Intro to MySQL

This module introduces SQL injection through `MySQL`, and it is crucial to learn more about `MySQL` and SQL to understand how SQL injections work and utilize them properly. Therefore, this section will cover some of MySQL/SQL's basics and syntax and examples used within MySQL/MariaDB databases.

Structured Query Language (SQL)

SQL syntax can differ from one RDBMS to another. However, they are all required to follow the ISO standard for Structured Query Language. We will be following the MySQL/MariaDB syntax for the examples shown. SQL can be used to perform the following actions:

- Retrieve data
- Update data
- Delete data
- Create new tables and databases
- Add / remove users
- Assign permissions to these users

Command Line

The `mysql` utility is used to authenticate to and interact with a MySQL/MariaDB database. The `-u` flag is used to supply the username and the `-p` flag for the password. The `-p` flag should be passed empty, so we are prompted to enter the password and do not pass it directly on the command line since it could be stored in cleartext in the bash_history file.

Intro to MySQL

```
MisaelMacias@htb[/htb]$ mysql -u root -p

Enter password: <password>
...SNIP...

mysql>
```

Again, it is also possible to use the password directly in the command, though this should be avoided, as it could lead to the password being kept in logs and terminal history:

Intro to MySQL

```
MisaelMacias@htb[/htb]$ mysql -u root -p<password>

...SNIP...

mysql>
```

Tip: There shouldn't be any spaces between '-p' and the password.

The examples above log us in as the superuser, i.e.,"`root`" with the password "`password`," to have privileges to execute all commands. Other DBMS users would have certain privileges to which statements they can execute. We can view which privileges we have using the SHOW GRANTS command which we will be discussing later.

When we do not specify a host, it will default to the `localhost` server. We can specify a remote host and port using the `-h` and `-P` flags.

Intro to MySQL

```
MisaelMacias@htb[/htb]$ mysql -u root -h docker.hackthebox.eu -P 3306 -p

Enter password:
...SNIP...

mysql>
```

Note: The default MySQL/MariaDB port is (3306), but it can be configured to another port. It is specified using an uppercase `P`, unlike the lowercase `p` used for passwords.

Note: To follow along with the examples, try to use the 'mysql' tool on your PwnBox to log in to the DBMS found in the question at the end of the section, using its IP and port. Use 'root' for the username and 'password' for the password.

Creating a database

Once we log in to the database using the `mysql` utility, we can start using SQL queries to interact with the DBMS. For example, a new database can be created within the MySQL DBMS using the CREATE DATABASE statement.

Intro to MySQL

```
mysql> CREATE DATABASE users;

Query OK, 1 row affected (0.02 sec)
```

MySQL expects command line queries to be terminated with a semi-colon. The example above created a new database named `users`. We can

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

MySQL expects command line queries to be terminated with a semi-colon. The example above created a new database named `users`. We can view the list of databases with SHOW DATABASES, and we can switch to the `users` database with the `USE` statement:

Intro to MySQL

```
mysql> SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| users              |
+--------------------+

mysql> USE users;

Database changed
```

SQL statements aren't case sensitive, which means 'USE users;' and 'use users;' refer to the same command. However, the database name is case sensitive, so we cannot do 'USE USERS;' instead of 'USE users;'. So, it is a good practice to specify statements in uppercase to avoid confusion.

Tables

DBMS stores data in the form of tables. A table is made up of horizontal rows and vertical columns. The intersection of a row and a column is called a cell. Every table is created with a fixed set of columns, where each column is of a particular data type.

A data type defines what kind of value is to be held by a column. Common examples are `numbers`, `strings`, `date`, `time`, and `binary data`. There could be data types specific to DBMS as well. A complete list of data types in MySQL can be found here. For example, let us create a table named `logins` to store user data, using the CREATE TABLE SQL query:

Code: sql

```sql
CREATE TABLE logins (
    id INT,
    username VARCHAR(100),
    password VARCHAR(100),
    date_of_joining DATETIME
    );
```

As we can see, the `CREATE TABLE` query first specifies the table name, and then (within parentheses) we specify each column by its name and its data type, all being comma separated. After the name and type, we can specify specific properties, as will be discussed later.

Intro to MySQL

```
mysql> CREATE TABLE logins (
    ->     id INT,
    ->     username VARCHAR(100),
    ->     password VARCHAR(100),
    ->     date_of_joining DATETIME
    ->     );
Query OK, 0 rows affected (0.03 sec)
```

The SQL queries above create a table named `logins` with four columns. The first column, `id` is an integer. The following two columns, `username` and `password` are set to strings of 100 characters each. Any input longer than this will result in an error. The `date_of_joining` column of type `DATETIME` stores the date when an entry was added.

Intro to MySQL

```
mysql> SHOW TABLES;

+-----------------+
| Tables_in_users |
+-----------------+
| logins          |
+-----------------+
1 row in set (0.00 sec)
```

A list of tables in the current database can be obtained using the `SHOW TABLES` statement. In addition, the DESCRIBE keyword is used to list the table structure with its fields and data types.

Intro to MySQL

```
mysql> DESCRIBE logins;

+-----------------+-------------+
| Field           | Type        |
+-----------------+-------------+
| id              | int         |
| username        | varchar(100) |
| password        | varchar(100) |
| date_of_joining | date        |
+-----------------+-------------+
4 rows in set (0.00 sec)
```

Table Properties

Within the `CREATE TABLE` query, there are many properties that can be set for the table and each column. For example, we can set the `id` column to auto-increment using the `AUTO_INCREMENT` keyword, which automatically increments the id by one every time a new item is added to the table:

Code: sql

```sql
    id INT NOT NULL AUTO_INCREMENT,
```

The `NOT NULL` constraint ensures that a particular column is never left empty 'i.e., required field.' We can also use the `UNIQUE` constraint to ensures that the inserted item are always unique. For example, if we use it with the `username` column, we can ensure that no two users will have the same username:

Code: sql

```
    username VARCHAR(100) UNIQUE NOT NULL,
```

Another important keyword is the `DEFAULT` keyword, which is used to specify the default value. For example, within the `date_of_joining` column, we can set the default value to Now(), which in MySQL returns the current date and time:

Code: sql

```
    date_of_joining DATETIME DEFAULT NOW(),
```

Finally, one of the most important properties is `PRIMARY KEY`, which we can use to uniquely identify each record in the table, referring to all data of a record within a table for relational databases, as previously discussed in the previous section. We can make the `id` column the `PRIMARY KEY` for this table:

Code: sql

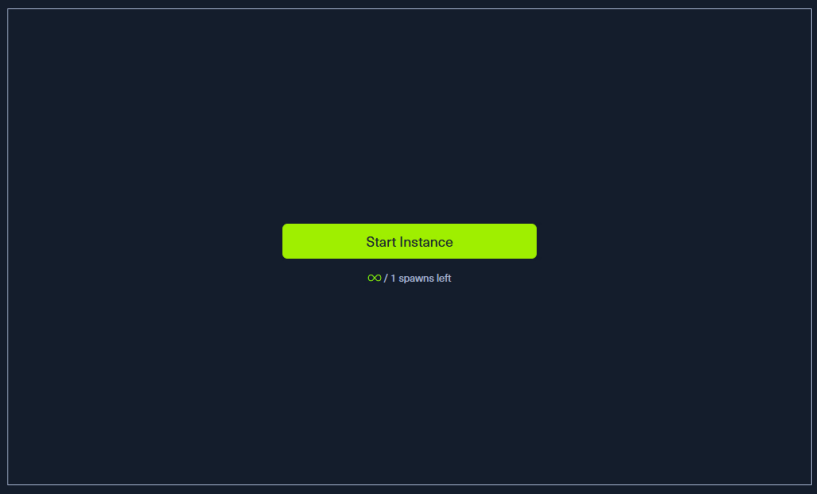
```
    PRIMARY KEY (id)
```

The final `CREATE TABLE` query will be as follows:

Code: sql

```
CREATE TABLE logins (
    id INT NOT NULL AUTO_INCREMENT,
    username VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    date_of_joining DATETIME DEFAULT NOW(),
    PRIMARY KEY (id)
    );
```

Note: Allow 10-15 seconds for the servers in the questions to start, to allow enough time for Apache/MySQL to initiate and run.



Waiting to start...