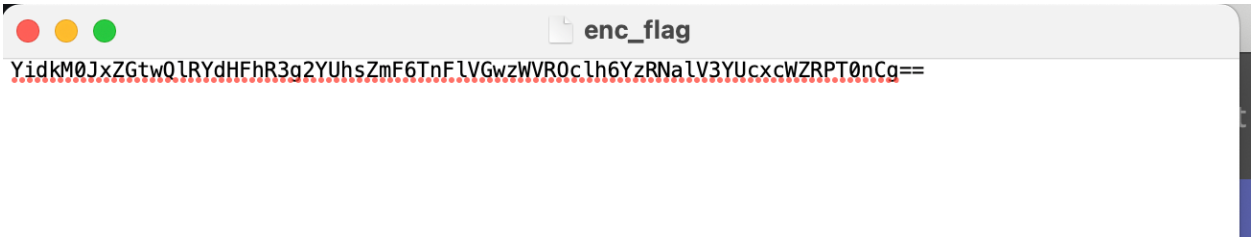
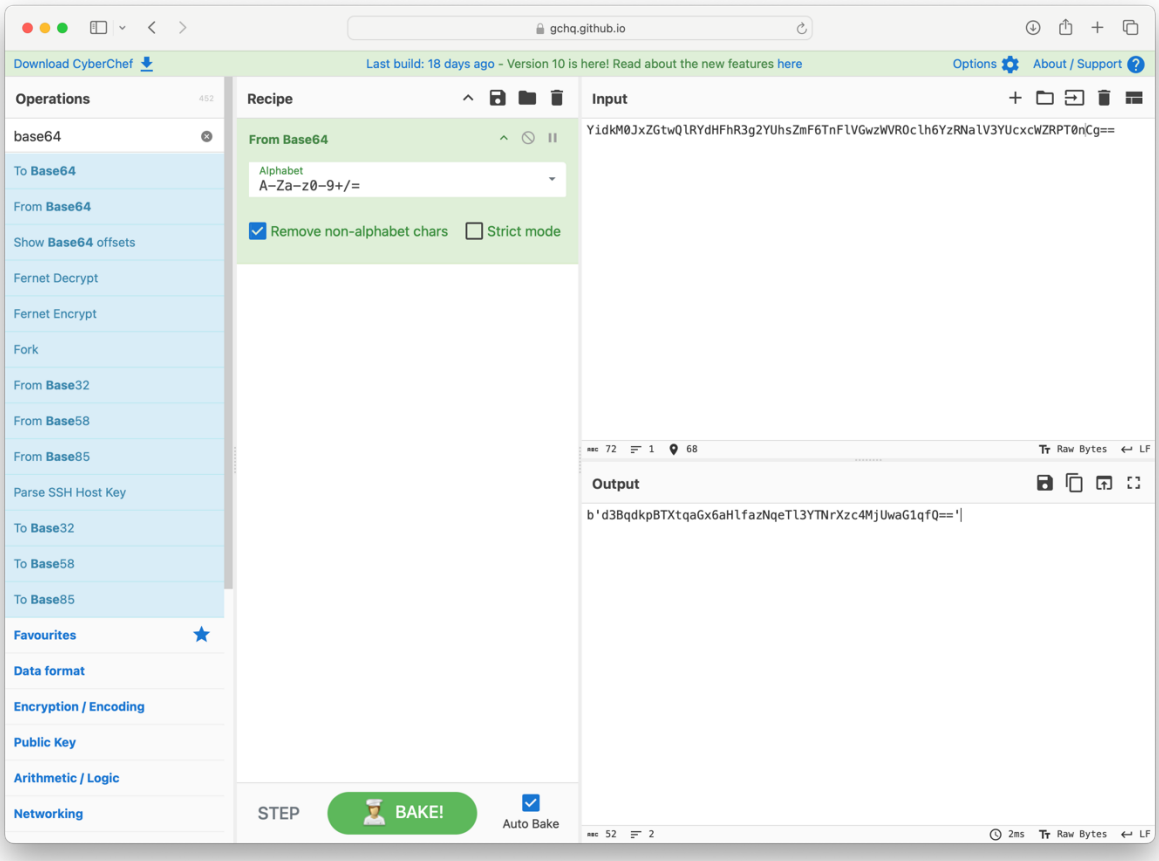
Ejercicio 1

The First step is to download the enc_flag archive which gives us the ciphertext

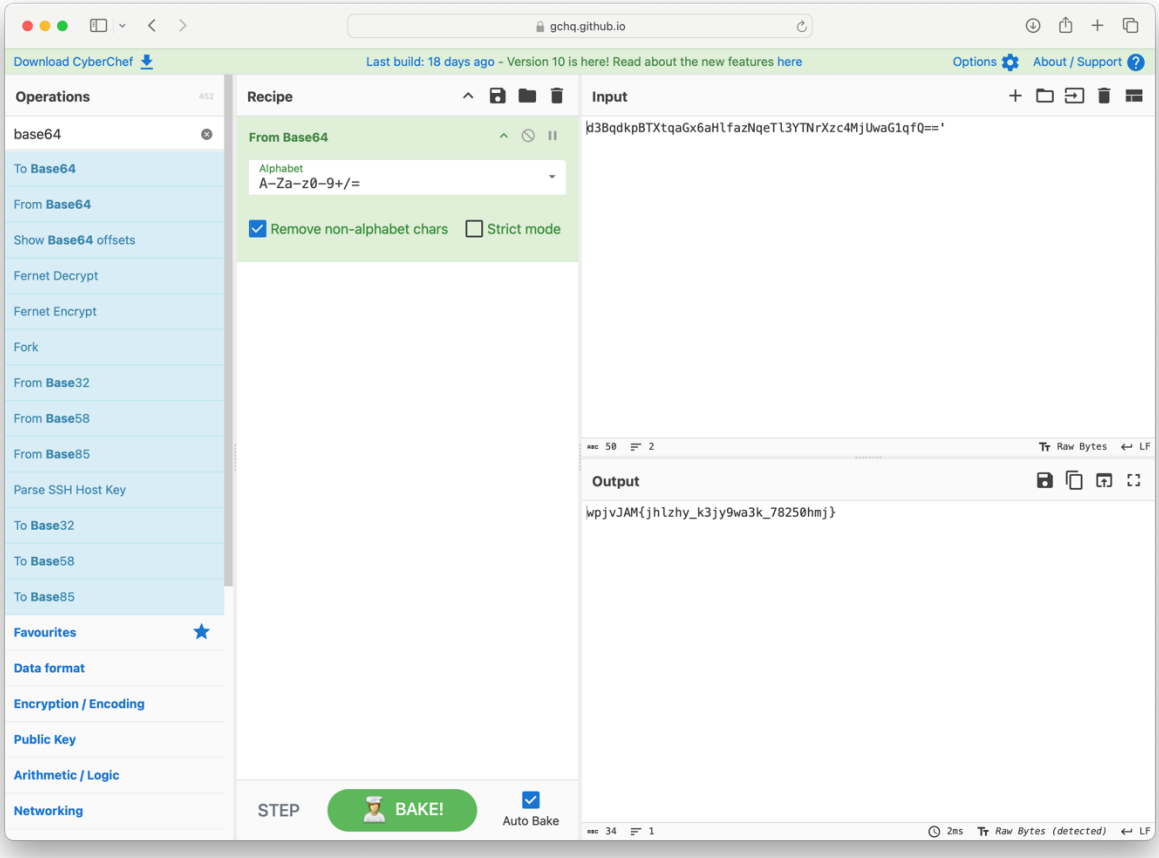


YidkM0JxZGtwQlRYdHFhR3g2YUhsZmF6TnFlVGwzWVROclh6YzRNalV3YUcxcWZRPT0nCg==

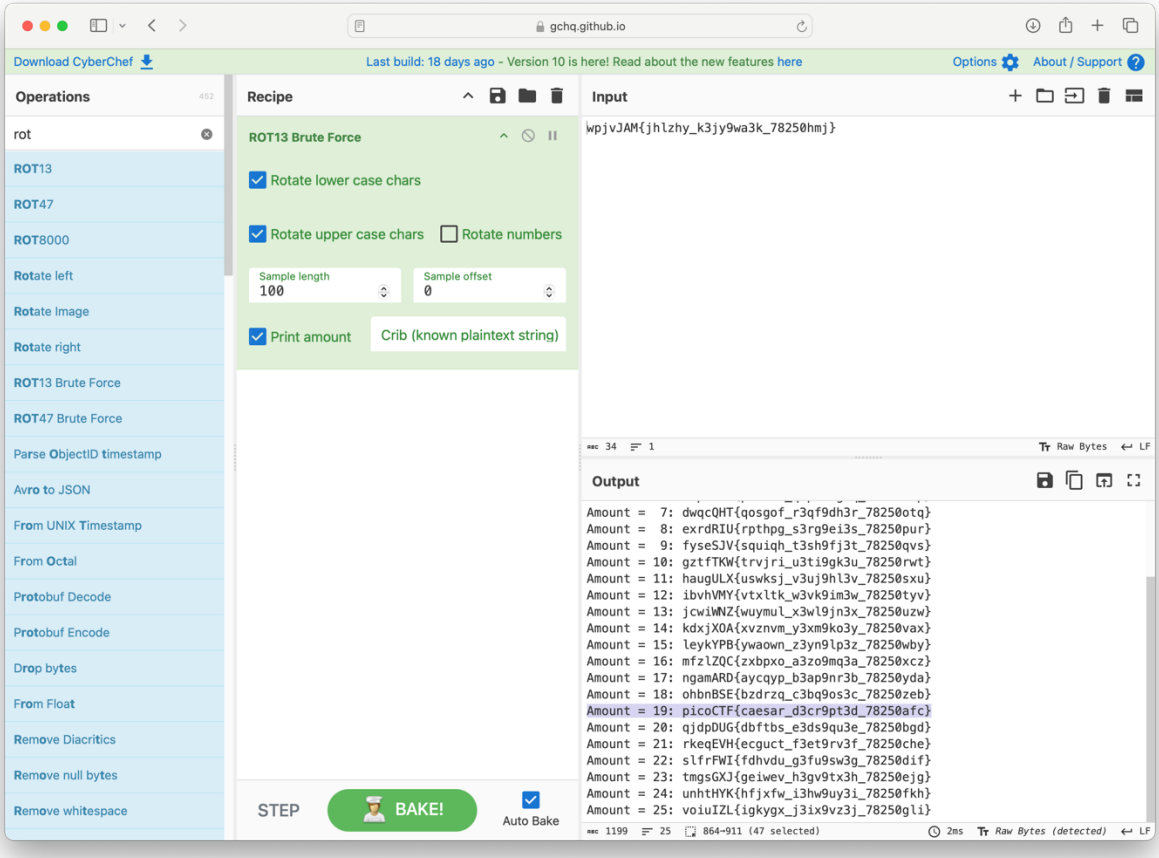
After reviewing the message we can see that the message is encoded in base64, we can assume that because it's a continuos string, it contains letters from (A-Z) with both lower and uppercases, hacing = at the end of the message is also a good giveaway since in base 64 its used as padding for data that does not firt into 3-byte chunks.



With that information we load the message into the CyberChef website and decode it using the Base 64 function.



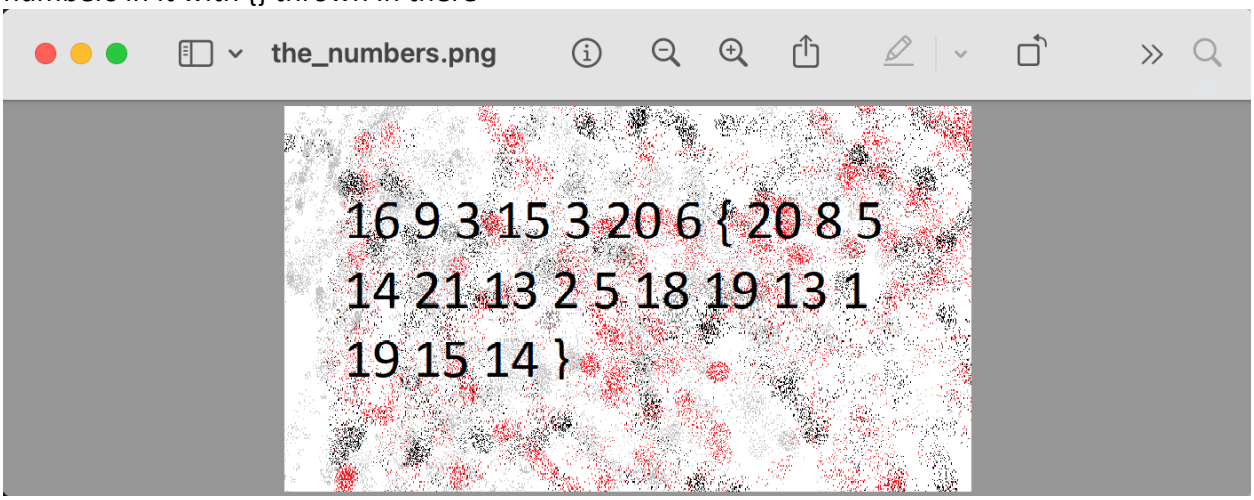
Once that is donde we get another encrypted message, looking closely at the message it only contains letters from the A-Z so it might be a Rotation Cypher or ROT, since we don't know which rotation has been used we try to brute force the problem and chose to use the ROT 13 brute Force function.



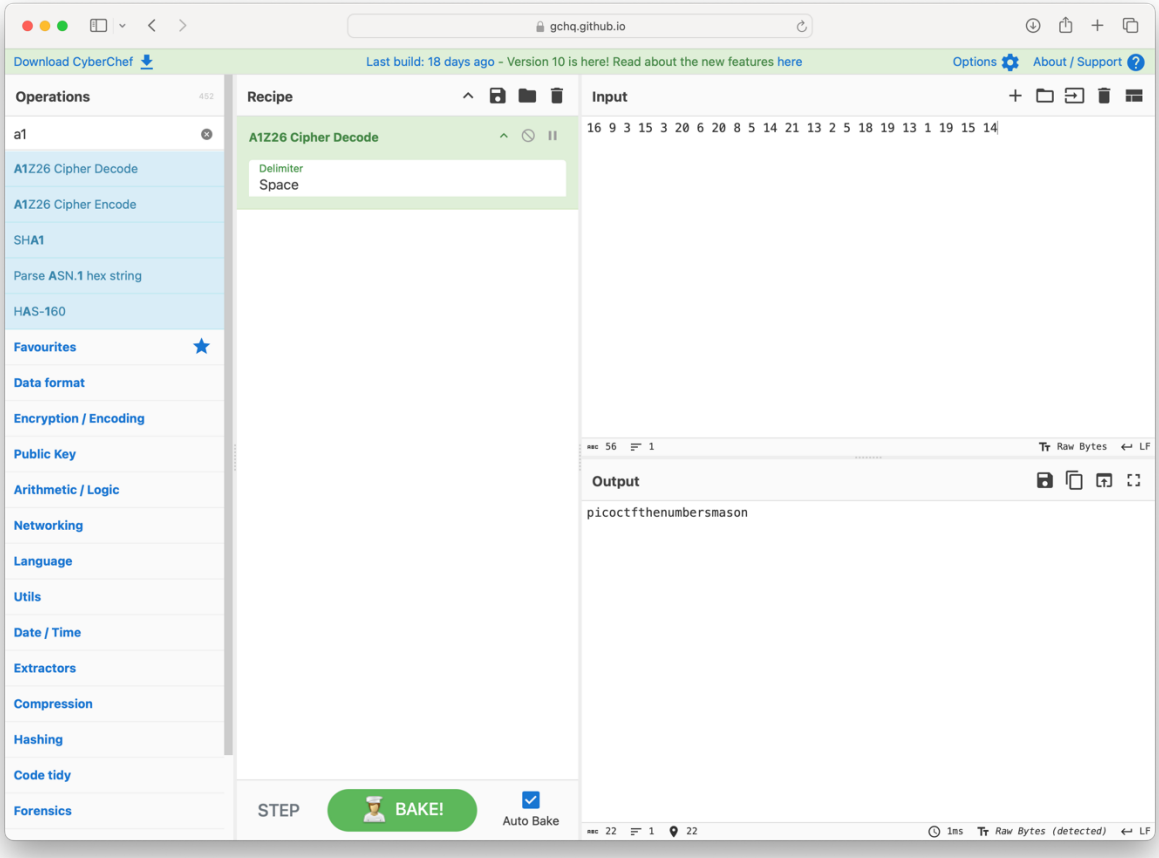
We check the different resultds and se that the amount 19 gives us a complete message that can be our solution therefore we copy that message and as we thought that is the message decoded.

Ejercicio 2

We download the file that comes with the problem and see its an image with a bunch of numbers in it with {} thrown in there

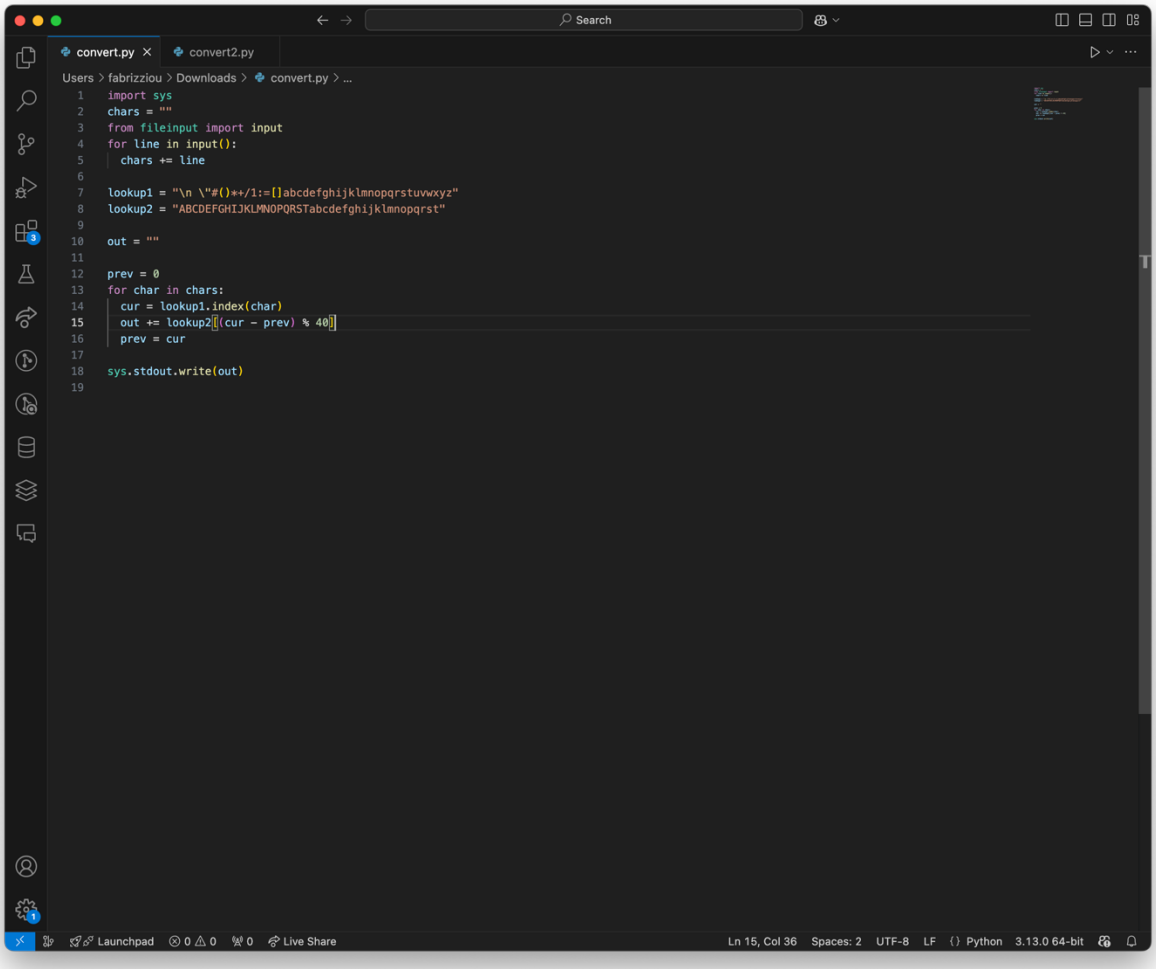


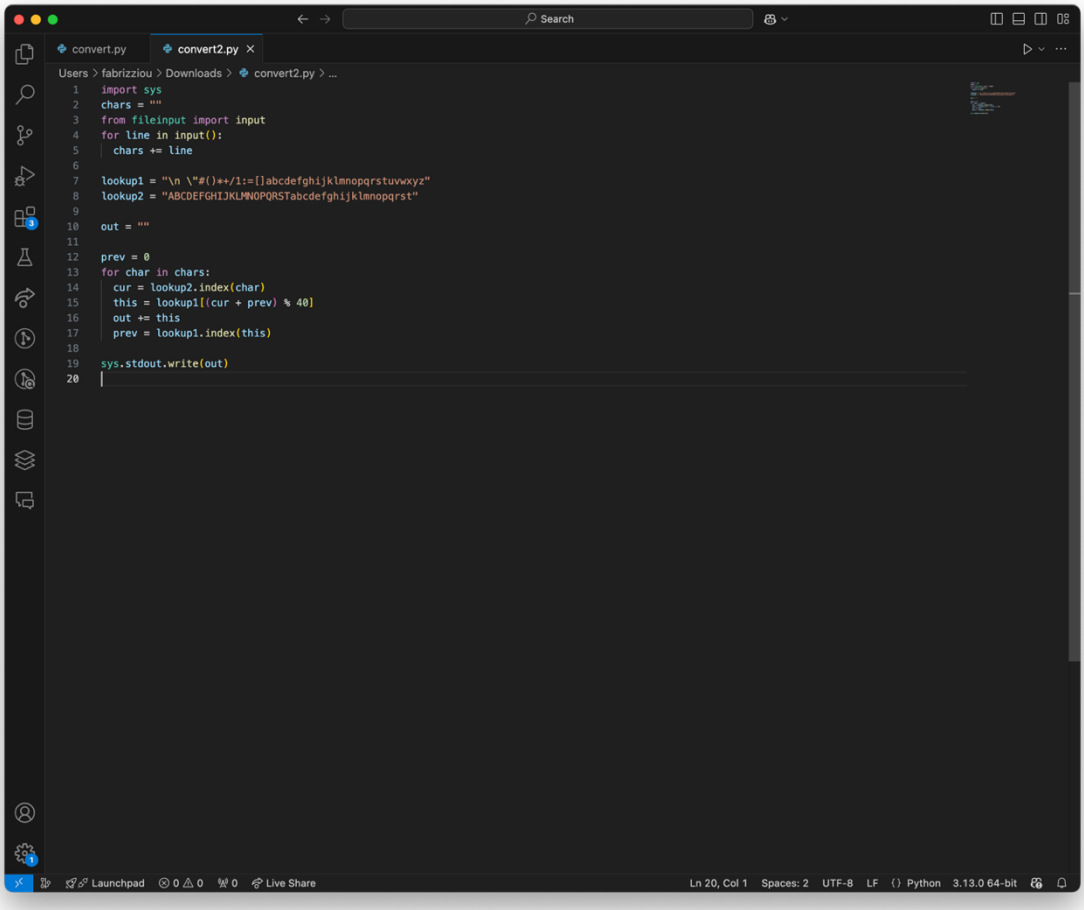
The first thing that comes to mind its that it is a numerical cypher with so we proceed to wirte the numbers down in the CyberChef page, once that is done we look for the cypher function that can help us which turns out to be A1Z26 Cipher Decode, this gives us the decoded message picoctfthenumbersmason as an aswer which proves to be correct once we tried it.



Ejercicio 3

Fisrt we download the files in the problem which are a ciphertext and the encoder convert.py





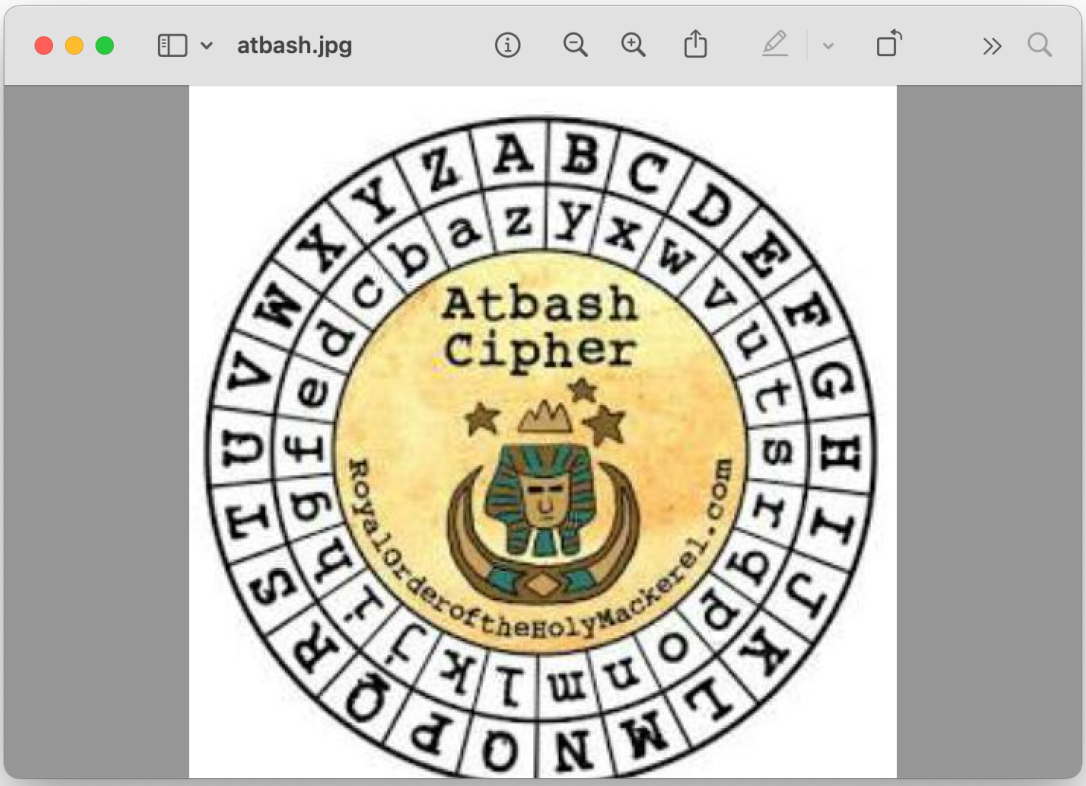
We proceed to change the code of convert.py so that instead of enconding it decodes the message, once that is done we can proceed to use the new code convert2.y to decode the ciphertext.

```
Downloads — -zsh — 90×30
(base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads %  echo abcd| python3 convert.py
OBBBd%
(base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % echo -n "OBBBd" | python3 convert2.p
y
abcd
(base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat ciphertext |python3 convert2.py
#asciiorder
#fortychars
#selfinput
#pythontwo

chars = ""
from fileinput import input
for line in input():
    chars += line
b = 1 / 1

for i in range(len(chars)):
    if i == b * b * b:
        print chars[i] #prints
        b += 1 / 1
(base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat ciphertext |python3 convert2.py
>file.py
```

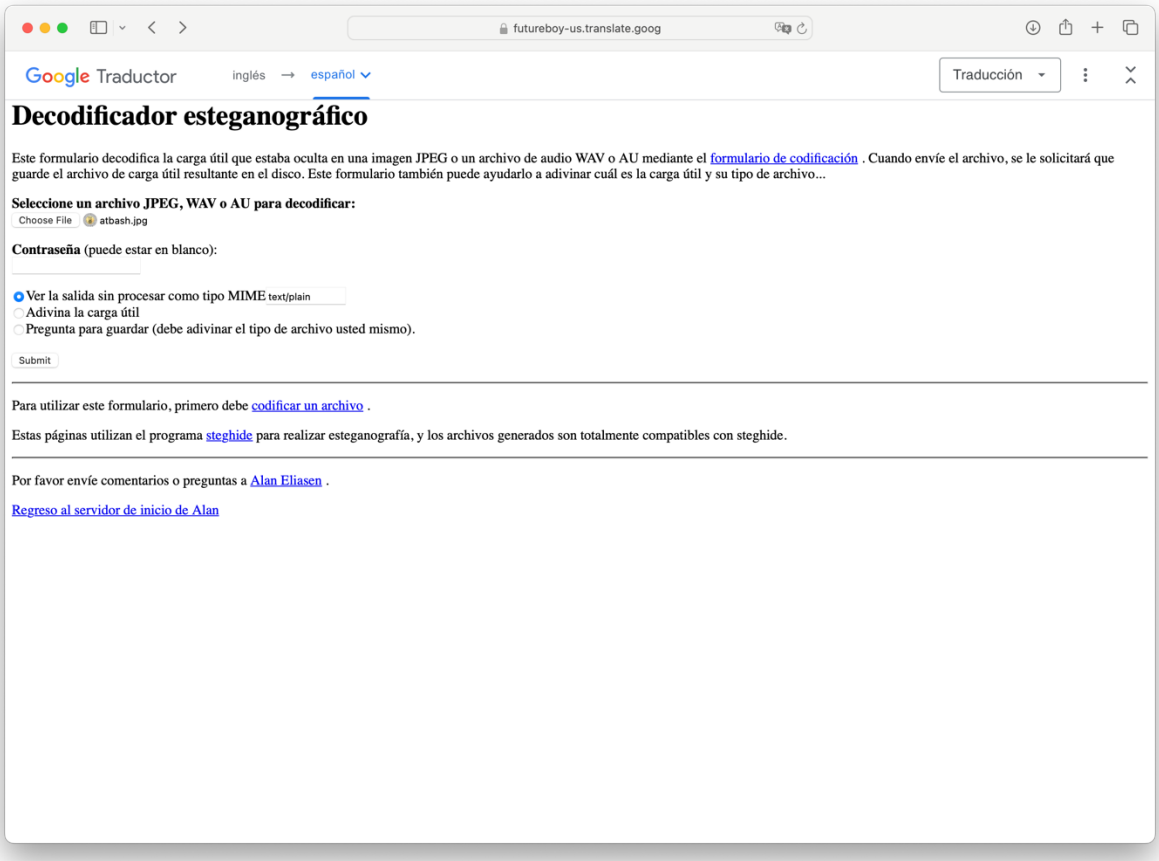
As seen in the image we procced to test if the new code can indeed decode the message by encoding abcd with the convert.py and using the encoded result in convert2.py once we made sure it decodes we proceded to pass the cyphertext which gave us a new encoded message we save said encoded message in a file.py and using that file as an argument for convert2.py we get the answer which is: picoCTF{adlibs}

Ejercicio 4

First we download the file from the problem which is a JPG with an Atbash Cipher as its shown in the image



The first step would be to get the ciphertext from that image and for that we used an Steganografic decoder



futureboy-us.translate.goog

Google Traductor inglés → español Traducción

Decodificador esteganográfico

Este formulario decodifica la carga útil que estaba oculta en una imagen JPEG o un archivo de audio WAV o AU mediante el formulario de codificación . Cuando envíe el archivo, se le solicitará que guarde el archivo de carga útil resultante en el disco. Este formulario también puede ayudarlo a adivinar cuál es la carga útil y su tipo de archivo...

Seleccione un archivo JPEG, WAV o AU para decodificar:

Choose File atbash.jpg

Contraseña (puede estar en blanco):

Ver la salida sin procesar como tipo MIME text/plain

Adivina la carga útil

Pregunta para guardar (debe adivinar el tipo de archivo usted mismo).

Submit

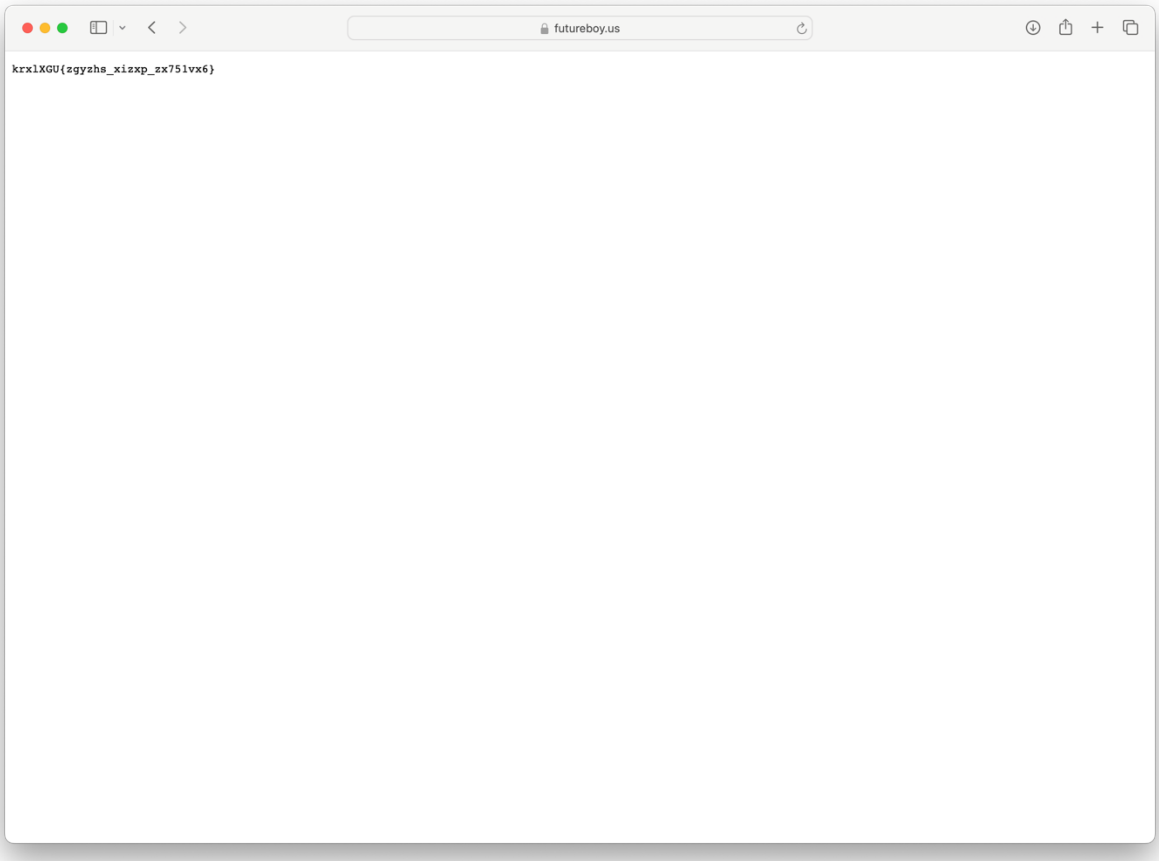
Para utilizar este formulario, primero debe codificar un archivo .

Estas páginas utilizan el programa steghide para realizar esteganografía, y los archivos generados son totalmente compatibles con steghide.

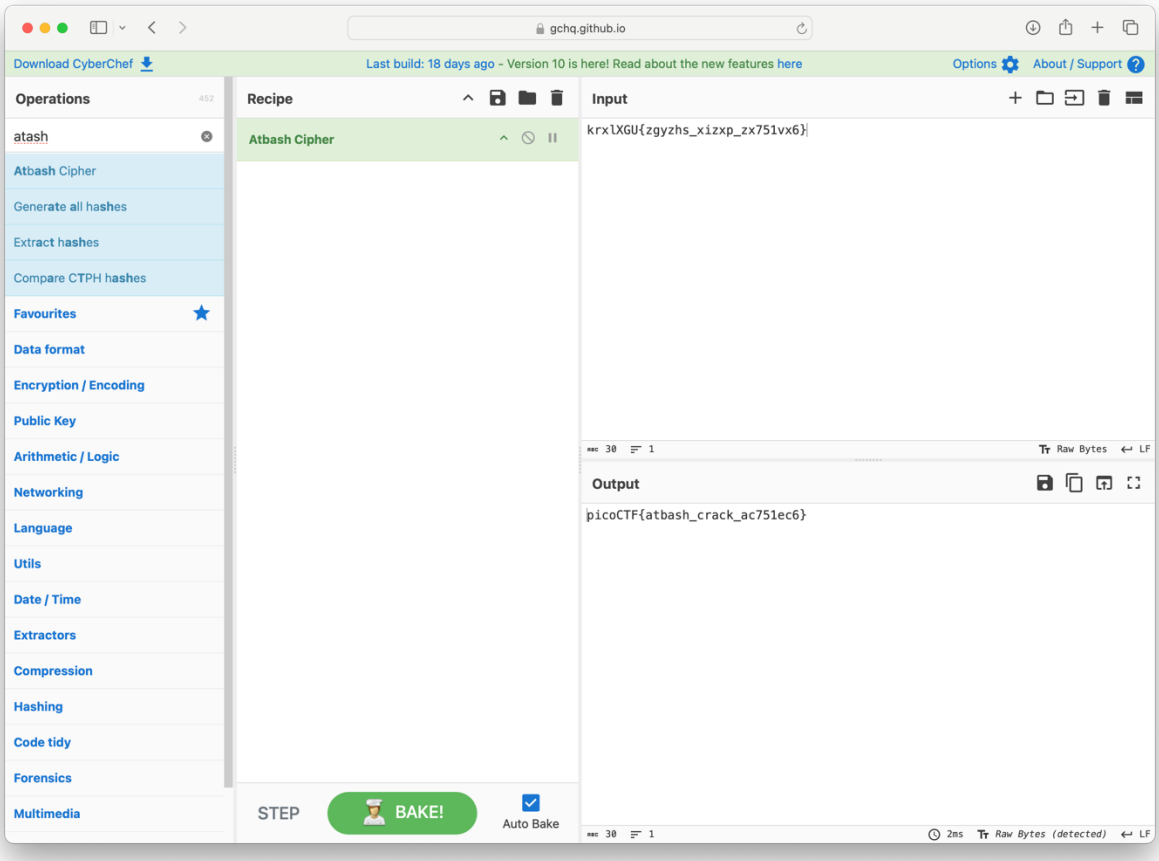
Por favor envíe comentarios o preguntas a Alan Eliasen .

Regreso al servidor de inicio de Alan

This decoder gave us the ciphertext which is shown in the next image.



Now that we have the ciphertext we can go to the CyberChef page to help us decode it using the fiunction decoder for Atbash CIpher



This gives us the message picoCTF{atbash_crak_ac751ec6}

Ejercicio 5

First of all we downlado both files te message and the password so that we can begin

rsa_oracle

Medium | Cryptography | picoCTF 2024 | browser_webshell_solvable

AUTHOR: GEOFFREY NJOGU

Description

Can you abuse the oracle?
An attacker was able to intercept communications between a bank and a fintech company. They managed to get the message (ciphertext) and the password that was used to encrypt the message. Additional details will be available after launching your challenge instance.

This challenge launches an instance on demand.
Its current status is:
NOT_RUNNING

Launch Instance

Hints

1 2 3 4

OpenSSL can be used to decrypt the message. e.g `openssl enc -aes-256-cbc -d ...`

1,970 users solved

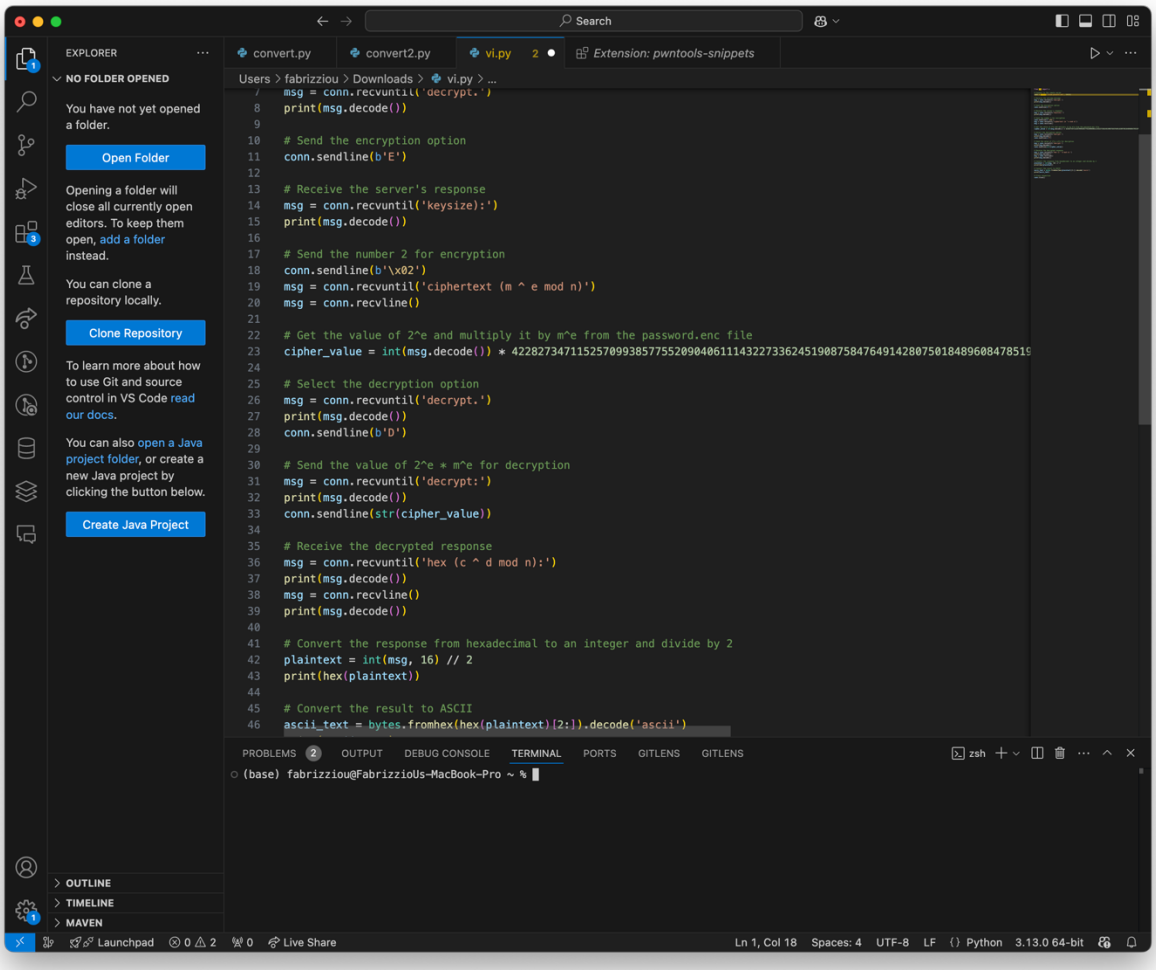
95% Liked

picoCTF{FLAG}

Submit Flag

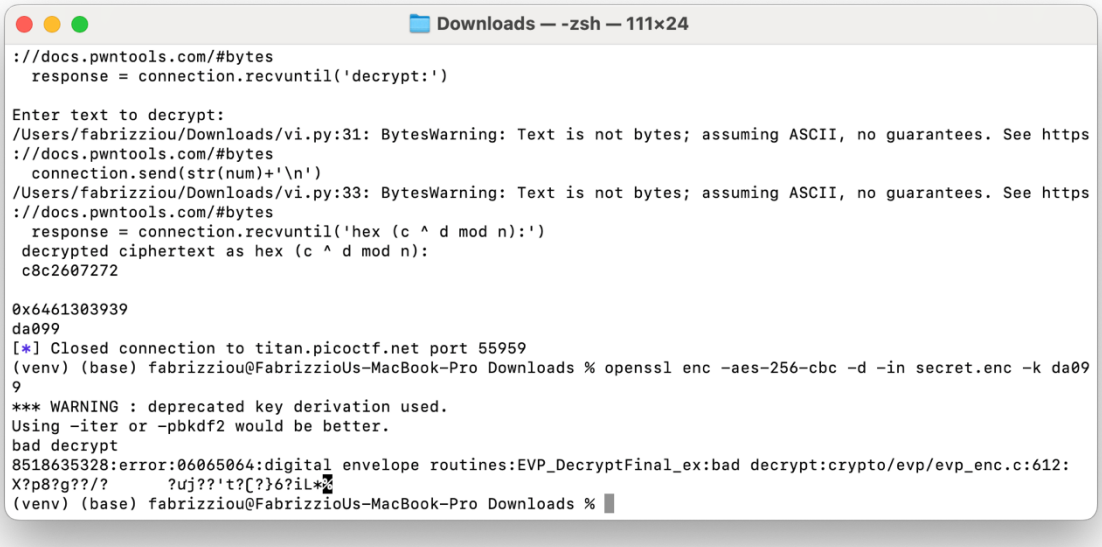
After that we need to lauch instance in the termina we use the nc titan.picoctf.net in my case 55959 and it shows us that we can encrypt or decrypt if we try the message it telss us that good try but be more creative.

```
Downloads — -zsh — 111×24
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat secret.enc
Salted__g???0' ^?       l??h??i???p2'E???Lc}5ق??H?|UV;??h@?=??%
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat password.enc
25751359509831173152345685228579952776621131280760718377634920697639897600186047338132659297722452922230462880 9
829872034354251737553818566230557737574693 4%
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % nc titan.picoctf.net 55959
****************************************
****************THE ORACLE***************
****************************************
what should we do for you?
E --> encrypt D --> decrypt.
E
enter text to encrypt (encoded length must be less than keysize):


encoded cleartext as Hex m: 0

ciphertext (m ^ e mod n) 0

what should we do for you?
E --> encrypt D --> decrypt.
```



To bypass this restriction we used a padding attack base in RSA multiplicative properties, to be more precise we tock the encrypted message and modified it multiplying it by a value of 2. Because RSA fallows the mathematical property D(E(m) *2^e mod n) =2 * m mod n

Now we ask the server to decrypt this modified ciphertext , and we get a scaled version of the decriptedtext dividing by 2 the results we can recover the actual text.



```
Downloads — -zsh — 111×24
://docs.pwntools.com/#bytes
  response = connection.recvuntil('decrypt:')

Enter text to decrypt:
/Users/fabrizziou/Downloads/vi.py:31: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https
://docs.pwntools.com/#bytes
  connection.send(str(num)+'\n')
/Users/fabrizziou/Downloads/vi.py:33: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https
://docs.pwntools.com/#bytes
  response = connection.recvuntil('hex (c ^ d mod n):')
 decrypted ciphertext as hex (c ^ d mod n):
 c8c2607272

0x6461303939
da099
[*] Closed connection to titan.picoctf.net port 55959
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % openssl enc -aes-256-cbc -d -in secret.enc -k da09
9
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
8518635328:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:crypto/evp/evp_enc.c:612:
X?p8?g??/?      ?uj??'t?[?}6?iL*%
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads %
```

As seen in this image I could not recover the plaintext in my computer for some reason it said the key was deprecated.

Upon reviewing the key that decrypted the Python code, we realized that both my classmates and I received the same answer, so I proceeded to use the decoding they came up with, obtaining the correct answer:

Answer: picoCTF({su((3ss_(r@ck1ng_r3@_da099d93}

The terminal image is proof that even using the hint provided by the exercise, openssl enc -aes-256-cbc -d. does not decrypt the message mentioned.