

Ejercicio 1

The screenshot displays the CyberChef web application interface. The browser address bar shows `gchq.github.io`. The top navigation bar includes links for "Download CyberChef", "Last build: 18 days ago - Version 10 is here! Read about the new features here", "Options", and "About / Support".

The main interface is divided into three primary sections:

- Operations:** A sidebar on the left containing a list of operations. The "base64" operation is selected and expanded, showing sub-options: "To Base64", "From Base64", "Show Base64 offsets", "Fernet Decrypt", "Fernet Encrypt", "Fork", "From Base32", "From Base58", "From Base85", "Parse SSH Host Key", "To Base32", "To Base58", and "To Base85". Below this list are sections for "Favourites", "Data format", "Encryption / Encoding", "Public Key", "Arithmetic / Logic", and "Networking".
- Recipe:** The central area shows a recipe configuration for "From Base64". It includes a dropdown menu set to "Alphabet" with the value "A-Za-z0-9+/", a checked checkbox for "Remove non-alphabet chars", and an unchecked checkbox for "Strict mode". At the bottom of this section is a green "BAKE!" button with a chef icon and an "Auto Bake" checkbox.
- Input:** The rightmost section contains the input text: `YidkM0JxZGtwQLRYdHFhR3g2YUhsZmF6TnFLVGwzWVR0clh6YzRNAUV3YUcxcWZRPT8nCG==`.

The output section at the bottom right displays the result of the operation: `b'd3BqdkpBTXtqaGx6aHlfazNqeTL3YTnrXzc4MjUwaG1qfQ=='|`. The interface also features status bars at the bottom of the Input and Output sections, showing character counts and encoding options like "Raw Bytes" and "LF".

Download CyberChef

Last build: 18 days ago - Version 10 is here! Read about the new features here

Options About / Support

Operations

base64

To Base64

From Base64

Show Base64 offsets

Fernet Decrypt

Fernet Encrypt

Fork

From Base32

From Base58

From Base85

Parse SSH Host Key

To Base32

To Base58

To Base85

Favourites

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Recipe

From Base64

Alphabet

A-Za-z0-9+/=

☒ Remove non-alphabet chars

☐ Strict mode

STEP

BAKE!

Auto Bake

Input

kd3BqdkpBtXtqaGx6aHlfazNqeTl3YTNrXzc4MjUwaG1qfQ=='

Output

wpjvJAM{jhlzhy_k3jy9wa3k_78250hmj}

Download CyberChef

Last build: 18 days ago - Version 10 is here! Read about the new features here

Options About / Support

Operations

rot

ROT13

ROT47

ROT8000

Rotate left

Rotate Image

Rotate right

ROT13 Brute Force

ROT47 Brute Force

Parse ObjectID timestamp

Avro to JSON

From UNIX Timestamp

From Octal

Protobuf Decode

Protobuf Encode

Drop bytes

From Float

Remove Diacritics

Remove null bytes

Remove whitespace

Recipe

ROT13 Brute Force

☒ Rotate lower case chars

☒ Rotate upper case chars ☐ Rotate numbers

Sample length 100 Sample offset 0

☒ Print amount Crib (known plaintext string)

STEP

BAKE!

Auto Bake

Input

wpjvJAM{jhLzhy_k3jy9wa3k_78250hmj}

Output

Amount = 7: dwqc0HT{qosgof_r3qf9dh3r_78250otq}

Amount = 8: exrdRIU{rpthpg_s3rg9ei3s_78250pur}

Amount = 9: fyseSJV{squlqh_t3sh9fj3t_78250qvs}

Amount = 10: gztFTKW{trvjri_u3ti9gk3u_78250rvt}

Amount = 11: haugULX{uswksj_v3uj9hl3v_78250sxu}

Amount = 12: ibvhVMY{vtxltk_w3vk9lm3w_78250tyv}

Amount = 13: jcw1WNZ{wuymu_l_x3wl9jn3x_78250uzw}

Amount = 14: kdxjXOA{xvznm_y3xm9ko3y_78250vax}

Amount = 15: leykYPB{ywaown_z3yn9lp3z_78250wby}

Amount = 16: mfzLZQC{zxbpxo_a3zo9mq3a_78250xcz}

Amount = 17: ngamARD{aycyp_b3ap9nr3b_78250yda}

Amount = 18: ohbnBSE{bzdrrq_c3bq9os3c_78250zeb}

Amount = 19: picoCTF{caesar_d3cr9pt3d_78250afc}

Amount = 20: qjdpDUG{dbftbs_e3ds9qu3e_78250bgd}

Amount = 21: rkeqEVH{ecguct_f3et9rv3f_78250che}

Amount = 22: slfrFWI{fdhvdu_g3fu9sw3g_78250dif}

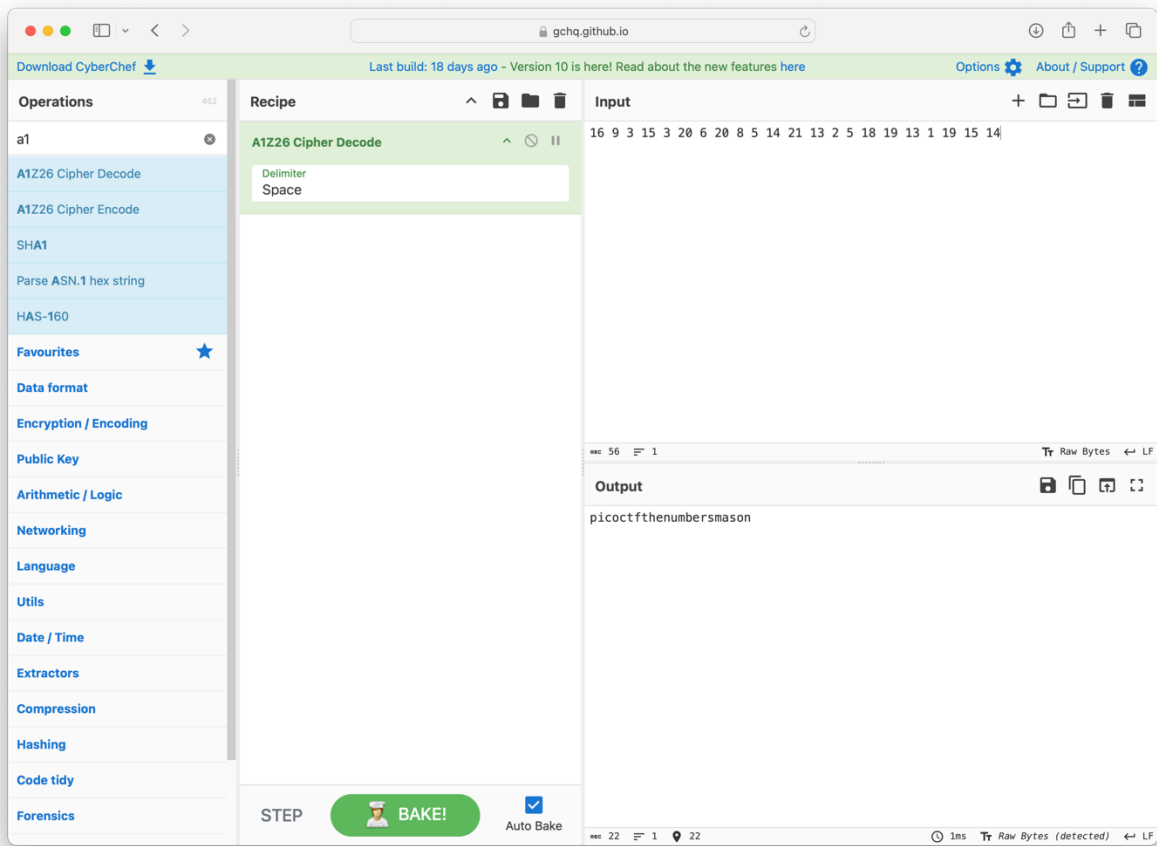
Amount = 23: tmsgGXJ{geiwev_h3gv9tx3h_78250ejg}

Amount = 24: unhthYK{hfjxfw_i3hw9uy3i_78250fkh}

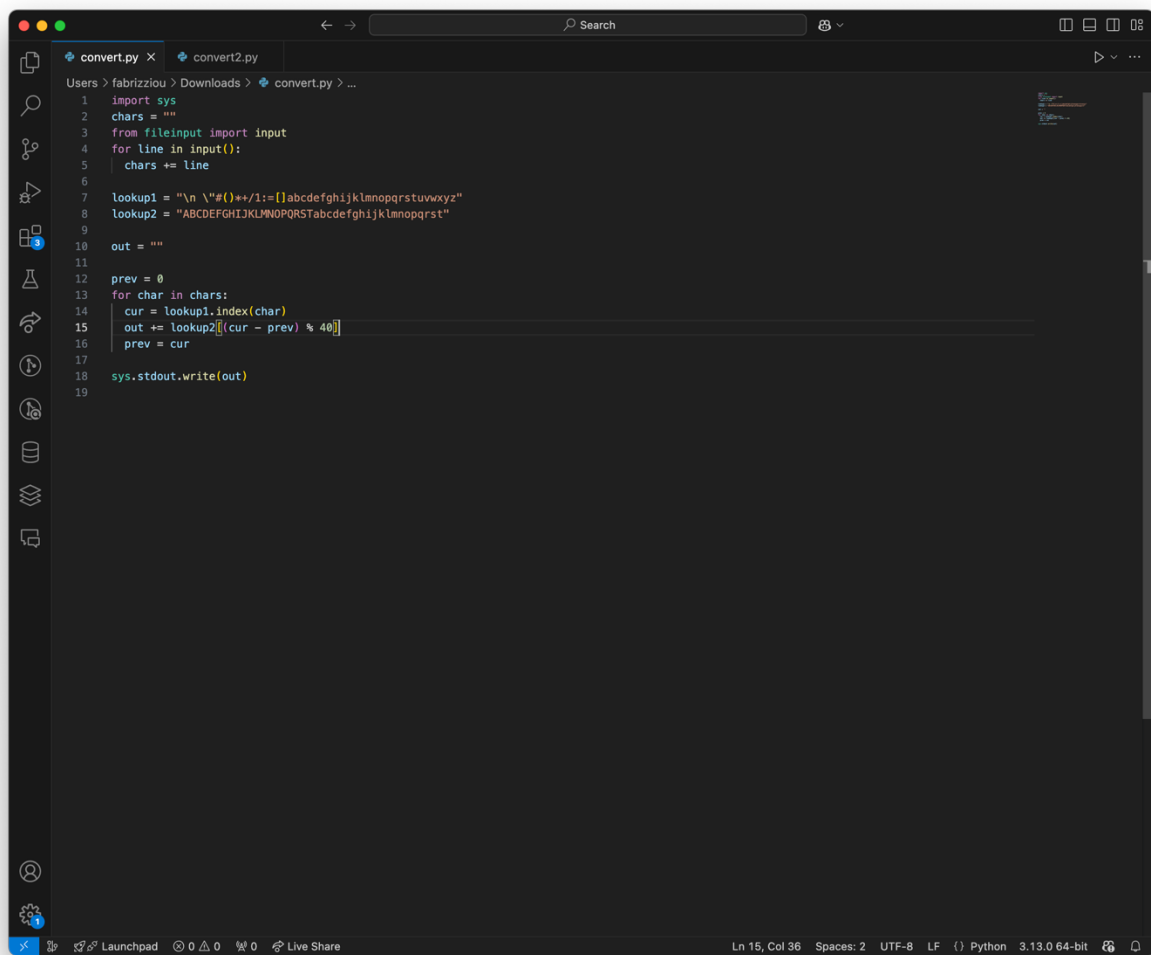
Amount = 25: voiuIZL{igkygx_j3ix9vz3j_78250gli}

864-911 (47 selected)

Ejercicio 2



Ejercicio 3



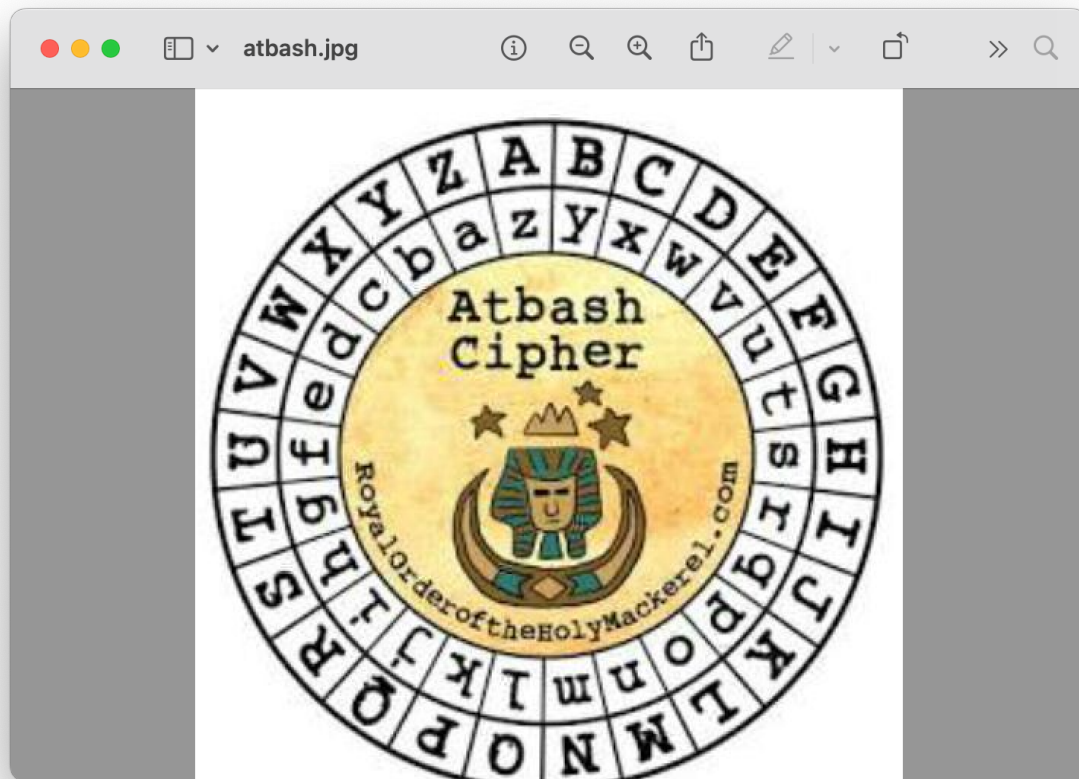
```
1 import sys
2 chars = ""
3 from fileinput import input
4 for line in input():
5     chars += line
6
7 lookup1 = "\n\n#()**/1:=[]abcdefghijklmnopqrstuvwxyz"
8 lookup2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst"
9
10 out = ""
11
12 prev = 0
13 for char in chars:
14     cur = lookup1.index(char)
15     out += lookup2[(cur - prev) % 40]
16     prev = cur
17
18 sys.stdout.write(out)
19
```

Ln 15, Col 36 Spaces: 2 UTF-8 LF () Python 3.13.0 64-bit

```
1 import sys
2 chars = ""
3 from fileinput import input
4 for line in input():
5     chars += line
6
7 lookup1 = "\n \\"#()*+,-./:;[]abcdefghijklmnopqrstuvwxyz"
8 lookup2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
9
10 out = ""
11
12 prev = 0
13 for char in chars:
14     cur = lookup2.index(char)
15     this = lookup1[(cur + prev) % 40]
16     out += this
17     prev = lookup1.index(this)
18
19 sys.stdout.write(out)
20
```

Respuesta picoCTF{adlibs}

Ejercicio 4



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).

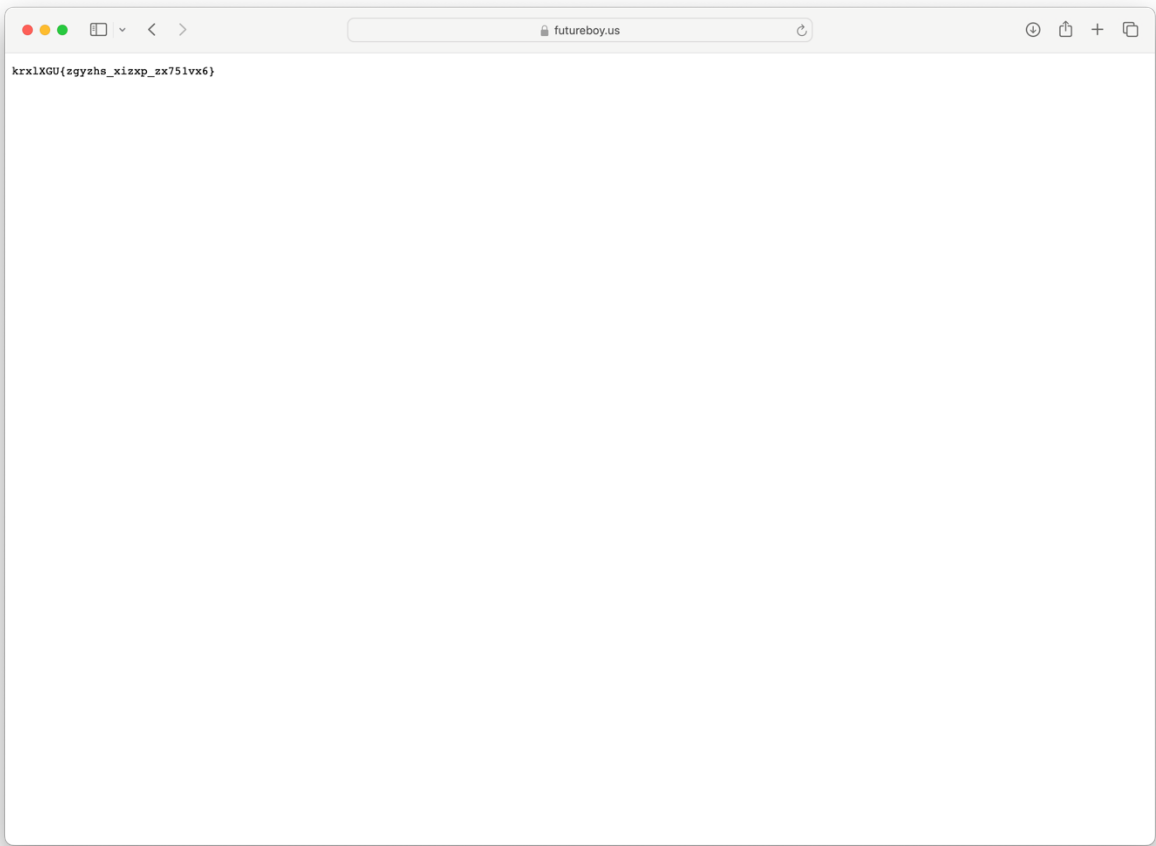
Submit

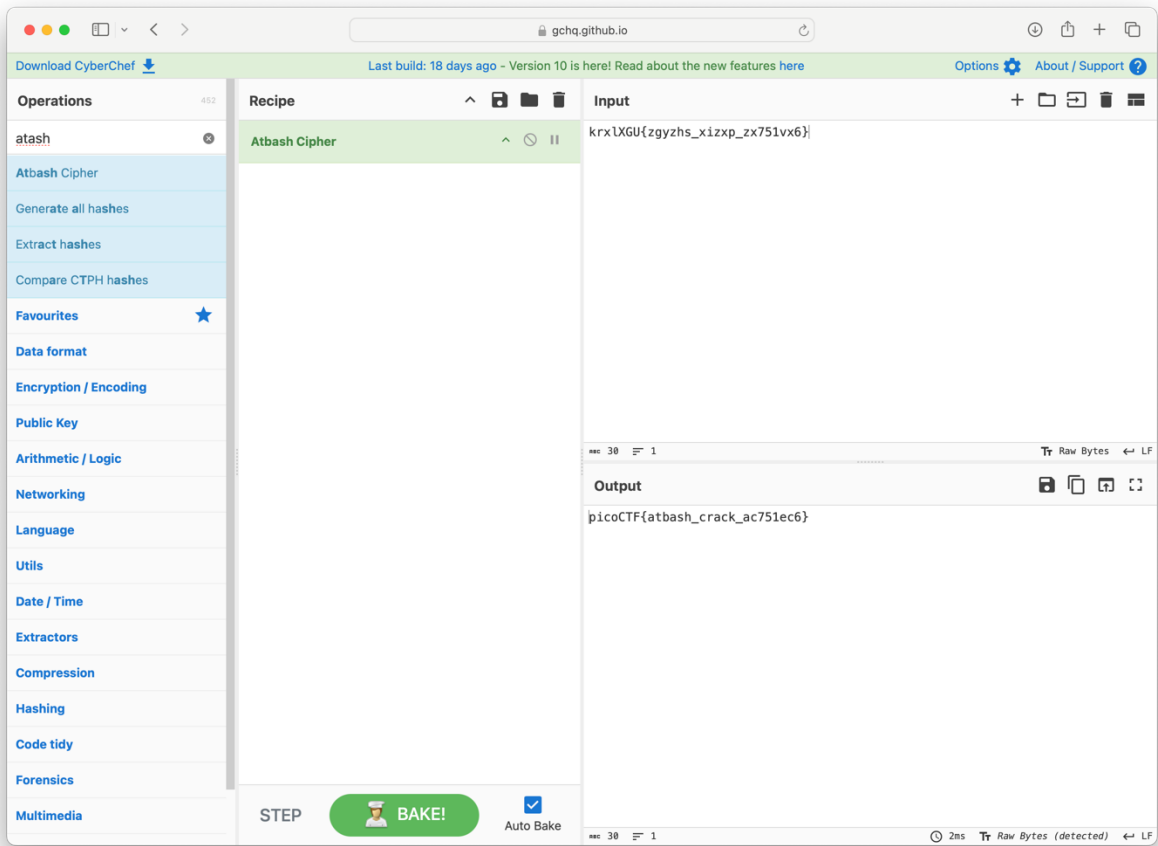
Para utilizar este formulario, primero debe [codificar un archivo](#) .

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 Eliassen](#) .

[Regreso al servidor de inicio de Alan](#)





Ejercicio 5

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



```
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat secret.enc  
Salted_g???0' ^?      l??h??i???p2E???Lc}5j??H?|UV;??h@?=???  
[  
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % cat password.enc  
257513595098311731523456852285799527766211312807607183776349206976398976001860473381326592977224529222304628809]  
8298720343542517375538185662305577375746934?  
(venv) (base) fabrizziou@FabrizzioUs-MacBook-Pro Downloads % nc titan.picocftf.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.
```

```
1 msg = conn.recvuntil('decrypt:')
2 print(msg.decode())
3
4 # Send the encryption option
5 conn.sendline(b'E')
6
7 # Receive the server's response
8 msg = conn.recvuntil('keysize:')
9 print(msg.decode())
10
11 # Send the number 2 for encryption
12 conn.sendline(b'\x02')
13 msg = conn.recvuntil('ciphertext (m ^ e mod n)')
14 msg = conn.recvline()
15
16 # Get the value of 2^e and multiply it by m^e from the password.enc file
17 cipher_value = int(msg.decode()) * 42282734711525709938577552090406111432273362451908758476491428075018489608478515
18
19 # Select the decryption option
20 msg = conn.recvuntil('decrypt:')
21 print(msg.decode())
22 conn.sendline(b'D')
23
24 # Send the value of 2^e * m^e for decryption
25 msg = conn.recvuntil('decrypt:')
26 print(msg.decode())
27 conn.sendline(str(cipher_value))
28
29 # Receive the decrypted response
30 msg = conn.recvuntil('hex (c ^ d mod n):')
31 print(msg.decode())
32 msg = conn.recvline()
33 print(msg.decode())
34
35 # Convert the response from hexadecimal to an integer and divide by 2
36 plaintext = int(msg, 16) // 2
37 print(hex(plaintext))
38
39 # Convert the result to ASCII
40 ascii_text = bytes.fromhex(hex(plaintext)[2:]).decode('ascii')
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS GITLENS

○ (base) fabrizziou@fabrizziou-MacBook-Pro ~ %

Por alguna razon me salia un error de clave deprecada, al intentarlo de nuevo seguia con el error, como hicimos el trabajo en grupo le pregunte a mis compañeros cual era la clave que daba el programa a todos nos dio da099

```
Downloads — -zsh — 111x24

: //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 %
```

al revisarla calve que decifro el codigo de python nos dimos cuenta que tanto a mis compañeros como a mi nos dio la misma respuesta asi que procedi a utilizar la decodificacion que les salio a ellos obteniendo la respuesta correcta la cual es:

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

La imagen del terminal es prueba de que aun utilizando la pista que el ejercicio proporciona `openssl enc -aes-256-cbc -d`. no decodifica el mensaje menciona que la clave esta deprecada.