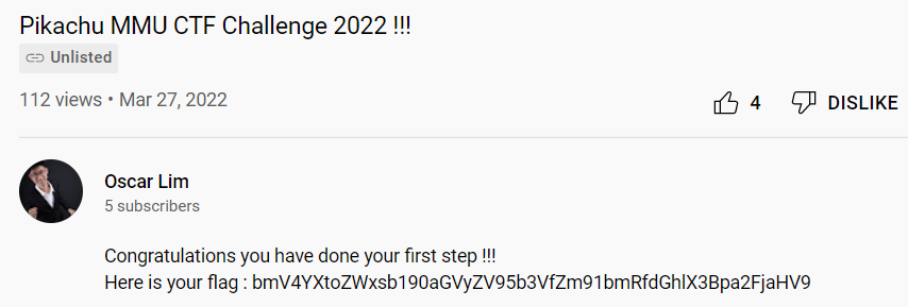
- Afif:
  - 

  - Audio file is given
  - Try to crack the stego file with *steghide*
  - Requires passphrase, and it has not been found yet
  - Unable to decode the audio file…
- Tan Shupei
  - The same approach as Afif was done using the passphrase “flag” but it did not decode the audio file.
  - Therefore another approach was done which is to use Adobe Audition to view the audio spectrum and a YouTube link was shown in the spectrogram.

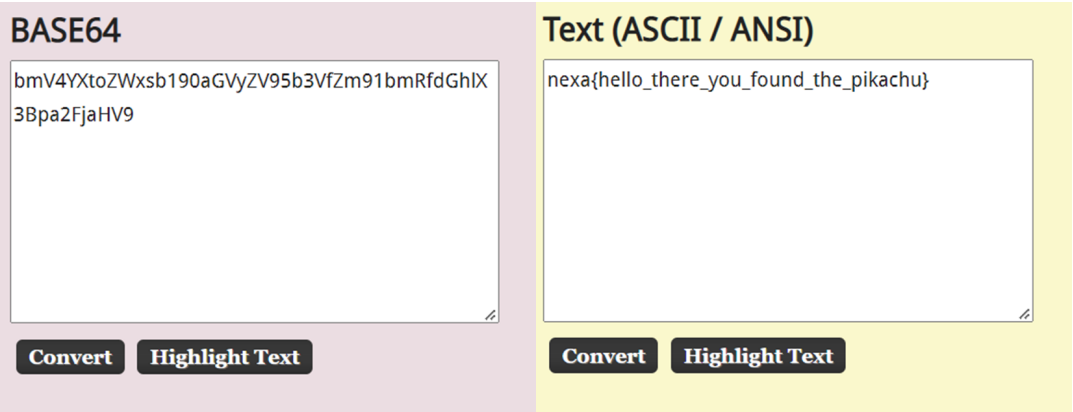


  - https://www.youtube.com/watch?v=_WAmLcJ9mxM
    The flag “bmV4YXtoZWxsb190aGVyZV95b3VfZm91bmRfdGhlX3Bpa2FjaHV9”

was then found in the video description box but it is not the exact flag yet.



- Therefore, a ASCII converter tool is used to convert the flag from Base64 to ASCII as shown below.



- Flag revealed: ***nexa{hello_there_you_found_the_pikachu}***

Web

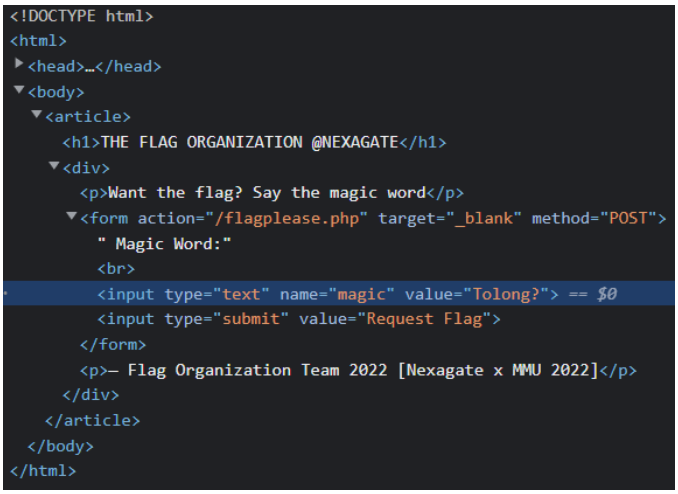
Say the MAGIC WORD!

Members involved: Tan Shupei, Afif, Khairul Hanie Hazierah Binti Mohd Azmi, Cheok Yi Xuan

Thought process/methodologies:

- Tan Shupei
  - The inspect of the page was checked but nothing strange was spotted.
  - Since the question is asking for magic word, words like “please”, “thank you”, “abracadabra” were used as an input but none revealed the flag.
  - An attempt to view flagplease.php was also made however it is not doable as PHP is a server-side programming language therefore flagplease.php is not viewable at all.
  - An attempt was also made to change the form method to GET but the flag still could not be revealed successfully.

- Afif:
  - Click on link given by the question
  - Link leads to a page requesting user input as *Request Flag*
  - 

  - By viewing the page source of the HTML file, the variable assignment *magic* might give a clue
  - 

  - No obvious hardcoded flag found anywhere
  - Unable to proceed from here…

- Khairul Hanie Hazierah Binti Mohd Azmi:
  - The link given was open and the elements and the sources were analyzed.
  - Based on the hint given, “wstg-inpv-03” indicates a HTTP verb tampering/parameter tampering.

    - A lot of trial and error occurred but the flag is nowhere to be found.

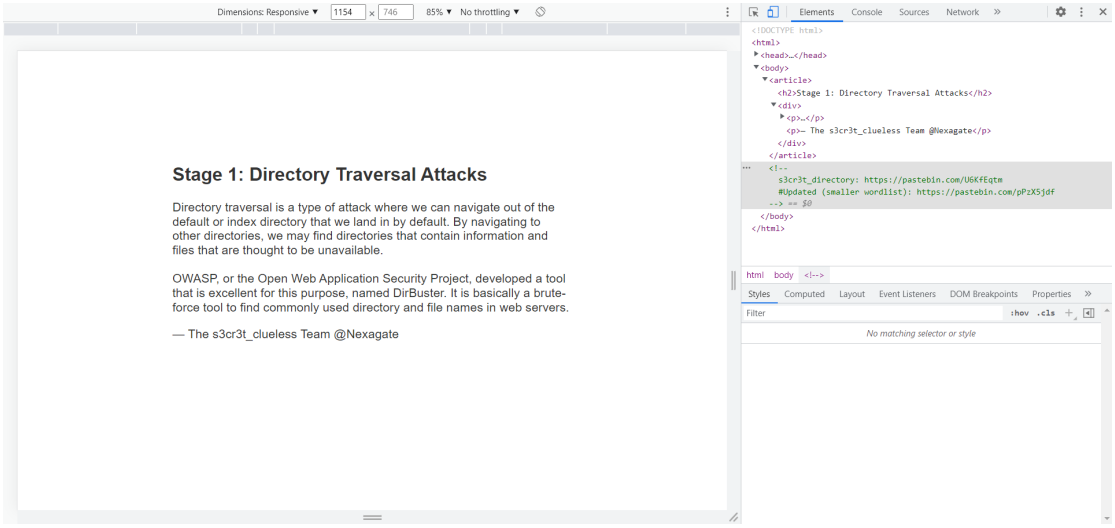
- Cheok Yi Xuan
    - Open the link provided
    - Tried multiple inputs such as “magic”, “help”, “please”, etc
    - Changed the form method using inspect element and attempt to submit
    - Tried using command console “curl” command with multiple different method calls

Clueless Wargames!

Members involved: Tan Shupei, Afif

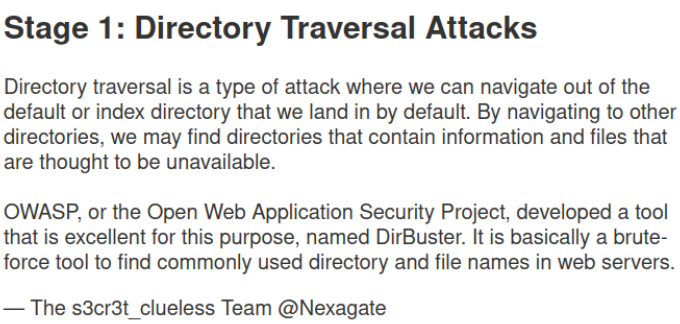
Thought process/methodologies:

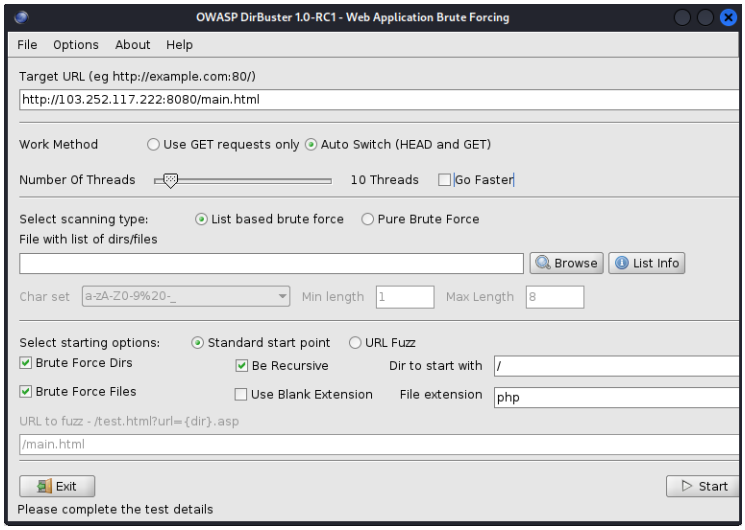
- Tan Shupei
    - The inspect of the page was checked and something was spotted in the code of lines that were commented.



`s3cr3t_directory: https://pastebin.com/U6KfEqtm`
`#Updated (smaller wordlist): https://pastebin.com/pPzX5jdf`

- Afif:





- Question mentions about a software named *DirBuster* that checks for common directory exploits in websites
- Run *DirBuster*, paste in the page's URL, and start the test
- Unable to continue from here due to many program errors in the brute forcing process…