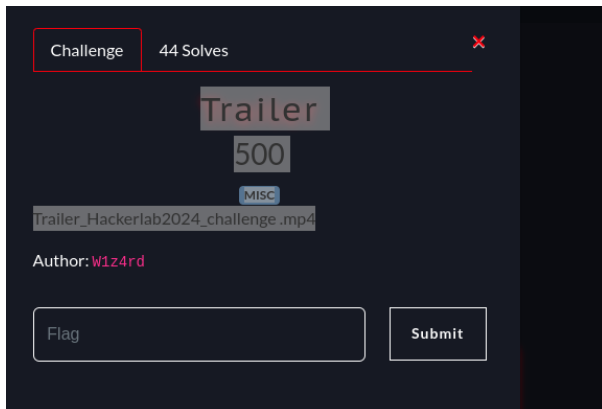


Preuve du challenge

Nom du challenge :QUALIFICATION STAGES

Auteur : **foundhack**



Etape :1

J'ai télécharger la vidéo et j'ai pris du temps pour regarder la vidéo à plusieurs reprises mais pour instant rien j'ai fait **exiftool** pour voir s'il a eu des données cachées dans la vidéo mais rien.

😞👤👤👤👤 je recommence encore la vidéo et j'ai commencé par traduire les phrases de la vidéo en français car je ne suis pas bon en anglais ça peut être des indices. Oui j'ai eu raison car avant Le message « **Are you up for the challenge ?** » juste à la fin de la vidéo je vois des petits pixels branche en haut dans la partie droite de la vidéo et je me dis **big big** c'est ça ma solution.

Etape:2 et Etape :3

Création du dossier resolve et je lance mon script « *J'ai capture tous les frames* »

```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge
(kali@kali)-[~/Vidéos]
$ cd Trailer_Hackerlab2024_challenge
(kali@kali)-[~/Vidéos/Trailer_Hackerlab2024_challenge]
$ mkdir resolve
(kali@kali)-[~/Vidéos/Trailer_Hackerlab2024_challenge]
$ ffmpeg -i Trailer_Hackerlab2024_challenge.mp4 -vf fps=60 resolve/%d.png
```

Etape:4

```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge

(kali@kali)-[~/Vidéos]
$ cd Trailer_Hackerlab2024_challenge

(kali@kali)-[~/Vidéos/Trailer_Hackerlab2024_challenge]
$ mkdir resolve

(kali@kali)-[~/Vidéos/Trailer_Hackerlab2024_challenge]
$ ffmpeg -i Trailer_Hackerlab2024_challenge.mp4 -vf fps=60 resolve/%.png
ffmpeg version 6.1.1-4+b1 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Debian 13.2.0-24)
  configuration: --prefix=/usr --extra-version=4+b1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/usr/include/x86_64-linux-gnu --arch=amd64 --enable-gpl --disable-stripping --disable-omx --enable-gnutls --enable-libaom --enable-libb
s2b --enable-libcaca --enable-libcdio --enable-libcodecs2 --enable-libdav1d --enable-libflite --enable-libfontconfig --enable-libfreety
pe --enable-libfribidi --enable-libgslang --enable-libgme --enable-libgsm --enable-libharfbuzz --enable-libmp3lame --enable-libmysofa
--enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-librubberband --enable-libshine --enable-lbsnappy --enable-libsox
r --enable-lbspeex --enable-libtheora --enable-libtwolame --enable-libvidstab --enable-libvorbis --enable-libvpx --enable-libwebp --e
nable-libx265 --enable-libxml2 --enable-libxvid --enable-libzimg --enable-opengl --enable-openc1 --enable-opengl --disable-sndio --ena
ble-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-lads
pa --enable-libbluray --enable-libjack --enable-libpulse --enable-librabbittmq --enable-librist --enable-lbsrt --enable-libssh --enabl
e-lbsvtav1 --enable-libx264 --enable-libzmq --enable-libzvti --enable-lv2 --enable-sdl2 --enable-libplacebo --enable-librav1e --enabl
e-pocketsphinx --enable-librsvg --enable-libjxl --enable-shared
  libavutil      58. 29.100 / 58. 29.100
  libavcodec     60. 31.102 / 60. 31.102
  libavformat    60. 16.100 / 60. 16.100
  libavdevice    60.  3.100 / 60.  3.100
  libavfilter     9. 12.100 /  9. 12.100
  libswscale     7.  5.100 /  7.  5.100
  libswresample  4. 12.100 /  4. 12.100
  libpostproc   57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'Trailer_Hackerlab2024_challenge.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf60.3.100
  Duration: 00:01:09.87, start: 0.000000, bitrate: 929 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(progressive), 1280x720 [SAR 1:1 DAR 16:9], 784 kb/s, 60 fps,
```

Etape 5

```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge

  libswscale     7.  5.100 /  7.  5.100
  libswresample  4. 12.100 /  4. 12.100
  libpostproc   57.  3.100 / 57.  3.100
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'Trailer_Hackerlab2024_challenge.mp4':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf60.3.100
  Duration: 00:01:09.87, start: 0.000000, bitrate: 929 kb/s
  Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(progressive), 1280x720 [SAR 1:1 DAR 16:9], 784 kb/s, 60 fps,
60 tbr, 15360 tbn (default)
  Metadata:
    handler_name     : VideoHandler
    vendor_id        : [0][0][0][0]
    encoder          : Lavc60.3.100 libx264
  Stream #0:1[0x2](und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 131 kb/s (default)
  Metadata:
    handler_name     : SoundHandler
    vendor_id        : [0][0][0][0]
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> png (native))
Press [q] to stop, [?] for help
Output #0, image2, to 'resolve/%.png':
  Metadata:
    major_brand      : isom
    minor_version    : 512
    compatible_brands: isomiso2avc1mp41
    encoder          : Lavf60.16.100
  Stream #0:0(und): Video: png, rgb24(pc, gbr/unknown/unknown, progressive), 1280x720 [SAR 1:1 DAR 16:9], q=2-31, 200 kb/s, 60 fps, 60
tbn (default)
  Metadata:
    handler_name     : VideoHandler
    vendor_id        : [0][0][0][0]
    encoder          : Lavc60.31.102 png
frame= 2871 fps= 31 q=-0.0 size=N/A time=00:00:47.85 bitrate=N/A speed=0.509x
```

Etape:6

c'est la fin du script

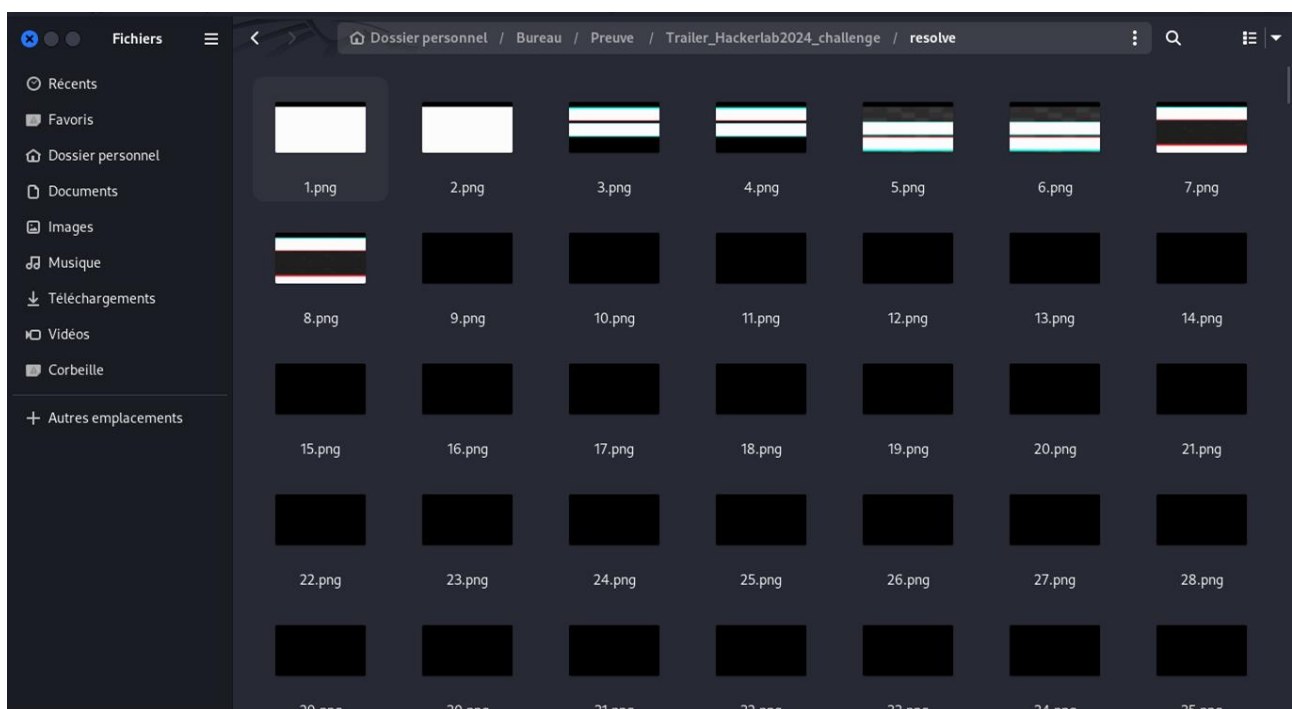
```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge

major_brand      : isom
minor_version    : 512
compatible_brands: isomiso2avc1mp41
encoder          : Lavf60.3.100
Duration: 00:01:09.87, start: 0.000000, bitrate: 929 kb/s
Stream #0:0[0x1](und): Video: h264 (High) (avc1 / 0x31637661), yuv420p(progressive), 1280x720 [SAR 1:1 DAR 16:9], 784 kb/s, 60 fps, 60 tbr, 15360 tbn (default)
Metadata:
  handler_name    : VideoHandler
  vendor_id       : [0][0][0][0]
  encoder         : Lavc60.3.100 libx264
Stream #0:1[0x2](und): Audio: aac (LC) (mp4a / 0x6134706D), 44100 Hz, stereo, fltp, 131 kb/s (default)
Metadata:
  handler_name    : SoundHandler
  vendor_id       : [0][0][0][0]
Stream mapping:
  Stream #0:0 -> #0:0 (h264 (native) -> png (native))
Press [q] to stop, [?] for help
Output #0, image2, to 'resolve/%d.png':
Metadata:
  major_brand      : isom
  minor_version    : 512
  compatible_brands: isomiso2avc1mp41
  encoder          : Lavf60.16.100
Stream #0:0(und): Video: png, rgb24(pc, gbr/unknown/unknown, progressive), 1280x720 [SAR 1:1 DAR 16:9], q=2-31, 200 kb/s, 60 fps, 60 tbn (default)
Metadata:
  handler_name    : VideoHandler
  vendor_id       : [0][0][0][0]
  encoder         : Lavc60.31.102 png
[out#0/image2 @ 0x5627e6335400] video:1743221kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: unknown
frame= 4192 fps= 38 q=-0.0 Lsize=N/A time=00:01:09.85 bitrate=N/A speed=0.627x

(kali@kali)~-[~/Vidéos/Trailer_Hackerlab2024_challenge]
```

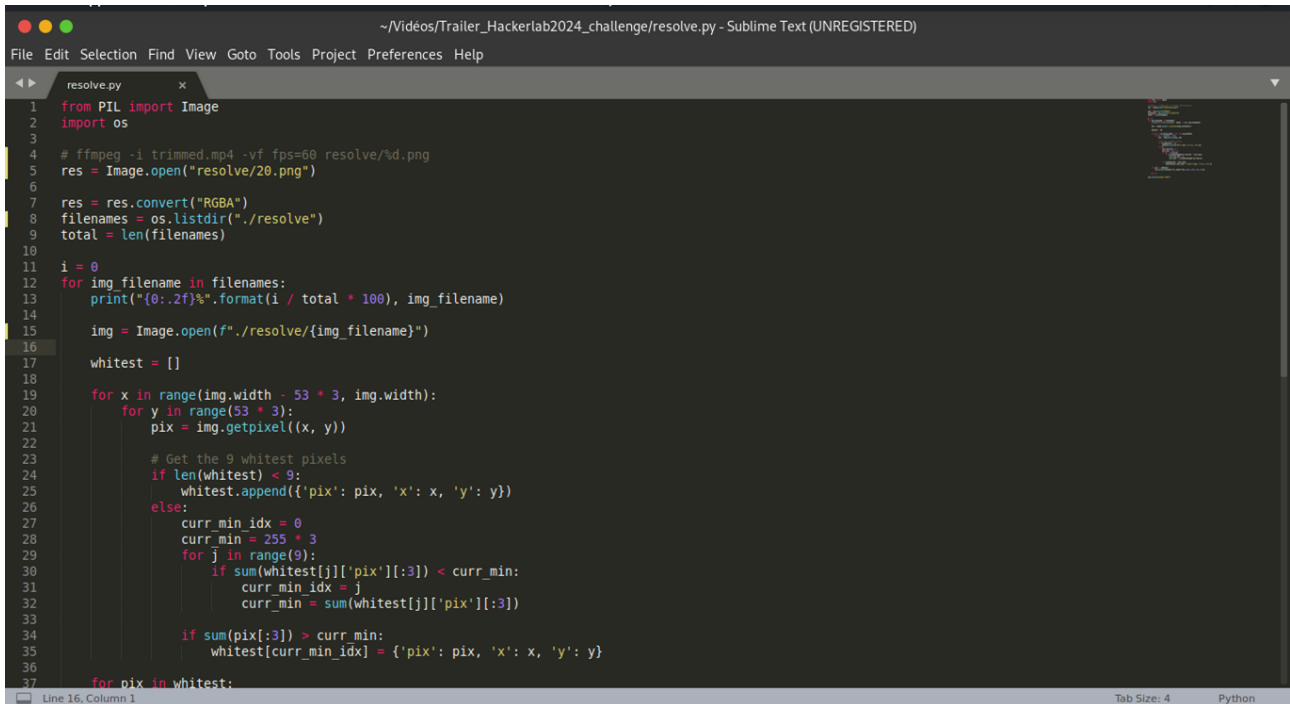
Etape :7

J'ai trouvé beaucoup d'images comme des captures d'ecrans



Etape :8

J'ai beaucoup réfléchi à comment faire et après quelques minutes de réflexion je trouve un script python qui m'aidera à regrouper les pixels blancs là



```
1 from PIL import Image
2 import os
3
4 # ffmpeg -i trimmed.mp4 -vf fps=60 resolve/%d.png
5 res = Image.open("resolve/20.png")
6
7 res = res.convert("RGB")
8 filenames = os.listdir("./resolve")
9 total = len(filenames)
10
11 i = 0
12 for img_filename in filenames:
13     print("{0:.2f}%".format(i / total * 100), img_filename)
14
15     img = Image.open(f"./resolve/{img_filename}")
16
17     whitest = []
18
19     for x in range(img.width - 53 * 3, img.width):
20         for y in range(53 * 3):
21             pix = img.getpixel((x, y))
22
23             # Get the 9 whitest pixels
24             if len(whitest) < 9:
25                 whitest.append({'pix': pix, 'x': x, 'y': y})
26             else:
27                 curr_min_idx = 0
28                 curr_min = 255 * 3
29                 for j in range(9):
30                     if sum(whitest[j]['pix'][:3]) < curr_min:
31                         curr_min_idx = j
32                         curr_min = sum(whitest[j]['pix'][:3])
33
34                 if sum(pix[:3]) < curr_min:
35                     whitest[curr_min_idx] = {'pix': pix, 'x': x, 'y': y}
36
37     for pix in whitest:
```

J'ai pris un fond noir pour que il n'y a pas de problème après la récupération de la nouvelle image car j'ai pris un fond blanc regarder ceux que j'avais comme problème



Je ne peux pas lire l'image car je vois que c'est impossible de le lire c'est pour cela que j'ai pris en final le noir.

Etape:9

Execution du script python

```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge

(kali@kali)~$ python3 resolve.p
python3: can't open file '/home/kali/Vidéos/Trailer_Hackerlab2024_challenge/resolve.p': [Errno 2] No such file or directory

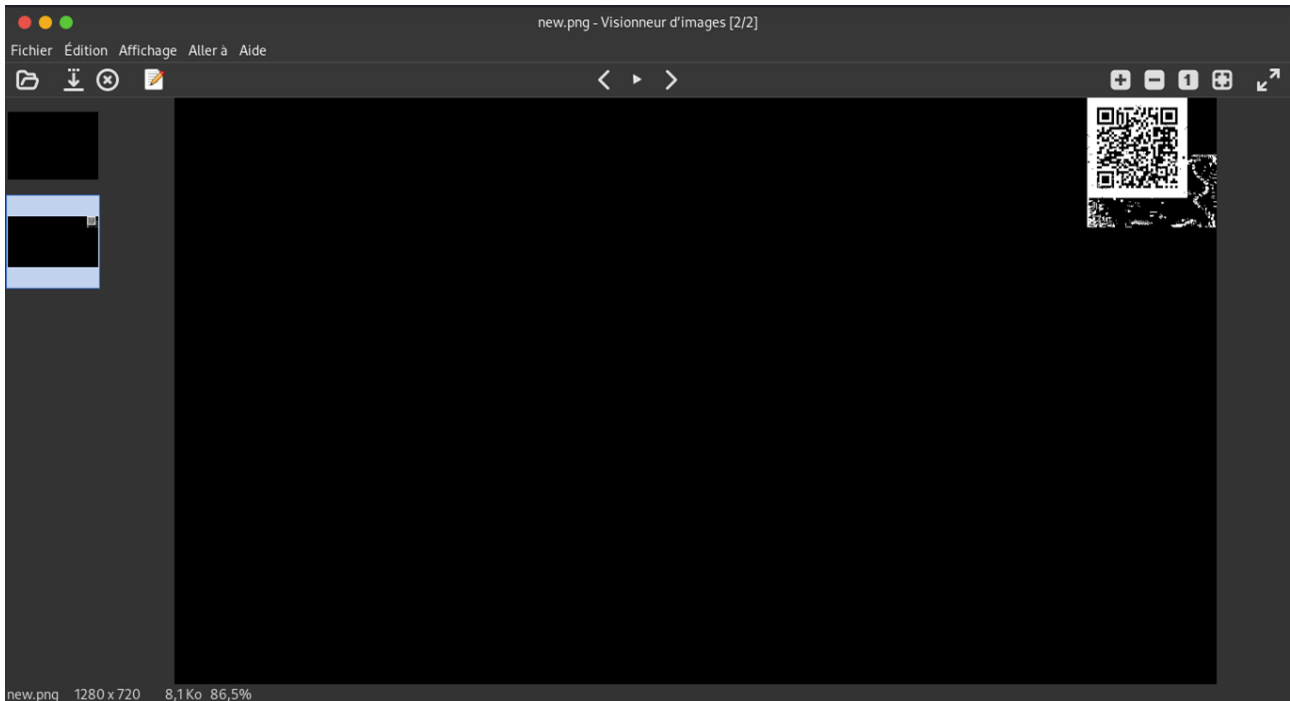
(kali@kali)~$ python3 resolve.py
0.00% 3417.png
0.02% 1567.png
0.05% 2105.png
0.07% 3972.png
0.10% 2518.png
0.12% 2335.png
0.14% 3441.png
0.17% 3001.png
0.19% 1343.png
0.21% 2981.png
0.24% 3687.png
0.26% 284.png
0.29% 3603.png
0.31% 3823.png
0.33% 90.png
0.36% 127.png
0.38% 1033.png
0.41% 414.png
0.43% 3471.png
0.45% 947.png
0.48% 621.png
0.50% 2434.png
0.52% 2905.png
0.55% 1417.png
0.57% 504.png
0.60% 1885.png
0.62% 2760.png
[]
```

```
kali@kali: ~/Vidéos/Trailer_Hackerlab2024_challenge

99.14% 3320.png
99.17% 97.png
99.19% 568.png
99.21% 1017.png
99.24% 2222.png
99.26% 2999.png
99.28% 2516.png
99.31% 2938.png
99.33% 3797.png
99.36% 924.png
99.38% 1093.png
99.40% 2985.png
99.43% 2037.png
99.45% 1509.png
99.48% 210.png
99.50% 2480.png
99.52% 291.png
99.55% 1910.png
99.57% 308.png
99.59% 273.png
99.62% 2534.png
99.64% 3221.png
99.67% 3444.png
99.69% 3667.png
99.71% 2508.png
99.74% 1706.png
99.76% 115.png
99.79% 1131.png
99.81% 1969.png
99.83% 608.png
99.86% 3876.png
99.88% 982.png
99.90% 3047.png
99.93% 3837.png
99.95% 270.png
99.98% 953.png
```

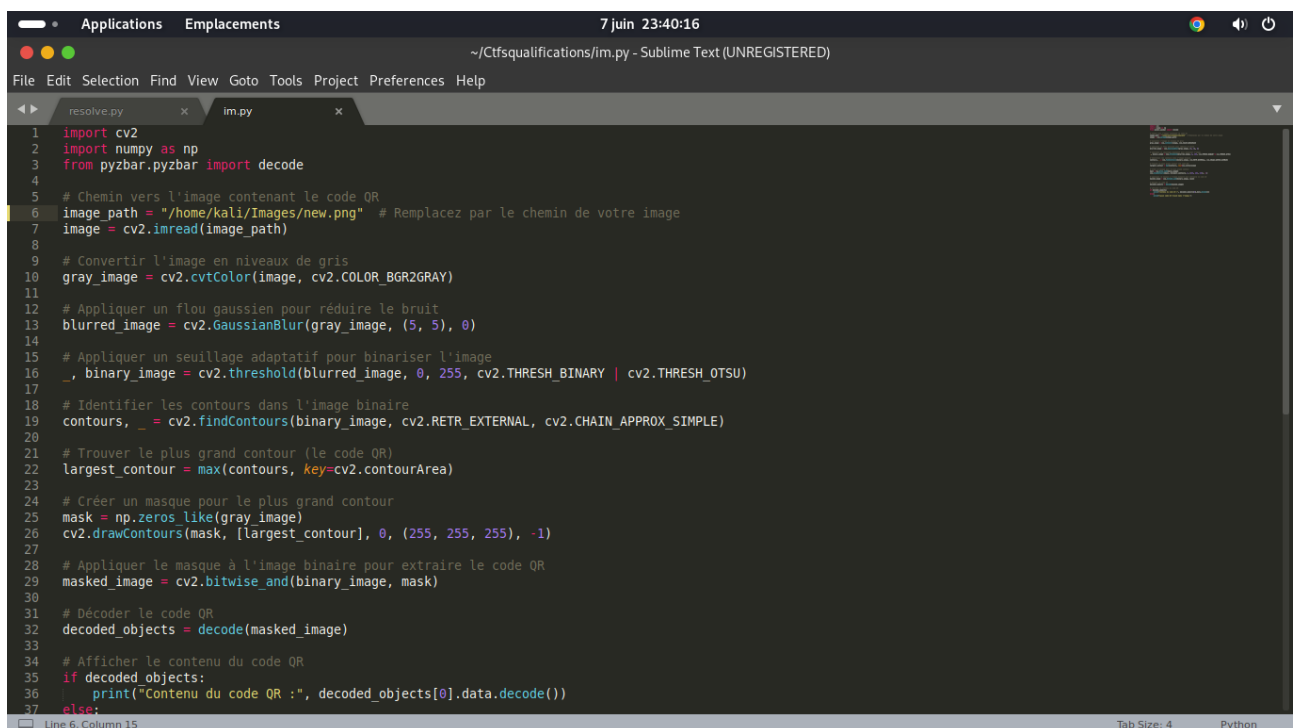
Etape:10

J'obtiens la nouvelle image et c'est lisible car c'est du code qr



Je me dis en même temps que c'est la fin du challenge que c'est sûr que le flag c'est le code QR mais `□□□□□` c'est maintenant que le travail veut bien commencer. J'écris vite un code python pour lire le code QR

Etape:11



et voici le résultat

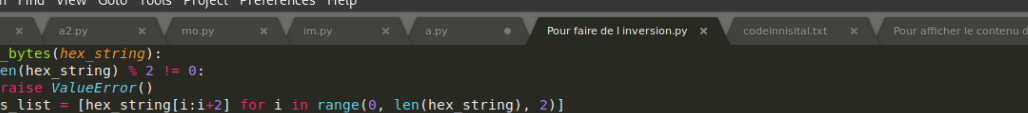
```
(kali㉿kali)-[~/Ctfsqualifications]
$ python3 im.py
Contenu du code QR : https://justpaste.it/1nhcv

(kali㉿kali)-[~/Ctfsqualifications]
$
```

Je me suis dit que le flag est dans le lien et je clique suis le lien mais à ma grand surprise c'est encore du boulot à faire encore

[illegible]

Hum j'ai vu beaucoup de données je me suis dit en même temps que c'est du hex qu'avec ça je peux avoir l'image mais c'est pas possible. Après avoir faire des recherche suis Google je trouve que mon **RAW DATA IMAGE** est un peu comme du jpeg Ex de format jpeg : **FF D8** mais avec mon RAW DATA IMAGE je constate que les données sont inversées de deux à deux .Je me suis dit en même temps pourquoi pas le faire avec python car c'est impossible de le faire à la main



```
def swap_bytes(hex_string):
    if len(hex_string) % 2 != 0:
        raise ValueError()
    bytes_list = [hex_string[i:i+2] for i in range(0, len(hex_string), 2)]
    swapped_bytes_list = []
    for i in range(0, len(bytes_list), 2):
        if i+1 < len(bytes_list):
            swapped_bytes_list.append(bytes_list[i+1])
            swapped_bytes_list.append(bytes_list[i])
    swapped_hex_string = ''.join(swapped_bytes_list)
    return swapped_hex_string

hex_string = "d8ffe0ff1000464a4649010000101000100000bfff4300030002020203020203020304030406040404040608050609060a08090a090"
swapped_hex_string = swap_bytes(hex_string)
print(swapped_hex_string)
```


Etape 12

[illegible]

Etape 13

j'ai encore écrire un code pour lire l'image

```
from io import BytesIO
from PIL import Image
import imghdr

def lire_image_de_hex(hex_image_string):
    # Convertir la chaîne hexadécimale en données binaires
    image_data = bytes.fromhex(hex_image_string)

    # Créer un flux d'octets à partir des données binaires
    image_stream = BytesIO(image_data)

    # Tenter de déterminer le format de l'image
    image_format = imghdr.what(image_stream)
    if not image_format:
        raise ValueError("Impossible de déterminer le format de l'image à partir de la chaîne hexadécimale fournie.")

    # Réinitialiser la position du flux d'octets à 0
    image_stream.seek(0)

    # Ouvrir l'image à partir du flux d'octets
    image = Image.open(image_stream)

    return image

# Chaîne hexadécimale de l'image (remplacez-la par votre propre chaîne)
hex_image_string = "ffd8ffe000104a46494600010100000100010000ffdb00430003020202020302030303040604040404080606050609080a0"
image = lire_image_de_hex(hex_image_string)
# Afficher l'image
image.show()
```



```
2108401117d900
(kali@kali)-[~/Bureau/Preuve ]
$
(kali@kali)-[~/Bureau/Preuve ]
$
(kali@kali)-[~/Bureau/Preuve ]
$ python3 Pour\ afficher\ le\ contenu\ du\ code\ hex.py
/home/kali/Bureau/Preuve /Pour afficher le contenu du code hex.py:3: DeprecationWarning: 'imghdr' is deprecated and slated for removal in Python 3.13
import imghdr
(kali@kali)-[~/Bureau/Preuve ]
$
(ristretto:2850): Gtk-WARNING **: 17:01:28.117: Theme file for WhiteSur-Dark has no name
(ristretto:2850): Gtk-WARNING **: 17:01:28.117: Theme file for WhiteSur-Dark has no directories
█
```

Etape 14 : j'ai une image qui est du code QR mais une partie est illisible



J'ai bien regarder le code QR car il à la main suis la partie gauche du code QR mais j'ai beaucoup cherche comment lire le code QR c'est là j'ai trouvé que je peux prendre une des carrées du haut en venue placée ça au niveau de la partie gauche et bigo bigo j'ai réussir le challenge !!!!!!! 😄😄😄 ☐☐☐ 😊😞



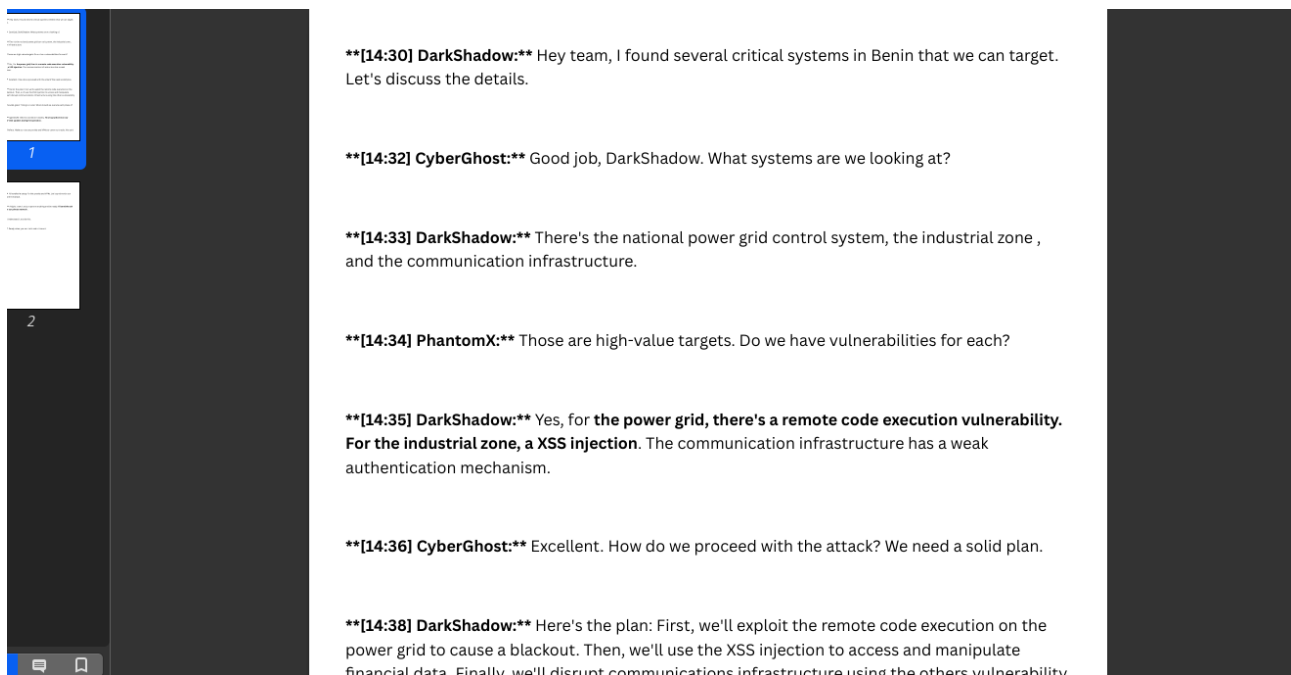
je le lire avec un outils de lecture de code QR en ligne



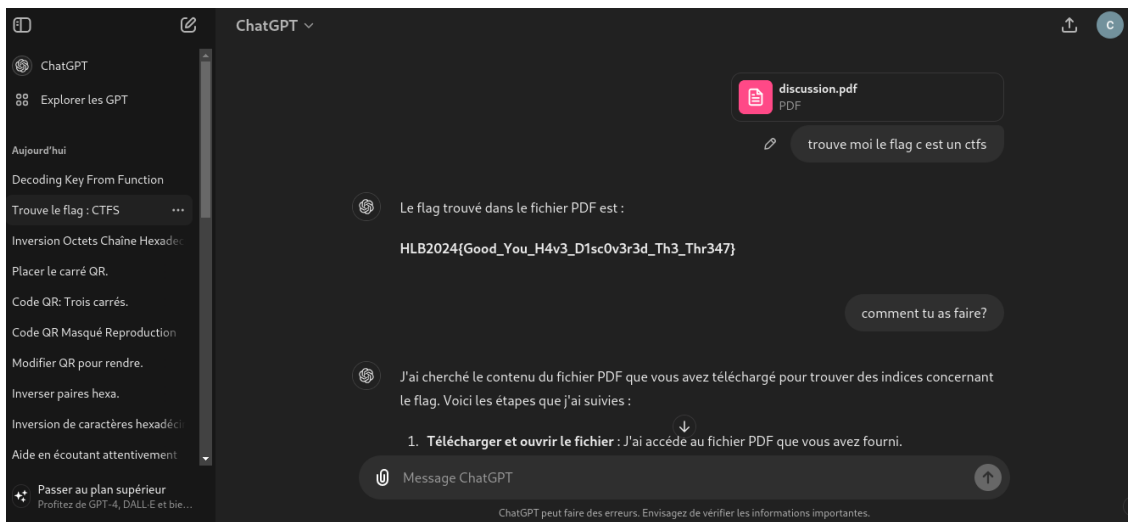
Merde merde je me dis encore donc ça me reste encore du travail à faire car je peut pas avoir les 500points sans beaucoup travailler quand j ai vu un lien mega file

Etape : 15

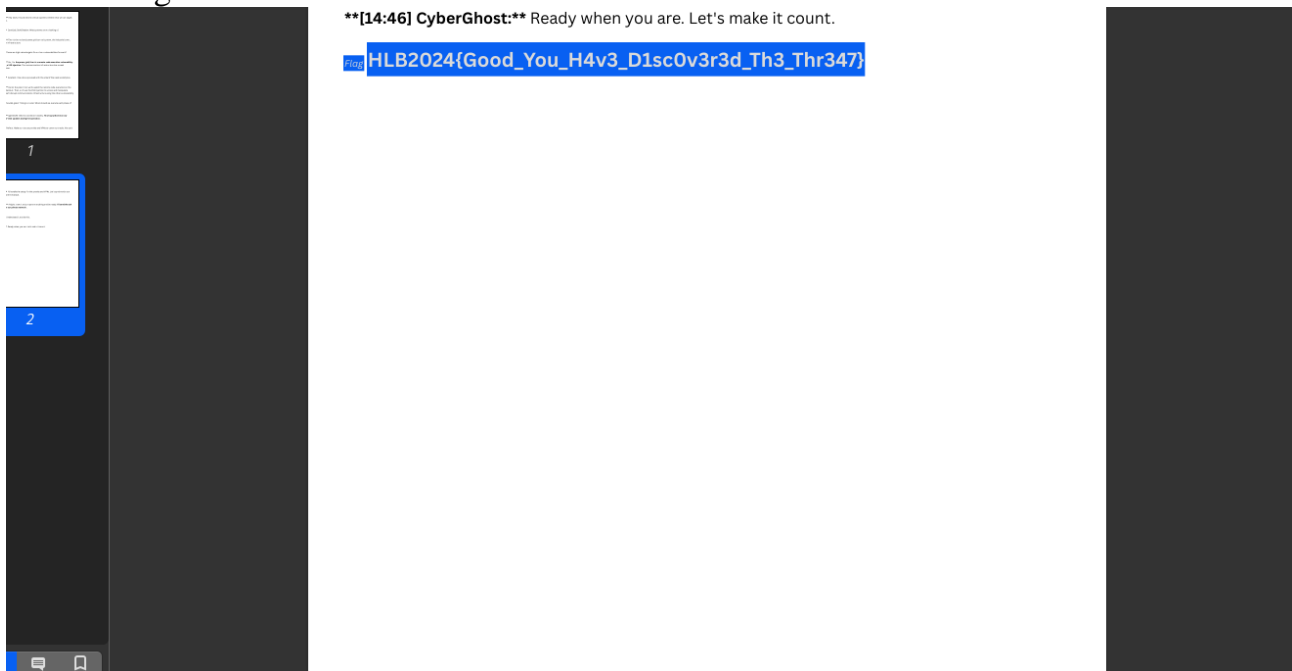
J'ai télécharger le fichier mega file la mais c'est du PDF



Comme moi en anglais je comprend rien et je suis déjà fatiguer du challenge j ai demander de l aide à gpt



Mais le flag c etait facile à trouver car le flag était en fond blanc donc on doit selectionner tout le texte d abord j avais pas remarquée cela car j avais ouvrir le pdf dans chrome mais voici le flag



Vraiment le challenge à été vraiment dure pour moi car quand tu termine une partie tu t attends à la fin mais ça te reste encore comme le mot anglais que j ai traduire qui dit cherche encore.....Vraiment chapeau au créateur de ceux challenge **W1z4rd**

Writeup ecris par [foundhack](#)