4. Crypto – 'I want it all'

For solving this challenge, you should know how XOR Encrypt&Decryption work.

I will not explain exclusively in the writeup, but you can refer to some other documents.

But for this, a big hint for you is the flag format. You can try using several beginning letters as the key.

```
from pwn import xor #pip install pwntools
flag                                                                =
bytes.fromhex('3b312857495b41441a115b3a0e5d1d6b32481f3a4d051f00
10')
print(xor(flag, 'VHC2022{'.encode()))
```

From its output, you can see:

```
b'mykeyis?LY\x18\x08>o/\x10d\x00\\\x08}7-{F'
```

Because XOR encryption is symmetric:

$$c = p \oplus k => c \oplus k = (p \oplus k) \oplus k = p$$

For c is ciphertext, p is plaintext and k is the key.

It's worth a try for you to use 'mykeyis?' as your key.

```
print(xor(flag, 'mykeyis?'.encode()))
```

```
b'VHC2022{wh0_w4nT_1t_4ll?}'
```

In this challenge, the encryptor made a mistake. When using XOR encryption, you have to use a key that at least has length at least equal to the plaintext. Or no, the key will be used again and again for the plaintext and a cryptanalysis is able to launch on this situation. So, you don't need a full-length key to decrypt the code. Moreover, because in here we predict a part of plaintext ("VHC2022{"), this attack is called **'known-plaintext attack'**.

Ah! In the solution I used a very useful library for this: pwntools. You should install it to your device too :>.