

M8. Tarea 3

Se dispone de una aplicación web donde las contraseñas de los clientes están almacenadas hasheadas en MD5 y sin saltear.

Como parte de un proceso de mejora de la seguridad se quiere modificar el almacenamiento de contraseñas para guardarlas resumidas con SHA256 y salteadas.

Con el fin de evitar resetear la contraseña a todos los usuarios; a nuestro equipo de desarrollo se nos ha pedido provisionar automáticamente aquellas contraseñas que actualmente seamos capaces de “revertir” para el máximo número de usuarios posibles.

Para poder descriptar las contraseñas hasheadas en MD5 y sin saltear, he utilizado los servicios de md5decrypt.net. Usando su api le he pasado todas las contraseñas que tenemos y haciendo una consulta en su diccionario de pares de contraseñas y hashes, he generado el fichero plan.txt con aquellas que ha encontrado, dejando en blanco las que no han sido encontradas.

Tengo un script, md5Decrypt.py en Python, que funciona de la siguiente manera:

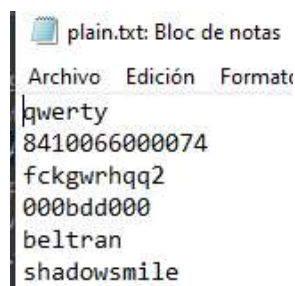
Prepara la url del api que voy a utilizar, a la cual le paso la url, el tipo de hash, el email que uso para acceder al api y el código secreto, que como se puede ver en la imagen no es el código real.

```
# Preparing the url
# Decrypt process using md5decrypt.net
url = "https://md5decrypt.net/en/Api/api.php"
hash_type = "md5"
email = "ikermacaya@gmail.com"
code = "ThisIsNotTheCode"
```

Posteriormente itera sobre el fichero de contraseñas y va comprobando una a una en el api. Aquellas en las que encuentra un valor lo añade al fichero plain.txt y aquellas que no encuentra deja un hueco en blanco:

```
# Read hash by hash and make the request to the API
for password in passwords:
    decryptPasswords = open("decryptPasswords.txt", "a")
    hash = password.split("\n")[0]
    result = requests.get(url + "?hash=" + hash + "&hash_type=" + hash_type)
    decryptPasswords.write(result.text)
    decryptPasswords.write("\n")
    decryptPasswords.close()
```

El fichero plan.txt generado:



```
plain.txt: Bloc de notas
Archivo Edición Formato
qwerty
8410066000074
fckgwrhqq2
000bdd000
beltran
shadowsmile
```

El segundo script, ecryptSHA256.py, también en Python es el que se encarga de hashear las contraseñas obtenidas en SHA256 pero esta vez salteadas.

Para ello he usado dos librerías de Python hashes y default_backend.

Este script genera un nuevo fichero: new_passwords.txt con los resultados de plain.txt con las contraseñas salteadas con una parte fija y con una dinámica. La parte fija es "Modulo8_Iker_" y la parte dinámica es el número de línea de la contraseña en el fichero: "_lineXXX"

El script funciona de la siguiente manera:

Lo primero itera y enumera el contenido del fichero plain.txt, usando en cada iteración el contenido de la línea y el número de la misma.

Si la longitud de la contraseña es mayor de 1, es decir, no esta en blanco, se genera un nuevo string al que se le añade los saltos mencionados anteriormente, fijo y el dinámico.

Esta string es hasheada a SHA256 con el método Hash de la librería hashes y su valor codificado en UTF8 se guarda en new_passwords.txt

En el caso de que la línea este en blanco, esta se mantiene así en el nuevo fichero.

```
for lineNumber, password in enumerate(passwords):
    if len(password) > 1:
        saltPassword = ("Modulo8_Iker_" + password.split("\n")[0] + "_line" + str(lineNumber))
        digest = hashes.Hash(hashes.SHA256(), default_backend())
        digest.update(saltPassword.encode('utf8'))
        newPasswords = open("new_passwords.txt", "a")
        newPasswords.write(digest.finalize().hex())
        newPasswords.write("\n")
        newPasswords.close()
    else:
        newPasswords = open("new_passwords.txt", "a")
        newPasswords.write("\n")
        newPasswords.close()
```

Fichero new_passwords.txt:

 new_passwords.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
2c554f41275edb29e34e40c7a8151b4555bb245f1782a346fd7bacfddb8bc40d
a773bf3b277355a69eb356caba5396edf5935ca7215f06b5813a0a9cf933f56a
67edbc00c7b9822d8187cd6685d426c4b9bb27273336b126e376cbdd0dd4fab0
6222757e3943272d59238a159fffe22171fd3cbc271e8b32836d91f49e07a8b9
8dc9f296685f97530689e816c6b92407e61b6fff61e5bfe1792e8ac223ac4d4c
781653e9bab2ae528810ea065de1835e6dc552ce51d4f9bf1a3eb1bfe36497b0
e00fce05006fcd7c053512810a5dd07c956788d7d9dff51019e2cd50ce4f4c9e
ef170a83b341a89c4244e60a759cf2757a0663da911c05c1b36a0a3bd3029e9a
324dca50f4dbbcc0a621aba2e0a82182111f2aea3328e6fd95faf5e51b6a78b4
773c03e731b923cd70d90c9d58caa42b56c6b9bca7ea07aba7cc9b7b54ebb522
```