

RESOLVIENDO LOS RETOS BÁSICOS DE ATENEA (CCN-CERT)

Vamos a resolver los retos Básicos de la plataforma <https://atenea.ccn-cert.cni.es>.

Taparé los flags para que al menos tengáis que molestaros en leer cómo los he resuelto.

RETO 1 – HASH

ENUNCIADO:

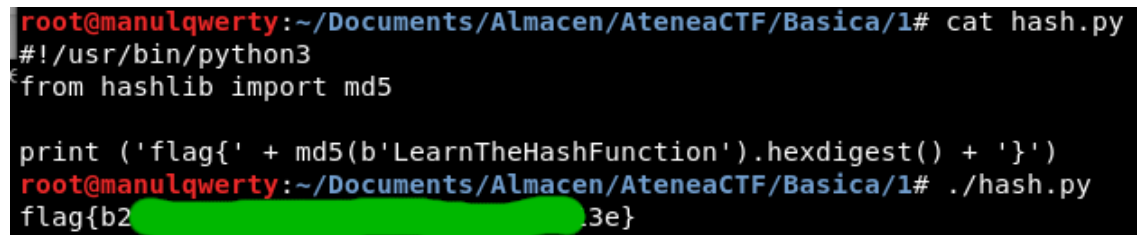
La contraseña para superar este reto es LearnTheHashFunction

Tendrás que calcular su hash md5 y ponerla en el formato de la plataforma, esto es: flag{md5}

SOLUCIÓN:

La resolución de este primer reto es muy sencilla, bastará con usar el módulo md5 de la librería hashlib de python

```
1#!/usr/bin/python3
2from hashlib import md5
3
4print ('flag{' + md5(b'LearnTheHashFunction').hexdigest() + '})'
```



```
root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/1# cat hash.py
#!/usr/bin/python3
from hashlib import md5

print ('flag{' + md5(b'LearnTheHashFunction').hexdigest() + '})'
root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/1# ./hash.py
flag{b2[REDACTED]3e}
```

RETO 2 – HASH 2

ENUNCIADO:

La contraseña para superar este reto es ThisIsAMoreSecureHashFunction

Tendrás que calcular su hash sha256 y posteriormente calcular su md5 para poder poner la solución en el formato de la plataforma, esto es: flag{md5}

SOLUCIÓN:

Este segundo reto es muy parecido al primero, esta vez usaremos 'update' de la librería hashlib para que quede más ordenado:

```

1#!/usr/bin/python3
2import hashlib
3
4s = hashlib.sha256()
5s.update(b"ThisIsAMoreSecureHashFunction")
6m = hashlib.md5()
7m.update(s.hexdigest().encode('utf-8'))
8print("flag{"+m.hexdigest()+"}")

```

```

root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/2# cat hash2.py
#!/usr/bin/python3
import hashlib

s = hashlib.sha256()
s.update(b"ThisIsAMoreSecureHashFunction")
m = hashlib.md5()
m.update(s.hexdigest().encode('utf-8'))
print("flag{"+m.hexdigest()+"}")
root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/2# ./hash2.py
flag{dd[REDACTED]b0}
root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/2#

```

RETO 3 – HASH 3

ENUNCIADO:

Para superar este reto deberás calcular la cadena de texto cuyo hash md5 se corresponde con el siguiente: 54f662a095fa3d5fbbdaac72d176701b

Una vez obtenida, deberás poner dicha cadena de texto en mayúsculas y calcular su hash md5 para poder enviar la solución siguiendo el formato de la plataforma: flag{md5}

SOLUCIÓN:

El método es utilizar john o hashcat desde nuestra máquina con un diccionario, para esto se suele usar el rockyou.txt

```

1echo '54f662a095fa3d5fbbdaac72d176701b' > hash.txt
2hashcat -m 0 -a 0 hash.txt /usr/share/wordlists/rockyou.txt
3# o bien
4john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

```

```

root@manulqwerty:/tmp# john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
masterofpuppets (?)
lg 0:00:00:00 DONE (2018-10-02 00:41) 33.33g/s 2616Kp/s 2616Kc/s 2616KC/s mattyb..martin5
Use the "--show" option to display all of the cracked passwords reliably
Session completed

```

Como veis, el resultado de crackear este hash es : masterofpuppets, vamos a convertirlo a mayúsculas y obtener el flag:

```

1#!/usr/bin/python3
2import hashlib
3
4m = hashlib.md5()
5m.update('masterofpuppets'.upper().encode('utf-8'))
6print("flag{"+m.hexdigest()+"}")

```

```

1  #!/usr/bin/python3
2  import hashlib
3
4  m = hashlib.md5()
5  m.update('masterofpuppets'.upper().encode('utf-8'))
6  print("flag{"+m.hexdigest()+"}")
7
Status root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/3# ./hash3.py
Compiler flag{f3[REDACTED]22}

```

RETO 4 – BASE64

ENUNCIADO:

Para superar este reto tendrás que decodificar el fichero adjunto y poner la contraseña en el formato de la plataforma, esto es: flag{md5}

Contenido del fichero:

```

1UmVjdWVyZGEgcXVlIGN1YW5kbyBjb2RpZm1jYXMgYWxnbyBlbiBiYXNlNjQgTk8gbG8gZXN0w6Fz
2IGNpZnJhbmRvLCBzaW5vIHF1ZSBzaW1wbGVtZW50ZSBsbyBlc3RlIHJldG8gZXM6IHJlY3VlcmRhcXVlYmFz
3CkxhIGNvbW50ZSBzaW50ZSBsbyBlc3RlIHJldG8gZXM6IHJlY3VlcmRhcXVlYmFz
4ZTY0Tk9lc2NpZnJhc0KCg==

```

SOLUCIÓN:

Para solucionar este reto bastaría con usar cualquier web que decodifique Base64 o con bash:

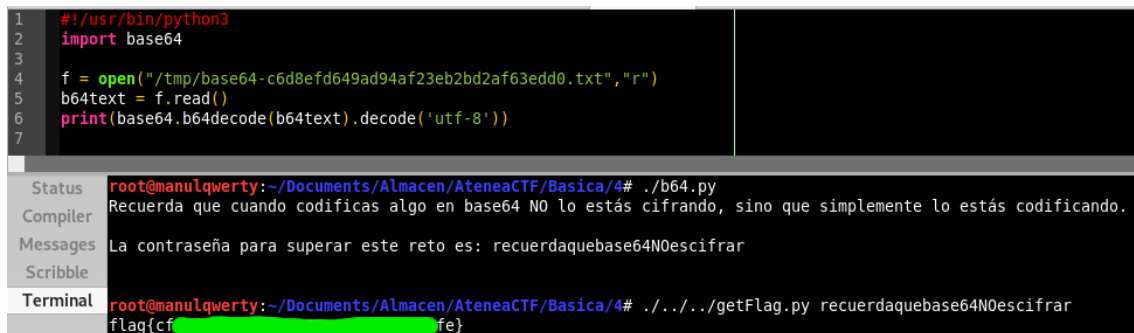
```

1 | base64 -d base64 -c6d8efd649ad94af23eb2bd2af63edd0.txt

```

Pero para seguir con la dinámica, también lo resolví con python:

```
1#!/usr/bin/python3
2import base64
3
4f = open("/tmp/base64-c6d8efd649ad94af23eb2bd2af63edd0.txt", "r")
5b64text = f.read()
6print(base64.b64decode(b64text).decode('utf-8'))
```



```
1#!/usr/bin/python3
2import base64
3
4f = open("/tmp/base64-c6d8efd649ad94af23eb2bd2af63edd0.txt", "r")
5b64text = f.read()
6print(base64.b64decode(b64text).decode('utf-8'))
```

Status root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/4# ./b64.py

Compiler Recuerda que cuando codificas algo en base64 NO lo estás cifrando, sino que simplemente lo estás codificando.

Messages La contraseña para superar este reto es: recuerdaquebase64N0escifrar

Scribble

Terminal root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/4# ../../getFlag.py recuerdaquebase64N0escifrar

```
flag{cf[REDACTED]fe}
```

RETO 5 – ASCII

ENUNCIADO:

Los códigos del 33 al 126 se conocen como caracteres imprimibles, y representan letras, dígitos, signos de puntuación y varios símbolos.

Para pasar este reto deberás encontrar los caracteres correspondientes a la siguiente codificación ASCII:

080 097 115 115 119 111 114 100 032 112 097 114 097 032 115 117 112 101 114 097 114 032 101 108
032 114 101 116 111 058 032 084 104 101 065 083 067 073 073 084 097 098 108 101 033

SOLUCIÓN:

Para resolver este reto debemos ir convirtiendo a char cada código ascii, para ello recorreremos la cadena que nos dan separando por los espacios. Usaremos la función split para esto:

```
#!/usr/bin/python3
1
2
3msg = ''
4text = '080 097 115 115 119 111 114 100 032 112 097 114 097 032 115 117 112 101 114 097 114 032 101 108 032 114 101 116 111 058 032 084 104 101 065 083 067 073 073 084 097 098 108 101 033'
5for i in text.split(' '):
6    msg += chr(int(i))
7print(msg)
```

```
1  #!/usr/bin/python3
2
3  msg = ''
4  text = ''080 097 115 115 119 111 114 100 032 112 097 114 097 032 115 117 112 101 114 097 114
5  032 101 108 032 114 101 116 111 058 032 084 104 101 065 083 067 073 073 084 097 098 108 101 033''
6  for i in text.split(' '):
7      msg += chr(int(i))
8  print (msg)
```

Status	root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/5# ./asc.py
Compiler	Password para superar el reto: TheASCIITable!
Messages	root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/5# ../../getFlag.py TheASCIITable! flag{ba[REDACTED]71}

RETO 6 – HEX

ENUNCIADO:

Para pasar este reto deberás decodificar la siguiente cadena hexadecimal:

50617373776f72643a2044346d7054686548337821

SOLUCIÓN:

Para resolver este reto podríamos usar cualquier web que nos convierta hexadecimal en ascii como: <https://www.rapidtables.com/convert/number/hex-to-ascii.html>

Os incluyo también la resolución usando python:

```
1#!/usr/bin/python3
2
3print(bytes.fromhex('50617373776f72643a2044346d7054686548337821').decode('utf-8'))
4#python 2 : print ('50617373776f72643a2044346d7054686548337821'.decode('hex'))
```

```
1  #!/usr/bin/python3
2
3  print(bytes.fromhex('50617373776f72643a2044346d7054686548337821').decode('utf-8'))
4  #python 2 : print ('50617373776f72643a2044346d7054686548337821'.decode('hex'))
5
```

Status	root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/6# ./h3x.py
Compiler	Password: D4mpTheH3x!
Messages	root@manulqwerty:~/Documents/Almacen/AteneaCTF/Basica/6# ../../getFlag.py D4mpTheH3x! flag{60[REDACTED]ce}

RETO 6 – JUEGOS DE GUERRA

ENUNCIADO:

Se tienen sospechas de que en la siguiente imagen está escondida, de alguna manera, una contraseña. ¿Eres capaz de localizarla?

Fichero: d2FyZ2FtZXMK-d2414dd3b174b6503d59a113c6f02537.jpg

SOLUCIÓN:

Nos enfrentamos al primer reto de esteganografía de la plataforma y ya se empiezan a complicar las cosas. Una de los primeros pasos al analizar un fichero es mirar los metadatos.

```
1 exiftool d2FyZ2FtZXMK-d2414dd3b174b6503d59a113c6f02537.jpg
```

En la última línea de la salida de este comando vemos:

Thumbnail Image: (Binary data 6338 bytes, use -b option to extract)

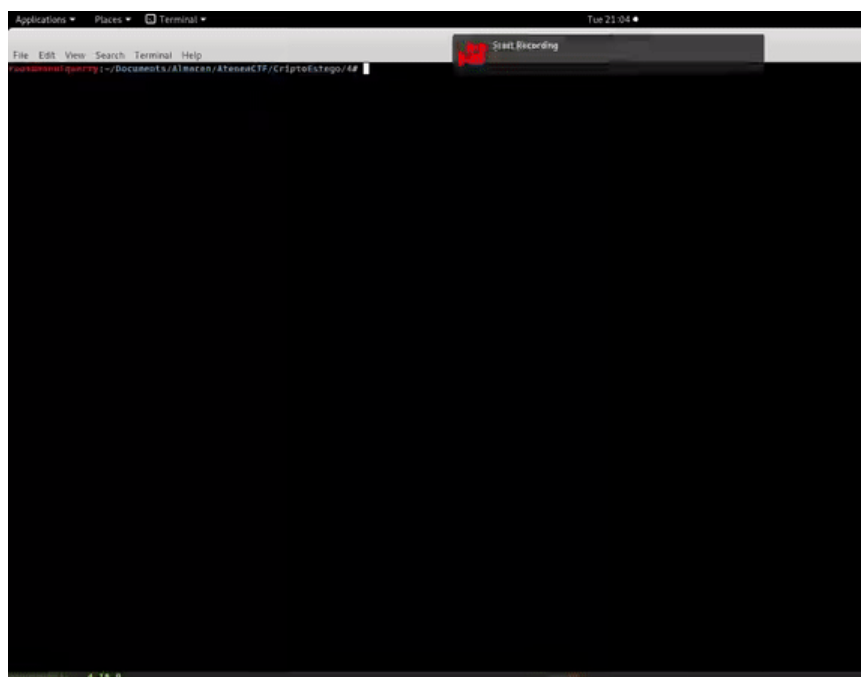
Para extraer este fichero:

```
1 exiftool d2FyZ2FtZXMK-d2414dd3b174b6503d59a113c6f02537.jpg -b -ThumbnailImage > output
```

En los metadatos del fichero extraído vemos que también hay una miniatura adjuntada (Thumbnail), vamos a hacer lo mismo que antes para extraerla

```
1 exiftool output -b -ThumbnailImage > output2
```

Una vez hemos extraído esta tercera imagen tardé un buen rato en ver que el siguiente paso estaba escondido también en los metadatos ya que los dos ficheros extraídos contienen unas coordenadas:



1 37°29'60"N 122°8'43.40W

Vamos a buscar en Google Maps estas coordenadas:



Vemos en el suelo la palabra 'HACK', esta será la clave del flag

```
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/4# ../../getFlag.py HACK  
flag{0a[REDACTED]f4}
```

RETO 7 – DURIN’S GATES

ENUNCIADO:

Tu objetivo será investigar el fichero adjunto .jpg y averiguar si está relacionado con una clave o mensaje de estas características.

Fichero: doors_of_durin-f686f3e1aa18d5e3f4261bea89a24c17.jpg

SOLUCIÓN:

Lo primero que vamos a hacer es abrir la imagen:



En la imagen vemos una famosa frase del Señor de los Anillos, esta nos da una pista así que vamos a buscar sobre la cita:

http://tolkiengateway.net/wiki/Doors_of_Durin

En esta fuente leemos que la contraseña de la que habla es: 'Mellon'

Tras ver esto pensé en steghide:

```
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# steghide extract -sf doors_of_durin-f686f3e1aa18d5e3f4261bea89a24c17.jpg
Enter passphrase:
wrote extracted data to "url.txt".
```

Vemos que el comando con la contraseña 'Mellon' nos extrae otro fichero: url.txt, vamos a echarle un ojo:

```
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# cat url.txt
https://pastebin.com/SgaSizcn
```

```
1 | https://pastebin.com/SgaSizcn
```

En este pastebin encontramos un texto cifrado en base64 bastante largo.

Vamos a descifrarlo y meterlo en otro fichero: key.enc

```
1 | base64 -d pastebin.txt > file.enc
```

Con el comando file vemos que se trata de un encriptado por openssl con contraseña.

Tras un buen rato buscando posibles contraseñas, volví a mirar los metadatos del fichero que nos entregan. En ellos vemos:

```
1 | Artist: 68913499125FAA
```

Vamos a probar a usar esa contraseña:

```
1 | openssl enc -aes-256-cbc -d -md MD5 -in file.enc -out file.txt -k 68913499125FAA
```

```
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# base64 -d pastebin.txt > file.enc
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# file file.enc
file.enc: openssl enc'd data with salted password
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# openssl enc -aes-256-cbc -d -in file.enc
-out file.txt
enter aes-256-cbc decryption password:
bad decrypt
140011784601792:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:../crypto/evp/evp_enc.c:536:
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# openssl enc -aes-256-cbc -d -md md5 -in file.enc -out file.txt
enter aes-256-cbc decryption password:
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# file file.txt
file.txt: ISO Media, MP4 Base Media v1 [ISO 14496-12:2003]
```

Como veis, el fichero obtenido es un video, vamos a verlo.

El video simplemente nos muestra que la clave es: Minas_Tirith_2017

```
key: Minas_Tirith_2017
root@manulqwerty:~/Documents/Almacen/AteneaCTF/CriptoEstego/5# ../../getFlag.py Minas_Tirith_2017
flag{c486904231261cc28cbff2f4ce7d6126}
```


RETO 8 – REALLY???

ENUNCIADO:

Durante el estudio del disco duro del ordenador de un sospechoso se ha encontrado un fichero cifrado mediante PGP. Al no encontrarse ninguna clave privada dentro del equipo se sospecha que dicho fichero esté cifrado mediante cifrado simétrico.

Por otro lado, todas las contraseñas obtenidas de varias cuentas del sospechoso (a partir de la investigación de su equipo) tienen las siguientes características:

- Son de longitud 6 o 7
- Sólo contienen letras minúsculas
- Sólo se utilizan estas letras: qwertyiopnmjk
- No se repite ninguna de las letras de la contraseña
- Algunas de ellas contienen únicamente un número entre estos: 013

Ninguna de esas contraseñas ha servido para descifrar el fichero, pero quizás haya sido cifrado con una contraseña con estas mismas características.

No sabemos si el contenido del fichero es relevante para la investigación, pero sólo hay una forma de averiguarlo...

¡Pista! La lengua materna del dueño del equipo es el inglés

SOLUCIÓN:

Tras leer el enunciado sabemos que necesitaremos un diccionario de palabras con solo 6 o 7 de estos caracteres:

qwertyiopnmjk013

Para ello obtendremos las palabras del rockyou que cumplan estos requisitos:

```
grep -x '[qwertyiopnmjk013]\{6,7\}' /usr/share/wordlists/rockyou.txt > word.txt
```

Una vez tenemos el diccionario podemos usar la herramienta <https://github.com/kholia/PGPCrack-NG>

```
root@manulqwerty:/tmp/6# STARTTIME=$(date +%s);cat word.txt | ./PGPCrack-NG/PGPCrack-NG message-2def3de75a007fa096097626a097930b.asc ;echo "Time elapsed: $((date +%s)-STARTTIME)"
Found Password : monkey3
Time elapsed: 16
```

Si la habéis probado sabréis que es bastante lenta, así que vamos a utilizar otra para este tipo de retos:

<https://github.com/manulqwerty/CrackGPG>

```

root@manulqwerty:/tmp/6# grep -x '[qwertyiopnmjk013]\{6,7\}' /usr/share/wordlists/rockyou.txt > word.txt

root@manulqwerty:/tmp/6# git clone https://github.com/manulqwerty/CrackGPG.git
Cloning into 'CrackGPG'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 26 (delta 9), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (26/26), done.
root@manulqwerty:/tmp/6# chmod +x CrackGPG/crackgpg.pyc
root@manulqwerty:/tmp/6# ./CrackGPG/crackgpg.pyc message-2def3de75a007fa096097626a097930b.asc word.txt

      CRACKGPG
                                by @manulqwerty

-----
Time elapsed: 4.82s - 117/51179 (0%)
[+] Password Found: monkey3
[+] Output File: gpg_output

```

Como veis es bastante más rápida

La ejecución de estos comandos nos devuelve un txt, sabiendo que es un reto de stego y tras probar varias técnicas vemos que con SNOW (<http://www.darkside.com.au/snow/>) obtenemos la información oculta:

`./snow -C gpg_output`

```

root@manulqwerty:/tmp/6# ./snow-20130616/snow gpg_output
Receta:

- Un poco de ron negro
- Una cucharada de azúcar
- Zumo de lima
- Una rama de canela
- Agua hirviendo

Se mezclan todos los ingredientes, añadiendo agua hirviendo hasta llenar el vaso.

Solución al reto: la bebida de la receta 2 veces seguidas (y en minúsculas!)

```

Tras leer la información oculta y buscar en google vemos que el flag es: groggrog



```

root@manulqwerty:/tmp/6# ./getFlag.py groggrog
flag{5...

```