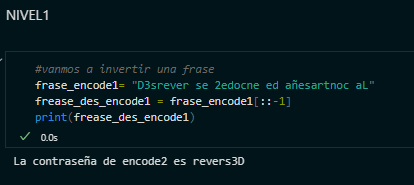
1. **ENCODE1**

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

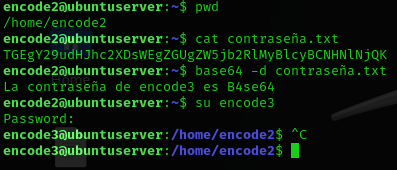


Nota: imagen de todos los niveles a conseguir



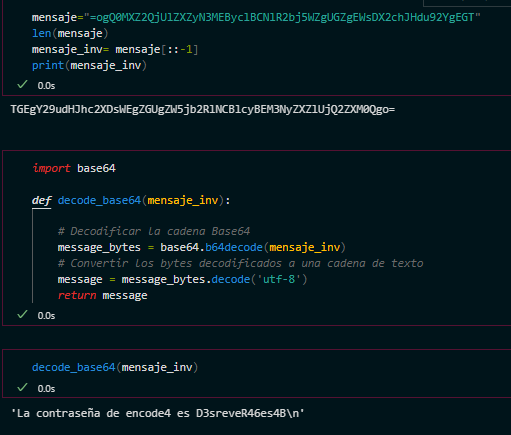
1. **ENCODE2**

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



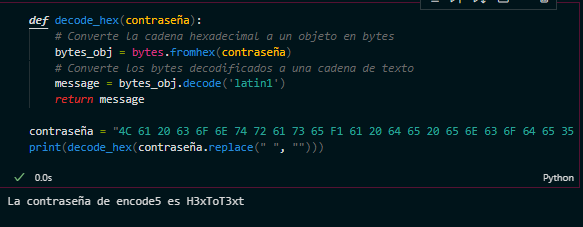
1. **ENCODE3**

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:



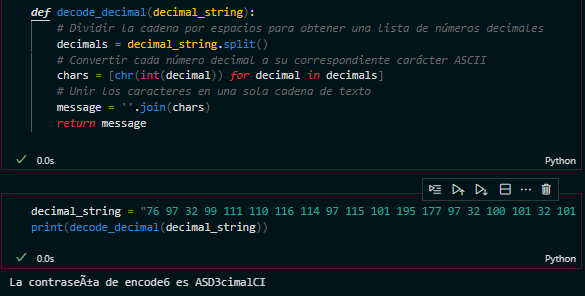
1. **ENCODE4**

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



1. **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.

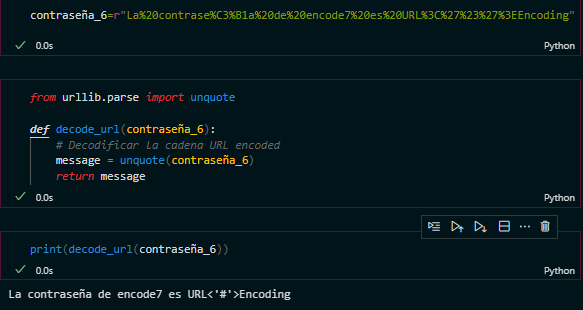


La función “chr” convierte en una lista de comprensión los números decimales a ASCII (estas listas se leen de atrás hacia delante) y con “. join” unimos todos los caracteres sin espacios en una cadena de texto que era la que había que decodificar.

1. **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.

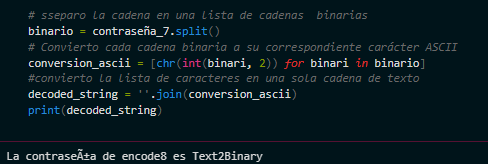
Vamos a intentar usar la función “unquote “de la librería de Python “urllib”:



1. **ENCODE7**

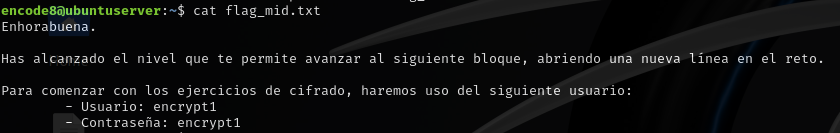
El código que nos encontramos es binario, el cual es conocido como lenguaje maquina compuesto por 0s y 1s.

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:



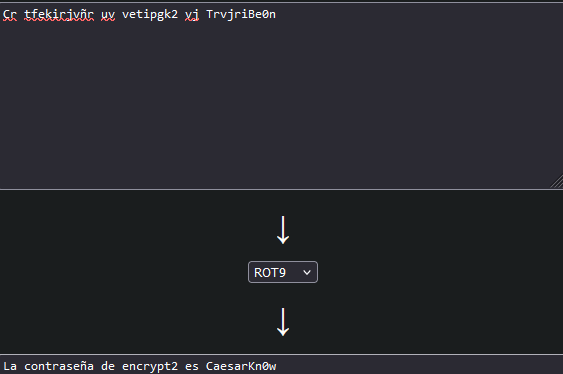
1. **ENCODE8 – ENCRYPT1**

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



1. **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),



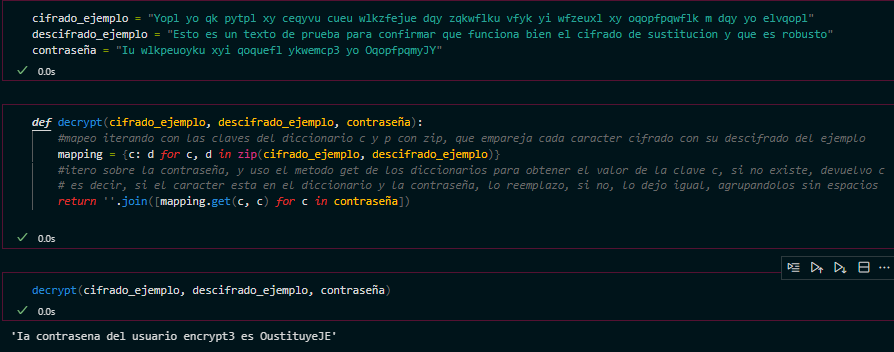
La contraseña de encrypt2 es CaesarKn0w

1. **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 |

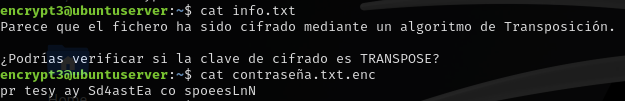
Finalmente, hacemos una función que coge los valores de los ejemplos y el mensaje a descifrar, mapeándolo y devolviendo un mensaje nuevo basado en un diccionario conseguido tras el mapeo. Si la letra de la contraseña esta en el diccionario la cambia y si no, la deja igual.



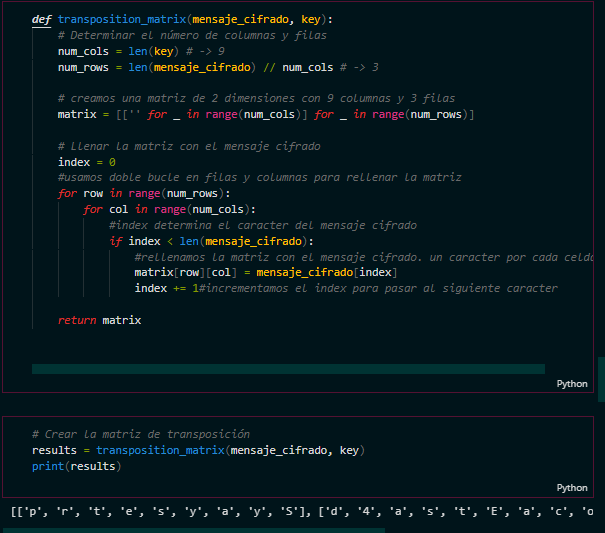
Con el mapeo simple no ha decodificado ciertos caracteres que son la I, O y la J en mayúsculas siendo sus valores L, S, M, por lo que la contraseña finalmente queda: **SustituyeME**

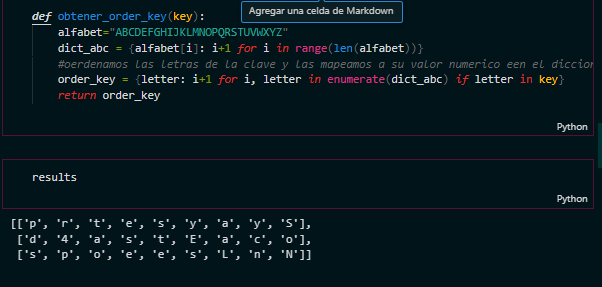
1. **ENCRYPT3**

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:

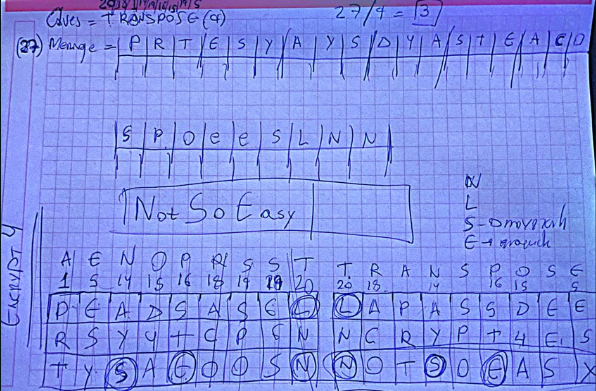


Intento descifrar por el método de transposición o también llamado permutación, cambiando las posiciones de los caracteres en función de la matriz resultante de dividir el número de caracteres del mensaje y la clave elegida, consiguiendo 2 diccionarios mediante funciones en Python, no llegando a conseguir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

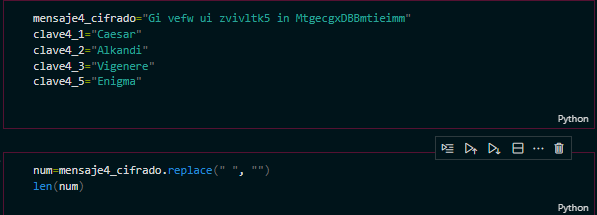




Por lo que finalmente opte por hacerlo manualmente, siendo el resultado final:



1. **ENCRYPT4**

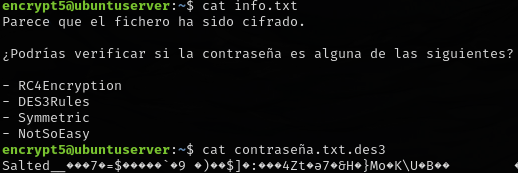


Este cifrado es un cifrado vigenere, descartando permutación con la única clave que sería posible, pero no tiene sentido el mensaje (con la clave alkadi). Por lo tanto, investigamos sobre las características del cifrado vigenere, son al igual que el de transposición se usa una palabra clave para cifrar, 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.

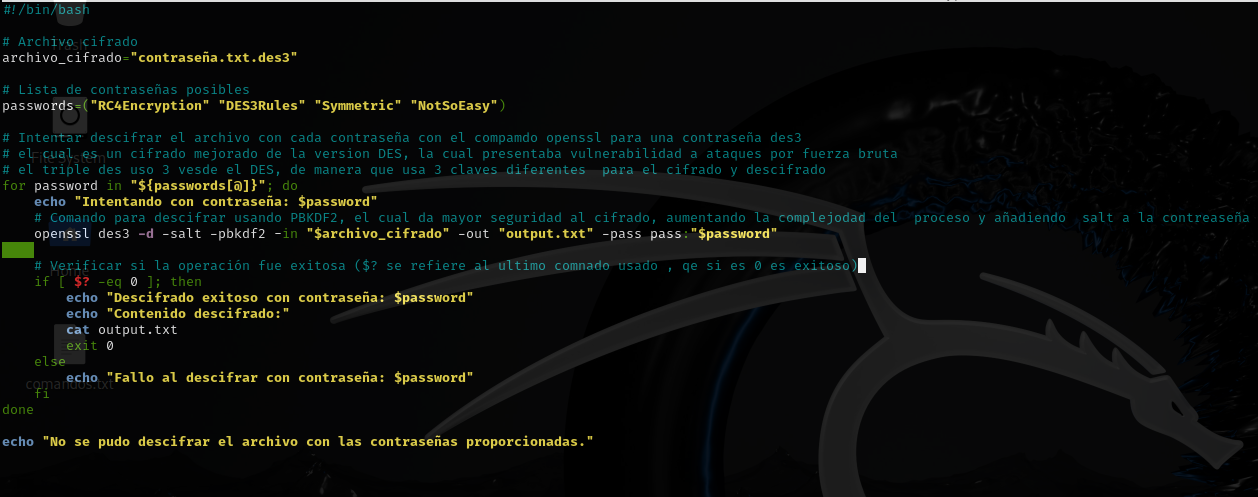


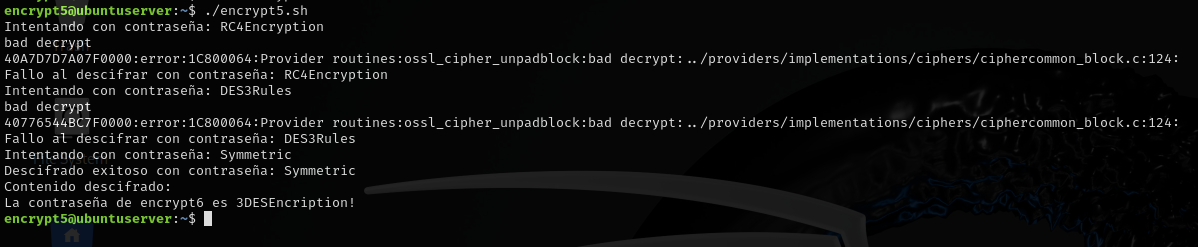
La pass de encrypt5 es EncryptITVigenere

1. **ENCRYPT5**



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

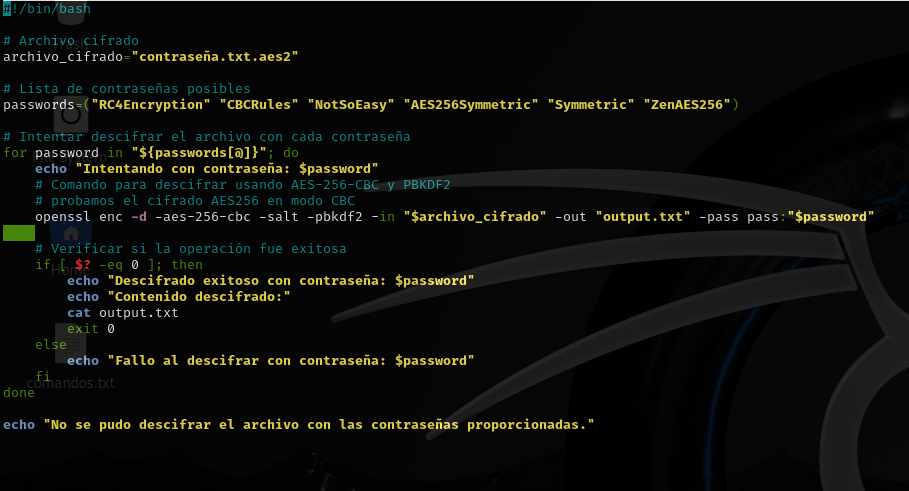




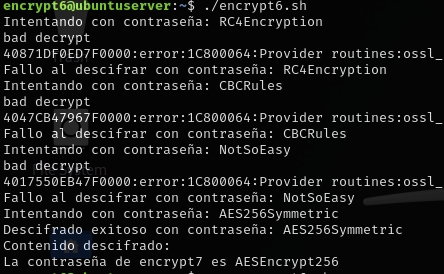
La contraseña de encrypt6 es 3DESEncription!

1. **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:

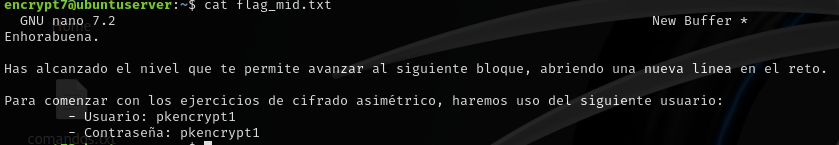


Resultado:



La contraseña de encrypt7 es AESEncrypt256

1. **ENCRYPT7 - PKENCRYPT1**



Usuario: pkencrypt1 y Contraseña: pkencrypt1

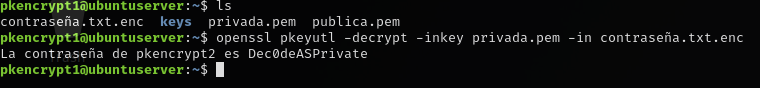
Muevo todos los archivos a la misma carpeta.



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.



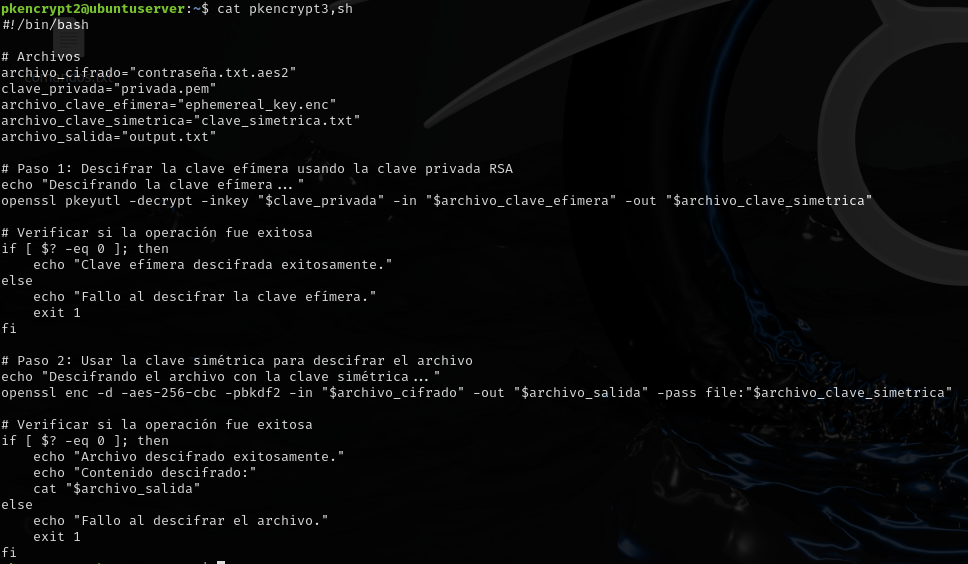
La contraseña de pkencrypt2 es Dec0deASPrivate

1. **PKENCRYPT2**

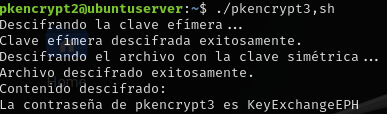
Una vez movidos todos los archivos a la carpeta principal, existe un archivo nuevo que parece ser una clave efímera, junto a las contraseñas publica y privada en RSA y el mensaje a descifrar.

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.

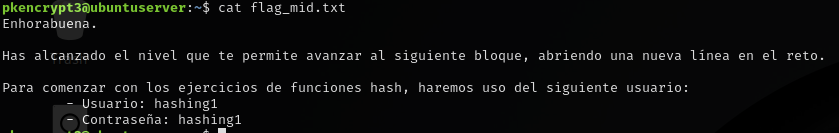


Ejecuto el script:



La contraseña de pkencrypt3 es KeyExchangeEPH

1. **PKENCRYPT3 – HASHING1**



Usuario: hashing1 y Contraseña: hashing1

1. **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.

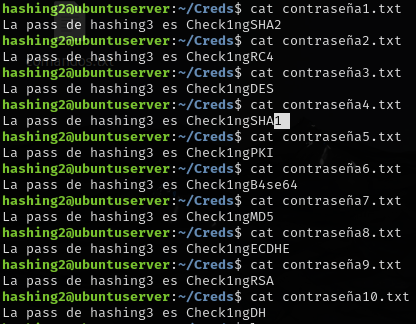
Si analizamos el hash vemos que tiene 32 caracteres, hexadecimales, por lo que muy probamente sea MD5, ya que, aunque es inseguro, es muy usado en almacenamiento de contraseñas.

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

1. **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:

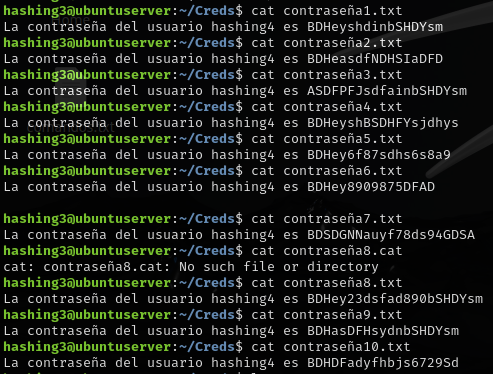
La pass de hashing3 es Check1ngSHA1

1. **HASHING3**

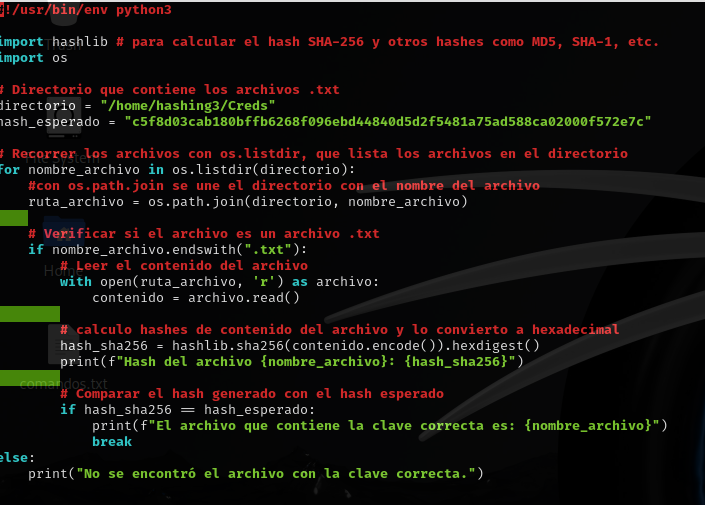
De nuevo, nos presentan un hash con 64 caracteres hexadecimales, siendo al algoritmo sha256, uno de los que produce hash de 64 caracteres (256 bits):

c5f8d03cab180bffb6268f096ebd44840d5d2f5481a75ad588ca02000f572e7c

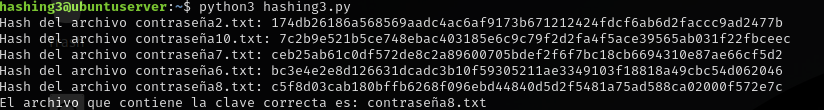
Vamos a observar nuevamente todas las contraseñas:



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



Resultado:



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

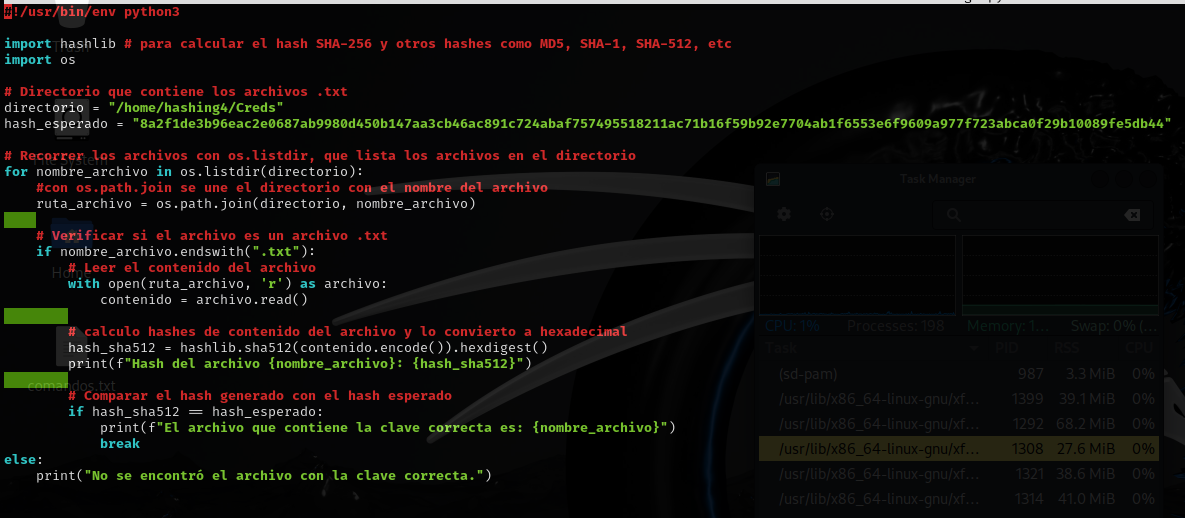
La contraseña del usuario hashing4 es BDHey23dsfad890bSHDYsm

1. **HASHING4**

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

8a2f1de3b96eac2e0687ab9980d450b147aa3cb46ac891c724abaf757495518211ac71b16f59b92e7704ab1f6553e6f9609a977f723abca0f29b10089fe5db44

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.



Resultado:



La contraseña del usuario hashing5 es BDHasDFHsydnbSHDYsm

1. **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



1. **HASHING6 - FINAL**

