# Project Black CTFs Challenge 2

https://projectblack.io/ctf/challenge2.txt

FLAGS:
PROJECTBLACK{Flag_1of4:_U_R_0fF_2_A_Gr3aT_$T@RT}
PROJECTBLACK{Flag_2of4:_1f_pYTh0N3_1$_s0_g00D,_WH3r3_Iz_PyTH0n4?}
PROJECTBLACK{Flag_3of4:_G0tT@_c@tch_Th3m_@11!_0n3_M0rE_2_G0!}
PROJECTBLACK{Flag_4of4:_C0nGr@t$!_U_R_@n_3l1t3_h@ck3r!!1!}

- Noticed = at the end of the text so likely base64 encoding.

PROJECTBLACK{Flag_1of4:_U_R_0fF_2_A_Gr3aT_$T@RT}

- Decode base64 first portion of text.

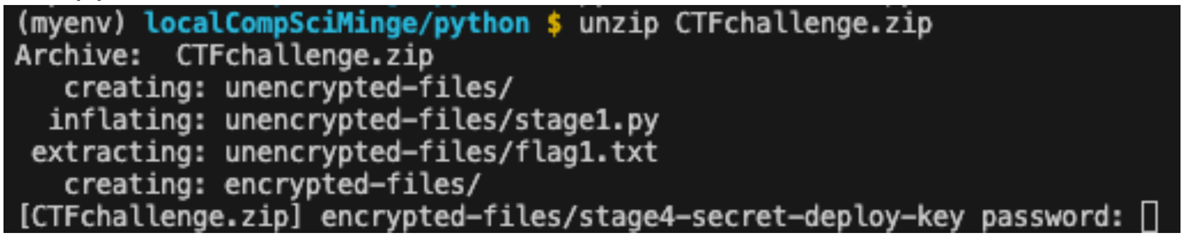
- Noticed PK in the decoded text. ZIP files.
- Wrote this script to write a CTFchallenge.zip to then later unzip.

```
import base64
import requests

content = requests.get("https://projectblack.io/ctf/challenge2.txt").text

with open("CTFchallenge.zip", "wb") as file:
    file.write(base64.b64decode(content))
```

- Unzipped files:



```
(myenv) localCompSciMinge/python $ unzip CTFchallenge.zip
Archive:  CTFchallenge.zip
   creating: unencrypted-files/
  inflating: unencrypted-files/stage1.py
 extracting: unencrypted-files/flag1.txt
   creating: encrypted-files/
[CTFchallenge.zip] encrypted-files/stage4-secret-deploy-key password:
```

- 
- Looked through and found in stage1.py: ENCRYPTED_PASSWORD = [
- 0b01010100,
- 0b01101000,
- 0b00110001,
- 0b00100100,
- 0b01011111,
- 0b00110001,
- 0b00100100,
- 0b01011111,
- 0b01101101,
- 0b01011001,
- 0b01011111,
- 0b00100100,
- 0b00110011,
- 0b01100011,

- 0b01010010,
- 0b01000101,
- 0b01110100,
- 0b01011111,
- 0b01110000,
- 0b01000000,
- 0b00100100,
- 0b00100100,
- 0b01110111,
- 0b00110000,
- 0b01110010,
- 0b01000100,
- ]

which is binary. Also found: expected_password = ''.join(map(chr, ENCRYPTED_PASSWORD)). So realised that its converted from binary to int and then to char.

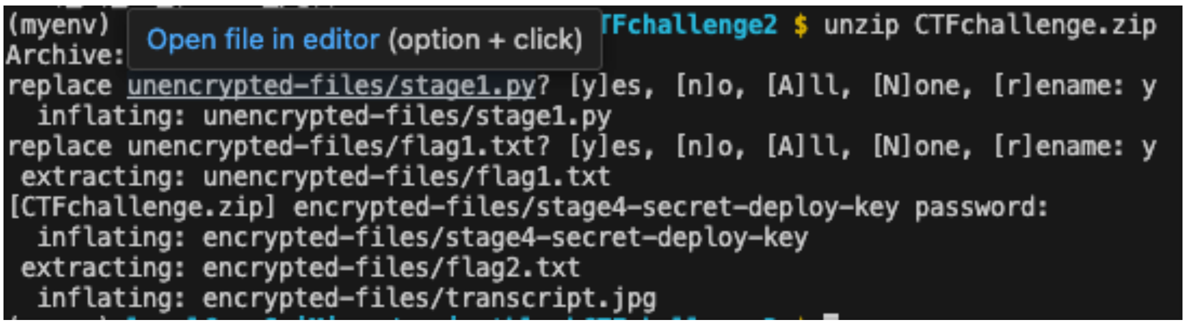
- Created script to decode that:

```
ENCRYPTED_PASSWORD_BINARY = [
    0b01010100,
    0b01101000,
    0b00110001,
    0b00100100,
    0b01011111,
    0b00110001,
    0b00100100,
    0b01011111,
    0b01101101,
    0b01011001,
    0b01011111,
    0b00100100,
    0b00110011,
    0b01100011,
    0b01010010,
    0b01000101,
    0b01110100,
    0b01011111,
    0b01110000,
    0b01000000,
    0b00100100,
    0b00100100,
    0b01110111,
    0b00110000,
    0b01110010,
    0b01000100,
]
```

```
ENCRYPTED_PASSWORD = []
for b in ENCRYPTED_PASSWORD_BINARY:
    ENCRYPTED_PASSWORD.append(chr((int(b))))

encrypted_password = ''.join(ENCRYPTED_PASSWORD)
print(encrypted_password)
```

- Decoding that gave me the password: Th1$_1$_mY_$3cREt_p@$$w0rD



```
(myenv) Open file in editor (option + click) TFchallenge2 $ unzip CTFchallenge.zip
Archive:
replace unencrypted-files/stage1.py? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: unencrypted-files/stage1.py
replace unencrypted-files/flag1.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
 extracting: unencrypted-files/flag1.txt
[CTFchallenge.zip] encrypted-files/stage4-secret-deploy-key password:
  inflating: encrypted-files/stage4-secret-deploy-key
 extracting: encrypted-files/flag2.txt
  inflating: encrypted-files/transcript.jpg
```

- Got the second flag by looking at flag2.txt

- Now need to ssh to remote system with private key
- Followed steps in the transcript.jpg and got the third flag.

- Followed this link also in the README: https://stage4-pb.github.io/
- Found the page with rows of dots and two buttons: correct and wrong.
  - Clicked buttons but nothing shows up on burp suite so no types of requests are sent out.
- Created an autoclicker to click "correct" button 5344 (number of dots) times to see if anything would happen but only empty.
- Assuming have to answer questions correctly to proceed.
- Tried getting html to create script to answer questions and click button simultaneously but body of the html from script does not have questions or rows of dots for some reason.

- Realised page likely using client-side rendering and html content is populated when JS executed.
- Used selenium to execute JS.
- Automated process of adding questions everytime new question is rendered to text file.
- Created autoclicker that would manually click the "correct" button when question is correct or "wrong" when question is wrong.
- Finished clicking but no flag.

- Realised the rows could be a binary message so wrote script to decode the binary message and got this:

IF9fX19fX19fX19fX19fX19fX19fX19fX19fX19fX19fX19fX19fX19fX19fX1
9fX19fX19fX19fX19fXwovIENvbmdyYXR1bGF0aW9ucywgeW91IGhhdm
UgZm91bmQgYWxsIGZvdXIgZmxhZ3MhIFlvdXIgbGFzdCAgXAp8IGZs
YWcgaXM6ICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgfAp8ICAgICAgICAgICAgICAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgfApcIFBST0pFQ1RC
TEFDS3tGbGFnXzRvZjQ6X0MwbkdyQHQkIV9VX1JfQG5fM2wxdDNfaE
BjazNyISExIX0gLwogLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS
0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tLS0tCiAgICAgICAgXCAgIF5fX1
4KICAgICAgICAgXCAgKG9vKVxfX19fX19fCiAgICAgICAgICAgIChfXylcI
CAgICAgIClcL1wKICAgICAgICAgICAgICAgIHx8LS0tLXcgfAogICAgICAgI
CAgICAgICAgfHwgICAgIHx8Cg==

- Looked like a strange base64 encoding given away by the padding = at the end.
- Decoded it and got the fourth flag:

```
 _____________________________________________________________
/ Congratulations, you have found all four flags! Your last  \
| flag is:                                                    |
|                                                             |
\ PROJECTBLACK{Flag_4of4:_C0nGr@t$!_U_R_@n_3l1t3_h@ck3r!!1!} /
 -------------------------------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```