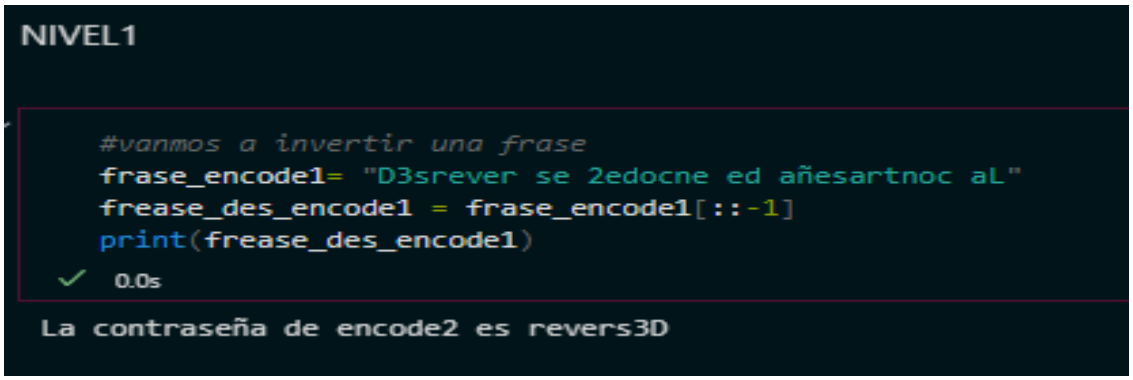
1. ENCODE1

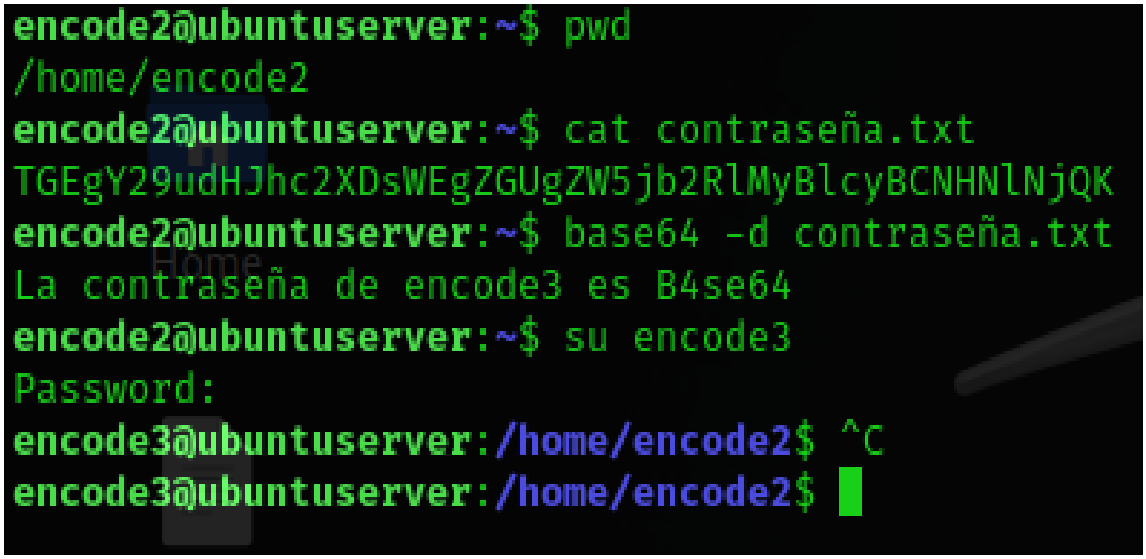
La frase presenta caracteres alfanuméricos por lo que se descarta el cifrado cesar. Parece estar al revés la frase:



Nota: imagen de todos los niveles a conseguir

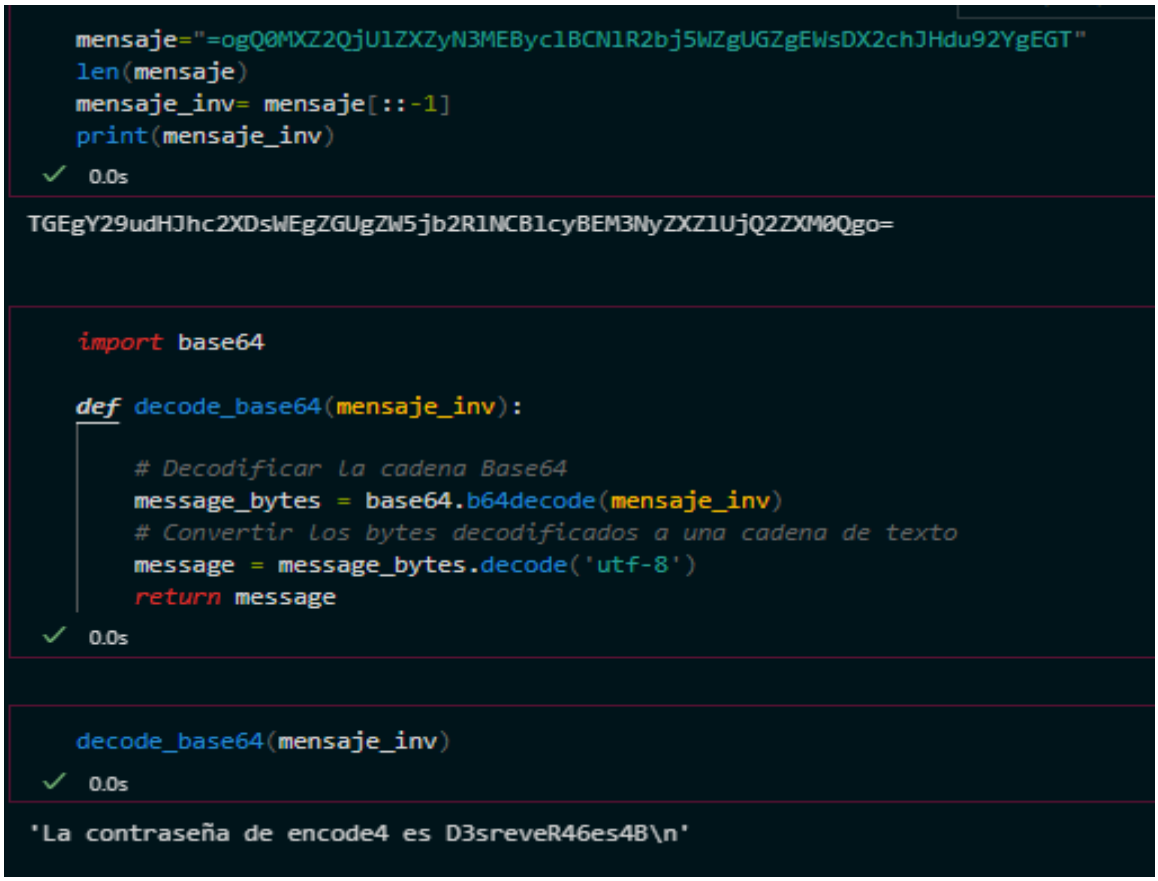
2. ENCODE2

La contraseña presenta signos de ser base64, ya que presenta caracteres alfanuméricos y su longitud total es múltiplo de 4:



3. ENCODE3

Se comprueba que la contraseña tiene 60 dígitos, pero al ejecutarse dice que no es base 64. Realizo probaturas, cambiando el símbolo “=” al final, y hasta intentando darles algún sentido a las letras. Finalmente revierto el orden de la contraseña, siendo válido en base64:



```
mensaje="=ogQ0MXZ2QjU1ZXZyN3MEByc1BCN1R2bj5WZgUGZgEWsDX2chJHdu92YgEGT"
len(mensaje)
mensaje_inv= mensaje[::-1]
print(mensaje_inv)
```

0.0s

```
TGEgY29udHJhc2XDsWEgZGUgZW5jb2R1NCB1cyBEM3NyZXZ1UjQ2ZXM0Qgo=
```

```
import base64

def decode_base64(mensaje_inv):

    # Decodificar la cadena Base64
    message_bytes = base64.b64decode(mensaje_inv)
    # Convertir los bytes decodificados a una cadena de texto
    message = message_bytes.decode('utf-8')
    return message
```

0.0s

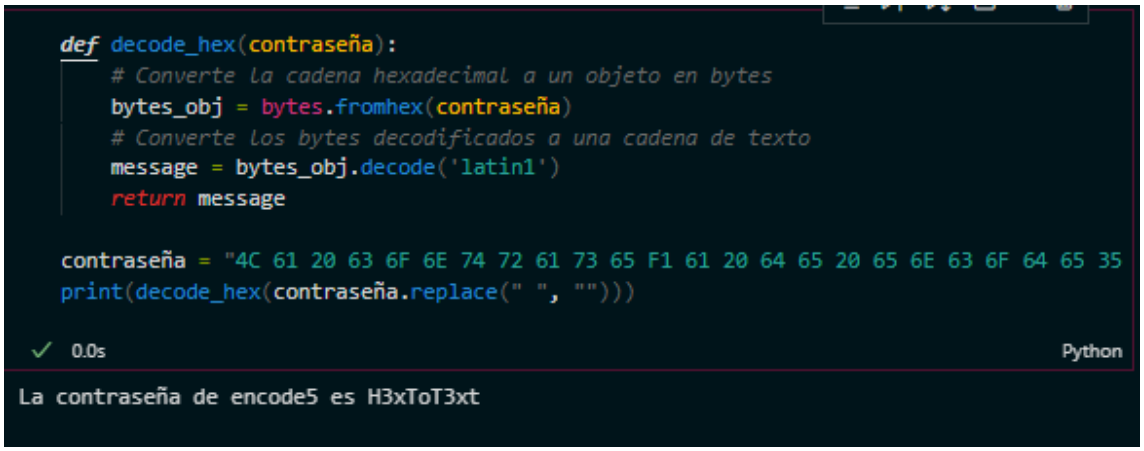
```
decode_base64(mensaje_inv)
```

0.0s

```
'La contraseña de encode4 es D3sreveR46es4B\n'
```

4. ENCODE4

La contraseña está en formato hexadecimal, por lo que vamos a probar pasarla a una cadena simple de texto:



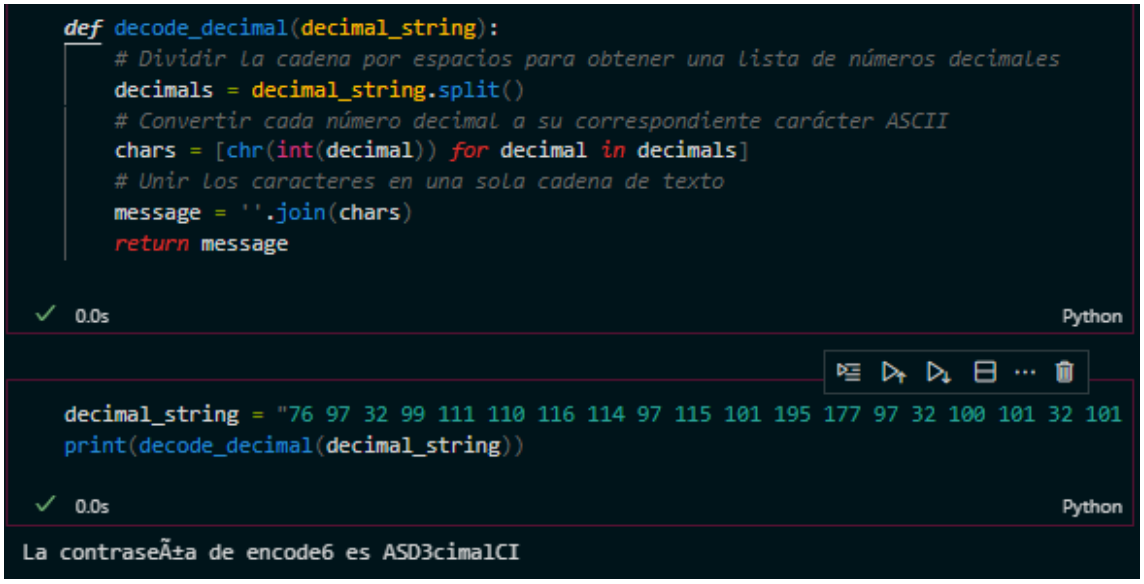
```
def decode_hex(contraseña):
    # Converte la cadena hexadecimal a un objeto en bytes
    bytes_obj = bytes.fromhex(contraseña)
    # Converte los bytes decodificados a una cadena de texto
    message = bytes_obj.decode('latin1')
    return message

contraseña = "4C 61 20 63 6F 6E 74 72 61 73 65 F1 61 20 64 65 20 65 6E 63 6F 64 65 35
print(decode_hex(contraseña.replace(" ", "")))
```

0.0s Python

```
La contraseña de encode5 es H3xToT3xt
```

5. ENCODE5

Aquí tenemos una cadena de números decimales y necesitamos pasarlo a una cadena de texto, por lo que vamos a pasar caDa número decimal a su correspondiente codificación en ASCII y después los unimos en una cadena.



```python
def decode_decimal(decimal_string):
    # Dividir la cadena por espacios para obtener una lista de números decimales
    decimals = decimal_string.split()
    # Convertir cada número decimal a su correspondiente carácter ASCII
    chars = [chr(int(decimal)) for decimal in decimals]
    # Unir los caracteres en una sola cadena de texto
    message = ''.join(chars)
    return message
```

✓ 0.0s Python

```python
decimal_string = "76 97 32 99 111 110 116 114 97 115 101 195 177 97 32 100 101 32 101
print(decode_decimal(decimal_string))
```

✓ 0.0s Python

```
La contraseÃ±a de encode6 es ASD3cimalCI
```

La función “chr” convierte en una lista de comprensión los números decimales a ASCII (estas listas se leen de atrás hacia delante) y con “. join” unimos todos los caracteres sin espacios en una cadena de texto que era la que había que decodificar.

6. ENCODE6

La contraseña parece estar codificada en “URL encoding”, con el que se codifican las URL, reemplazando los caracteres alfanuméricos en porcentajes seguidos de dos dígitos hexadecimales.

Vamos a intentar usar la función “unquote “de la librería de Python “urllib”:

```python
contraseña_6=r"La%20contrase%C3%B1a%20de%20encode7%20es%20URL%3C%27%23%27%3EEncoding"
```

✓ 0.0s Python

```python
from urllib.parse import unquote

def decode_url(contraseña_6):
    # Decodificar la cadena URL encoded
    message = unquote(contraseña_6)
    return message
```

✓ 0.0s Python

```python
print(decode_url(contraseña_6))
```

✓ 0.0s Python

```
La contraseña de encode7 es URL<'#'>Encoding
```

7. ENCODE7

El código que nos encontramos es binario, el cual es conocido como lenguaje maquina compuesto por 0s y 1s.

Usaré el comando “chr” para convertir de binario los 2 tipos de valores (0 y 1) que lo componen, para finalmente unirlo todo, en una cadena de texto con un “join”, decodificando la cadena:

```
    # sseparo la cadena en una lista de cadenas  binarias
    binario = contraseña_7.split()
    # Convierto cada cadena binaria a su correspondiente carácter ASCII
    conversion_ascii = [chr(int(binari, 2)) for binari in binario]
    #convierto la lista de caracteres en una sola cadena de texto
    decoded_string = ''.join(conversion_ascii)
    print(decoded_string)

La contraseÃ±a de encode8 es Text2Binary
```

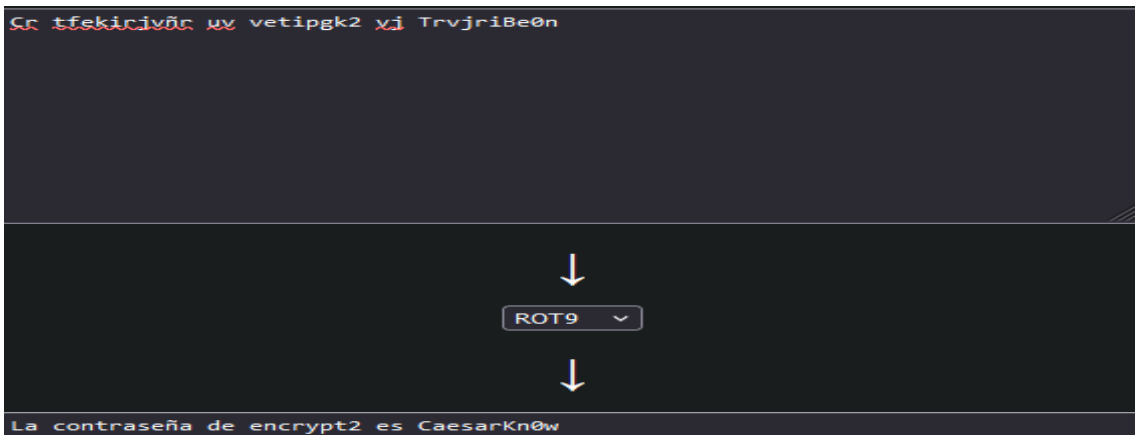
8. ENCODE8 – ENCRYPT1

Aquí encontramos un archivo llamado “flag_mid.txt”, que nos abre camino a otro nivel:

```
encode8@ubuntuserver:~$ cat flag_mid.txt
Enhorabuena.

Has alcanzado el nivel que te permite avanzar al siguiente bloque, abriendo una nueva línea en el reto.

Para comenzar con los ejercicios de cifrado, haremos uso del siguiente usuario:
        - Usuario: encrypt1
        - Contraseña: encrypt1
```

9. ENCRYPT1

Parece que nos encontramos ante un cifrado de sustitución (cesar), para ello vamos a usar un recurso disponible en https://rot13.com/, indicando que es rot9, lo que significa que cada digito se desplaza en un ciclo de 0 a 9, es decir cada digito + 9 en un ciclo de 10(0 -9) y el desencriptado sería el proceso inverso: digito -9 en un ciclo de 10 (0 a -9),



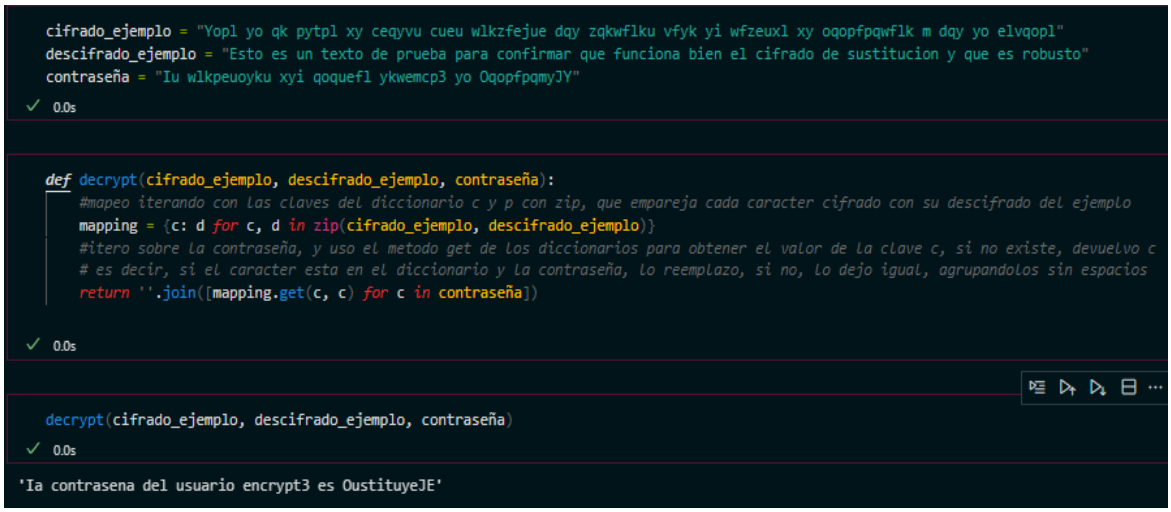
La contraseña de encrypt2 es CaesarKn0w

10. ENCRYPT2

Realizo un mapeo de las frases de ejemplos

Y	o	p	l	q	k	t	x	c	e	y	v	u	w	z	f	j	d	m	CIFRADO
E	s	t	o	u	n	x	d	p	r	a	b	i	f	c	m	q	g	y	DESCIFRADO

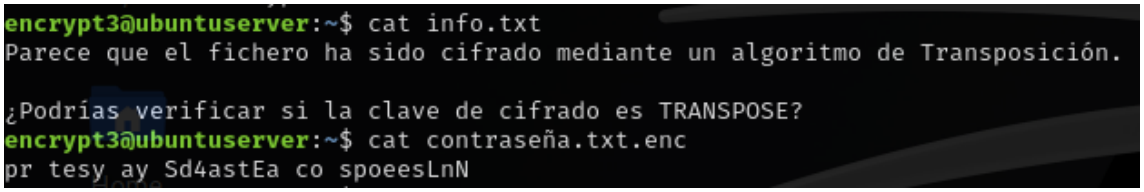
Finalmente, hacemos una función que coge los valores de los ejemplos y el mensaje a descifrar, mapeándolo y devolviendo un mensaje nuevo basado en un diccionario conseguido tras el mapeo. Si la letra de la contraseña esta en el diccionario la cambia y si no, la deja igual.



Con el mapeo simple no ha decodificado ciertos caracteres que son la I, O y la J en mayúsculas siendo sus valores L, S, M, por lo que la contraseña finalmente queda: **SustituyeME**

11. ENCRYPT3

Aquí parece que tenemos un cifrado de transposición, en el que la posición de los caracteres en el texto se cambia siguiendo un sistema, que en este caso es, la clave es “TRANSPOSE”. Si es así, esta clave determinará el numero de columnas en la tabla donde se escribirá el texto en filas, obteniendo el texto cifrado leyendo las columnas en orden. Para descifrar es revertir el proceso:



Intento descifrar por el método de transposición o también llamado permutación, cambiando las posiciones de los caracteres en función de la matriz resultante de dividir el número de caracteres del mensaje y la clave elegida, consiguiendo 2 diccionarios mediante funciones en Python, no llegando a conseguir descifrar el mensaje mediante otra 3 función.

```python
def transposition_matrix(mensaje_cifrado, key):
    # Determinar el número de columnas y filas
    num_cols = len(key) # -> 9
    num_rows = len(mensaje_cifrado) // num_cols # -> 3

    # creamos una matriz de 2 dimensiones con 9 columnas y 3 filas
    matrix = [['' for _ in range(num_cols)] for _ in range(num_rows)]

    # Llenar la matriz con el mensaje cifrado
    index = 0
    #usamos doble bucle en filas y columnas para rellenar la matriz
    for row in range(num_rows):
        for col in range(num_cols):
            #index determina el caracter del mensaje cifrado
            if index < len(mensaje_cifrado):
                #rellenamos la matriz con el mensaje cifrado. un caracter por cada celd
                matrix[row][col] = mensaje_cifrado[index]
                index += 1#incrementamos el index para pasar al siguiente caracter

    return matrix
```

Python

```python
# Crear la matriz de transposición
results = transposition_matrix(mensaje_cifrado, key)
print(results)
```

Python

```
[['p', 'r', 't', 'e', 's', 'y', 'a', 'y', 'S'], ['d', '4', 'a', 's', 't', 'E', 'a', 'c', 'o
```

```python
def obtener_order_key(key):
    alfabet="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    dict_abc = {alfabet[i]: i+1 for i in range(len(alfabet))}
    #oerdenamos las letras de la clave y las mapeamos a su valor numerico een el diccio
    order_key = {letter: i+1 for i, letter in enumerate(dict_abc) if letter in key}
    return order_key
```

Agregar una celda de Markdown

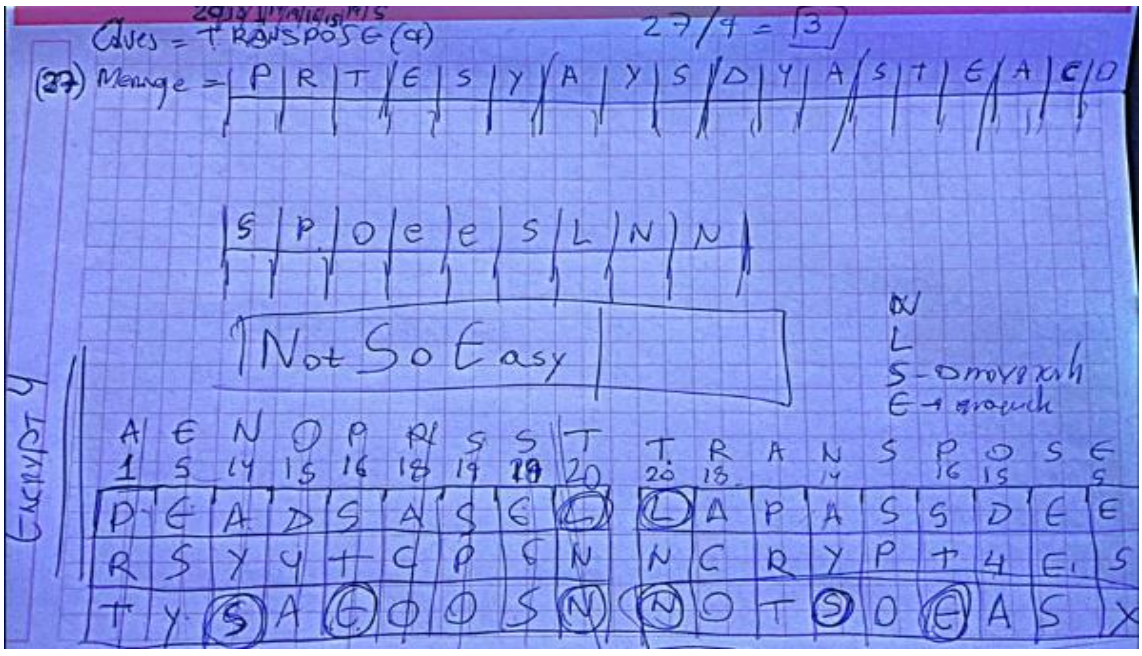
Python

```python
results
```

Python

```
[['p', 'r', 't', 'e', 's', 'y', 'a', 'y', 'S'],
 ['d', '4', 'a', 's', 't', 'E', 'a', 'c', 'o'],
 ['s', 'p', 'o', 'e', 'e', 's', 'L', 'n', 'N']]
```

Por lo que finalmente opte por hacerlo manualmente, siendo el resultado final:



12. ENCRYPT4

```python
mensaje4_cifrado="Gi vefw ui zvivltk5 in MtgecgxDBBmtieimm"
clave4_1="Caesar"
clave4_2="Alkandi"
clave4_3="Vigenere"
clave4_5="Enigma"
```

Python

```python
num=mensaje4_cifrado.replace(" ", "")
len(num)
```

Python

Este cifrado es un cifrado vigenere, descartando permutación con la única clave que sería posible, pero no tiene sentido el mensaje (con la clave alkadi). Por lo tanto, investigamos sobre las características del cifrado vigenere, son al igual que el de transposición se usa una palabra clave para cifrar, pero la diferencia es que en transposición solo hay un cambio de posiciones de letras y en el vigenere usa la clave para aplicar una serie de cifrados cesar al texto desplazando las letras un numero de posiciones. Además, el vigenere tiene patrones repetidos por al rehusar la clave, que transposición no ocurre, por lo que procedo a convertir las letras del mensaje y la clave a sus valores numéricos. Después restaremos los valores de la clave de los valores del mensaje cifrado, para finalmente convertir los valores resultantes de nuevo a letras. Este recurso web realiza el proceso automáticamente.

La pass de encrypt5 es EncryptITVigenere

13. ENCRYPT5

```
encrypt5@ubuntuserver:~$ cat info.txt
Parece que el fichero ha sido cifrado.

¿Podrías verificar si la contraseña es alguna de las siguientes?

- RC4Encryption
- DES3Rules
- Symmetric
- NotSoEasy
encrypt5@ubuntuserver:~$ cat contraseña.txt.des3
Salted__•••7•=$••••••`•9 •)••$]•:•••4Zt•a7•&H•}Mo•K\U•B••
```

El mensaje a desencriptar parece que es del tipo DES3, una versión mejorada del tipo DES.

```
#!/bin/bash

# Archivo cifrado
archivo_cifrado="contraseña.txt.des3"

# Lista de contraseñas posibles
passwords=("RC4Encryption" "DES3Rules" "Symmetric" "NotSoEasy")

# Intentar descifrar el archivo con cada contraseña con el compamdo openssl para una contraseña des3
# el cual es un cifrado mejorado de la version DES, la cual presentaba vulnerabilidad a ataques por fuerza bruta
# el triple des uso 3 vesde el DES, de manera que usa 3 claves diferentes  para el cifrado y descifrado
for password in "${passwords[@]}"; do
    echo "Intentando con contraseña: $password"
    # Comando para descifrar usando PBKDF2, el cual da mayor seguridad al cifrado, aumentando la complejodad del  proceso y añadiendo  salt a la contreaseña
    openssl des3 -d -salt -pbkdf2 -in "$archivo_cifrado" -out "output.txt" -pass pass:"$password"

    # Verificar si la operación fue exitosa ($? se refiere al ultimo comnado usado , qe si es 0 es exitoso)
    if [ $? -eq 0 ]; then
        echo "Descifrado exitoso con contraseña: $password"
        echo "Contenido descifrado:"
        cat output.txt
        exit 0
    else
        echo "Fallo al descifrar con contraseña: $password"
    fi
done

echo "No se pudo descifrar el archivo con las contraseñas proporcionadas."
```

```
encrypt5@ubuntuserver:~$ ./encrypt5.sh
Intentando con contraseña: RC4Encryption
bad decrypt
40A7D7D7A07F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrypt:../providers/implementations/ciphers/ciphercommon_block.c:124:
Fallo al descifrar con contraseña: RC4Encryption
Intentando con contraseña: DES3Rules
bad decrypt
40776544BC7F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad decrypt:../providers/implementations/ciphers/ciphercommon_block.c:124:
Fallo al descifrar con contraseña: DES3Rules
Intentando con contraseña: Symmetric
Descifrado exitoso con contraseña: Symmetric
Contenido descifrado:
La contraseña de encrypt6 es 3DESEncription!
encrypt5@ubuntuserver:~$
```

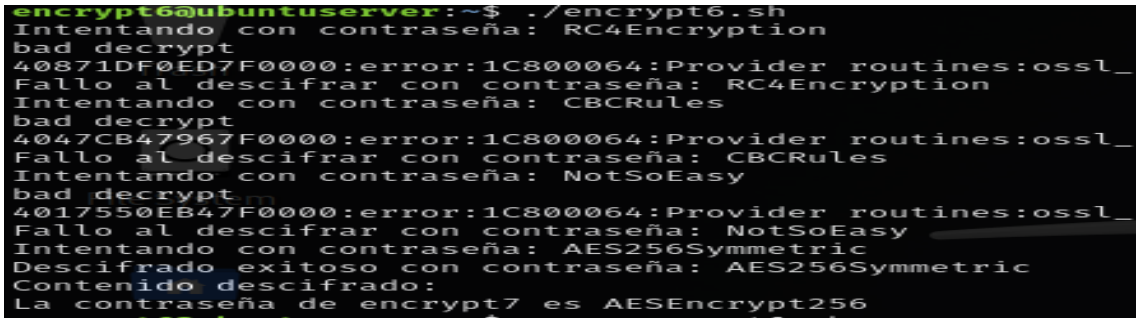
La contraseña de encrypt6 es 3DESEncription!

14. ENCRYPT6

Como anteriormente, parece que la contraseña es tipo AES, y si analizamos las contraseñas, las que mas encajan con este cifrado son el AES256Symmetric y zenAES256, lo que induce a pensar que el cifrado es tipo AES, por observación de las contraseñas que sen de tipo 256 y que es el método más usado actualmente: AES256 CBC. Este cifrado proporciona más seguridad al usar e cifrado por bloques y asegurando la aleatoriedad, pudiendo el mismo texto plano cifrarse de maneras diferentes. Por lo que, adaptamos el script anterior:

```
#!/bin/bash

# Archivo cifrado
archivo_cifrado="contraseña.txt.aes2"

# Lista de contraseñas posibles
passwords=("RC4Encryption" "CBCRules" "NotSoEasy" "AES256Symmetric" "Symmetric" "ZenAES256")

# Intentar descifrar el archivo con cada contraseña
for password in "${passwords[@]}"; do
    echo "Intentando con contraseña: $password"
    # Comando para descifrar usando AES-256-CBC y PBKDF2
    # probamos el cifrado AES256 en modo CBC
    openssl enc -d -aes-256-cbc -salt -pbkdf2 -in "$archivo_cifrado" -out "output.txt" -pass pass:"$password"

    # Verificar si la operación fue exitosa
    if [ $? -eq 0 ]; then
        echo "Descifrado exitoso con contraseña: $password"
        echo "Contenido descifrado:"
        cat output.txt
        exit 0
    else
        echo "Fallo al descifrar con contraseña: $password"
    fi
done

echo "No se pudo descifrar el archivo con las contraseñas proporcionadas."
```

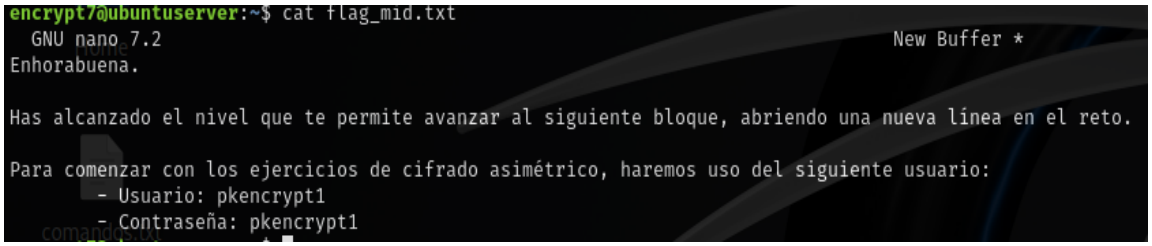
Resultado:



```
encrypt6@ubuntuserver:~$ ./encrypt6.sh
Intentando con contraseña: RC4Encryption
bad decrypt
40871DF0ED7F0000:error:1C800064:Provider routines:ossl_
Fallo al descifrar con contraseña: RC4Encryption
Intentando con contraseña: CBCRules
bad decrypt
4047CB47967F0000:error:1C800064:Provider routines:ossl_
Fallo al descifrar con contraseña: CBCRules
Intentando con contraseña: NotSoEasy
bad decrypt
4017550EB47F0000:error:1C800064:Provider routines:ossl_
Fallo al descifrar con contraseña: NotSoEasy
Intentando con contraseña: AES256Symmetric
Descifrado exitoso con contraseña: AES256Symmetric
Contenido descifrado:
La contraseña de encrypt7 es AESEncrypt256
```

La contraseña de encrypt7 es AESEncrypt256

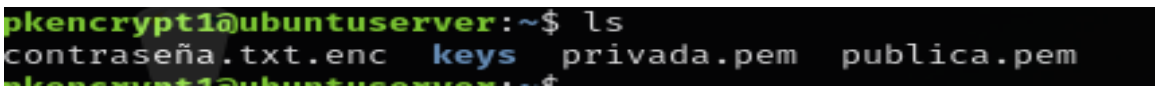
15. ENCRYPT7 - PKENCRYPT1



```
encrypt7@ubuntuserver:~$ cat flag_mid.txt
  GNU nano 7.2                                                New Buffer *
Enhorabuena.

Has alcanzado el nivel que te permite avanzar al siguiente bloque, abriendo una nueva línea en el reto.

Para comenzar con los ejercicios de cifrado asimétrico, haremos uso del siguiente usuario:
        - Usuario: pkencrypt1
        - Contraseña: pkencrypt1
```

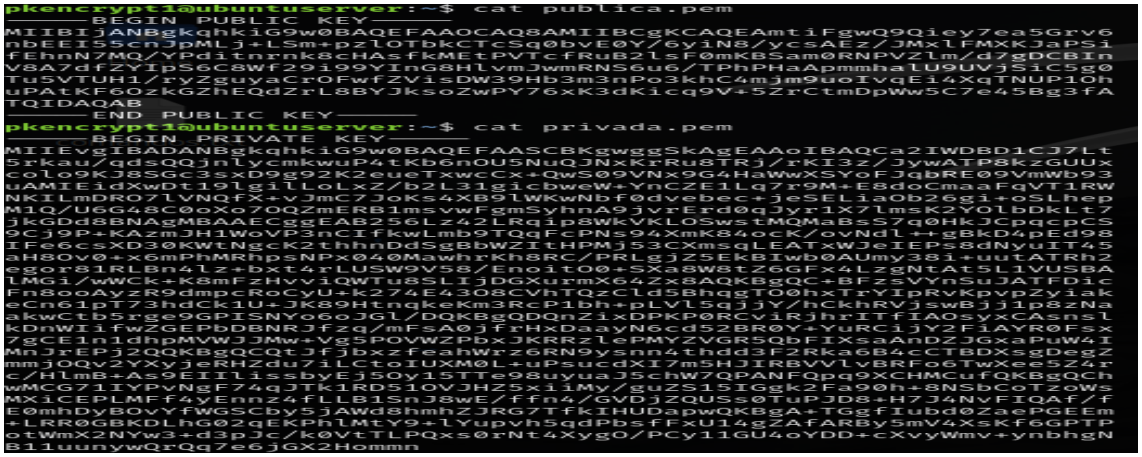
Usuario: pkencrypt1 y Contraseña: pkencrypt1

Muevo todos los archivos a la misma carpeta.

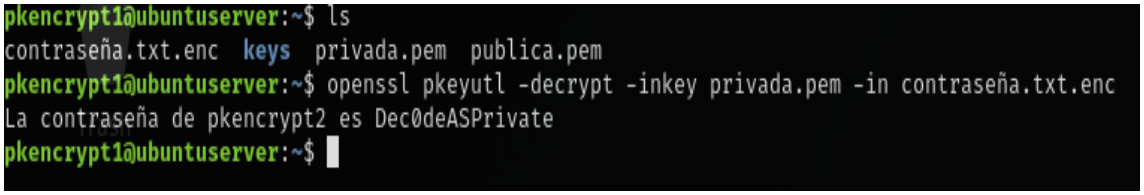


```
pkencrypt1@ubuntuserver:~$ ls
contraseña.txt.enc  keys  privada.pem  publica.pem
```

Analizo los archivos: “.pem”, que representan la clave pública y privada cifrada en RSA.



Utilizo el comando “openssl” “pkeyutl” ya que los archivos han sido cifrados en RSA, siendo muy usado con “pkeyutl” en operaciones con claves públicas, con “-decrypt” para iniciar el proceso de desencriptado y, finalmente usamos “-inkey”, que especifica donde esta la clave privada con la que desencripta el mensaje.



La contraseña de pkencrypt2 es Dec0deASPrivate

16. PKENCRYPT2

Una vez movidos todos los archivos a la carpeta principal, existe un archivo nuevo que parece ser una clave efímera, junto a las contraseñas publica y privada en RSA y el mensaje a descifrar.

En primer lugar, uso la clave privada para desencriptar la clave efímera, la cual me dará como resultado una clave simétrica.

Después, usamos la clave simétrica para desencriptar el mensaje, el cual es tipo AES, ya que viene con la misma terminación que uno anterior (aes2), en el que use “aes256” con “cbc” y “pkbdf2”, por lo que probaremos igual.

```
pkencrypt2@ubuntuserver:~$ cat pkencrypt3,sh
#!/bin/bash

# Archivos
archivo_cifrado="contraseña.txt.aes2"
clave_privada="privada.pem"
archivo_clave_efimera="ephemereal_key.enc"
archivo_clave_simetrica="clave_simetrica.txt"
archivo_salida="output.txt"

# Paso 1: Descifrar la clave efímera usando la clave privada RSA
echo "Descifrando la clave efímera..."
openssl pkeyutl -decrypt -inkey "$clave_privada" -in "$archivo_clave_efimera" -out "$archivo_clave_simetrica"

# Verificar si la operación fue exitosa
if [ $? -eq 0 ]; then
    echo "Clave efímera descifrada exitosamente."
else
    echo "Fallo al descifrar la clave efímera."
    exit 1
fi

# Paso 2: Usar la clave simétrica para descifrar el archivo
echo "Descifrando el archivo con la clave simétrica..."
openssl enc -d -aes-256-cbc -pbkdf2 -in "$archivo_cifrado" -out "$archivo_salida" -pass file:"$archivo_clave_simetrica"

# Verificar si la operación fue exitosa
if [ $? -eq 0 ]; then
    echo "Archivo descifrado exitosamente."
    echo "Contenido descifrado:"
    cat "$archivo_salida"
else
    echo "Fallo al descifrar el archivo."
    exit 1
fi
```

Ejecuto el script:

```
pkencrypt2@ubuntuserver:~$ ./pkencrypt3,sh
Descifrando la clave efímera...
Clave efímera descifrada exitosamente.
Descifrando el archivo con la clave simétrica...
Archivo descifrado exitosamente.
Contenido descifrado:
La contraseña de pkencrypt3 es KeyExchangeEPH
```

La contraseña de pkencrypt3 es KeyExchangeEPH

17. PKENCRYPT3 – HASHING1

```
pkencrypt3@ubuntuserver:~$ cat flag_mid.txt
Enhorabuena.

Has alcanzado el nivel que te permite avanzar al siguiente bloque, abriendo una nueva línea en el reto.

Para comenzar con los ejercicios de funciones hash, haremos uso del siguiente usuario:
        - Usuario: hashing1
        - Contraseña: hashing1
```

Usuario: hashing1 y Contraseña: hashing1

18. HASHING1

En este caso tenemos una carpeta con muchas contraseñas, y parece que solo una es válida para pasar al siguiente nivel. Se muestra un hash del fichero correcto siendo 9f75f653a20dba0796f5011dddc34aaa.

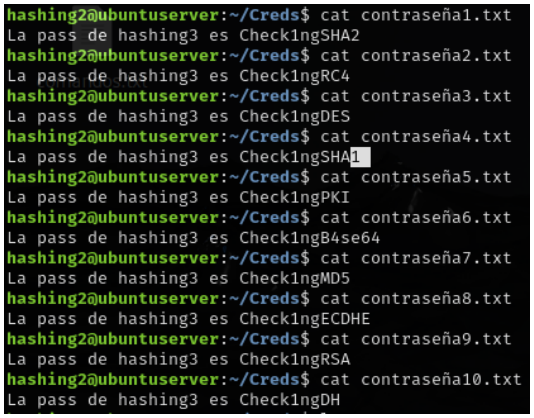
Si analizamos el hash vemos que tiene 32 caracteres, hexadecimales, por lo que muy probamente sea MD5, ya que, aunque es inseguro, es muy usado en almacenamiento de contraseñas.

Por lo que procedo a leer los archivos de las contraseñas, hallando el archivo contraseña6.txt con la terminación MD5, y como tengo 2 opciones antes del bloque compruebo si es la contraseña y correcto:

La pass de hashing2 es Check1ngMD5

19. HASHING2

Aquí nos encontramos un hash 26ed6139d311e851d4efa906bfc78e90f970cedd, el cual presenta 40 caracteres, hexadecimales el cual produce 160 bit con ese número de caracteres, por lo cual es muy probable que sea un hash SHA1.



Una vez observada todas las contraseñas, vemos que la contraseña4.txt termina en SHA1, así que probamos y es la correcta:

La pass de hashing3 es Check1ngSHA1

20. HASHING3

De nuevo, nos presentan un hash con 64 caracteres hexadecimales, siendo al algoritmo sha256, uno de los que produce hash de 64 caracteres (256 bits):

c5f8d03cab180bffb6268f096ebd44840d5d2f5481a75ad588ca02000f572e7c

Vamos a observar nuevamente todas las contraseñas:

```
hashing3@ubuntuserver:~/Creds$ cat contraseña1.txt
La contraseña del usuario hashing4 es BDHeyshdinbSHDYsm
hashing3@ubuntuserver:~/Creds$ cat contraseña2.txt
La contraseña del usuario hashing4 es BDHeasdfNDHSIaDFD
hashing3@ubuntuserver:~/Creds$ cat contraseña3.txt
La contraseña del usuario hashing4 es ASDFPFJsdfainbSHDYsm
hashing3@ubuntuserver:~/Creds$ cat contraseña4.txt
La contraseña del usuario hashing4 es BDHeyshBSDHFYsjdhys
hashing3@ubuntuserver:~/Creds$ cat contraseña5.txt
La contraseña del usuario hashing4 es BDHey6f87sdhs6s8a9
hashing3@ubuntuserver:~/Creds$ cat contraseña6.txt
La contraseña del usuario hashing4 es BDHey8909875DFAD

hashing3@ubuntuserver:~/Creds$ cat contraseña7.txt
La contraseña del usuario hashing4 es BDSDGNNauyf78ds94GDSA
hashing3@ubuntuserver:~/Creds$ cat contraseña8.cat
cat: contraseña8.cat: No such file or directory
hashing3@ubuntuserver:~/Creds$ cat contraseña8.txt
La contraseña del usuario hashing4 es BDHey23dsfad890bSHDYsm
hashing3@ubuntuserver:~/Creds$ cat contraseña9.txt
La contraseña del usuario hashing4 es BDHasDFHsydnbSHDYsm
hashing3@ubuntuserver:~/Creds$ cat contraseña10.txt
La contraseña del usuario hashing4 es BDHDFadyfhbjs6729Sd
```

En este caso, ninguna me sugiere que sea la correcta por lo que procedemos con Python:

```python
#!/usr/bin/env python3

import hashlib # para calcular el hash SHA-256 y otros hashes como MD5, SHA-1, etc.
import os

# Directorio que contiene los archivos .txt
directorio = "/home/hashing3/Creds"
hash_esperado = "c5f8d03cab180bffb6268f096ebd44840d5d2f5481a75ad588ca02000f572e7c"

# Recorrer los archivos con os.listdir, que lista los archivos en el directorio
for nombre_archivo in os.listdir(directorio):
    #con os.path.join se une el directorio con el nombre del archivo
    ruta_archivo = os.path.join(directorio, nombre_archivo)

    # Verificar si el archivo es un archivo .txt
    if nombre_archivo.endswith(".txt"):
        # Leer el contenido del archivo
        with open(ruta_archivo, 'r') as archivo:
            contenido = archivo.read()

        # calculo hashes de contenido del archivo y lo convierto a hexadecimal
        hash_sha256 = hashlib.sha256(contenido.encode()).hexdigest()
        print(f"Hash del archivo {nombre_archivo}: {hash_sha256}")

        # Comparar el hash generado con el hash esperado
        if hash_sha256 == hash_esperado:
            print(f"El archivo que contiene la clave correcta es: {nombre_archivo}")
            break
else:
    print("No se encontró el archivo con la clave correcta.")
```

Resultado:

```
hashing3@ubuntuserver:~$ python3 hashing3.py
Hash del archivo contraseña2.txt: 174db26186a568569aadc4ac6af9173b671212424fdcf6ab6d2faccc9ad2477b
Hash del archivo contraseña10.txt: 7c2b9e521b5ce748ebac403185e6c9c79f2d2fa4f5ace39565ab031f22fbceec
Hash del archivo contraseña7.txt: ceb25ab61c0df572de8c2a89600705bdef2f6f7bc18cb6694310e87ae66cf5d2
Hash del archivo contraseña6.txt: bc3e4e2e8d126631dcadc3b10f59305211ae3349103f18818a49cbc54d062046
Hash del archivo contraseña8.txt: c5f8d03cab180bffb6268f096ebd44840d5d2f5481a75ad588ca02000f572e7c
El archivo que contiene la clave correcta es: contraseña8.txt
```

Procedo a realizar un script en Python buscando el hash del archivo que contine la contraseña resultando que es la contraseña 8:

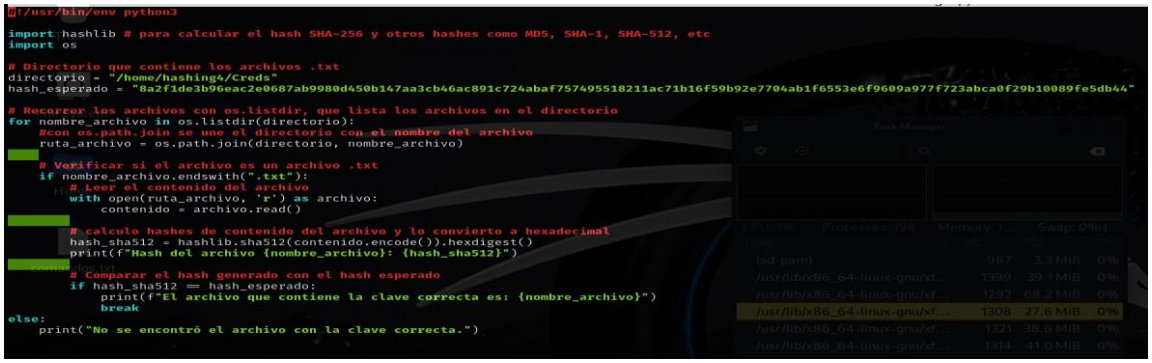
La contraseña del usuario hashing4 es BDHey23dsfad890bSHDYsm

21. HASHING4

De nuevo nos pregunta por el hash del fichero que contiene la contraseña, en este caso que coincida con el hash:

8a2f1de3b96eac2e0687ab9980d450b147aa3cb46ac891c724abaf75749551821
1ac71b16f59b92e7704ab1f6553e6f9609a977f723abca0f29b10089fe5db44

Este hash tiene 128 caracteres en caracteres hexadecimales, el cual, que coincide con el sha512 con la que genera 512 bits, generándose cada 4 bits un digito hexadecimal.



```
#!/usr/bin/env python3

import hashlib # para calcular el hash SHA-256 y otros hashes como MD5, SHA-1, SHA-512, etc
import os

# Directorio que contiene los archivos .txt
directorio = "/home/hashing4/Creds"
hash_esperado = "8a2f1de3b96eac2e0687ab9980d450b147aa3cb46ac891c724abaf75749551821 1ac71b16f59b92e7704ab1f6553e6f9609a977f723abca0f29b10089fe5db44"

# Recorrer los archivos con os.listdir, que lista los archivos en el directorio
for nombre_archivo in os.listdir(directorio):
    #con os.path.join se une el directorio con el nombre del archivo
    ruta_archivo = os.path.join(directorio, nombre_archivo)

    # Verificar si el archivo es un archivo .txt
    if nombre_archivo.endswith(".txt"):
        # Leer el contenido del archivo
        with open(ruta_archivo, 'r') as archivo:
            contenido = archivo.read()

        # calculo hashes de contenido del archivo y lo convierto a hexadecimal
        hash_sha512 = hashlib.sha512(contenido.encode()).hexdigest()
        print(f"Hash del archivo {nombre_archivo}: {hash_sha512}")

        # Comparar el hash generado con el hash esperado
        if hash_sha512 == hash_esperado:
            print(f"El archivo que contiene la clave correcta es: {nombre_archivo}")
            break
else:
    print("No se encontró el archivo con la clave correcta.")
```

Resultado:

```
hashing4@ubuntuserver:~$ python3 hashing4.py
Hash del archivo contraseña2.txt: e686b6250f4ac0cf27ed6c6be970095766ec6e744b62e4bec3bf9a0c8dfb3beaeba9877e5b0a8ff98a3f6ded17770675a4c599d0a602f0c0908b033f6c130b3a
Hash del archivo contraseña10.txt: b014b9daa8d25d2d05a7d0d0aea0a22c5d43d38bfb1a67f659338bbd7b213399344ba7f6c82a73abae3320f82aa5b544c6f02112dcb91fe96cd29529ebb25546
Hash del archivo contraseña7.txt: 055a7be66944b5814fb83937367300b7d0564494ba8bba5b41a1bfe7ef4457c578ad57fe4f71b252ef69e9b64bdf93ff7d10dc8c9d8eb85e63687f1e16a42449
Hash del archivo contraseña6.txt: 0e19444951159f61d4df80eda4b7da88130d9382313ef56b8ca15ea237e94962f6138e991e7d5dda4d11f201c1fc8f253ce8c42e94037fab6cc7e4a2ce59ed4e
Hash del archivo contraseña8.txt: 54e4f40c80b1e901fa23b9feaca8095f88ae062a4eb1b3d2f9405a100a5863425b12eabc51b7798b0488826042564962 3c939fa63f9a61bb0a13c8ba5df16fc8
Hash del archivo contraseña5.txt: aeaef199d0518fd753833d9706e25918449ac9982fd4a15e2cbdbee1b183cd222494e08566f606ba6b4196b32e7e16285aa79e5af0ad477e7db70f2251a7b5da
Hash del archivo contraseña4.txt: 64952b076a71bc73d493f2c29c72e8a1c520326ac74da27a3579d5fc5ac726908f24c72bdf0c0cdf87d8a24ee8c5cd5aab945b23ae715d23823ee172dfb8b786
Hash del archivo contraseña3.txt: 640c84254aa9ef0dabb95f646bffaa38e92735658d6d7b464d8c55ae38e602bf9adf55e9c2c88a1008dacc59bb84069093c4fabd9d1a9d7cb28e011fceba6ea3
Hash del archivo contraseña1.txt: b8451acdfc10de1b5098661a7d0824718684a1bf54c52b45c59c9a6435173b906a5c10b01015c299c8ce5ed59d27067e36f93215e4aede2e70b2fd9ab31099e1
Hash del archivo contraseña9.txt: 8a2f1de3b96eac2e0687ab9980d450b147aa3cb46ac891c724abaf75749518211ac71b16f59b92e7704ab1f6553e6f9609a977f723abca0f29b10089fe5db44
El archivo que contiene la clave correcta es: contraseña9.txt
```

La contraseña del usuario hashing5 es BDHasDFHsydnbSHDYsm

22. HASHING5

Únicamente nos muestra un archivo con un hash con 32 caracteres hexadecimales, y al final viene con .md5, por lo que es un hash de este tipo:

0192023a7bbd73250516f069df18b500

Procedo a descifrar a través de servicios web, siendo la contraseña: admin123

23. HASHING6 - FINAL