Shadowfin Writeup

Comenzando con el reto, se nos proporcionan dos archivos: **shadowfin.py** y **out.txt.enc**.

Descripción del reto:
"¿Podrás ver más allá de las sombras y encontrar lo que está escondido? Con la combinación perfecta de ingenio y deducción, te espera un enigma simple, pero que esconde una pieza clave: la flag."

Este mensaje nos hace sospechar que el reto implica algún tipo de **cifrado clásico o modificación ligera del contenido**.

shadowfin.py

shadowfin.py > ...

```python
import random

a = 5
b = 8


def affine_encrypt(text, a, b):
    encrypted = []
    for char in text:
        if char.isalpha():
            base = 65 if char.isupper() else 97
            encrypted.append(chr((a * (ord(char) - base) + b) % + base))
        else:
            encrypted.append(char)
    return ''.join(encrypted)

with open('flag.txt.enc', 'w') as f:
    f.write(scrambled_flag)


def random_noise():
    return ''.join([chr(random.randint(33, 126)) for _ in range(20)])

print(random_noise())
```

tras revisarlo detenidamente, vemos que implementa una **versión del cifrado Afín (Affine Cipher)**. Este tipo de cifrado toma un texto claro y lo transforma mediante una fórmula basada en **aritmética modular**, utilizando dos variables: a y b.

Sin embargo, **detectamos un error** en la fórmula del cifrado afín. La línea:

```
  encrypted.append(chr((a * (ord(char) - base) + b) % + base))
lse:
```

corrigiendo el código nos queda de la siguiente manera:

dec.py > ...

```python
def affine_decrypt(ciphertext, a, b):
    # Calculamos el inverso multiplicativo de 'a' módulo 26, que es 21 para a=5.
    a_inv = 21
    decrypted = []
    for char in ciphertext:
        if char.isalpha():
            base = 65 if char.isupper() else 97
            # Fórmula inversa del cifrado afín
            decrypted.append(chr((a_inv * ((ord(char) - base) - b) % 26) + base))
        else:
            decrypted.append(char)
    return ''.join(decrypted)

# El texto cifrado proporcionado en el reto
ciphertext = "UryNyzc{ciuy_ihhwvc_swfrcp}"

# Parámetros de cifrado: a = 5, b = 8
a = 5
b = 8

flag = affine_decrypt(ciphertext, a, b)
flag
```

Una vez listo podemos obtener nuestra flag!!

ShyByte{easy_affine_cipher}