Fruits

Reconocimiento inicial

El primer paso para la resolución de la máquina es el reconocimiento. Comenzaremos lanzando un ping para verificar si la máquina está activa y comprobar su TTL. Esto nos dará una idea del sistema operativo al que nos estamos enfrentando. En este caso, el TTL es de 64, lo que indica que probablemente se trate de una máquina Linux.

A continuación, realizamos un escaneo de puertos con `nmap` para identificar cuáles están abiertos. Usamos el siguiente comando:

```shell
sudo nmap --min-rate 5000 -sS -n -Pn -p- -v 192.168.0.132 -oG allports
```

- `--min-rate 5000`: Aumenta la velocidad de escaneo enviando al menos 5000 paquetes por segundo.
- `-sS`: Realiza un escaneo SYN para detectar puertos.
- `-n`: Evita la resolución DNS.
- `-Pn`: Omite el ping para no depender de ICMP.
- `-p-`: Escanea todos los puertos (1 al 65535).
- `-oG allports`: Guarda los resultados en formato `grepable` para un análisis más sencillo.

Resultado: Se detectaron los puertos 22 (SSH) y 80 (HTTP) como abiertos.

Después, realizamos un escaneo de servicios y versiones en estos puertos específicos con el comando:

```shell
sudo nmap -sCV -p22,80 192.168.0.132 -oN targeted
```

- `-sCV`: Detecta servicios, versiones y posibles scripts vulnerables.
- `-p22,80`: Limita el escaneo a los puertos abiertos identificados previamente.
- `-oN targeted`: Guarda los resultados en un archivo legible.

Resultados obtenidos:

SHELL

```
File: allPorts

# Nmap 7.94SVN scan initiated Sat Jan  4 19:39:18 2025 as: nmap --min-
rate 5000 -sS -n -Pn -p- -v -oG allports 192.168.0.132
# Ports scanned: TCP(65535;1-65535) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 192.168.0.132 ()  Status: Up
Host: 192.168.0.132 ()  Ports: 22/open/tcp//ssh///, 80/open/tcp//http///
Ignored State: closed (65533)
# Nmap done at Sat Jan  4 19:39:20 2025 -- 1 IP address (1 host up)
scanned in 1.82 seconds
```

SHELL

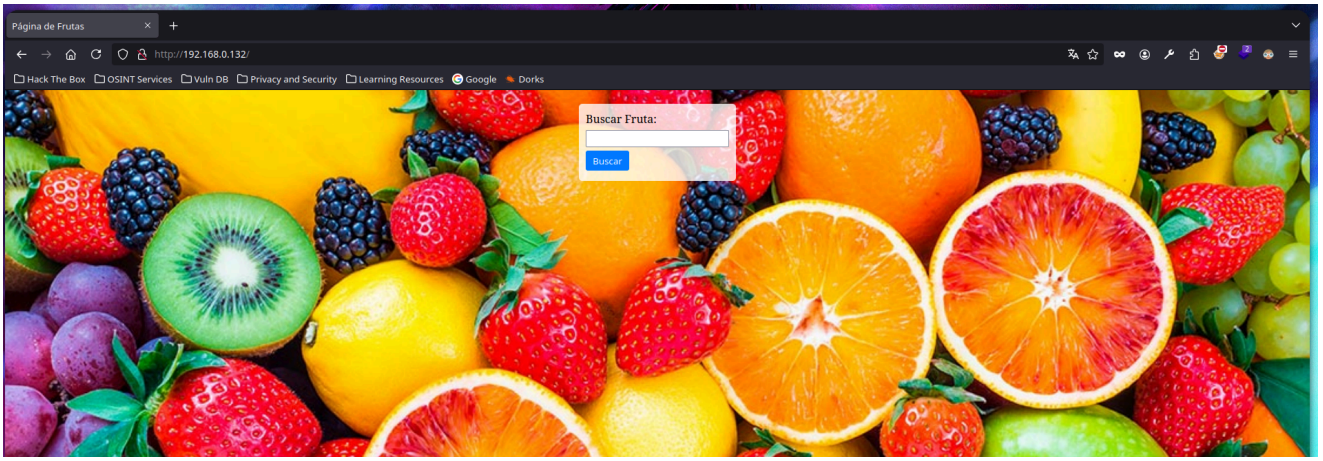
```
File: targeted

# Nmap 7.94SVN scan initiated Sat Jan  4 19:14:56 2025 as: nmap -sCV -
p22,80 -oN targeted 192.168.0.132
Nmap scan report for 192.168.0.132
Host is up (0.00040s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     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)
|_  256 68:16:a7:3a:63:0c:8b:f6:ba:a1:ff:c0:34:e8:bf:80 (ED25519)
80/tcp open  http    Apache httpd 2.4.57 ((Debian))
|_http-server-header: Apache/2.4.57 (Debian)
|_http-title: P\xC3\xA1gina de Frutas
MAC Address: 00:0C:29:95:B6:FF (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Sat Jan  4 19:15:03 2025 -- 1 IP address (1 host up)
scanned in 7.21 seconds
```

Podemos observar que el puerto 80 está abierto. Al visitar la página web asociada, nos encontramos con la siguiente interfaz:



Análisis del panel

He intentado interactuar con el panel, pero no parece llevar a ninguna parte útil. Siempre que se utiliza, la página redirige a:

`http://192.168.0.132/buscar.php?busqueda=naranja`

En esta URL, el parámetro `busqueda` cambia según el término ingresado. Esto sugiere que podríamos estar ante un punto de entrada para inyección de parámetros, pero primero debemos realizar más reconocimiento.

Escaneo inicial con Wfuzz

Antes de proceder con pruebas de inyección, revisé el código fuente de la página, pero no encontré información relevante ni elementos ocultos.

A continuación, realicé un escaneo de directorios con **Wfuzz** para intentar identificar rutas o archivos adicionales en el servidor web. Utilicé el siguiente comando:

```shell
wfuzz -c -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hc=404,403 -u http://192.168.0.132/FUZZ
```

- `-c`: Muestra la salida en color para facilitar su lectura.
- `-t 200`: Lanza 200 hilos concurrentes para acelerar el proceso.
- `-w`: Especifica la ruta del diccionario utilizado para el escaneo.
- `--hc=404,403`: Excluye respuestas con códigos de estado HTTP 404 (no encontrado) y 403 (prohibido).
- `-u http://192.168.0.132/FUZZ`: Especifica la URL objetivo, utilizando `FUZZ` como marcador para probar diferentes rutas.

Escaneo con Gobuster

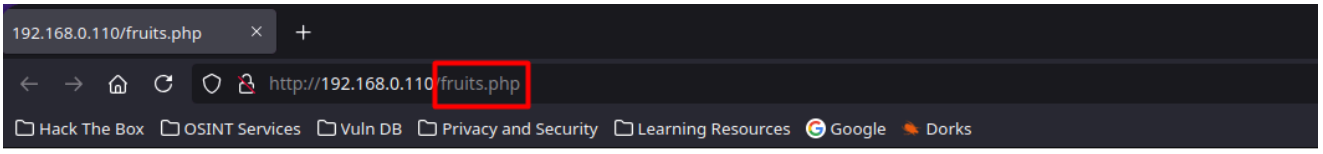
Tras no encontrar ningún directorio útil con **Wfuzz**, decidí cambiar de herramienta y realizar un escaneo más enfocado en archivos específicos (como `.php`, `.html` o `.js`) utilizando **Gobuster**. Con este enfoque, logré identificar un archivo llamado `fruits.php`:

```
❯ gobuster dir -u http://192.168.0.110/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,html,js

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.0.110/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,js
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 1811]
/.html                (Status: 403) [Size: 278]
/.php                 (Status: 403) [Size: 278]
/fruits.php           (Status: 200) [Size: 1]
/server-status        (Status: 403) [Size: 278]
Progress: 882184 / 882188 (100.00%)
===============================================================
Finished
===============================================================
```

Exploración inicial de `fruits.php`

Al acceder a `fruits.php`, la página parece estar vacía y no muestra ningún contenido relevante:



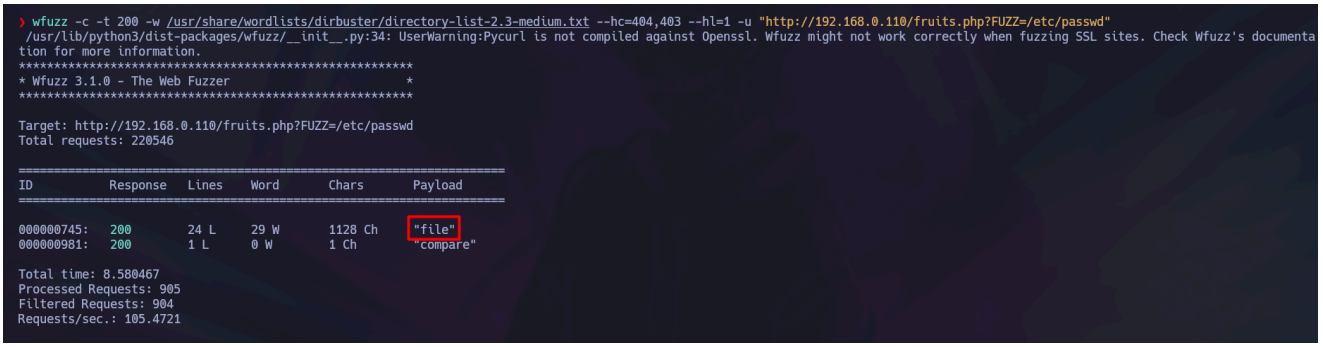
Siguiente paso: Identificación de parámetros útiles

En este punto, no encontré información aparente en la interfaz ni en el código fuente de la página. Para avanzar, consulté con ChatGPT, quien sugirió probar posibles parámetros en la URL del archivo para ver si se pueden manipular. Una herramienta útil para este propósito es **Wfuzz**.

Mi objetivo era intentar listar los usuarios de la máquina objetivo accediendo al archivo `/etc/passwd`. Para ello, utilicé el siguiente comando:

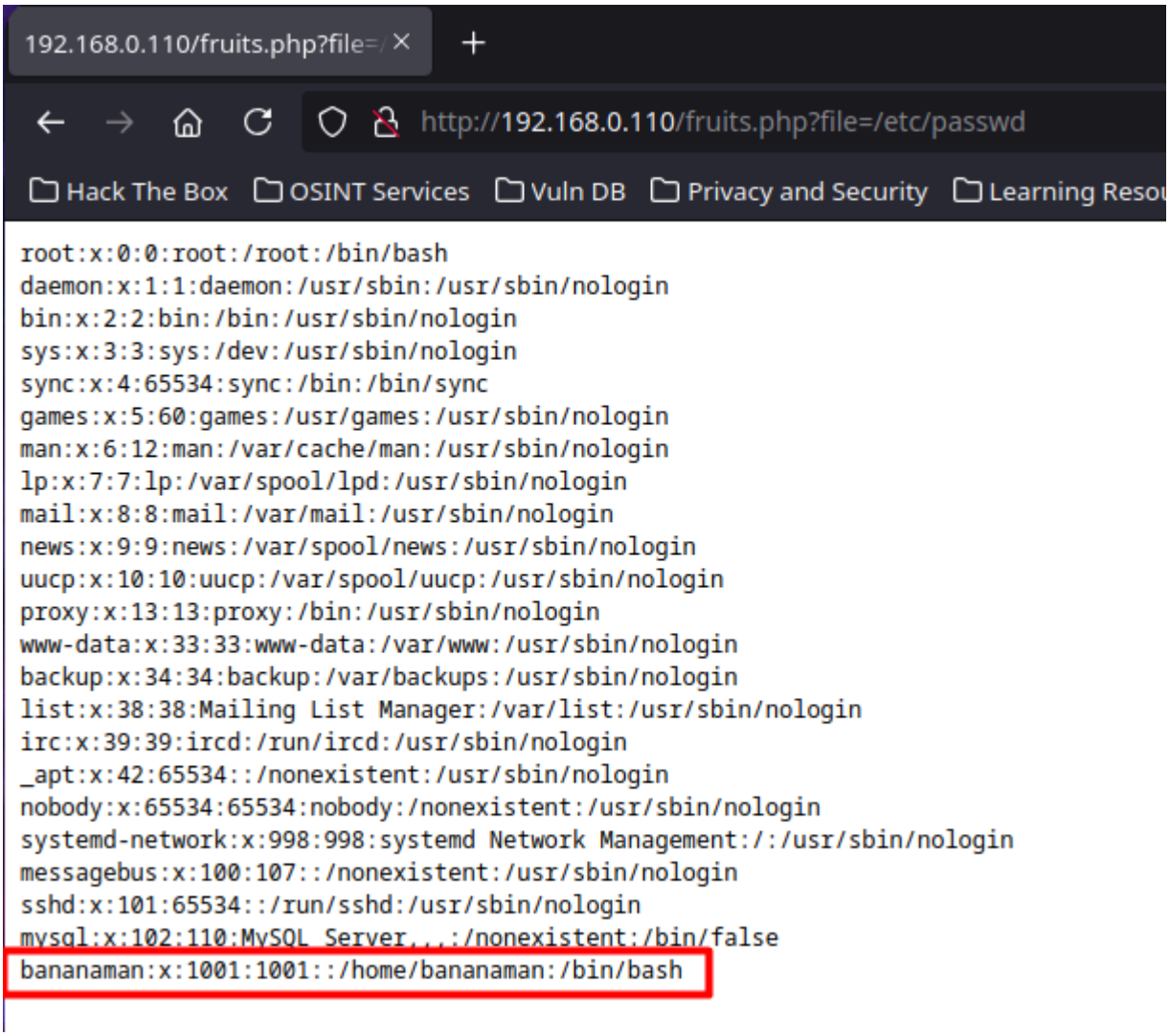
```shell
wfuzz -c -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --hc=404,403 --hl=1 -u http://192.168.0.110/fruits.php?FUZZ=/etc/passwd
```

- `FUZZ`: Lugar donde se insertan las palabras del diccionario para probar diferentes parámetros.
- `--hl=1`: Filtra las respuestas que tienen una longitud de 1 línea, ya que suelen ser páginas vacías o errores.



Explotación del Parámetro `file`

Tras realizar el escaneo con **Wfuzz**, descubrimos que el parámetro `file` en la URL es vulnerable. Esto nos permitió listar el contenido del archivo `/etc/passwd`, que contiene información sobre los usuarios del sistema:



Identificación de un Usuario

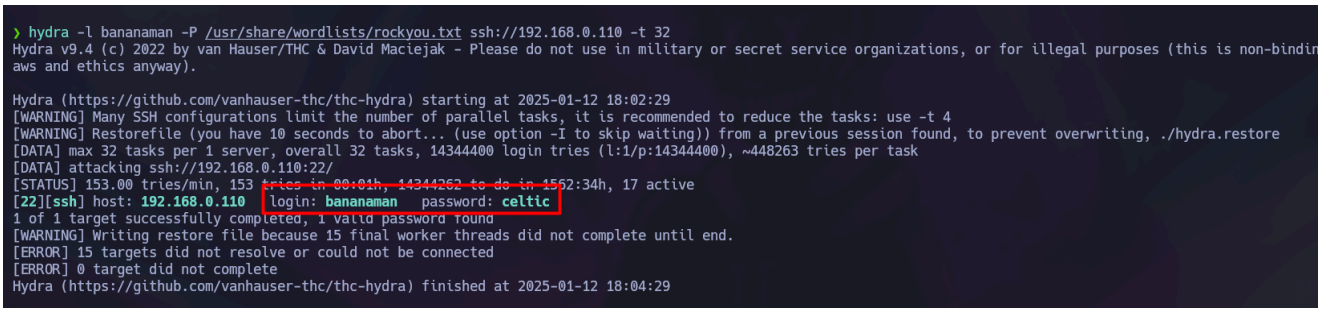
Gracias a la información obtenida de `/etc/passwd`, logramos identificar un usuario válido en la máquina objetivo. Como verificamos al inicio del reconocimiento, el puerto **22 (SSH)** está abierto. Esto nos brinda la oportunidad de intentar un ataque de fuerza bruta para acceder al sistema mediante este servicio.

Ataque de Fuerza Bruta con Hydra

Utilizamos **Hydra** para realizar un ataque de fuerza bruta contra el servicio SSH, empleando el nombre de usuario identificado previamente y una lista de contraseñas. El comando utilizado fue el siguiente:

```shell
hydra -l <usuario> -P /ruta/a/wordlist.txt ssh://192.168.0.132
```

- `-l <usuario>`: Define el nombre del usuario.
- `-P /ruta/a/wordlist.txt`: Especifica la lista de contraseñas que se probarán.
- `ssh://192.168.0.132`: Indica el protocolo y la dirección IP del objetivo.



Acceso y Flag

Con estas credenciales, finalmente pudimos acceder a la máquina mediante SSH. Una vez dentro, localizamos la flag y completamos el objetivo.

Configuración de la TTY para un Entorno Adecuado

Tras acceder a la máquina mediante SSH, es recomendable sanitizar la TTY para trabajar en un entorno más cómodo y funcional. Esto se logra ejecutando los siguientes comandos:

```shell
export TERM=xterm-256color
source /etc/skel/.bashrc
```

- `export TERM=xterm-256color`: Configura el terminal para usar colores y capacidades avanzadas.
- `source /etc/skel/.bashrc`: Carga el archivo de configuración predeterminado para establecer un entorno de trabajo más completo.

Ajuste de Dimensiones de la TTY

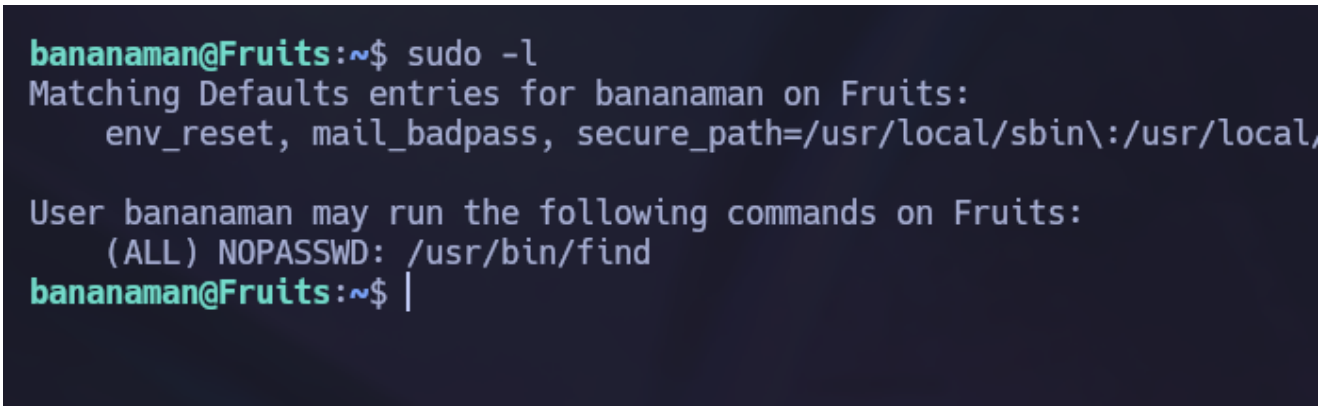
Para asegurarnos de que las dimensiones del terminal sean consistentes con nuestra pantalla, verificamos y ajustamos el tamaño con el comando:

```shell
stty size
```

Este comando muestra el número de filas y columnas del terminal actual. En mi caso, el tamaño fue de **35 rows x 184 columns**, pero debes ajustarlo según las dimensiones de tu terminal.

Preparación para la Escalada de Privilegios

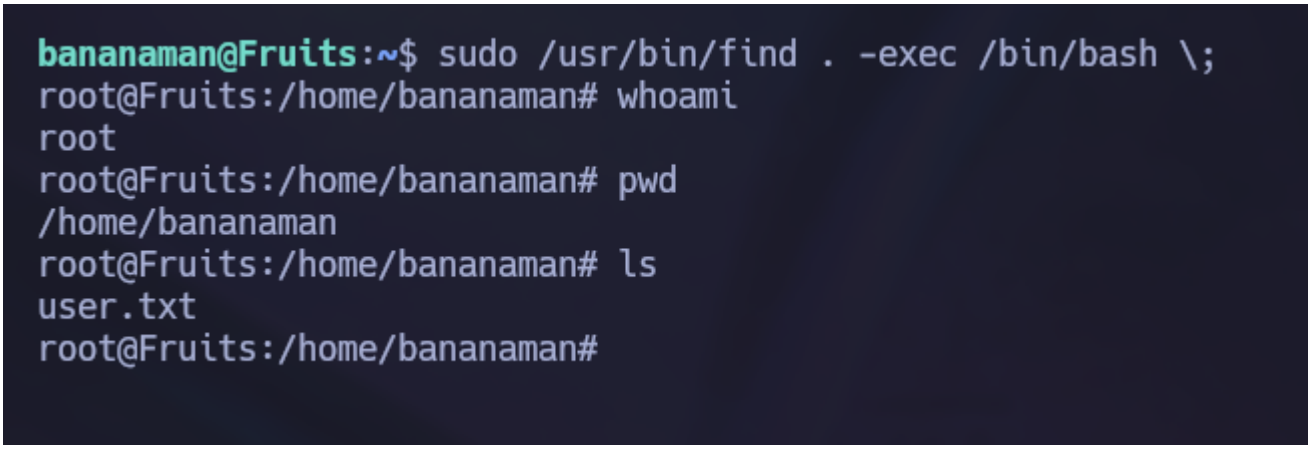
Con la TTY configurada correctamente, el siguiente paso es verificar si tenemos privilegios especiales en la máquina. Para ello, listamos los permisos de `sudoers` con el comando:



En este caso, descubrimos que el comando `find` puede ejecutarse como **sudo** sin necesidad de proporcionar una contraseña:

```shell
User <usuario> may run the following commands on <máquina>:
    (ALL) NOPASSWD: /usr/bin/find
```

El comando `find` incluye el parámetro `-exec`, el cual permite ejecutar un comando arbitrario en el sistema. Aprovechando esto, podemos generar una shell de Bash con privilegios de **root** y, de esta manera, acceder a la **flag**.



Y así hemos completado esta maquina de nuestros amigos de The Hackers Labs