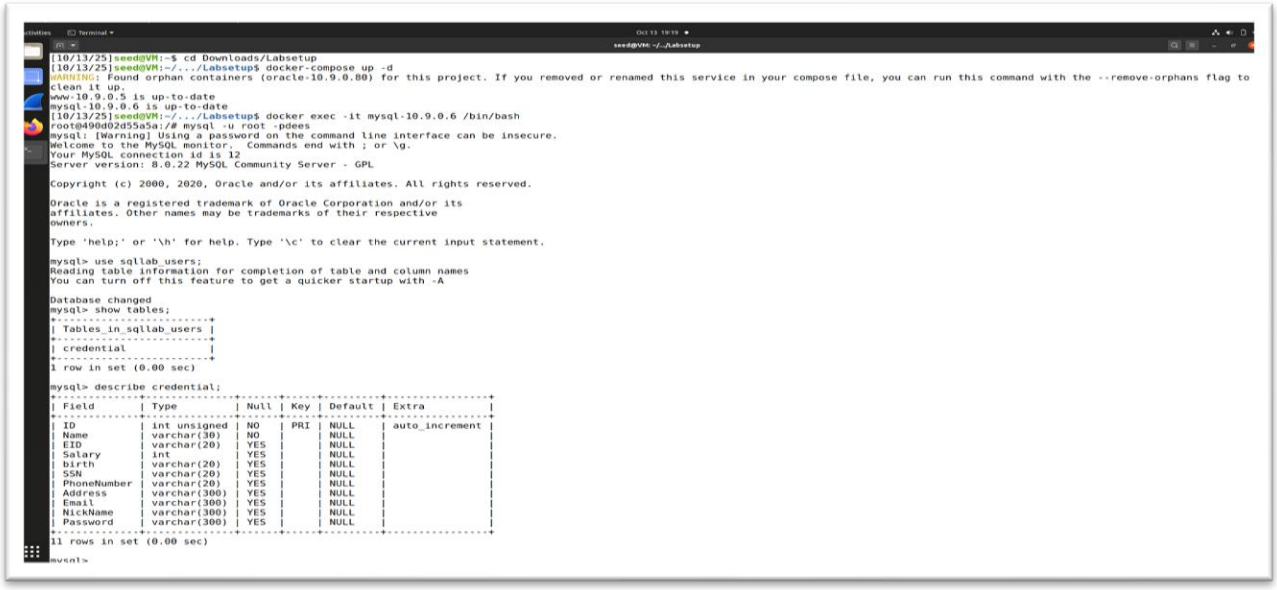
LAB- 02

By :- Faraz Ahmed

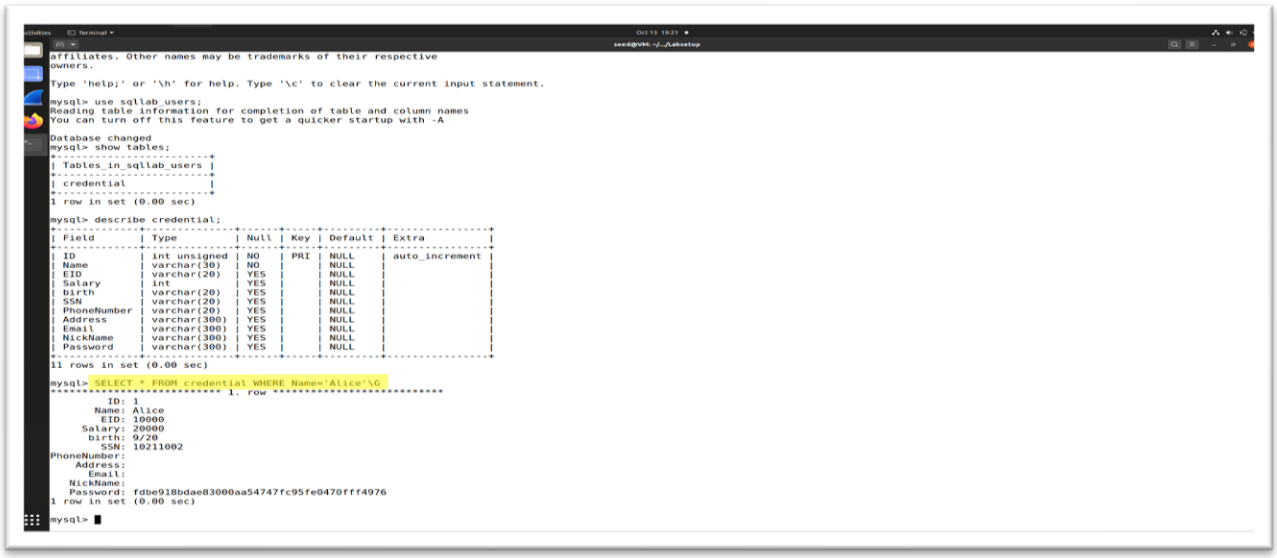
Task 1

- Firstly, we downloaded new labsetup for this assignment and run all dockers inside it. Then we gain access to MySQL container using “docker exec -it mysql-10.9.0.6 /bin/bash”, then login to MySQL using “mysql -u root -pdees”, then access the database sqllab_users and then display table inside it where we can observe credential which we then use “describe credential” to display it as observed in Screenshot 1.



Screenshot 1: Different credentials for every employees.

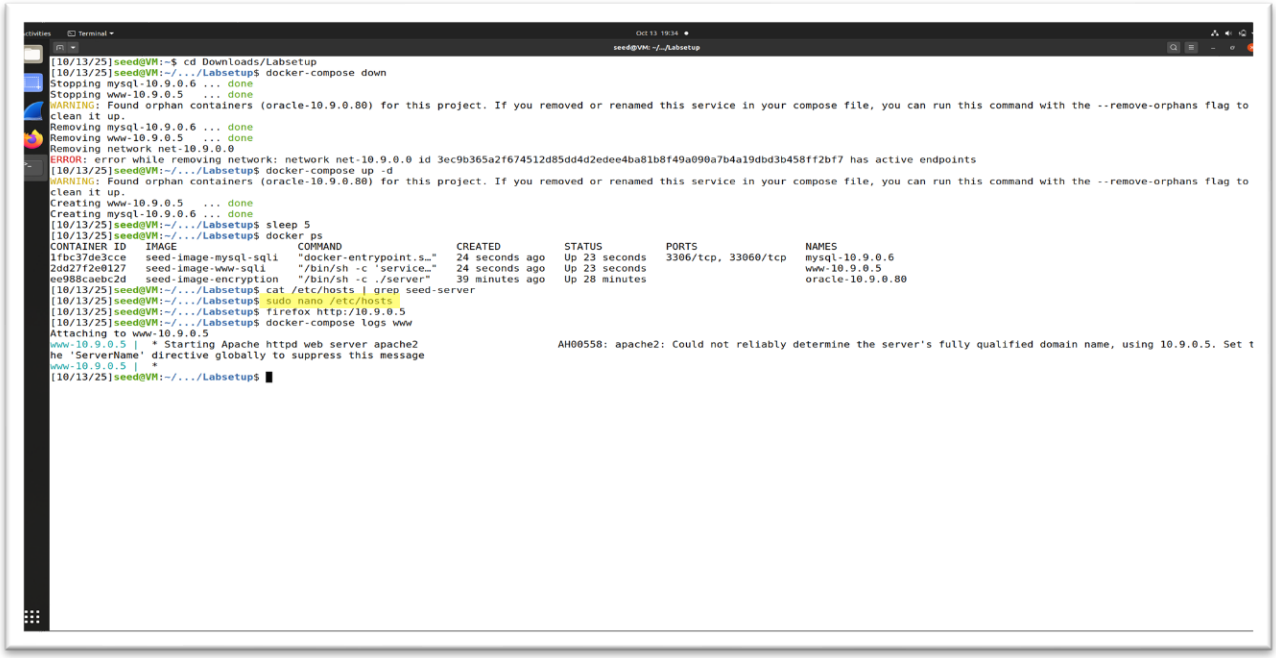
- Then it particularly gets information of Alice using “SELECT * FROM credential WHERE Name=’Alice’\G where \G display everything vertically. (as highlighted in Screenshot 2)



Screenshot 2: Profile information of employee Alice.

Task 2

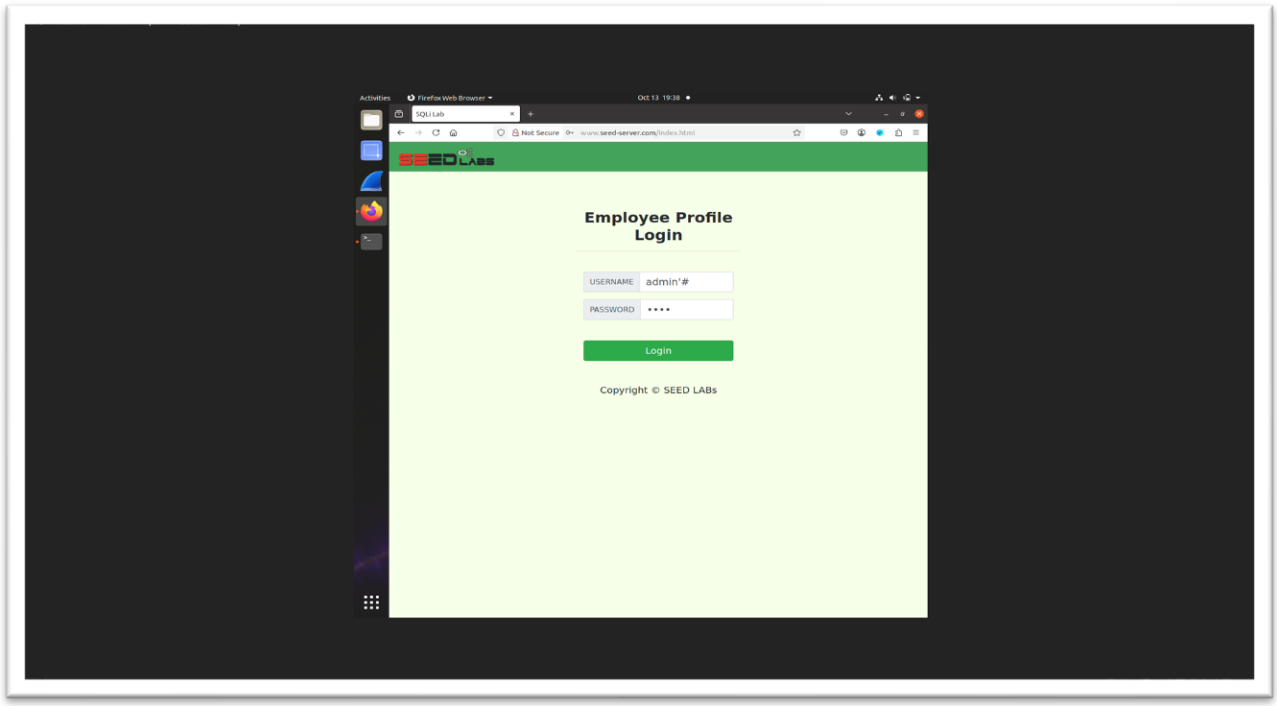
- In order to access "http://www.seed-server.com", we need that in /etc/hosts using nano /etc/hosts as highlighted in Screenshot 3 and we can observe the added website highlighted in Screenshot 4.



Screenshot 3: "nano /etc/hosts" command to add http://www.seed-server.com.

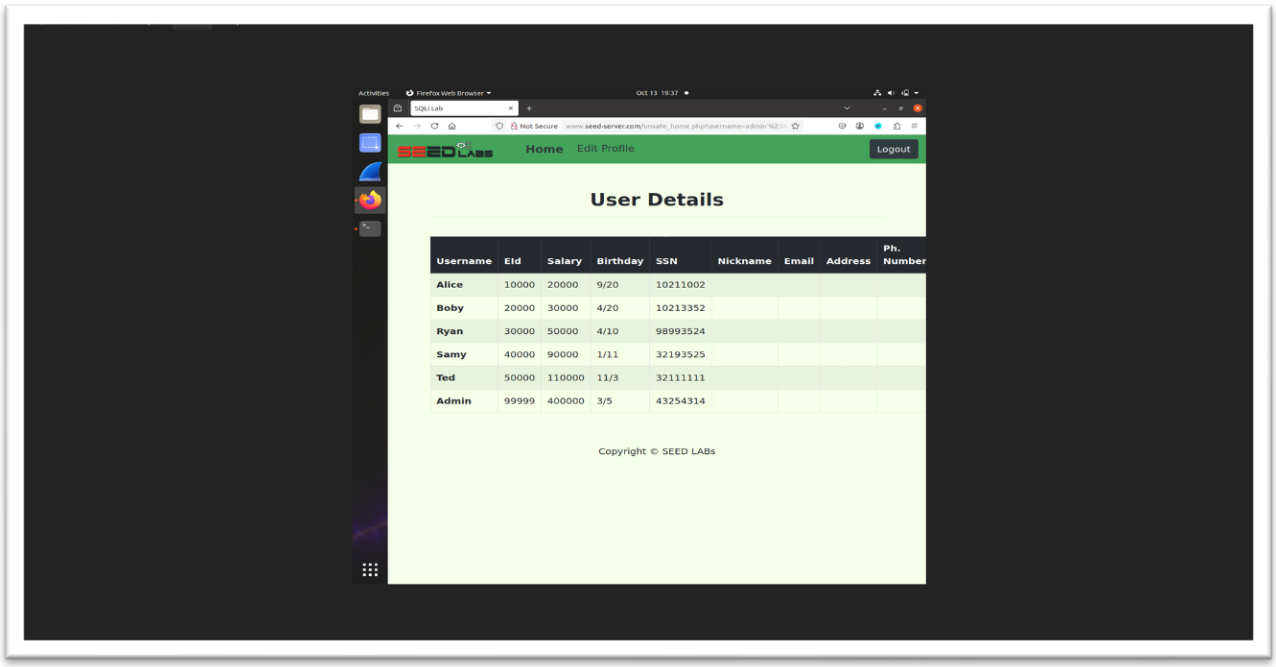
Screenshot 4: http://www.seed-server.com added in /etc/hosts.

- Now, we enter SQL injection as admin'# in Username and test (it could be anything or blank) in password as observed in Screenshot 5.



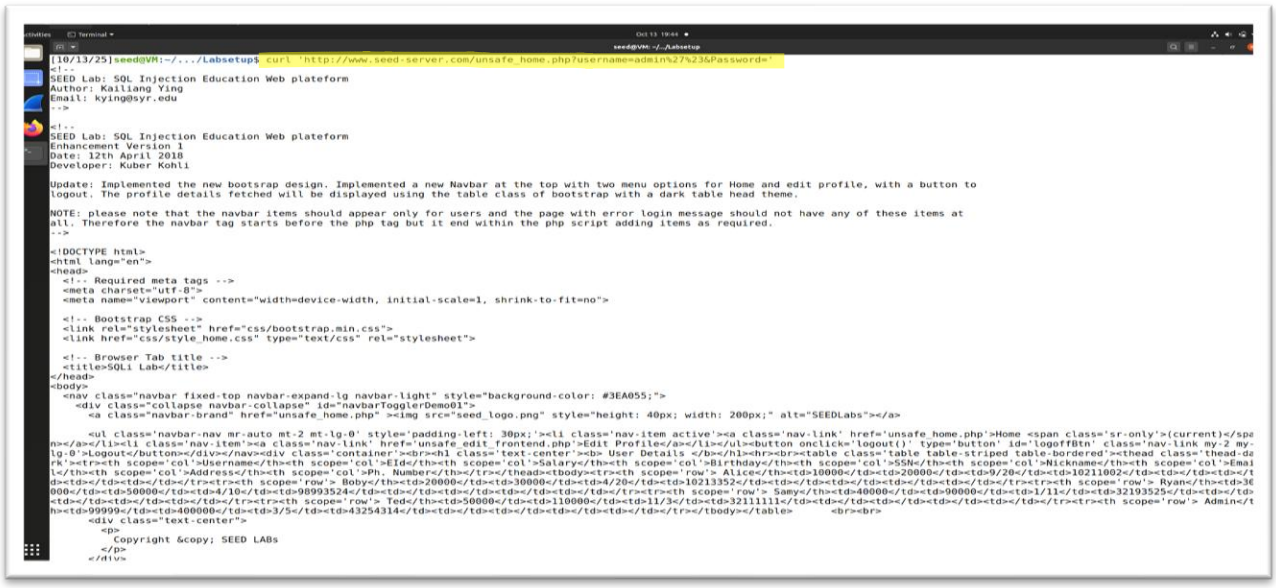
Screenshot 5: Injecting SQL in Profile login using "admin'#".

- After entering sql injection, we can observe all sensitive details of different users as observed in Screenshot 6.



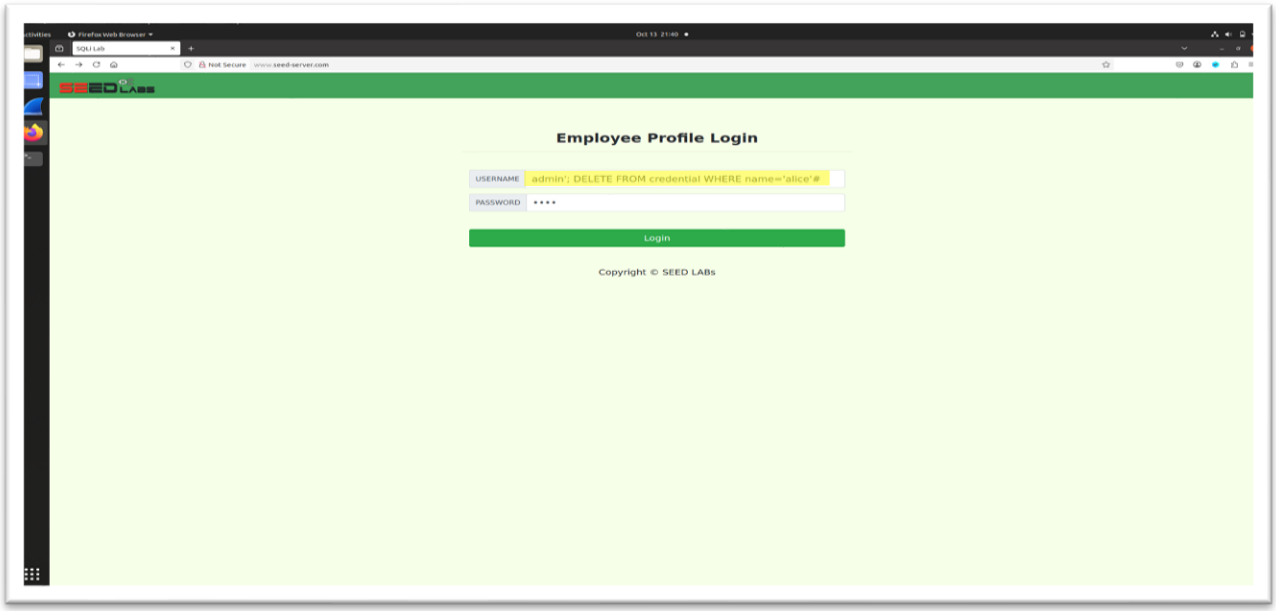
Screenshot 6: details of employees after successful SQL injection.

- After that, we enter “curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%23&Password='” where ‘ is %27 and # is %23 in command line to do SQL injection from there as observed successful attack in Screenshot 7.

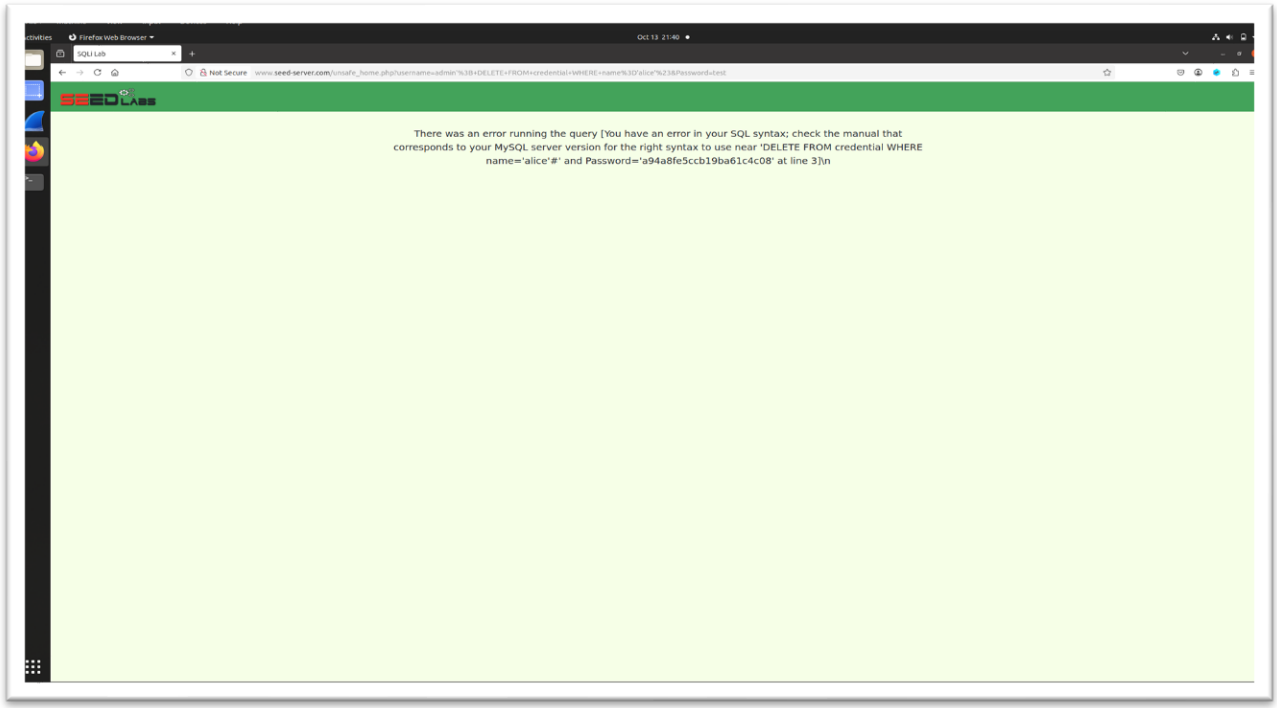


Screenshot 7: using “curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%23&Password=” command.

- Now we will enter “admin'; DELETE FROM credential WHERE name='alice'#” in username as highlighted in Screenshot 8 but this SQL injection fails as we can observe in Screenshot 9.

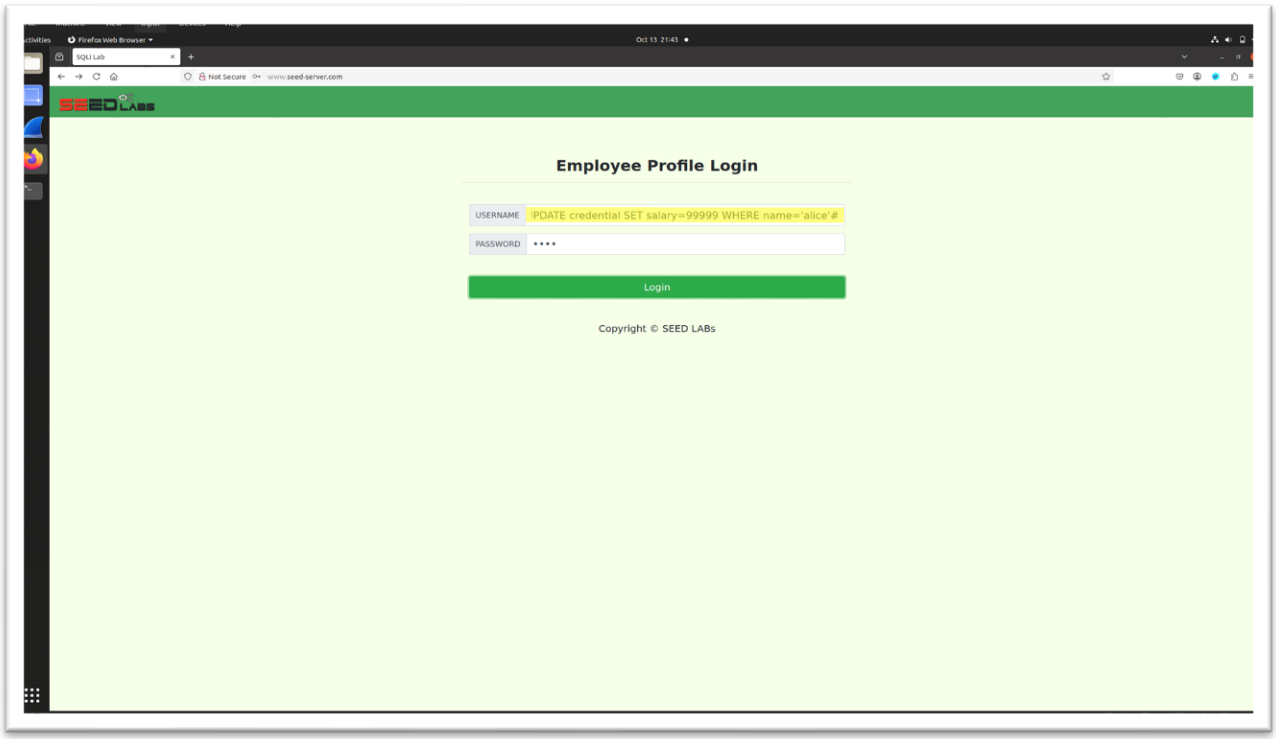


Screenshot 8: inputting “admin'; DELETE FROM credential WHERE name='alice'#” in user login

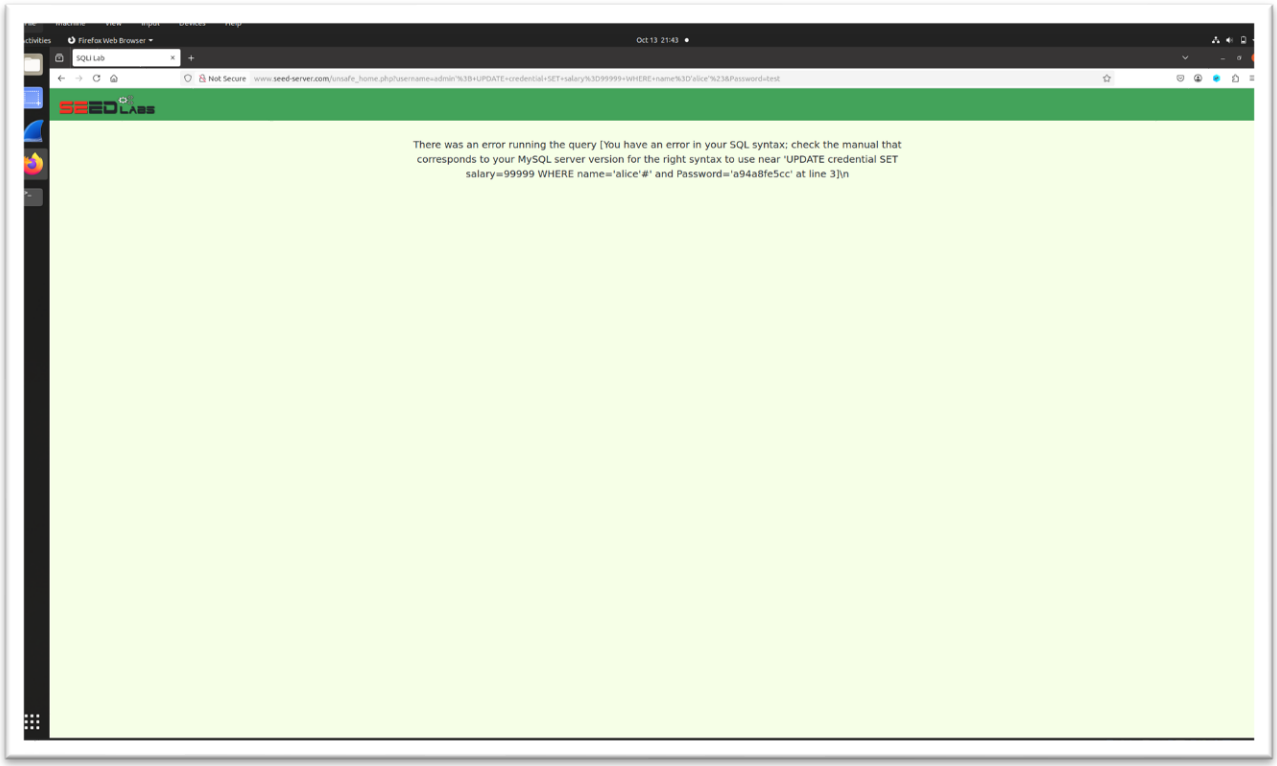


Screenshot 9: Message of failure of SQL injection.

- Here's another SQL query of "admin'; UPDATE credential SET salary=99999 WHERE name='alice'#" (as highlighted in Screenshot 10) which fails as shown in Screenshot 11.



Screenshot 10: inputting "admin'; UPDATE credential SET salary=99999 WHERE name='alice'#" in user login.



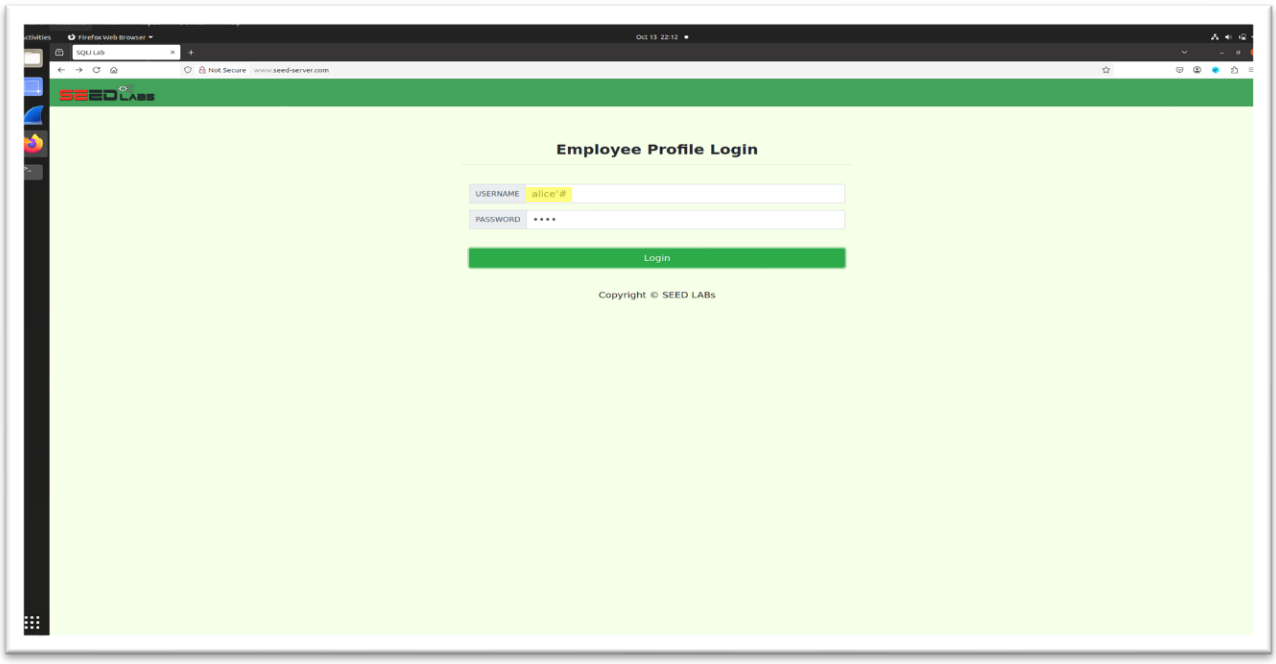
Screenshot 11: Message of failure of SQL injection.

- This fails because attack PHP's mysqli::query() function in PHP is designed to execute only ONE SQL statement per call. It does not support executing multiple SQL statements separated by semicolons, even if they are syntactically valid SQL code. This is a defense-in-depth mechanism at the API level that prevents attackers from chaining multiple dangerous commands like DROP TABLE or DELETE.
- But this cannot be a proper solution as:-

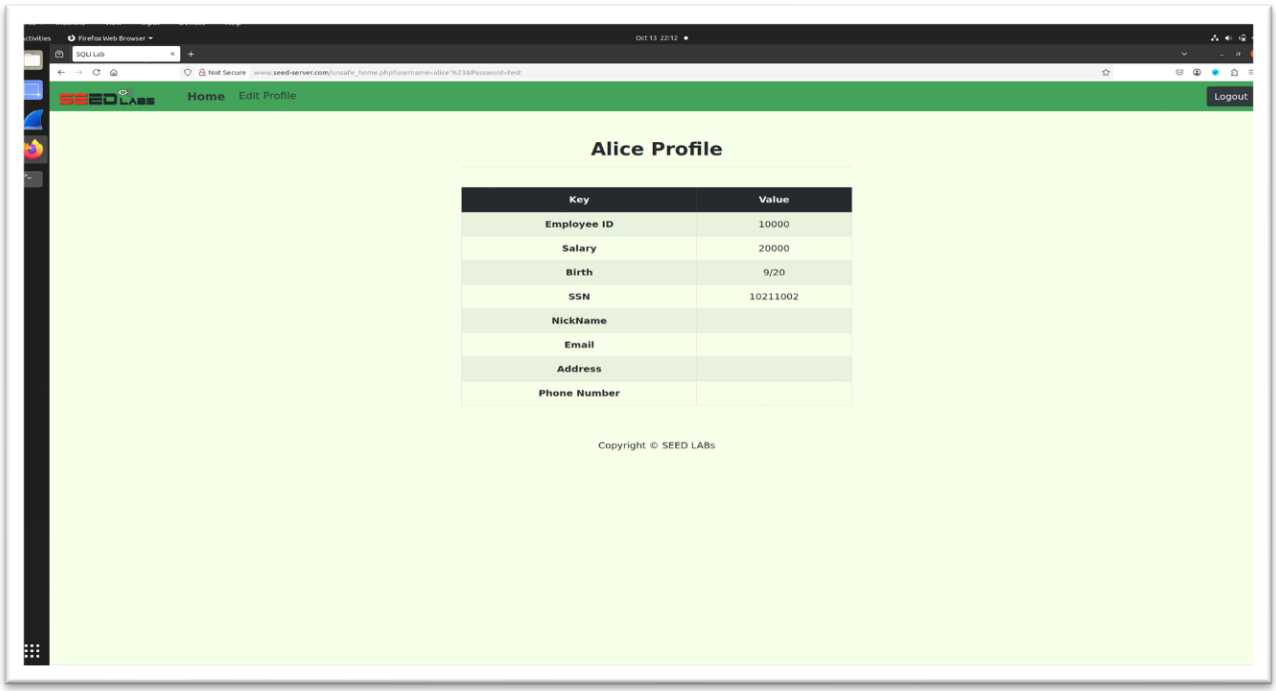
1. Attackers can still use single-statement attacks.
2. It still doesn't protect us from INSERT, SELECT and UPDATE injection attacks.
3. Don't forget that it's in API format and may not exist in other languages/frameworks.

Task 3

- To modify the salary of Alice, we need to access her account. So, for that we will use “alice’#” as highlighted in Screenshot 12 and we can observe that original salary of Alice is 20000 from Screenshot 13.

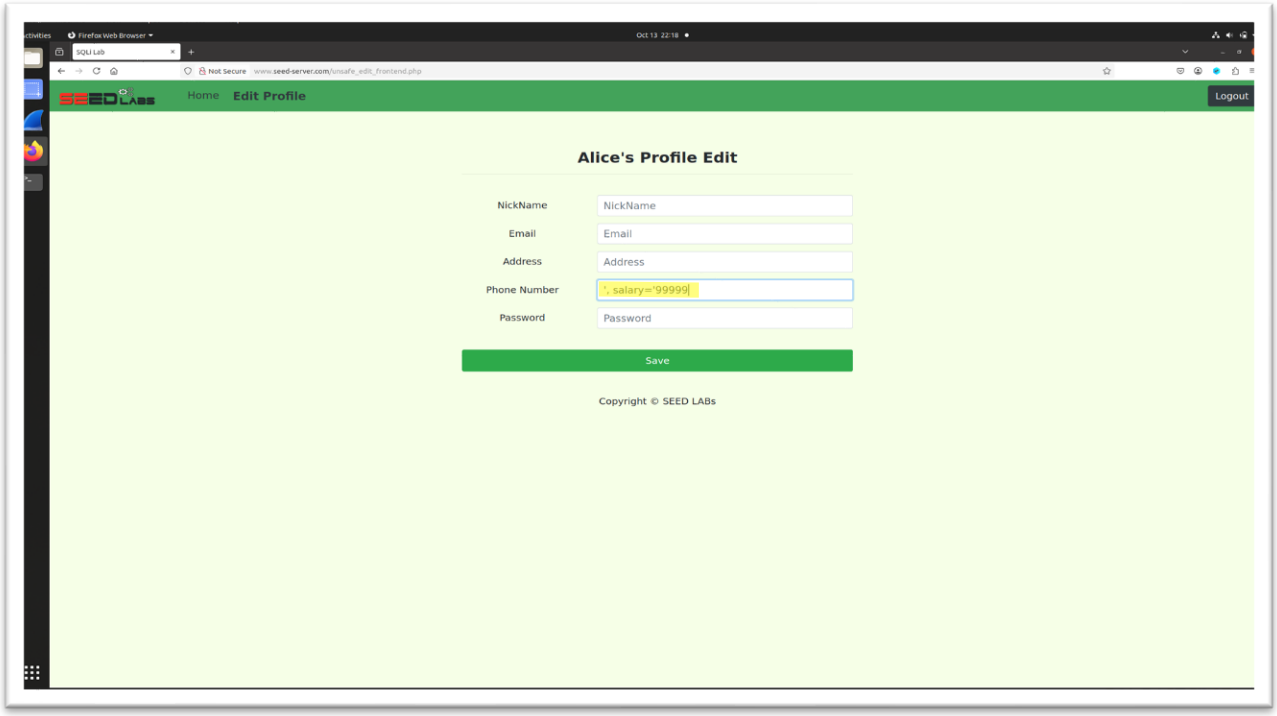


Screenshot 12: Injecting SQL in Profile login using “alice’#”.

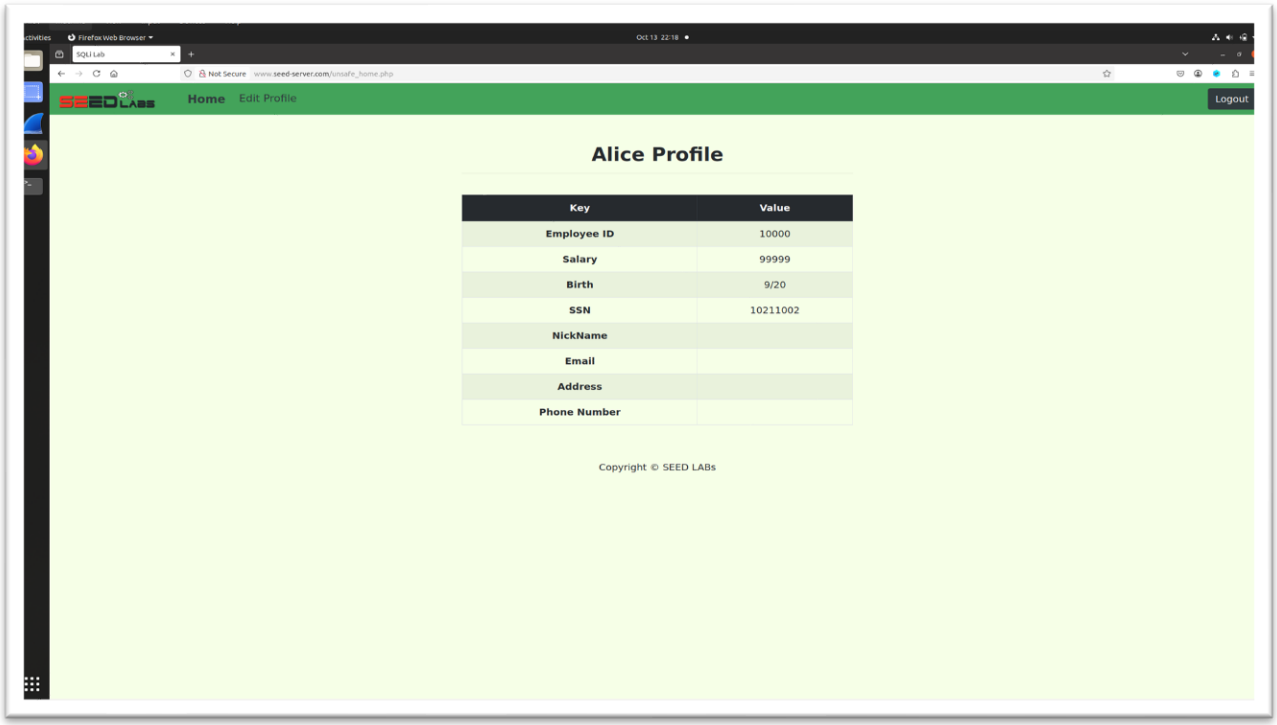


Screenshot 13: Details of Alice Profile by successful SQL injection.

- Now, we will select Edit Profile and then in Phone Number, we will enter “’, salary=’99999” (as highlighted Screenshot 14) which will edit her salary from 20000 to 99999 as observed in Screenshot 15.

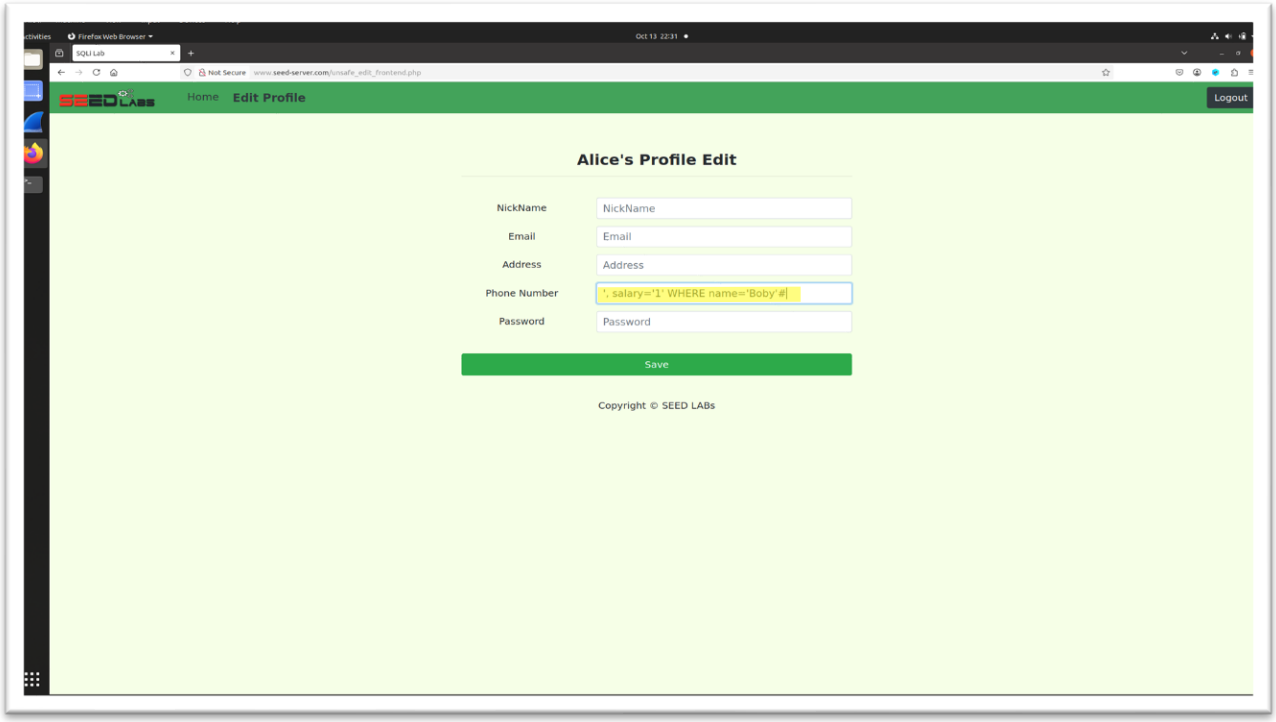


Screenshot 14: using “’, salary=’99999” to perform SQL injection.

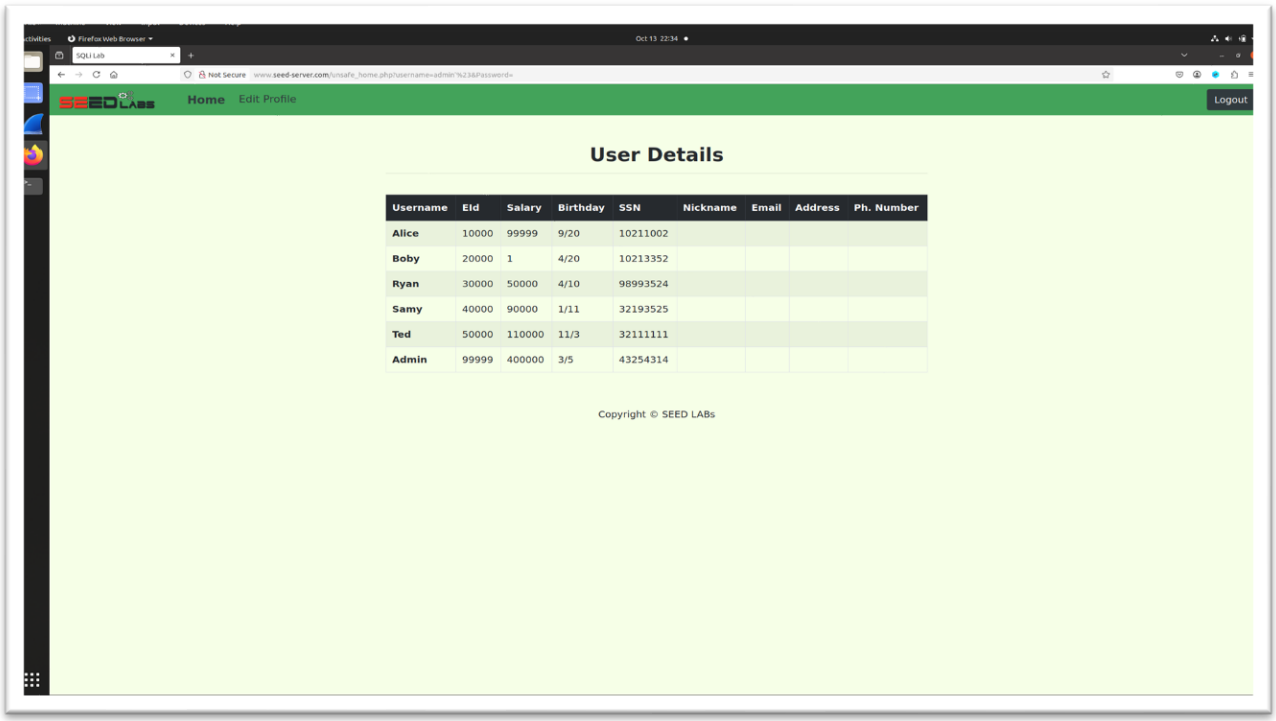


Screenshot 15: Information of Alice’s Profile.

- Now to edit someone else's salary, we can do that from Alice's Profile by simply entering "', salary='1' WHERE name='Boby'#" (as highlighted Screenshot 16) which will change Boby's salary to 1 as observed in Screenshot 17.

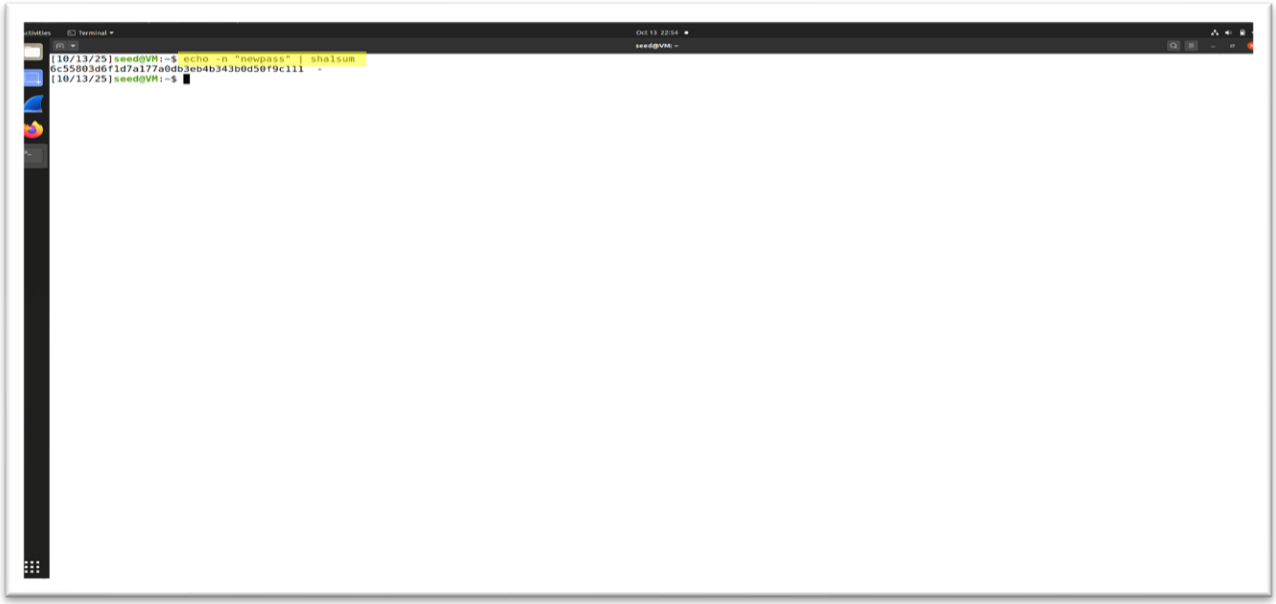


Screenshot 16: using "', salary='1' WHERE name='Boby'#" for SQL injection.



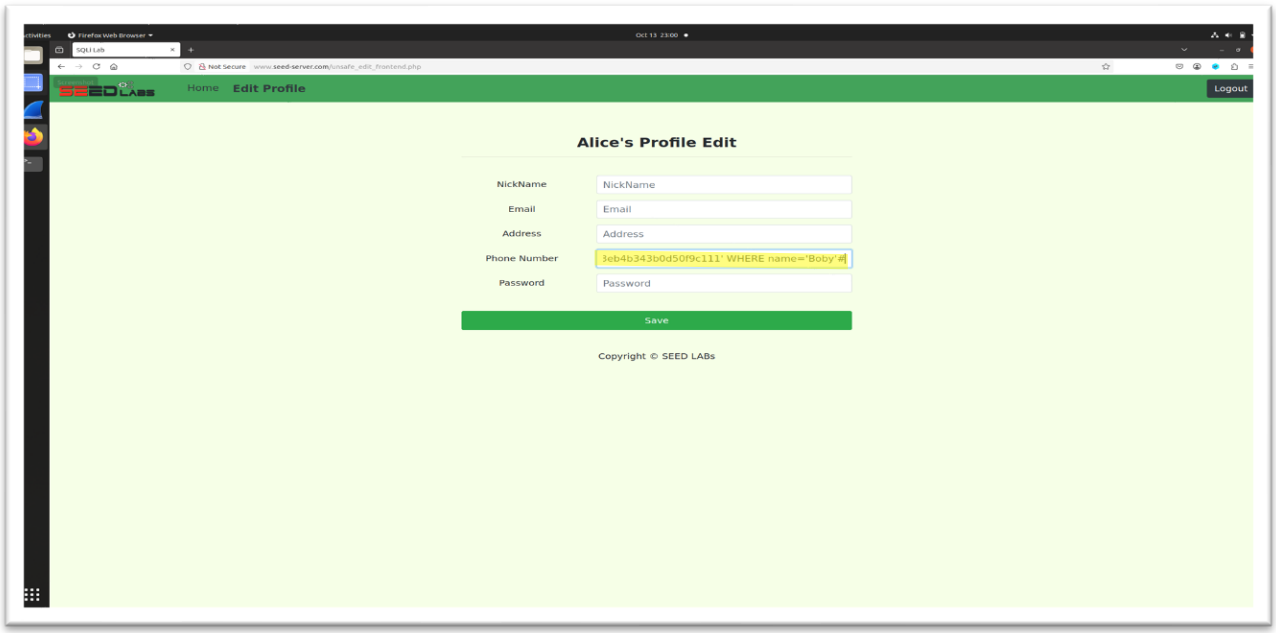
Screenshot 17: Details of every employees by successful SQL injection.

- Now, to modify other's password, we need a SHA1 hash of the new password. So, we enter "echo -n "newpass" | sha1sum" in terminal which gives us hash as highlighted in Screenshot 18.



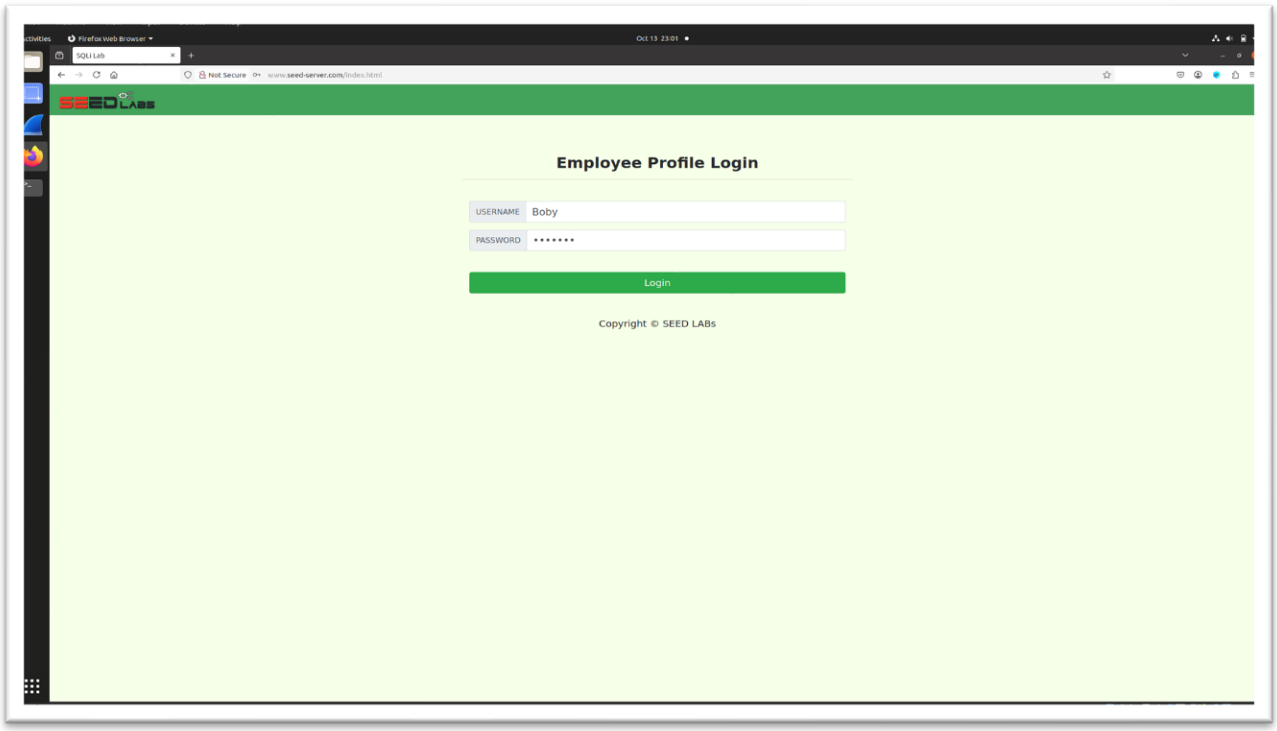
Screenshot 18: "echo -n "newpass" | sha1sum" command to get password hash.

- Then we input SQL query in phone number column in Alice profile- ', password='6c55803d6f1d7a177a0db3eb4b343b0d50f9c111' WHERE name='Boby'# as highlighted in Screenshot 19.

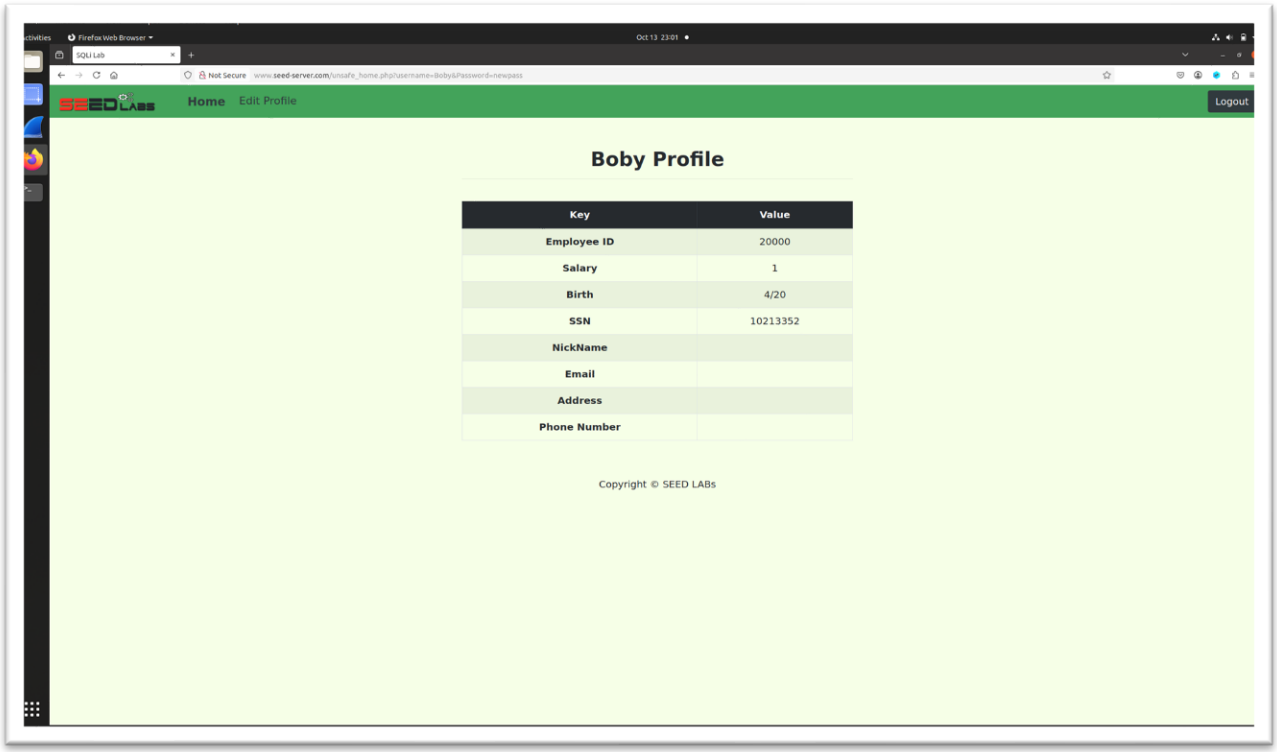


Screenshot 19: "', password='6c55803d6f1d7a177a0db3eb4b343b0d50f9c111' WHERE name='Boby'#" SQL injection to change password.

- That SQL code will change Boby's password and we can observe I entered new password "newpass" in Screenshot 20 and got into the account as observed in Screenshot 21.



Screenshot 20: Entering new password in Boby's login credentials.



Screenshot 21: Successful Boby's login using new password.

Task 4

- As we can see in Screenshot 22 that edit contains inside php file of website using “nano unsafe.php” and replace it with “$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn

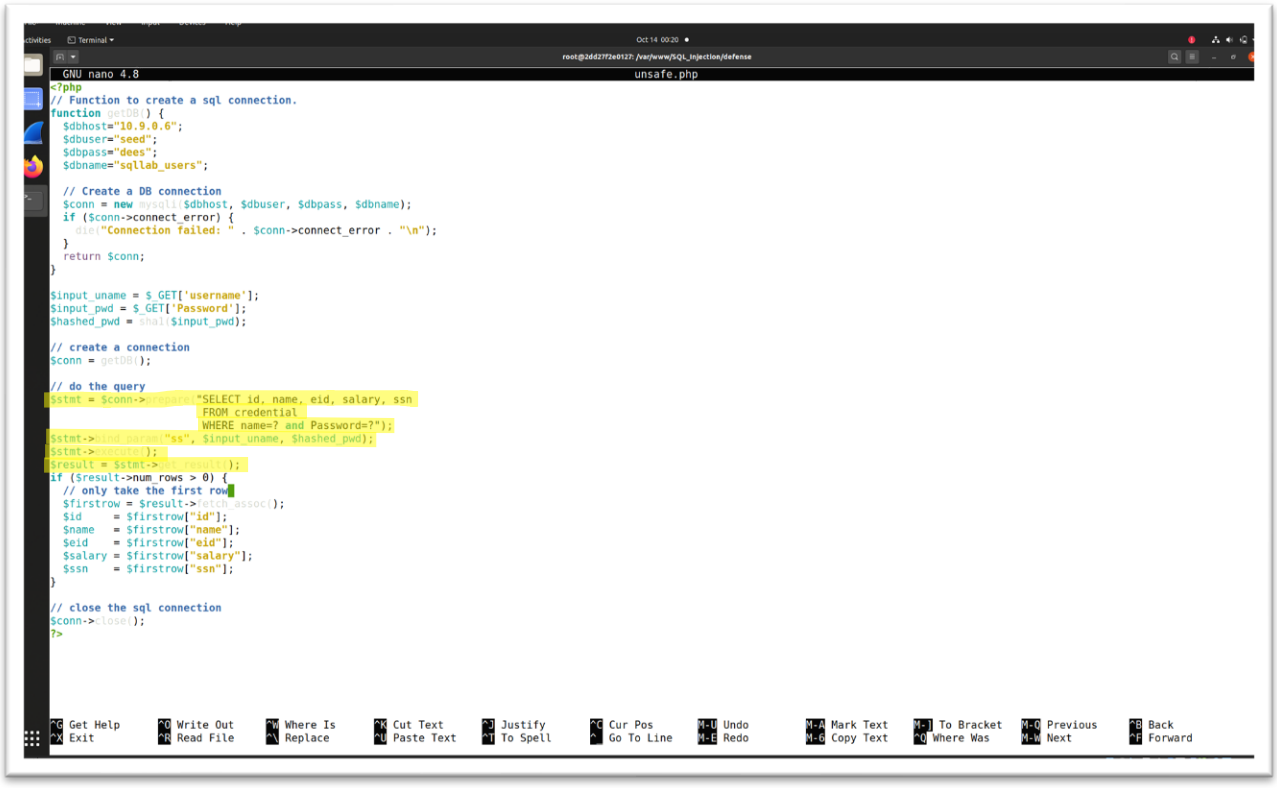
FROM credential

WHERE name=? and Password=?");

$stmt->bind_param(“ss”, $input_uname, $hashed_pwd);

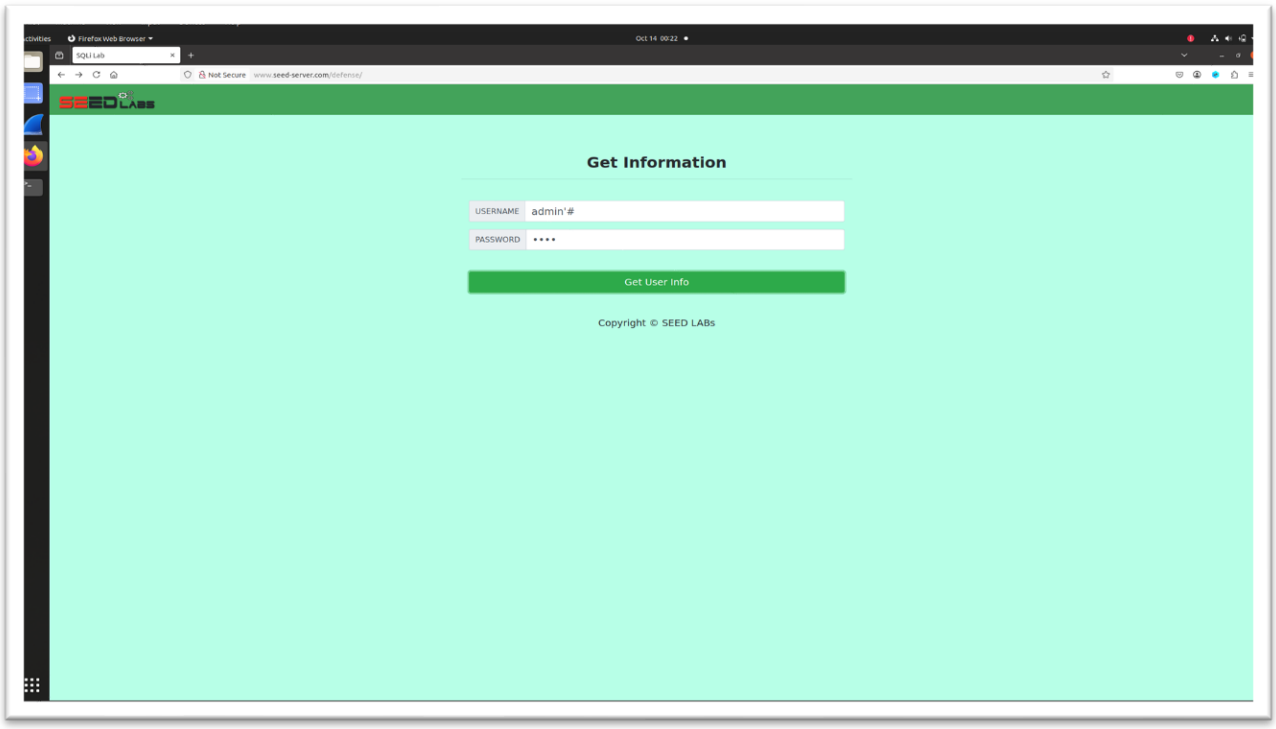
$stmt->execute();

$result = $stmt->get_result();” which is highlighted in Screenshot 22 which will block most of the SQL injection attacks as we can see from two example which I gave.

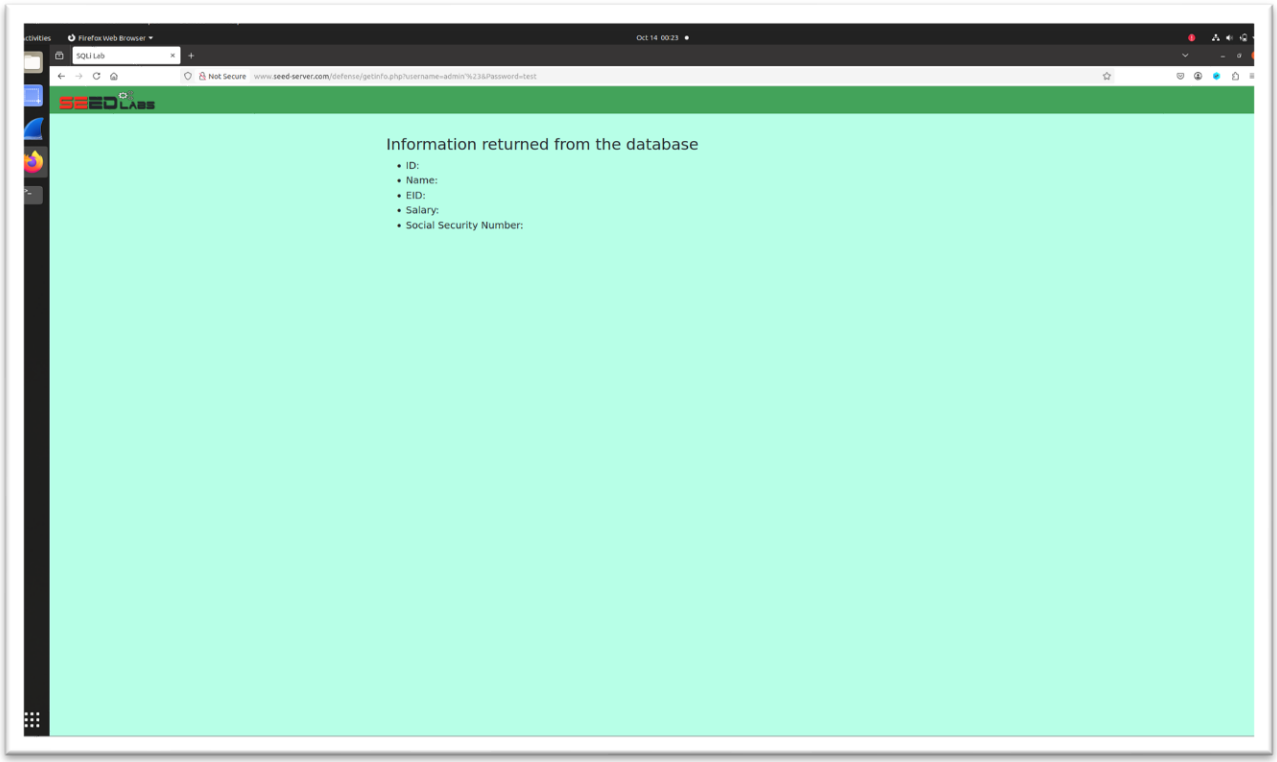


Screenshot 22: editing unsafe.php to protect against SQL injection.

- Example 1 is SQL query “admin’#” not working and failed as observed in Screenshot 23 and 24.

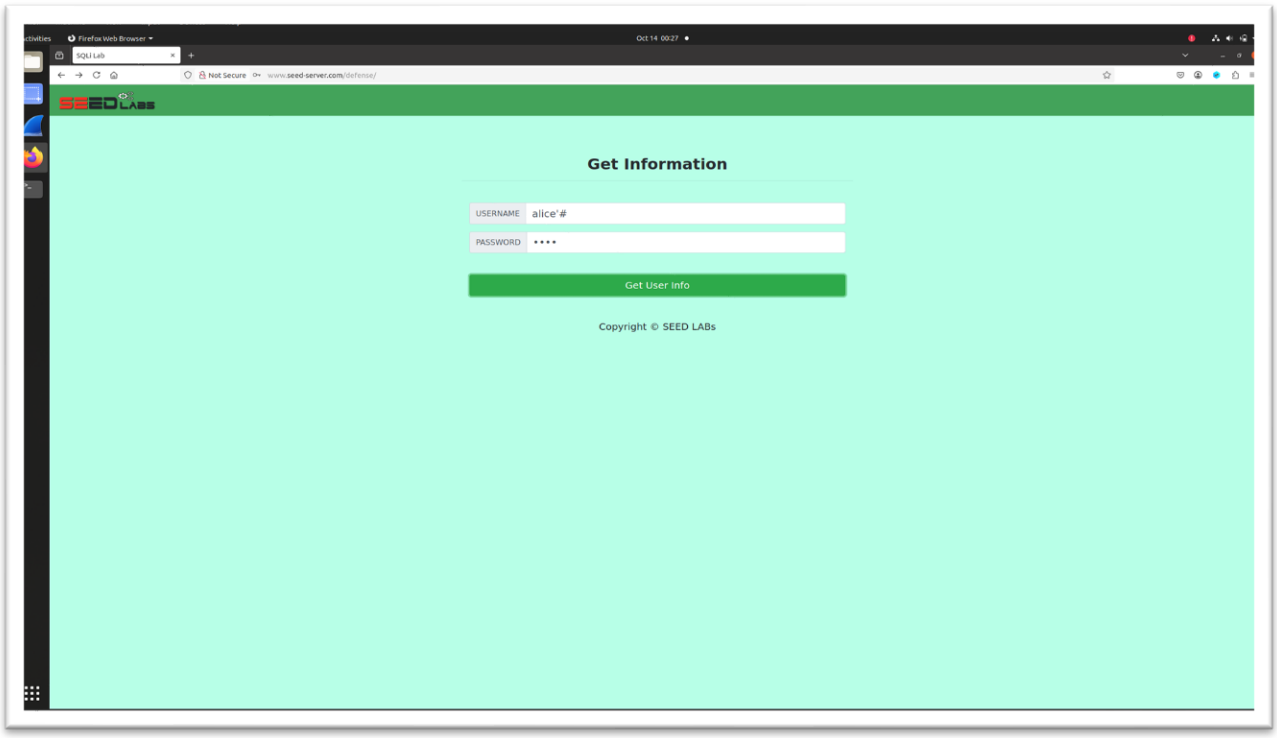


Screenshot 23: Example 1- inputting “admin’#” SQL injection.

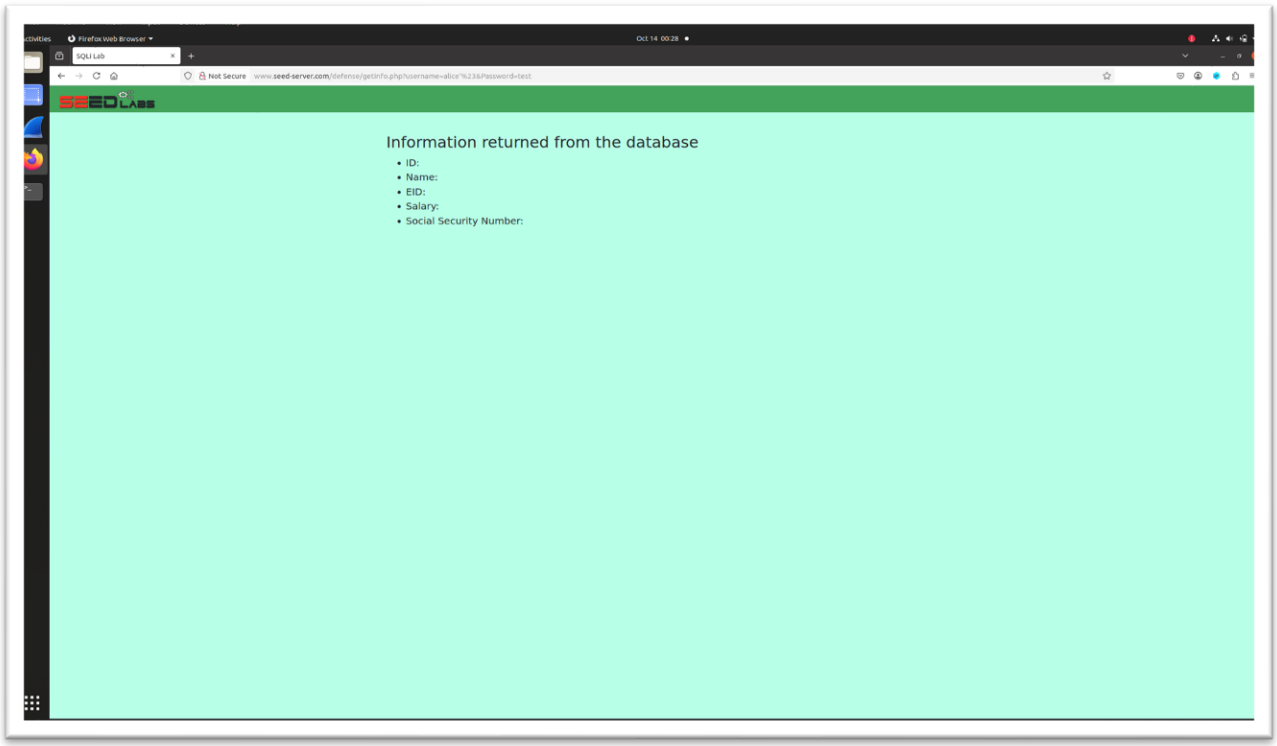


Screenshot 24: Unsuccessful login into the admin’s account.

- Example 2 is SQL query “alice’#” not working and failed as observed in Screenshot 25 and 26.



Screenshot 25: Example 2- inputting “alice’#” SQL injection.



Screenshot 26: Unsuccessful login into Alice’s account.