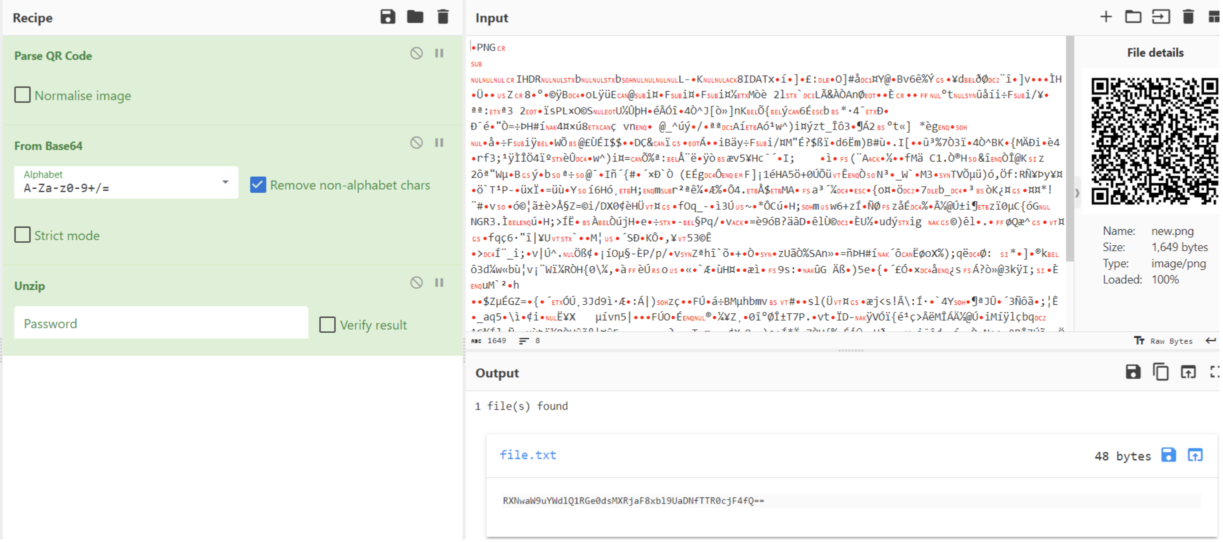
CTF solution

Stage 1: Obfuscated Python code
serial-number (currentYear + MD5 hash value for B2C3D4E5.DAT): 2024-ba5d5a8219784516fb3a1043bd093e25
password for the exe file (QR code Loader - B2C3D4E5.DAT AES encrypted):
Str34mC1ph3r_0v3rl04d

The player needs to reverse engineer the QDecryptor.exe to get the XORed key, command line argument, and how to use it. They also need to identify the correct file from the rest, which can be done by looking at the number in the encrypted file header. The Loader uses a less-known Win32 API function called SystemFunction032 for RC4 decryption.

```
F:\Projects\CTF\Player_file>QDecryptor.exe -d -i 2C3D4E5F.DAT -o new.png -k
Encrypt0r_K3yMast3r!
[*] Filename: 2C3D4E5F.DAT
[*] Encryption key: Encrypt0r_K3yMast3r!
[*] output filename: new.png
[*] Start Writing 1649 bytes to new.png.
[*] File has been saved successfully at new.png
[OK] Process has been done. Exit.
```

Then the player should Parse the QR code. This can be done via Python or Cyberchief.



Parse QR Code
☐ Normalise image

From Base64
Alphabet: A-Za-z0-9+/=
☑ Remove non-alphabet chars
☐ Strict mode

Unzip
Password
☐ Verify result

Name: new.png
Size: 1,649 bytes
Type: image/png
Loaded: 100%

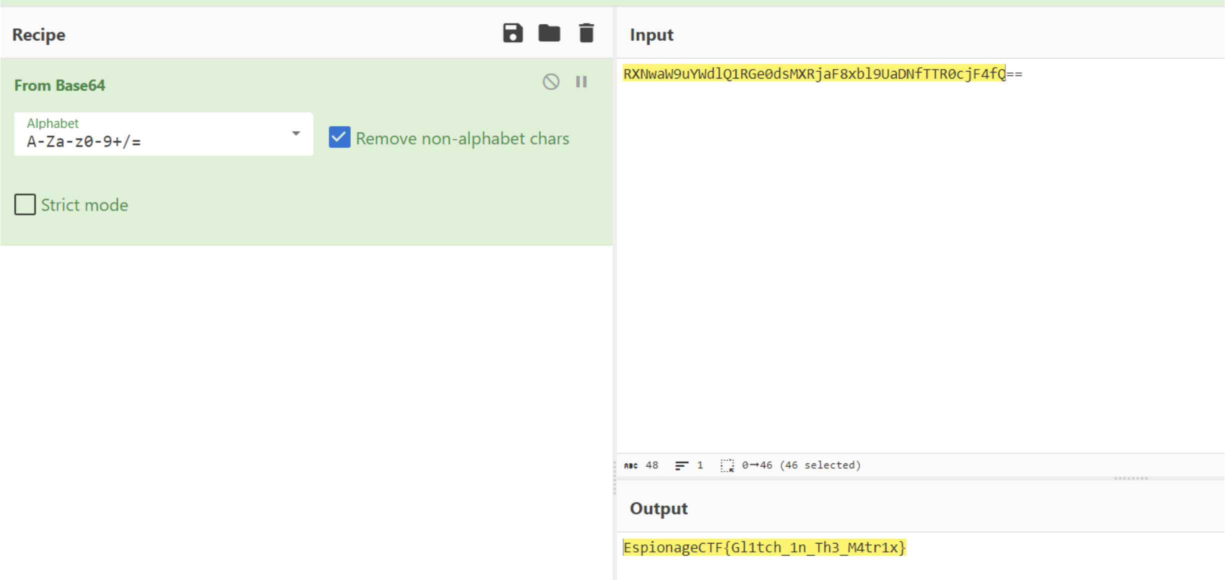
1649 8

```
1 file(s) found

file.txt                                                    48 bytes

RXNwaW9uYWdlQ1RGe0dsMXRjaF8xbl9UaDNfTTR0cjF4fQ==
```

The final will be file.txt inside this QR code:
it contains base64 flag:



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars

□ Strict mode

Input

```
RXNwaW9uYWdlQ1RGe0dsMXRjaF8xbl9UaDNfTTR0cjF4fQ==
```

```
48   1   0→46 (46 selected)
```

Output

```
EspionageCTF{Gl1tch_1n_Th3_M4tr1x}
```

EspionageCTF{Gl1tch_1n_Th3_M4tr1x}