CTF CHALLENGE

CYBER FORENSICS

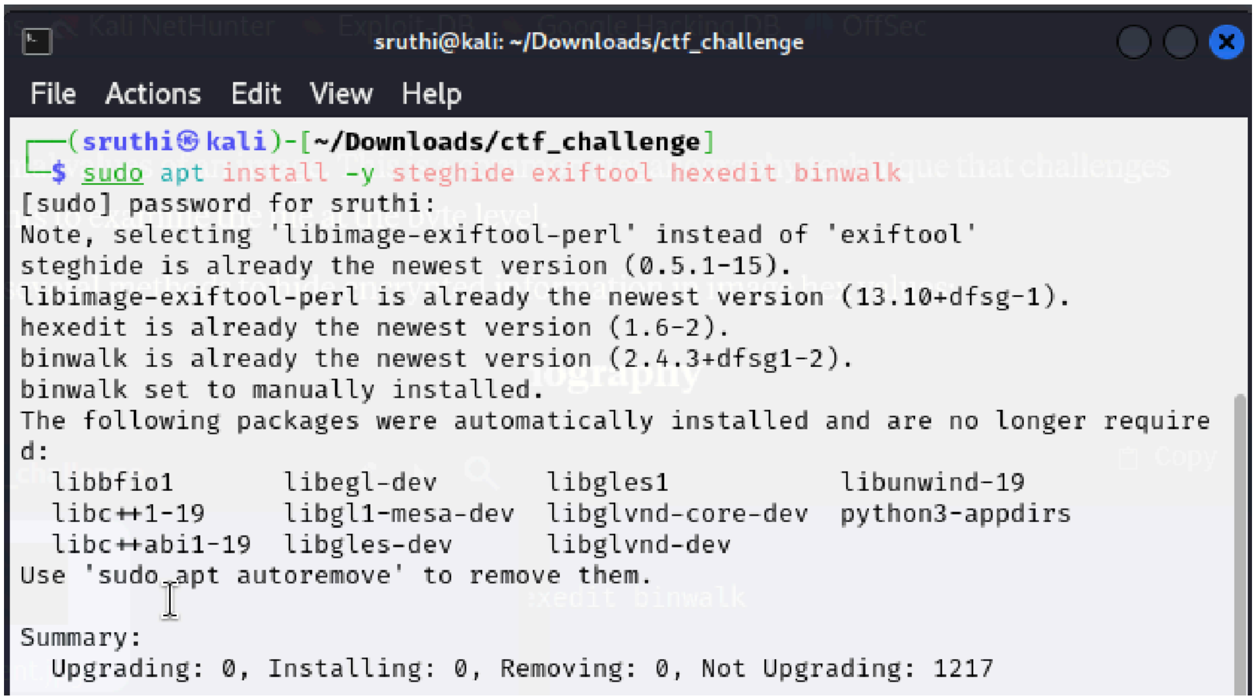
SRUTHI THANDOJU

Creation of CTF Challenge :

Creating image-based CTF challenge where the flag is hidden within the hexadecimal values of an image. This is the common steganography technique that challenges to examine the file at byte level.

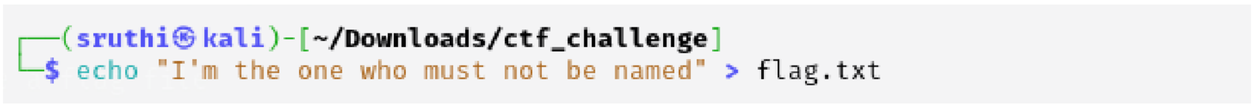
We can encrypt the information in image hex values.

1. **Installing all the modules required in the machine.**



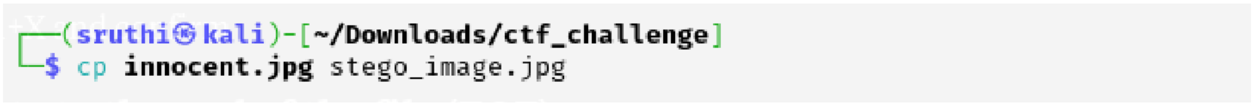
```
┌──(sruthi㉿kali)-[~/Downloads/ctf_challenge]
└─$ sudo apt install -y steghide exiftool hexedit binwalk
[sudo] password for sruthi:
Note, selecting 'libimage-exiftool-perl' instead of 'exiftool'
steghide is already the newest version (0.5.1-15).
libimage-exiftool-perl is already the newest version (13.10+dfsg-1).
hexedit is already the newest version (1.6-2).
binwalk is already the newest version (2.4.3+dfsg1-2).
binwalk set to manually installed.
The following packages were automatically installed and are no longer require
d:
  libbfio1        libegl-dev       libgles1           libunwind-19
  libc++1-19      libgl1-mesa-dev  libglvnd-core-dev  python3-appdirs
  libc++abi1-19   libgles-dev      libglvnd-dev
Use 'sudo apt autoremove' to remove them.

Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1217
```

2. **Creating a flag file information with in the folder.**



```
┌──(sruthi㉿kali)-[~/Downloads/ctf_challenge]
└─$ echo "I'm the one who must not be named" > flag.txt
```

3. **Making a copy of the image file for safety.**



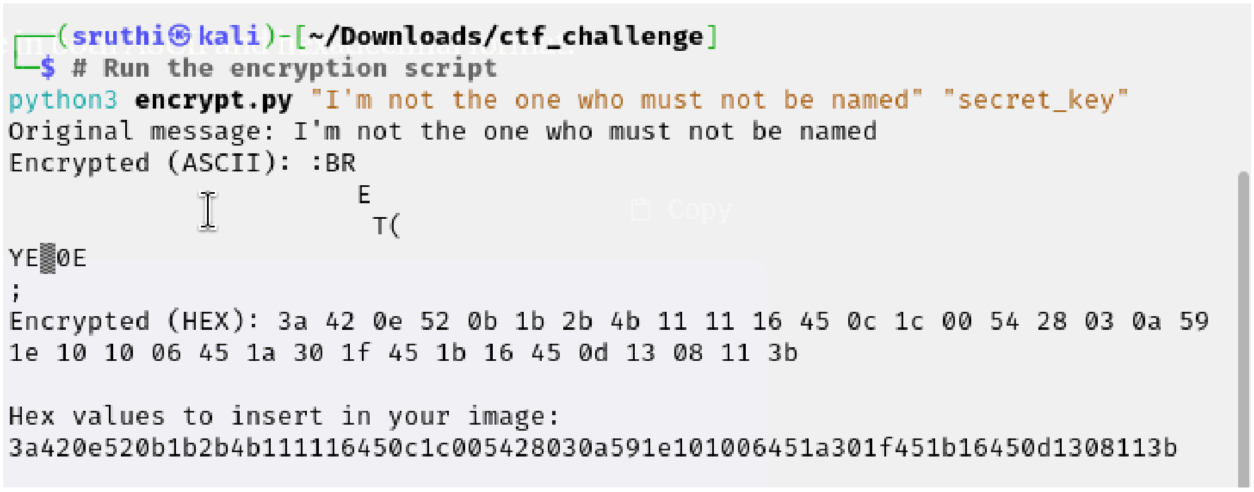
```
┌──(sruthi㉿kali)-[~/Downloads/ctf_challenge]
└─$ cp innocent.jpg stego_image.jpg
```

4. **Let's create a simple encryption script which encrypts the secret message. Here I'm using XOR encryption**

   encryption script :

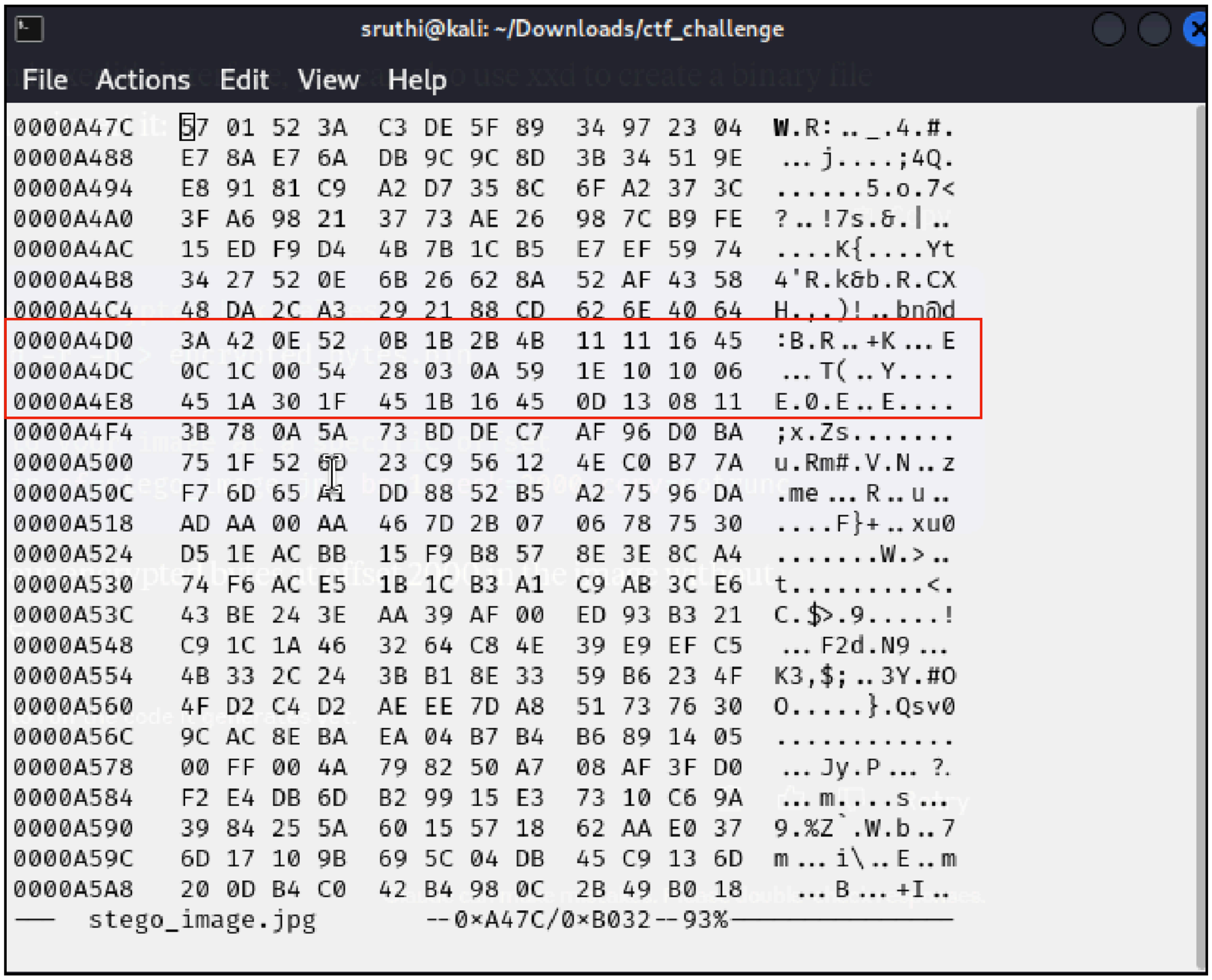
```
#!/usr/bin/env python3
import sys
def xor_encrypt(message, key):
    for i in range(len(message)):
        encrypted += chr(ord(message[i]) ^ ord(key[i % len(key)]))
    return encrypted
def string_to_hex(text):
    return ' '.join([hex(ord(c))[2:].zfill(2) for c in text])
if len(sys.argv) != 3:
    print("Usage: python3 encrypt.py <message> <key>")
    sys.exit(1)
message = sys.argv[1]
key = sys.argv[2]
encrypted = xor_encrypt(message, key)
hex_result = string_to_hex(encrypted)
print("Original message:", message)
print("Encrypted (ASCII):", encrypted)
print("Encrypted (HEX):", hex_result)
print("\nHex values to insert in your image:")
print(hex_result.replace(' ', ‘'))
```

5. **Encrypting the message below. Displaying the ASCII and 10100HEX values to insert in your image.**



```
┌──(sruthi㉿kali)-[~/Downloads/ctf_challenge]
└─$ # Run the encryption script
python3 encrypt.py "I'm not the one who must not be named" "secret_key"
Original message: I'm not the one who must not be named
Encrypted (ASCII): :BR
                                 E
                                  T(
YE▒0E
;
Encrypted (HEX): 3a 42 0e 52 0b 1b 2b 4b 11 11 16 45 0c 1c 00 54 28 03 0a 59
1e 10 10 06 45 1a 30 1f 45 1b 16 45 0d 13 08 11 3b

Hex values to insert in your image:
3a420e520b1b2b4b111116450c1c005428030a591e101006451a301f451b16450d1308113b
```

6. Inserting hex values in the image.


sruthi@kali: ~/Downloads/ctf_challenge
File Actions Edit View Help

```
0000A47C  57 01 52 3A  C3 DE 5F 89  34 97 23 04  W.R:.._.4.#.
0000A488  E7 8A E7 6A  DB 9C 9C 8D  3B 34 51 9E  ...j....;4Q.
0000A494  E8 91 81 C9  A2 D7 35 8C  6F A2 37 3C  ......5.o.7<
0000A4A0  3F A6 98 21  37 73 AE 26  98 7C B9 FE  ?..!7s.&.|..
0000A4AC  15 ED F9 D4  4B 7B 1C B5  E7 EF 59 74  ....K{....Yt
0000A4B8  34 27 52 0E  6B 26 62 8A  52 AF 43 58  4'R.k&b.R.CX
0000A4C4  48 DA 2C A3  29 21 88 CD  62 6E 40 64  H.,.)!..bn@d
0000A4D0  3A 42 0E 52  0B 1B 2B 4B  11 11 16 45  :B.R..+K...E
0000A4DC  0C 1C 00 54  28 03 0A 59  1E 10 10 06  ...T(..Y....
0000A4E8  45 1A 30 1F  45 1B 16 45  0D 13 08 11  E.0.E..E....
0000A4F4  3B 78 0A 5A  73 BD DE C7  AF 96 D0 BA  ;x.Zs.......
0000A500  75 1F 52 6D  23 C9 56 12  4E C0 B7 7A  u.Rm#.V.N..z
0000A50C  F7 6D 65 A1  DD 88 52 B5  A2 75 96 DA  .me...R..u..
0000A518  AD AA 00 AA  46 7D 2B 07  06 78 75 30  ....F}+..xu0
0000A524  D5 1E AC BB  15 F9 B8 57  8E 3E 8C A4  .......W.>..
0000A530  74 F6 AC E5  1B 1C B3 A1  C9 AB 3C E6  t.........<.
0000A53C  43 BE 24 3E  AA 39 AF 00  ED 93 B3 21  C.$>.9.....!
0000A548  C9 1C 1A 46  32 64 C8 4E  39 E9 EF C5  ...F2d.N9...
0000A554  4B 33 2C 24  3B B1 8E 33  59 B6 23 4F  K3,$;..3Y.#O
0000A560  4F D2 C4 D2  AE EE 7D A8  51 73 76 30  O.....}.Qsv0
0000A56C  9C AC 8E BA  EA 04 B7 B4  B6 89 14 05  ............
0000A578  00 FF 00 4A  79 82 50 A7  08 AF 3F D0  ...Jy.P...?.
0000A584  F2 E4 DB 6D  B2 99 15 E3  73 10 C6 9A  ...m....s...
0000A590  39 84 25 5A  60 15 57 18  62 AA E0 37  9.%Z`.W.b..7
0000A59C  6D 17 10 9B  69 5C 04 DB  45 C9 13 6D  m...i\..E..m
0000A5A8  20 0D B4 C0  42 B4 98 0C  2B 49 B0 18   ...B...+I..
---  stego_image.jpg     --0×A47C/0×B032--93%----------------
```

Solving the CTF challenge :

So the challenge would be asked as below,

- **Encrypted Hex Challenge**
  An agent has hidden encrypted information within the image file.

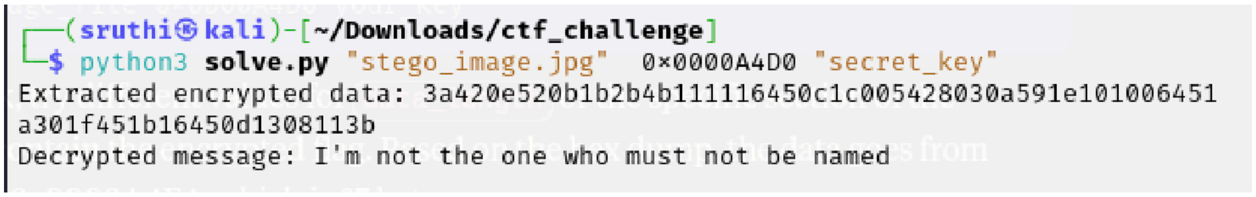
  Mission:
  - Examine the hexadecimal structure of the image.
  - Find the hidden encrypted data.
  - Decrypt the message using XOR decryption.
  - The encryption key is hidden somewhere in plain sight.

Submit the Flag.
Hint: The encryption key might be related to the simple key.

1. Creating a solution verification tool using python script.

```
import sys
def xor_decrypt(encrypted, key):
    decrypted = ""
    for i in range(len(encrypted)):
      decrypted += chr(ord(encrypted[i]) ^ ord(key[i % len(key)]))
    return decrypted
def extract_hex_data(filename, start_offset, length):
    with open(filename, 'rb') as f:
        f.seek(start_offset)
        data = f.read(length)
return data
def hex_to_string(hex_bytes):
    return ''.join([chr(b) for b in hex_bytes])
if len(sys.argv) != 4:
    print("Usage: python3 solve.py <image_file> <offset> <key>")
    sys.exit(1)
image_file = sys.argv[1]
offset = int(sys.argv[2])
key = sys.argv[3]

data_length = len("CTF{h3x_st3g0_m4st3r}")

encrypted_bytes = extract_hex_data(image_file, offset, data_length)
encrypted_text = hex_to_string(encrypted_bytes)

decrypted = xor_decrypt(encrypted_text, key)

print("Extracted encrypted data:", encrypted_bytes.hex())
print("Decrypted message:", decrypted)
```

2. After decrypting the hex dump using above python script, I should be getting something like below,



```
┌──(sruthi㉿kali)-[~/Downloads/ctf_challenge]
└─$ python3 solve.py "stego_image.jpg"  0x0000A4D0 "secret_key"
Extracted encrypted data: 3a420e520b1b2b4b111116450c1c005428030a591e101006451
a301f451b16450d1308113b
Decrypted message: I'm not the one who must not be named
```

3. So, I got the exact message which was encrypted by me in the creating part.