Writeup Tchetoula EQUIPE N4t10n

Dans un premier temps, nous avons un fichier zip protégé par un mot de passe et un fichier password.txt. A l'aide des infos sur ce repos github https://github.com/FrancoisCapon/Base64SteganographyTools on a arrive à décoder le password.txt et on obtient **Bt1@70ski#T0%pqax9bp.** On se sert de ce dernier pour décoder le fichier zip.

A l'intérieur se trouve un fichier serveur.py. Apres quelques recherches de ce code sur internet, on tombe sur ce code. Il s'agit pratiquement du même challenge avec quelques différences

“”””””

```
from Crypto.Util.Padding import unpad
from randcrack import RandCrack
from Crypto.Cipher import AES
from pwn import *
import os
import logging

logging.disable()

HOST = os.environ.get("HOST", "lazy-platform.challs.teamitaly.eu")
PORT = int(os.environ.get("PORT", 15004))


def getrandbytes(rc: RandCrack, n: int) -> bytes:
        if n % 4 != 0:
        return os.urandom(n)
        return b"".join(rc.predict_getrandbits(32).to_bytes(4, "little") for _ in range(n // 4))


if _name_ == "_main_":
        rc = RandCrack()

        conn = remote(HOST, PORT)

        for _ in range(624 // (32 // 4 + 16 // 4)):
        conn.sendlines([b"1", os.urandom(4).hex().encode()])

        conn.recvuntil(b"Key: ")
        key = bytes.fromhex(conn.recvline(False).decode())

        conn.recvuntil(b"IV: ")
        iv = bytes.fromhex(conn.recvline(False).decode())

        for i in range(0, len(key), 4):
        rc.submit(int.from_bytes(key[i:i+4], "little"))
```

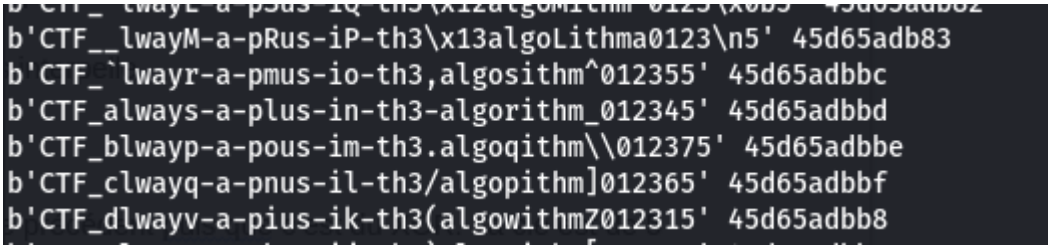
```
for i in range(0, len(iv), 4):
rc.submit(int.from_bytes(iv[i:i+4], "little"))
conn.sendline(b"3")

conn.recvuntil(b"Ciphertext: ")
ciphertext = bytes.fromhex(conn.recvline(False).decode())

key = getrandbytes(rc, 32)
iv = getrandbytes(rc, 16)

print(unpad(
AES.new(key, AES.MODE_CBC, iv).decrypt(ciphertext),
AES.block_size
).decode())
```

“””””

On adapte le code à notre contexte et on tombe sur cette sortie
b'\x06\x82\x1c\x84\xdc)\xa1;\xa2\xceh\xb7w\xab\xd10\xa5w\xb2\xd3h\xa22\xe8\x90$\xba=\xb4\xcf,\xa22\xb6\xe2u\xe7h\xe8\x89p'

Dans server.py il y a cette partie qui nous interpelle
for i in range(len(FLAG)):
FLAG[i] = FLAG[i]^key[i%len(key)]

Il s’agit donc de décoder la sortie du code précédent puisque c’est du XOR. La clé est de 5 caractères. Nous connaissons les quatres premiers caractères du message en clair qui sont “CTF_”. Pour retrouver la clé, nous avons improvisé pour le 5è caractère en ajoutant tout les caractères de la table ascii. Donc avec cela, nous avons généré toutes les clés possibles. Une fois cela fait, on essaye de décoder la sortie précédente à l’aide de toutes ces clés et on obtient ceci:



Le flag est donc **CTF_always-a-plus-in-th3-algorithm_012345**