

Unknown

Description : Is this file a binary in disguise?

Author : Oxrakesh

File : challenge

Analysis the file:

file challenge

```
→ pybyte git:(master) x file challenge
challenge: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2
, BuildID[sha1]=f6af5bc244c001328c174a6abf855d682aa7401b, for GNU/Linux 2.6.32, stripped
```

It is elf binary, dynamically linked and stripped.

Strings challenge

Nothing interest

readelf -S challenge


```
[23] .data          PROGBITS          00000000000040d280 0000c280
0000000000000004 0000000000000000 WA      0      0      1
[24] .bss           NOBITS           00000000000040d2a0 0000c284
000000000000173f0 0000000000000000 WA      0      0     32
[25] .comment       PROGBITS          0000000000000000 0000c284
0000000000000059 0000000000000001 MS      0      0      1
[26] pydata         PROGBITS          0000000000000000 0000c2dd
0000000000006a8dfe 0000000000000000      0      0      1
[27] .shstrtab      STRTAB           0000000000000000 006b50db
00000000000000fa 0000000000000000      0      0      1
```

The challenge binary contain pydata section. So it is python file that compile to elf. So decompile this file by using pyinstxtractor.

Pyinstxtractor.py challenge

It gives python bytecode file.

Uncompyle6 challenge.pyc

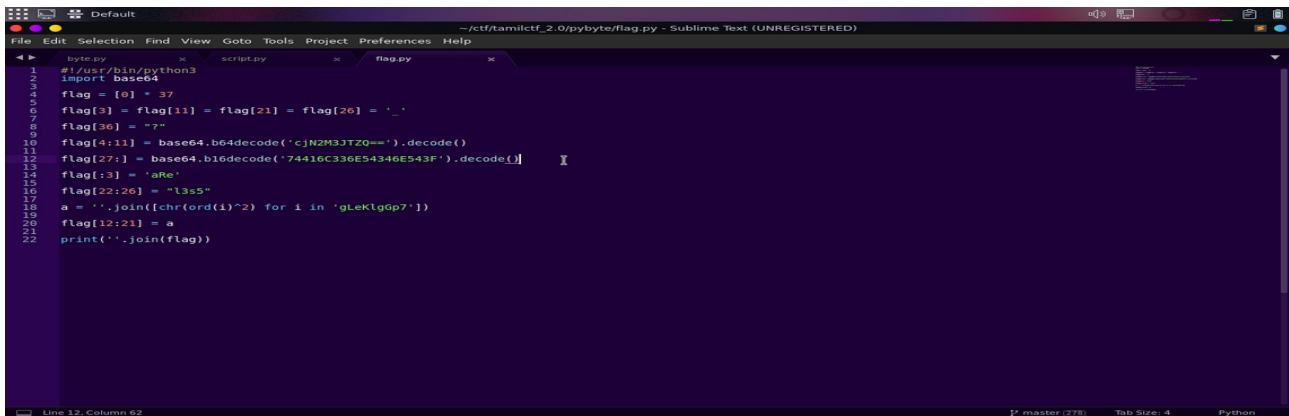
[illegible]

```
#!/usr/bin/python3
from binascii import
import base64

print(bytearray.fromhex('74696d652e736c656570283029206966206372696e67655b335d203d3d20275f2720656c736520696466632829').decode())
print(''.join(chr(i) for i in [116, 105, 109, 101, 46, 115, 108, 101, 101, 112, 40, 48, 41, 32, 105, 102, 32, 99, 114, 105, 110, 103, 101, 91,
7
print(bytearray.fromhex('74696d652e736c656570283029206966206372696e67655b32315d203d3d20275f2720656c736520696466632829').decode())
8
print(''.join(chr(i) for i in [116, 105, 109, 101, 46, 115, 108, 101, 101, 112, 40, 48, 41, 32, 105, 102, 32, 99, 114, 105, 110, 103, 101, 91,
9
print(bytearray.fromhex('74696d652e736c656570283029206966206372696e67655b2d315d203d3d20273f2720656c736520696466632829').decode())
10
print(base64.b64decode('dglTz5SzbGvLcGwKSBpZ1B1YXNlNjQuYjY0ZW5jb2RlKGNyaW5nZS9yOjEXXS5lbmNvZGUo3V0Z104jYkpLmRlY29kZSgpID09ICJjak4yTTNKVfPRTPE
11
print(base64.b64decode('dglTz5SzbGvLcGwKSBpZ1B1YXNlNjQuYjY0ZW5jb2RlKGNyaW5nZS9yOjEXXS5lbmNvZGUo3V0Z104jYkpLmRlY29kZSgpID09ICJjak4yTTNKVfPRTPE
12
print(''.join(chr(i) for i in [116, 105, 109, 101, 46, 115, 108, 101, 101, 112, 40, 49, 41, 32, 105, 102, 32, 99, 114, 105, 114, 105, 103, 100, 32, 61,
13
enig = bytearray.fromhex('27272e0a6f696e285b20636872286f72642809295e322920666f72206920696e206372696e67655b31323a32315d205d29')
14
print(enig)
15
print(''.join(chr(i) for i in [116, 105, 109, 101, 46, 115, 108, 101, 101, 112, 40, 49, 41, 32, 105, 102, 32, 101, 110, 105, 103, 100, 32, 61,
16
print(bytearray.fromhex('7072696e742822436f72726563742070617373776f7264212121215c6e2229').decode())
17
18
print(''.join(bytearray.fromhex('696e7075742822456e746572207468652070617373776f7264203a202229').decode()))
19
print(''.join([ chr(i) for i in [117, 100, 107, 110, 99, 100, 105, 40, 100, 102, 106, 105, 99, 41, 32, 105, 102, 32, 108, 101, 110, 40, 100, 105, 103, 100, 32, 61,
20
```

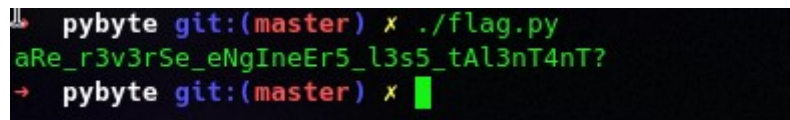
```
→ pybyte git:(master) x ./script.py
time.sleep(0) if cringe[3] == '_' else idfc()
time.sleep(0) if cringe[11] == '_' else idfc()
time.sleep(0) if cringe[21] == '_' else idfc()
time.sleep(0) if cringe[26] == '_' else idfc()
time.sleep(0) if cringe[-1] == '?' else idfc()
time.sleep(0) if base64.b64encode(cringe[4:11].encode('utf-8')).decode() == 'cjN2M3JTZQ==' else idfc()
time.sleep(1) if base64.b16encode(cringe[27:].encode('utf-8')).decode() == '74416C336E54346E543F' else idfc()
time.sleep(1) if cringe[:3] + cringe[22:26] == 'aRel3s5' else idfc()
bytearray(b"\\'\\.join([ chr(ord(i)^2) for i in cringe[12:21] ])"')
time.sleep(1) if enigd == 'gLeKlg6p7' else idfc()
print("Correct password!!!!\n")
input("Enter the password : ")
udkncdi(dfjic) if len(dfjic) == 37 else print("\nWrong flag\n")
```

There are some condition which check the password. Make a script.



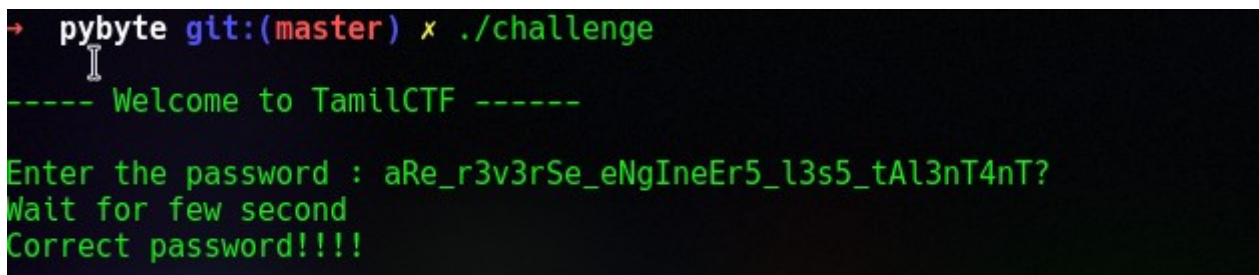
```
1 #!/usr/bin/python3
2 import base64
3
4 flag = [0] * 37
5 flag[3] = flag[11] = flag[21] = flag[26] = '_'
6 flag[36] = "7"
7
8 flag[4:11] = base64.b64decode('cJN2M3JTZQ==').decode()
9
10 flag[27:] = base64.b16decode('74416C336E54346E543F').decode()
11
12 flag[:3] = 'aRe'
13
14 flag[22:26] = 'l3s5'
15
16 a = ''.join([chr(ord(i)^2) for i in 'gLeKlg0p7'])
17
18 flag[12:21] = a
19
20 print(''.join(flag))
```

Execute the script :



```
pybyte git:(master) x ./flag.py
aRe_r3v3rSe_eNgIneEr5_l3s5_tAl3nT4nT?
pybyte git:(master) x
```

Run the binary with this password:



```
pybyte git:(master) x ./challenge
----- Welcome to TamilCTF -----
Enter the password : aRe_r3v3rSe_eNgIneEr5_l3s5_tAl3nT4nT?
Wait for few second
Correct password!!!!
```

Yeah !!! We found the correct password.