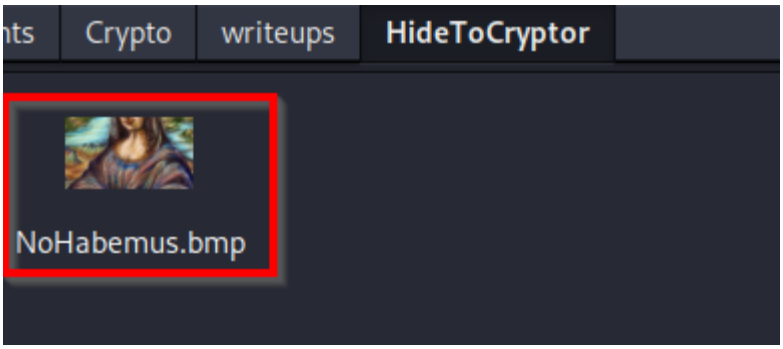
HideToCryptor Writeup

Comenzando con el reto, se nos proporciona una imagen llamada **NoHabemus.bmp**. La descripción del reto hace énfasis en **objetos ocultos**, lo que nos lleva a intuir que el desafío está relacionado con **esteganografía.**

Descripción del reto:

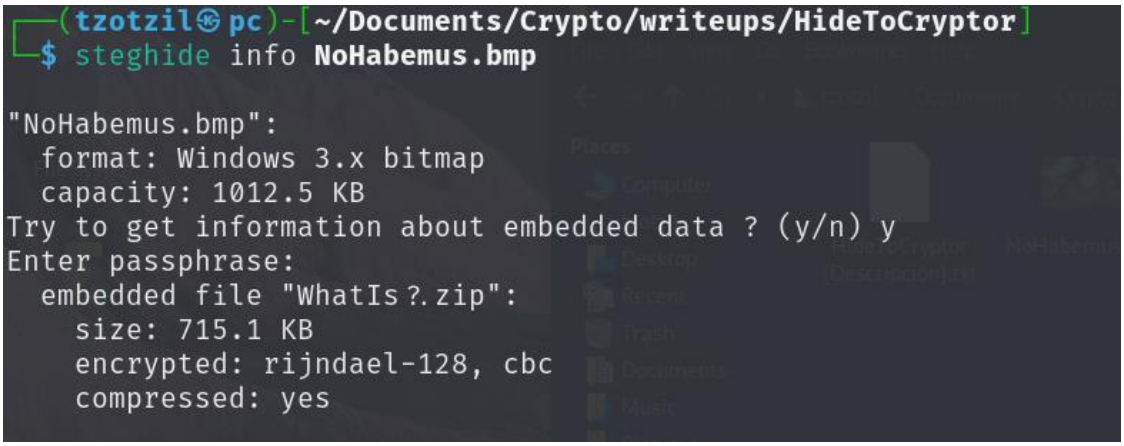
"Has llegado hasta aquí, pero recuerda: en Halloween nada es lo que parece. Cuando la oscuridad se cierne, los secretos ocultos salen a la luz. ¿Podrás ver más allá de las apariencias y encontrar aquello que se esconde entre las sombras? Todo lo que necesitas ya está frente a ti... solo queda unir las piezas del enigma. Mucha suerte, buscador de lo oculto. Las respuestas no esperan para siempre."



Nos encontramos con una imagen en formato **BMP**, que no es lo más común para una simple foto. **Las BMP suelen ser usadas para ocultar datos**, ya que al no tener compresión es más fácil meter información extra sin alterar la apariencia de la imagen.

Utilizando **steghide,** confirmamos que la efectivamente cuenta con un archivo oculto llamado

"WhatIs?.*zip*"



Descomprimimos el ZIP y encontramos **tres archivos**:

```
┌──(tzotzil㉿pc)-[~/Documents/Crypto/writeups/HideToCryptor]
└─$ unzip WhatIs\?.zip
Archive:  WhatIs?.zip
  inflating: 337354415f6e4f5f65736c345f4c4c3476335f4368346c333333.jpeg
  inflating: flag.txt.enc
  inflating: decifra.py
```

Revisamos el contenido de los archivos.

decifra.py

```
tzotzil → ~/Documentos/HideToCryptor $ strings decifra.py
def calcular(numero):
    return numero * 42 + 7
def operacion(texto):
    basura = [chr(ord(c) ^ 0x23) for c in texto]
    return "".join(basura)
def inicializar_componentes():
    componente_1 = "ABCDEFGHIJKLMNOPQRSTUVWXY.....................................................
..............."
    componente_2 = "QWERTYUIOPASDFGHJKLZXCVBNMpoiuytrewqlkjhgfdsamnbvcxz"
    return componente_1, componente
def proceso_complejo_para_descifrar(entrada, componente_1, componente
    mapeo = str.maketrans(componente_2, componente_2)
    salida = entrada.translate(mapeo)
resultado = calcular(3)
texto_ = operacion("EsteTextocifrado")
mensaje = input("Introduce el mensaje cifrado: ")
alfabeto1, alfabeto2 = inicializar_componentes()
print("Mensaje descifrado:", proceso_complejo_para_descifrar(mensaje, alfabe1, alfabeto2))
```

flag.txt.enc

```
┌──(tzotzil㉿pc)-[~/Documents/Crypto/writeups/HideToCryptor]
└─$ cat flag.txt.enc
IGSQ AOR HTFLQLZT JXT ZT SG RTPQKOQ DXN LTFEOSSG? HGK LXHXTLZG JXT FG, RTWTKOQL OFZTFZQK QHKTFRTK BGK BGK BGK BGK p284119402t0u03u1i84uipy8330884957
 JXT TL LIQ256?

┌──(tzotzil㉿pc)-[~/Documents/Crypto/writeups/HideToCryptor]
└─$
```

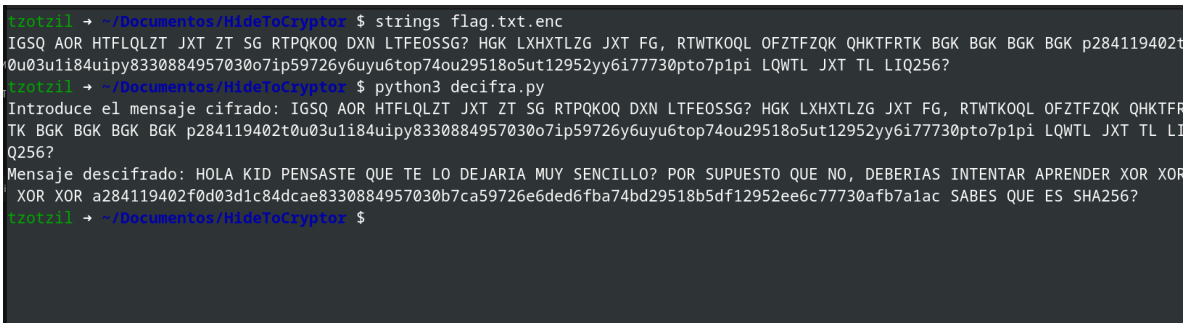
337354415f6e4f5f65736c345f4c4c3476335f4368346c333333.jpeg

El archivo decifra.py parece implementar una especie de **cifrado ATBASH**, pero con un **alfabeto personalizado**. Nos lanzamos a intentar descifrar el contenido del archivo flag.txt.enc usando este código. Sin embargo, el script tenía **algunos errores** que tuvimos que corregir.

cifra.py > ...

```python
def calcular(numero):
    return numero * 42 + 7

def operacion(texto):
    basura = [chr(ord(c) ^ 0x23) for c in texto]
    return "".join(basura)

def inicializar_componentes():
    componente_1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
    componente_2 = "QWERTYUIOPASDFGHJKLZXCVBNMpoiuytrewqlkjhgfdsamnbvcxz"
    return componente_1, componente_2

def proceso_complejo_para_descifrar(entrada, componente_1, componente_2):
    mapeo = str.maketrans(componente_2, componente_1)
    salida = entrada.translate(mapeo)
    return(salida)


resultado = calcular(3)
texto_ = operacion("EsteTextocifrado")


mensaje = input("Introduce el mensaje cifrado: ")
alfabeto1, alfabeto2 = inicializar_componentes()


print("Mensaje descifrado:", proceso_complejo_para_descifrar(mensaje, alfabeto1, alfabeto2))
```

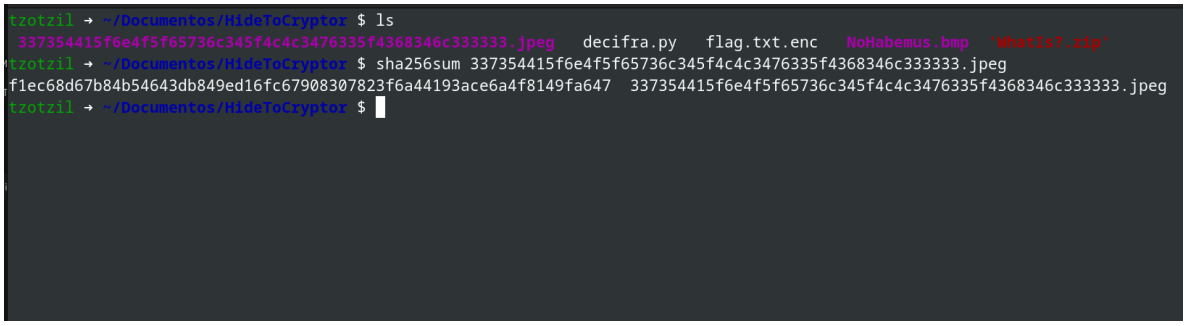
Una vez que arreglamos los bugs del script y lo ejecutamos, obtenemos este texto descifrado:



a284119402f0d03d1c84dcae8330884957030b7ca59726e6ded6fba74bd29518b5df12952ee6c77730afb7a1ac

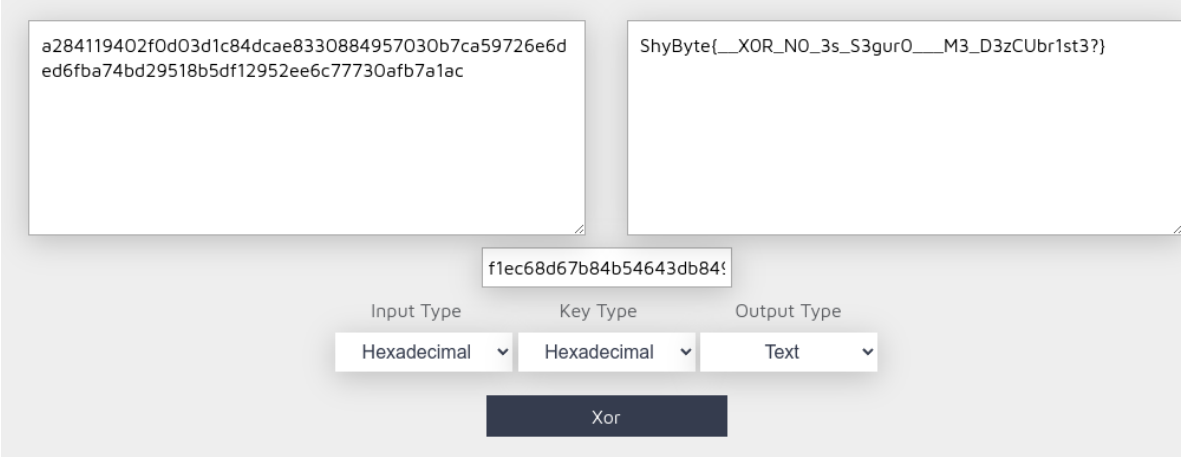
Las nuevas instrucciones mencionan que la **flag** está cifrada con una **operación XOR**. Además, nos sugieren que la **llave para esta operación** podría ser el **hash SHA-256** de uno de los archivos en el reto. ¡Bingo! Recordamos que en el archivo ZIP teníamos una **imagen.**

Calculamos el SHA-256 de la imagen y obtenemos:



f1ec68d67b84b54643db849ed16fc67908307823f6a44193ace6a4f8149fa647

Para completar el reto, usamos una **herramienta online** que permite realizar operaciones XOR. Ingresamos tanto el **texto descifrado** como el **hash SHA-256** como entrada. Y... ¡boom! Después de hacer el XOR, **encontramos la flag**.



ShyByte{__X0R_N0_3s_S3gur0___M3_D3zCUbr1st3?}