Darth Stuff WRITEUP

The challenge presents us two different channels. Each channel requires a default password: `this_is_darth_stuff` , then a number `p` (prime) which was equal for both the channels were prompted, followed by a number, which were different every time, then we have to insert something and the channel answer prompting CFB (a cipher block mode of operation), a base64 encoded string of 16 bytes (we can reasonably suppose that this is an IV or a key) and another base64 (we can reasonably suppose that this is the ciphertext). The process is a Diffie-Hellman key agreement, by performing the key agreement with each channel we can use the recovered key and the given IV to decrypt the ciphertexts, by putting together the two plaintexts we obtain the flag.

```python
from pwn import *
from Crypto.Cipher import AES
from base64 import b64decode

PASSWORD = "this_is_darth_stuff"

int_byte_length = lambda integer: -(integer.bit_length()//-8)

# base canonica per DH
g = 2

# stabilisco una connessione con i 2 server
r1 = remote("gamebox1.reply.it", 9999)
r2 = remote("gamebox1.reply.it", 9998)

# autenticazione
r1.recvuntil(": ")
r2.recvuntil(": ")
r1.sendline(PASSWORD.encode())
r2.sendline(PASSWORD.encode())

# recupero il modulo
r1.recvuntil("p: ")
p1 = r1.recvline().decode().strip()
r2.recvuntil("p: ")
p2 = r2.recvline().decode().strip()
```

```python
assert p1 == p2

p = p1

p = int(p[2:], 16)

# costruisco il mio esponente privato
b = 1342695364645384855984855582244833957357590802143969347423306461702299072119

# effettuo il key agreement
r1.recvuntil("says: ")
r2.recvuntil("says: ")

g1 = r1.recvline().decode().strip()
g2 = r2.recvline().decode().strip()

r1.recvuntil("? ")
r2.recvuntil("? ")

r1.sendline(str(hex(pow(g, b, p))))
r2.sendline(str(hex(pow(g, b, p))))

# ricevo IV e Ciphertext
r1.recvuntil("CFB\n")
r2.recvuntil("CFB\n")

iv1 = r1.recvline().decode().strip()
iv2 = r2.recvline().decode().strip()

ciphertext1 = r1.recvline().decode().strip()
ciphertext2 = r2.recv().decode().strip()

# costruisco le chiavi con i valori ottenuti dal key agreement
key1 = pow(int(g1[2:], 16), b, p)
key1 = key1.to_bytes(int_byte_length(key1), "big")[:16]

key2 = pow(int(g2[2:], 16), b, p)
key2 = key2.to_bytes(int_byte_length(key2), "big")[:16]

# decritto la flag
enc1 =  AES.new(key1, AES.MODE_CFB, iv=b64decode(iv1))
enc2 =  AES.new(key2, AES.MODE_CFB, iv=b64decode(iv2))
```

```
flag1 = enc1.decrypt(b64decode(ciphertext1)).decode()
flag2 = enc2.decrypt(b64decode(ciphertext2)).decode()

flag = f"{flag1}{flag2}"

print(f"FLAG: {flag}") # {FLG:fir5t_ha1f_0f_f14g_4nd_s3c0nd_ha1f_0f_f14g}
```