Hispasec]

DOCUMENTACIÓN

Writeup UAM

Octubre 2023

Informe técnico

Cláusula legal

Documentación

Informe técnico

Índice

1. Datos del reto. **3**

1.1. Especificaciones técnicas. 3

2. Proceso de resolución del reto. **4**

2.1. Flag 1 4

2.2. Flag 2 8

1. Datos del reto.

1.1. Especificaciones técnicas.

A continuación se detalla el alcance y el conjunto de especificaciones del reto, así como las normativas de aplicación.

Alcance	
Activo/s:	**flag.png**
Categoría:	Misc.
Fecha de inicio:	27/10/2023.
Fecha de fin:	10/11/2023.
Cumplimiento normativo:	Las contraseñas y otra información sobre los usuarios no son almacenadas en cumplimiento con la ley de protección de datos (Reglamento General de Protección de Datos, RGPD). Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.

2. Proceso de resolución del reto.

A continuación se detalla la manera intencionada en la que se espera que el usuario resuelva el reto. Cada entrada viene encabezada por la descripción detallada del punto en concreto, acompañado de los pasos a seguir para reproducirlo.

2.1. Flag 1

Descripción

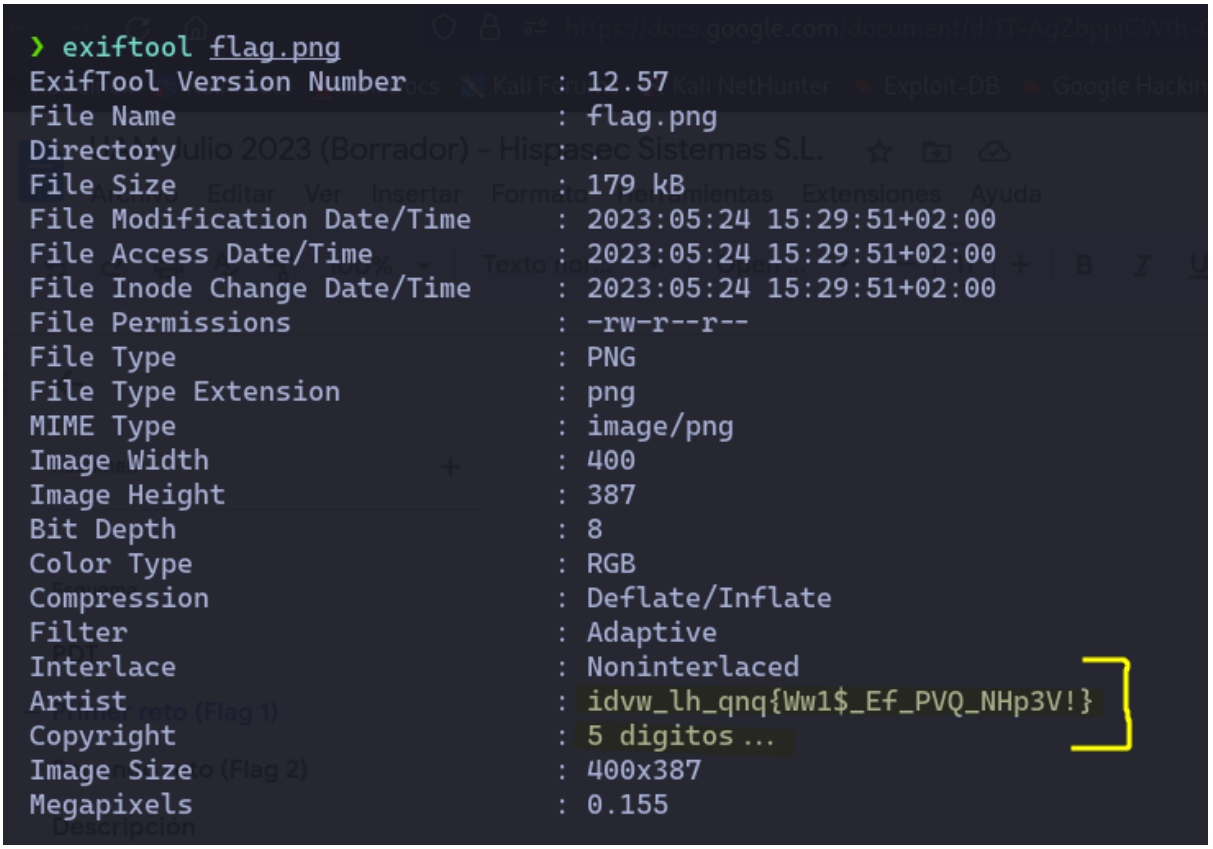
`El equipo de forenses, guiados por la astucia y el ingenio de Pepe, deberá descifrar un intrigante cifrado antiguo, donde deberán desentrañar los secretos ocultos en cada enigma y demostrar su dominio con los antiguos cifrados.`

- `formato de la flag: this_is_uam{}`

Resolución

Al comenzar el reto obtendremos la siguiente imagen.

Examinando los metadatos obtendremos dos datos interesantes, los cuales son *Artist* y *Copyright*.



Observando el formato de cifrado se podrá deducir que es de tipo rotatorio.

Dada la pista de 5 dígitos, y que solo existe un cifrado de rotación de caracteres que emplee clave, se podrá deducir que el cifrado empleado es de tipo *Vigenere*.

Dicho esto, para poder "romper" la contraseña, podremos realizar un script en python que filtre por la cadena de caracteres "**this_is_uam**" al comienzo de la rotación de palabras, para poder conseguir la clave, y por consiguiente, la flag.

El script en cuestión es el siguiente:

Python

```python
from string import ascii_lowercase
import re
import sys
import time
from itertools import product

def get_combinations():
  allowed_chars = ascii_lowercase
  combinations = set()

  for combination in product(allowed_chars, repeat=5):
    combinations.add(''.join(combination))

  return combinations

def decode_vinegere(cipher, key):
  clear_text = ''

  max_key = len(key)
  i = 0
  for letter in cipher:
    i = 0 if i%max_key == 0 else i
    if letter.isalpha():
      number = (ord(letter) - ord('a') - (ord(key[i]) - ord('a'))) % len(ascii_lowercase)
      clear_text += chr(number + ord('a'))
      i += 1
    else:
      clear_text += letter

  return clear_text, key

if __name__ == '__main__':
  cipher = "idvw_lh_qnq{Ww1$_Ef_PVQ_NHp3V!}"
  combinations = get_combinations()
  init = time.time()
  for combination in combinations:
    result, key = decode_vinegere(cipher, combination)
    print("Breaking key: [key] {}".format(key), end="\r")
    if re.search("^this_is_uam", result):
      end = time.time()
      print("\nTiempo empleado:{:.2f} segundos".format(end-init))
      time.sleep(1)
      sys.exit(0)

  print("\nKey not found!")
  sys.exit(1)
```

Al ejecutarlo, obtendremos la clave para descifrar *Vigenere*.

```
❯ python3 broken_vinegere.py
Breaking key: [key] pwned
Tiempo empleado:16.00 segundos
```

Ya solo queda usar la clave para descifrar *Vigenere*, para ello podremos ir a una web como **Cyberchef**.

Output

```
this_is_uam{Th1$_Is_LSB_RUl3S!}
```

Con esto conseguimos la **flag1**.

2.2. Flag 2

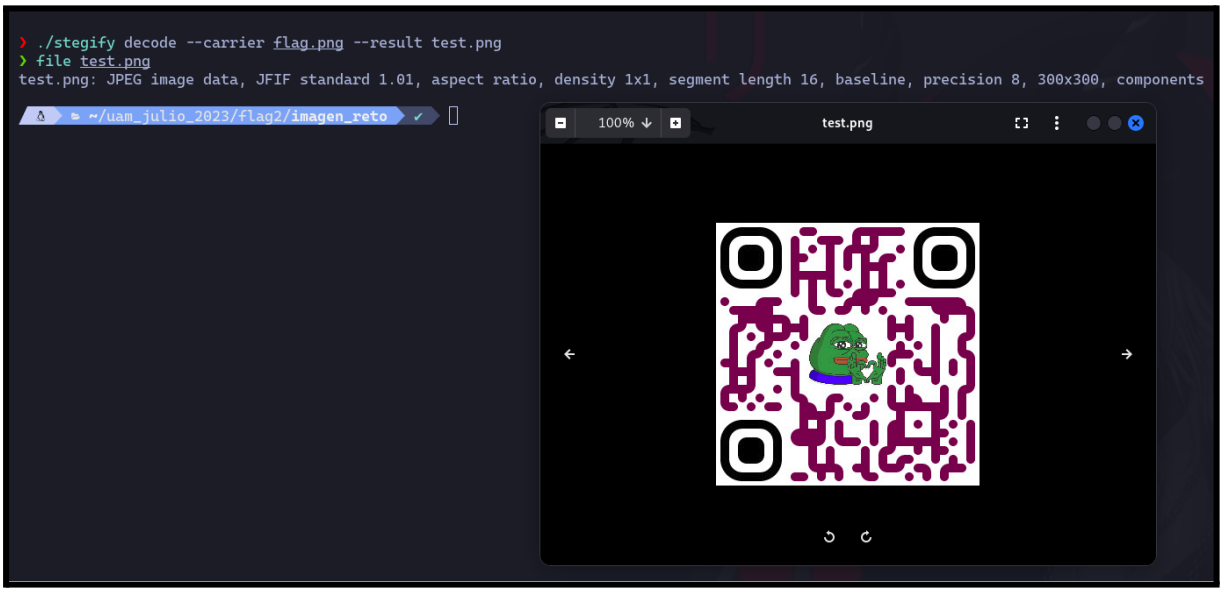
Descripción

En las profundidades se encuentra un tesoro oculto, esperando ser descubierto por aquellos valientes que desafíen los laberintos cibernéticos. Con la determinación de Pepe como guía, se desvelarán arcaicos secretos, revelando un legado perdido que aguarda ser rescatado de las sombras de los bits.

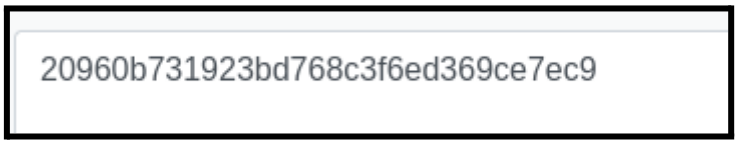
Resolución

Para la segunda flag tendremos que utilizar una herramienta **stegify**, que se encarga de esconder una imagen dentro de otra mediante la técnica **LSB** (*Least Significant Bit*).

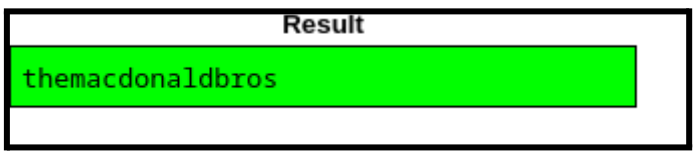
- **stegify decode --carrier flag.png --result output.png**



Una vez tenemos la imagen, si la escaneamos podremos sacar un *hash* **md5**.



Podremos crackearlo de forma sencilla, obteniendo la clave para usar en **Steghide**.



Ya con la clave, podremos extraer la flag de la imagen.

```
❯ steghide extract -sf test.png
Anotar salvoconducto:
anot los datos extrados e/"flag.txt".
❯ cat flag.txt
       File: flag.txt
   1   UAM{08c65b97844c45e5606aac0709954134}
 ~/uam_julio_2023/flag2/imagen_reto ✔
```

Otra opción alternativa sería el uso de **stegseek** para crackear el propio salvoconducto de la imagen.

```
❯ stegseek --crack test.png
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "themacdonaldbros"
[i] Original filename: "flag.txt".
[i] Extracting to "test.png.out".

 ~/uam_julio_2023/flag2/imagen_reto ✔
```

```
❯ cat test.png.out
       File: test.png.out
   1   UAM{08c65b97844c45e5606aac0709954134}
```

Hispasec]

Hispasec Sistemas S.L.
C/ Severo Ochoa, 10 - 29590, Málaga
Telf: (+34) 952 020 494
info@hispasec.com