EG CTF 2023

Category: FORENSICS

Challenge name: AutoBot

Points: 566

Challenge description:

Once upon a time, in a distant land, a group of archaeologists were on a dig in search of ancient artifacts. As they were sifting through the dirt, they suddenly came across something metallic and shiny. As they dug deeper, they realized that it was a robot!

Many parts of the robot were missing. Only 3 was found. Still, it is corrupted. Can you fix it?

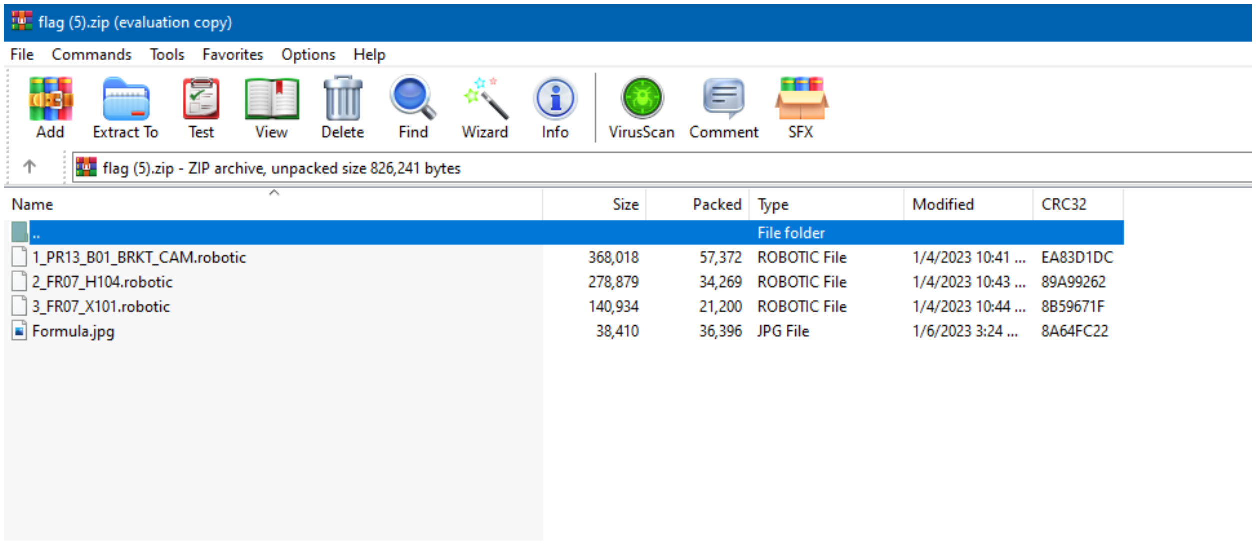
Note: Submit flag with EG{}

Given Files: flag.zip

Given Hints: none

SOLUTION

Let’s start by downloading the file from the challenge and unzipped it.



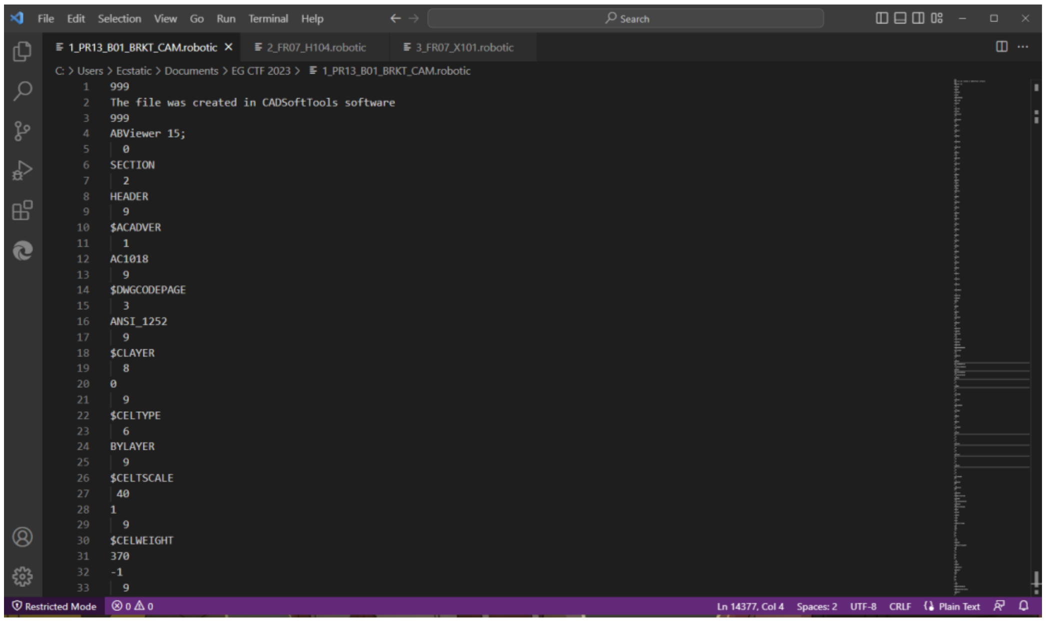
flag (5).zip (evaluation copy)

flag (5).zip - ZIP archive, unpacked size 826,241 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
1_PR13_B01_BRKT_CAM.robotic	368,018	57,372	ROBOTIC File	1/4/2023 10:41 ...	EA83D1DC
2_FR07_H104.robotic	278,879	34,269	ROBOTIC File	1/4/2023 10:43 ...	89A99262
3_FR07_X101.robotic	140,934	21,200	ROBOTIC File	1/4/2023 10:44 ...	8B59671F
Formula.jpg	38,410	36,396	JPG File	1/6/2023 3:24 ...	8A64FC22

Figure 1 Files in zip

It appears that it’s an unusual type of file (**.robotic**). In order to further analyze it I’m going to open it in a text editor.



C: > Users > Ecstatic > Documents > EG CTF 2023 > 1_PR13_B01_BRKT_CAM.robotic

```
999
The file was created in CADSoftTools software
999
ABViewer 15;
  0
SECTION
  2
HEADER
  9
$ACADVER
  1
AC1018
  9
$DWGCODEPAGE
  3
ANSI_1252
  9
$CLAYER
  8
0
  9
$CELTYPE
  6
BYLAYER
  9
$CELTSCALE
 40
1
  9
$CELWEIGHT
370
-1
  9
```

Figure 2 In text view

Next, I would just try to find the flag, so I pressed Ctrl+F to bring up the search window and type in “**EG**”

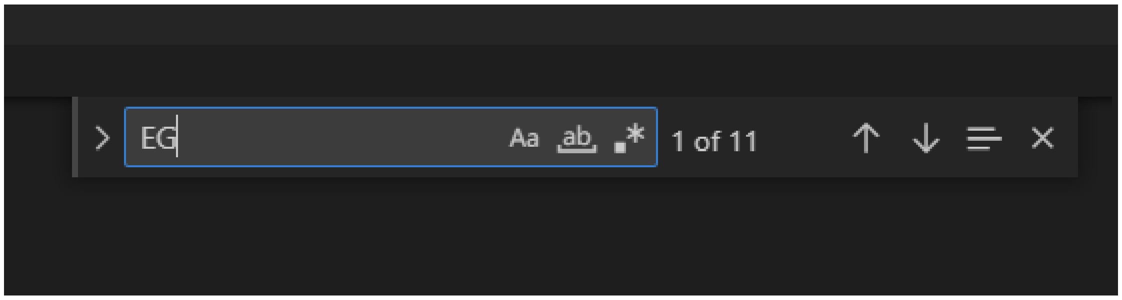
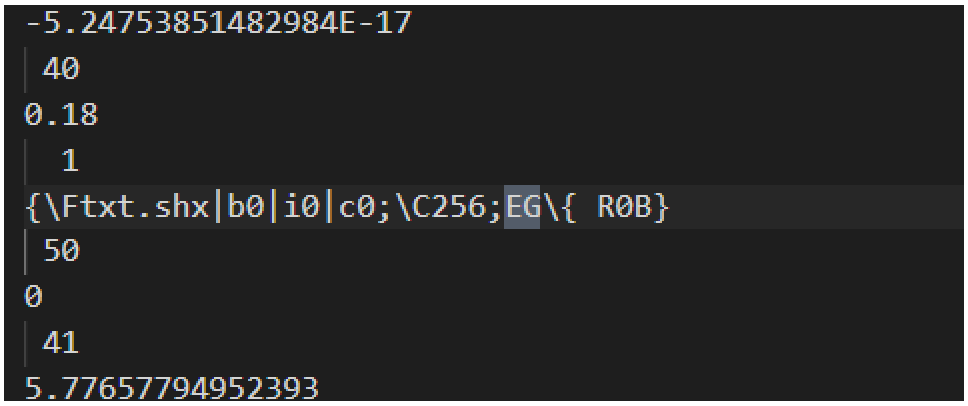


Figure 3 Find the flag

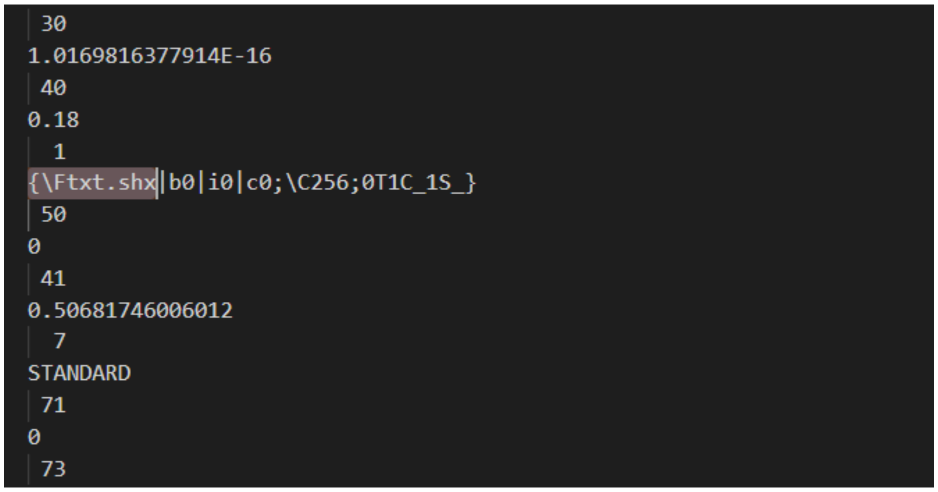
After scrolling down, we eventually found this, which is close to our flag format but shorter. From the description it says “***Many parts of the robots are missing. Only 3 has been found***”. This probably means that the flag is separated into those 3 files in the zip that we found earlier.



```
-5.24753851482984E-17
 40
0.18
  1
{\Ftxt.shx|b0|i0|c0;\C256;EG\{ R0B}
 50
0
 41
5.77657794952393
```

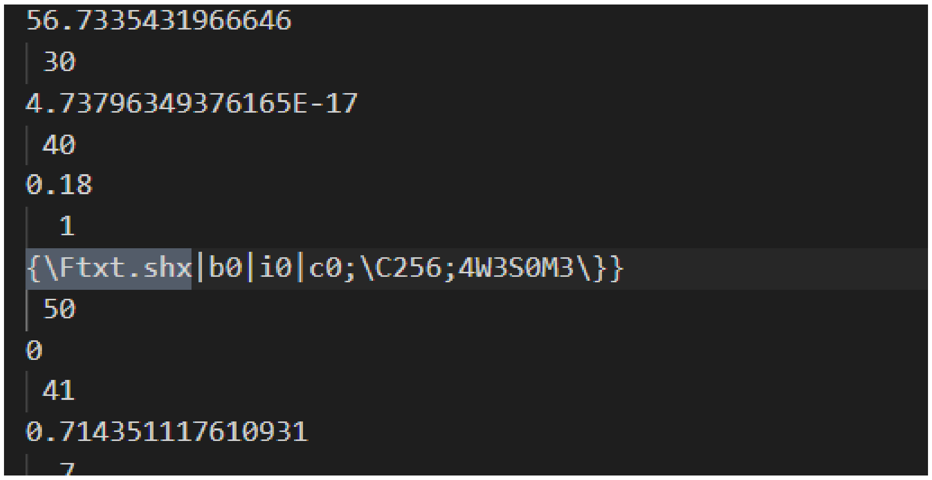
Figure 4 Flag?

At this point onwards the flag probably resides in the same text format so in the 2nd file, I typed in “***{\Ftxt.shx***” in the search window which led us straight to another portion of the flag. We’re going to do the same for the 3rd file.



```
 30
1.0169816377914E-16
 40
0.18
  1
{\Ftxt.shx|b0|i0|c0;\C256;0T1C_1S_}
 50
0
 41
0.50681746006012
  7
STANDARD
 71
0
 73
```

Figure 5 2nd portion of the flag



```
56.7335431966646
 30
4.73796349376165E-17
 40
0.18
  1
{\Ftxt.shx|b0|i0|c0;\C256;4W3S0M3\}}
 50
0
 41
0.714351117610931
```

Figure 6 Last portion of the flag

Combine all of them and you will get:

EG{R0B0T1C_1S_4W3S0M3}

Challenge name: Broken Oyen

Points: 616

Challenge description:

Oyen falling in love with a White cat. After a long journey Oyen lost its memory. Can u help oyen recover its memory!!!?

~ Not 8 Character, but 9! ~

Note: Submit flag with EG{}

Given Files: cat.zip

Given Hints:

1. **pass (----^%%%%) 9 char 👍**

SOLUTION

Let’s start with downloading the files given and unzipped it. We would get 3 files, one zip file and 2 jpeg files.

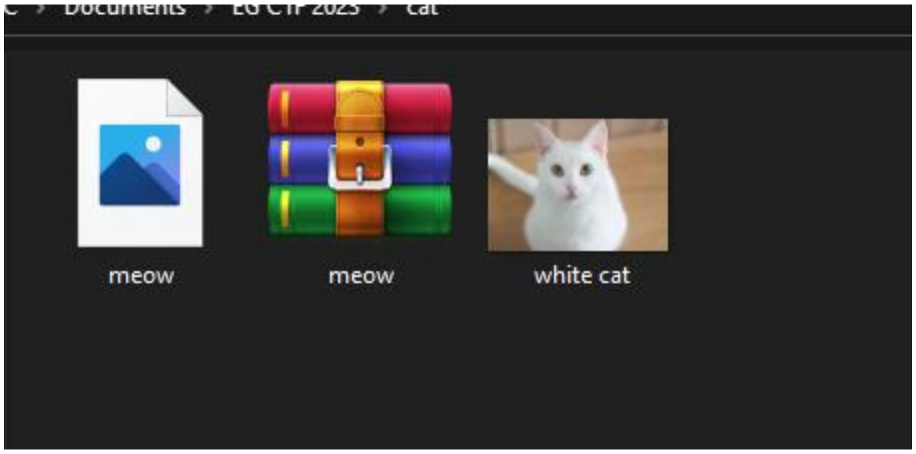
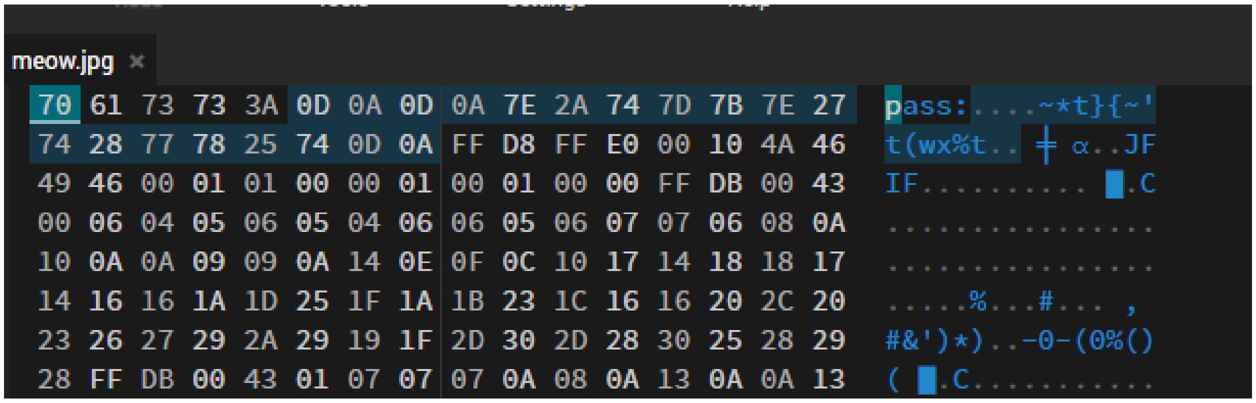


Figure 7 cat files

I’m going to analyze the “**meow.jpeg**” 1st and it seems like the file is corrupted. Therefore, I’m going to use a hex editor to check the headers.



```
70 61 73 73 3A 0D 0A 0D 0A 7E 2A 74 7D 7B 7E 27   pass:....~*t}{~'
74 28 77 78 25 74 0D 0A FF D8 FF E0 00 10 4A 46   t(wx%t..╪ α..JF
49 46 00 01 01 00 00 01 00 01 00 00 FF DB 00 43   IF..........█.C
00 06 04 05 06 05 04 06 06 05 06 07 07 06 08 0A   ................
10 0A 0A 09 09 0A 14 0E 0F 0C 10 17 14 18 18 17   ................
14 16 16 1A 1D 25 1F 1A 1B 23 1C 16 16 20 2C 20   .....%...#... ,
23 26 27 29 2A 29 19 1F 2D 30 2D 28 30 25 28 29   #&')*)..-0-(0%()
28 FF DB 00 43 01 07 07 07 0A 08 0A 13 0A 0A 13   ( █.C..........
```

Figure 8 meow hex

There is a passcode at the start of the hex, now we need to extract it and decode it. Then we get something like this:

“**~*t}{~'t(wx%t**” - ROT 47 = OYENLOVEWHITE

We can use this pass to open our meow.zip.

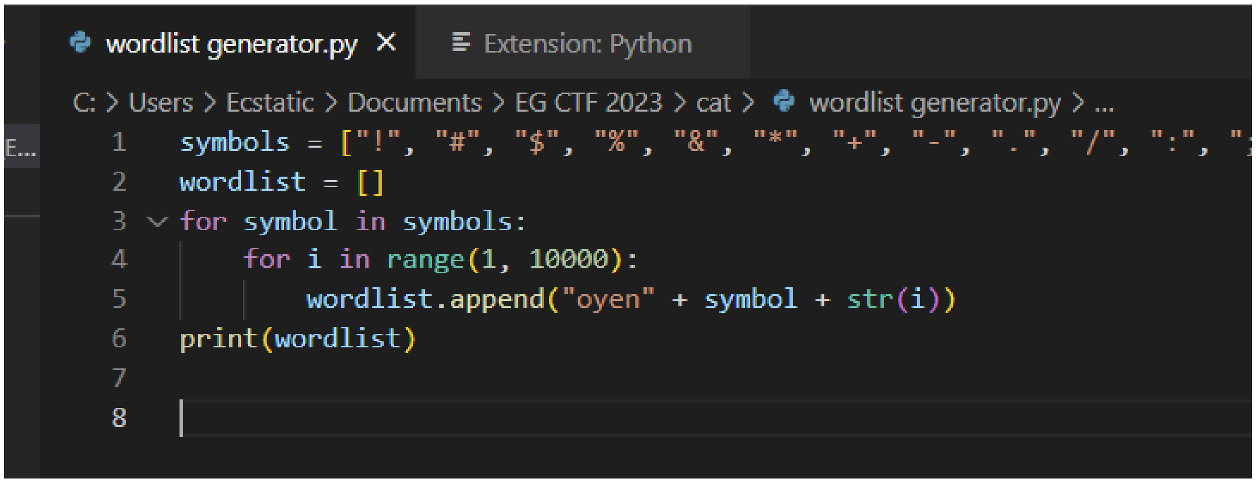
On the meow.zip we have 2 files, 1 zip file and 1 text file. The zip file is password protected and the clue is in the text file.

"THE LAST THINGS I REMEMBER MY PASSWORD CONTAIN MY NAME SYMBOL AND NUMBER." – quote from oyen.txt

From the hint and clue above here we can conclude that the password for zip file is:

oyen+symbols+numbers(1-9999)

We are going to use brute force in order the open the zip files, but 1st we have to create a customized our wordlist. We can achieve this with python programming.



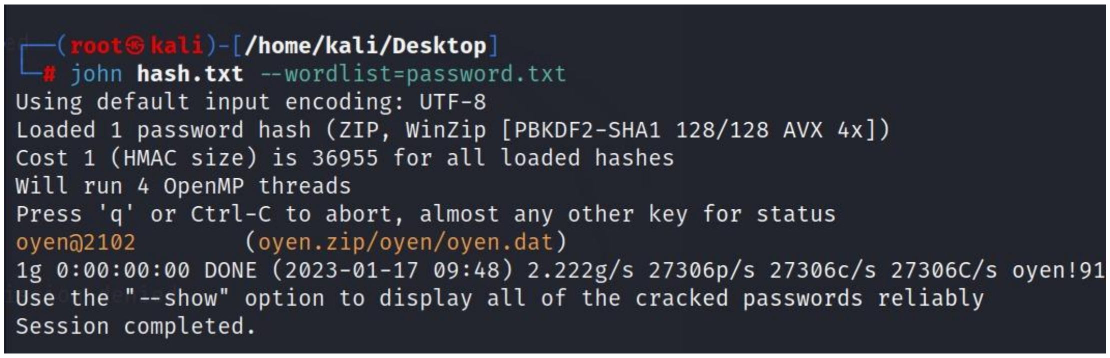
wordlist generator.py
Extension: Python
C: > Users > Ecstatic > Documents > EG CTF 2023 > cat > wordlist generator.py > ...

```
symbols = ["!", "#", "$", "%", "&", "*", "+", "-", ".", "/", ":", "
wordlist = []
for symbol in symbols:
    for i in range(1, 10000):
        wordlist.append("oyen" + symbol + str(i))
print(wordlist)

```

Figure 9 wordlist with python

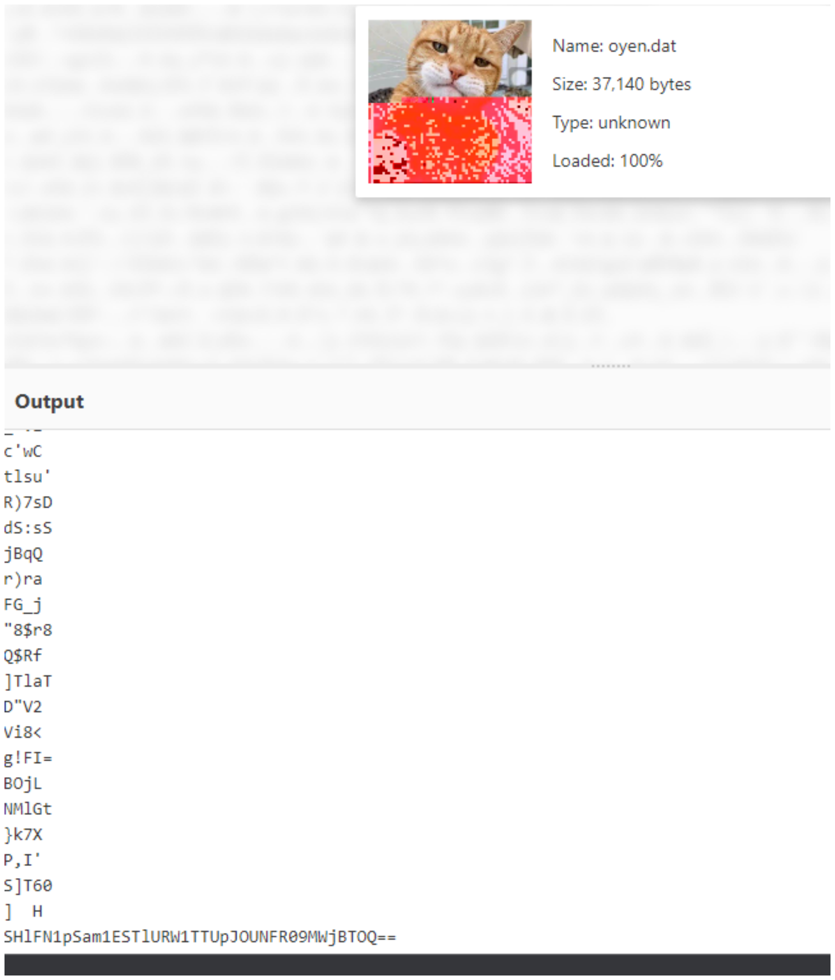
After that we can use the wordlist to use for our next tool called **John the Ripper**.



```
┌──(root㉿kali)-[/home/kali/Desktop]
└─# john hash.txt --wordlist=password.txt
Using default input encoding: UTF-8
Loaded 1 password hash (ZIP, WinZip [PBKDF2-SHA1 128/128 AVX 4x])
Cost 1 (HMAC size) is 36955 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
oyen@2102        (oyen.zip/oyen/oyen.dat)
1g 0:00:00:00 DONE (2023-01-17 09:48) 2.222g/s 27306p/s 27306c/s 27306C/s oyen!91
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 10 cracked password

After that, we get a file named oyen.dat that is partially “broken” so I’m going to further analyze this with **cyberchef**


Name: oyen.dat

Size: 37,140 bytes

Type: unknown

Loaded: 100%

Output

```
c'wC
tlsu'
R)7sD
dS:sS
jBqQ
r)ra
FG_j
"8$r8
Q$Rf
]TlaT
D"V2
Vi8<
g!FI=
BOjL
NMlGt
}k7X
P,I'
S]T60
]  H
SHlFN1pSam1ESTlURW1TTUpJOUNFR09MWjBTOQ==
```

Figure 11 oyen.dat

We have an encoded text, from here I’m going to decode it 4 times to get the flag.

SHlFN1pSam1ESTlURW1TTUpJOUNFR09MWjBTOQ== - Base64

HyE7ZRjmDI9TEmSMJI9CEGOLZ0S9 - ROT 13

UlR7MEwzQV9GRzFZWV9PRTBYM0F9 - Base64

RT{0L3A_FG1YY_OE0X3A} - ROT

EG{0Y3N_ST1LL_BR0K3N}