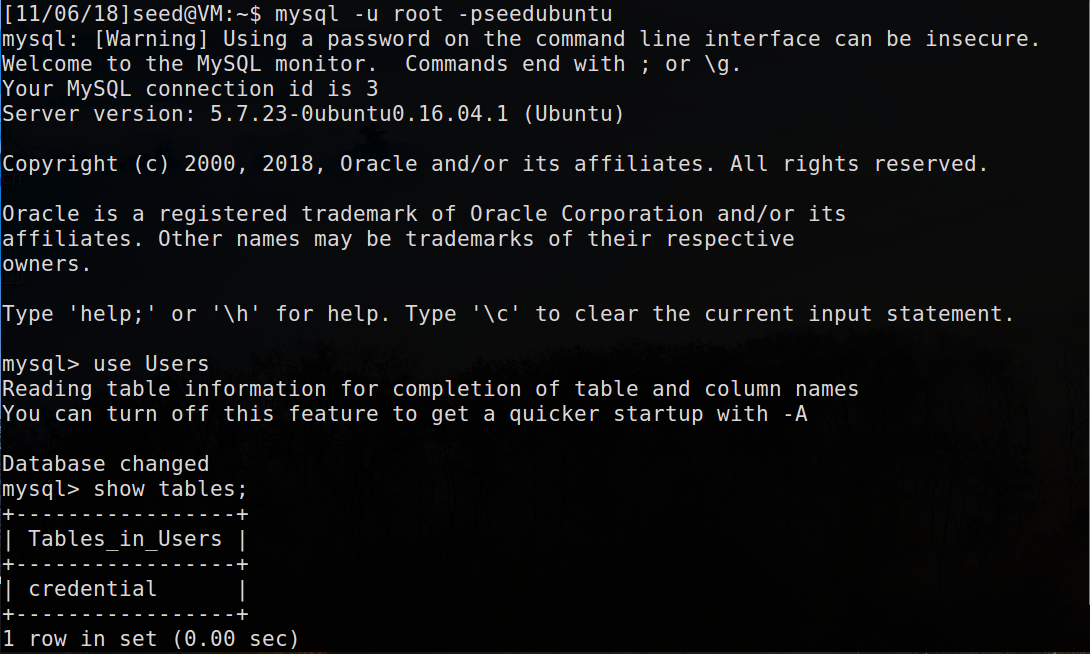
**SQL Injection Lab**

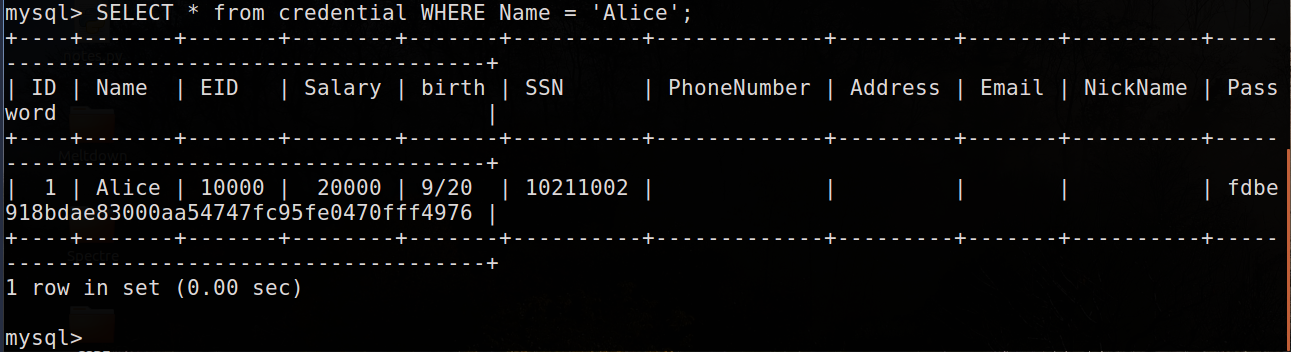
**Karan Amrutesh**

**Task 1: Get Familiar with SQL Statements:**

* Executing the commands given in the lab manual:



* Printing all the profile information of Alice:

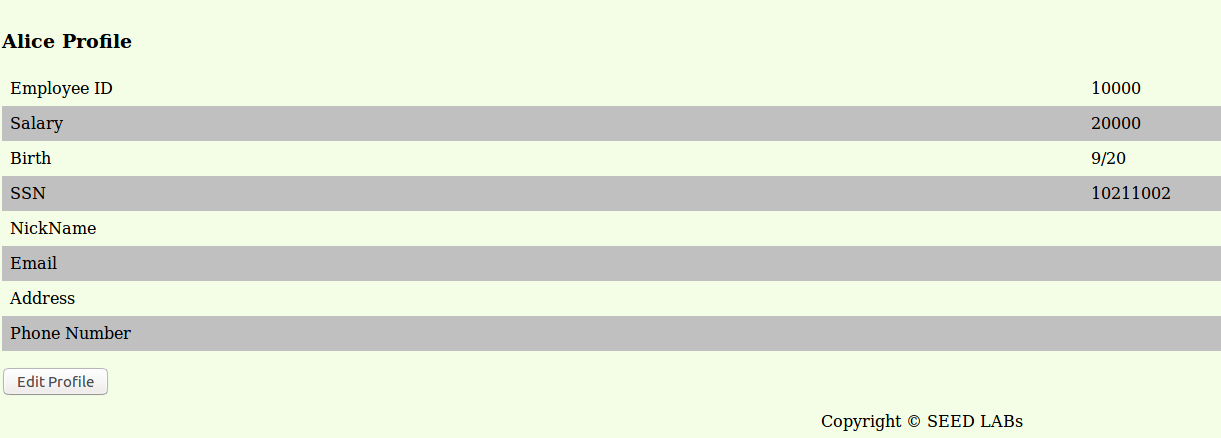


* Here we use the Where clause to compare the /name column to Alice and get her information.

**Task 2: SQL Injection Attack on SELECT Statement**

* By entering the following in the Employee ID, we get all the information from the database as the OR condition is always true and it retrieves an employee’s information as shown in the next snapshot.
* The first quote is to close the quote of the comparison of Employee ID and then put an OR which returns true always and then use # to comment out the rest of the line.



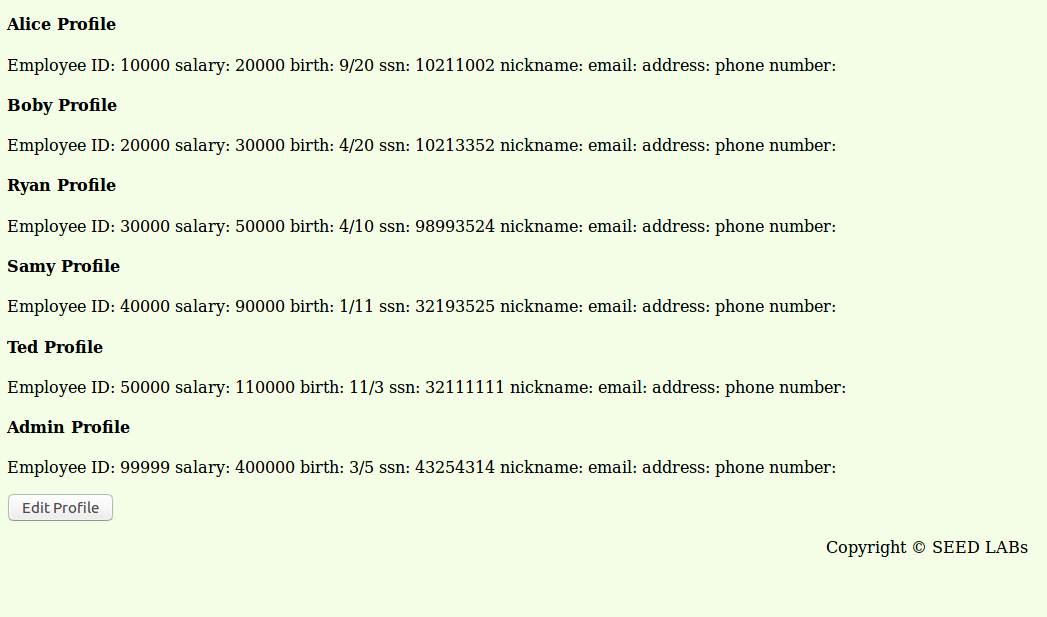


**Task 2.1: SQL Injection Attack from webpage**

* We enter the following text in order to obtain all employee records as an admin.



* We first close the quote and write and OR clause to match the name field as ‘admin’ and then comment rest of the line with #.
* We need not enter anything in the password field as the password comparison is on the same line in the sql query and is commented out by the # that we entered in the ID field.
* We can see that all the employee records are obtained:



**Task 2.2: SQL Injection Attack from command line.**

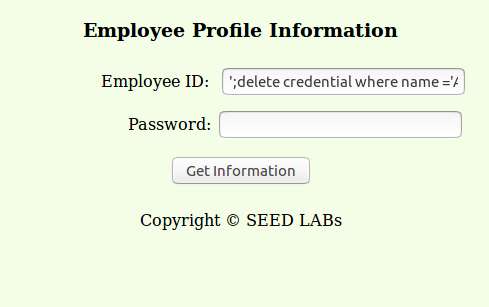
* We use the curl command in the command line to get access the employee records:

***curl 'www.seedlabsqlinjection.com/unsafe\_credential.php?EID=%27%20OR%20name=%27admin%27%23&Password='***

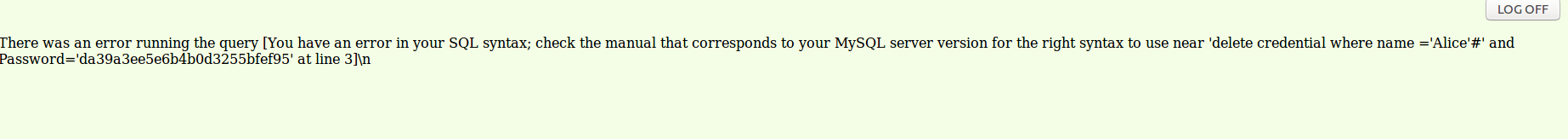
* The SQL statement used here is same as the one used in the previous task in the web page.
* We just encode the special characters in the command as they may be misinterpreted.
* So here we close the quote after EID= by %27 and give a space (%20) followed by OR statement and so on.
* After the admin name comparison we comment the rest of the line using ‘#’ (%23).



**Task 2.3: Append a new SQL statement**



* When we try to do this, we get an error.



* This is because the mysqli extension in PHP does not allow multiple queries to run in database server.

**Task 3: SQL Injection Attack on UPDATE Statement**

**Task 3.1: Modify your own salary**

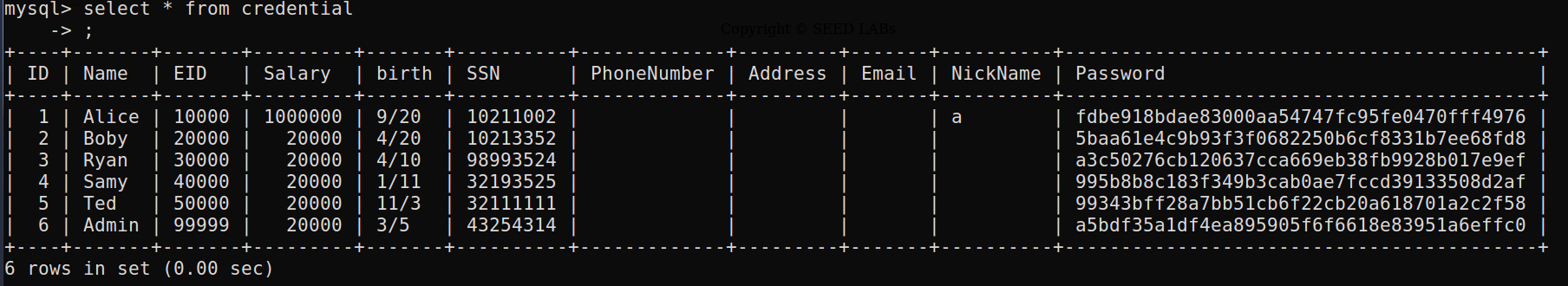
* We login into Alice’s account using SQL injection as follows:

*a', salary = 1000000 where name ='Alice' #*



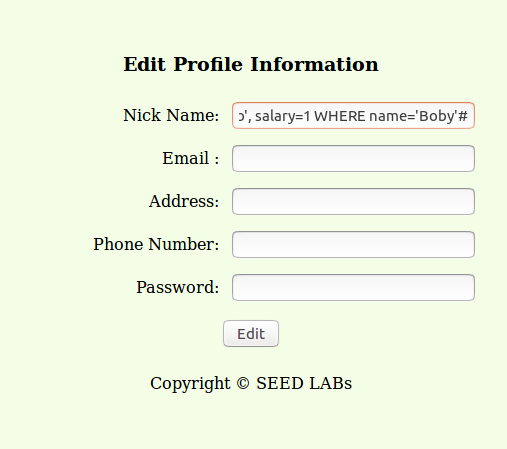
* We have to specify the Where clause to change only Alice’s salary and not everyone’s.
* In the Nickname field we just enter a nickname, close the quote and also set a salary field to any value we want followed by ‘#’ to comment rest of the fields in the command.
* We can see that the salary is successfully updated:



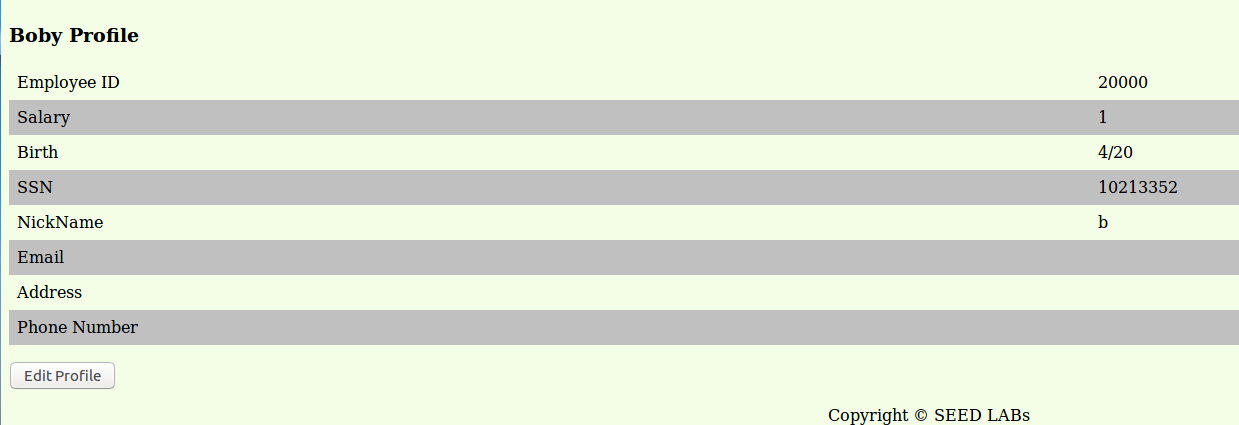


**Task 3.2: Modify other people’ salary**

* In Alice’s edit profile page, we update Boby’s salary by giving setting the salary field to 1 and specifying the WHERE clause to check name = Boby and then comment rest of the line.

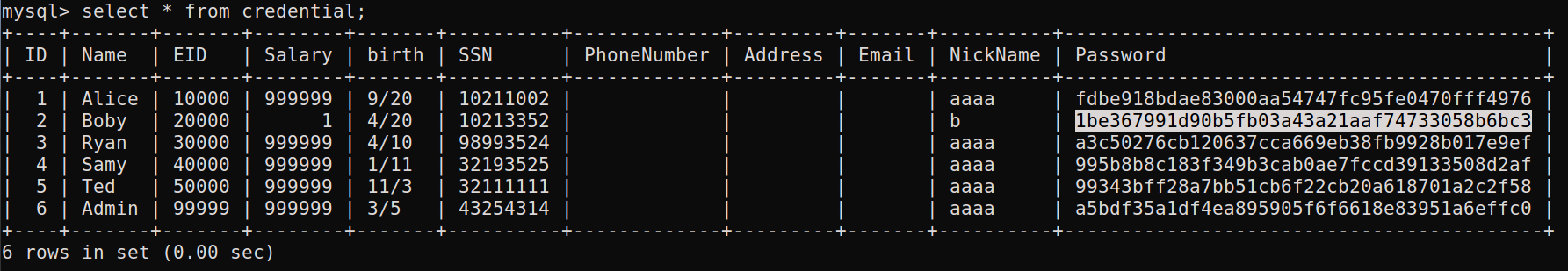


* After logging into Boby’s account, we can see that Boby’s salary is changed to 1:



**Task 3.3: Modify other people’ password.**

* Before changing the password:

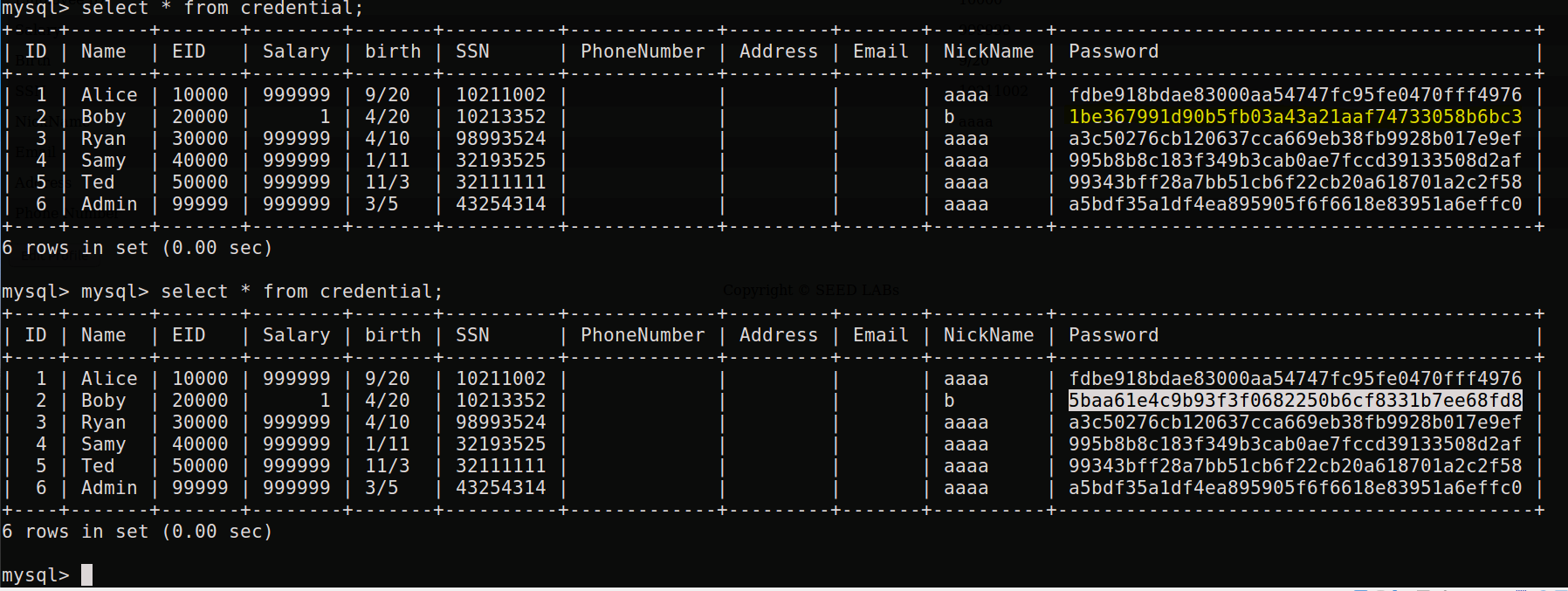


* In Alice’s edit page we enter the following in the nickname field:

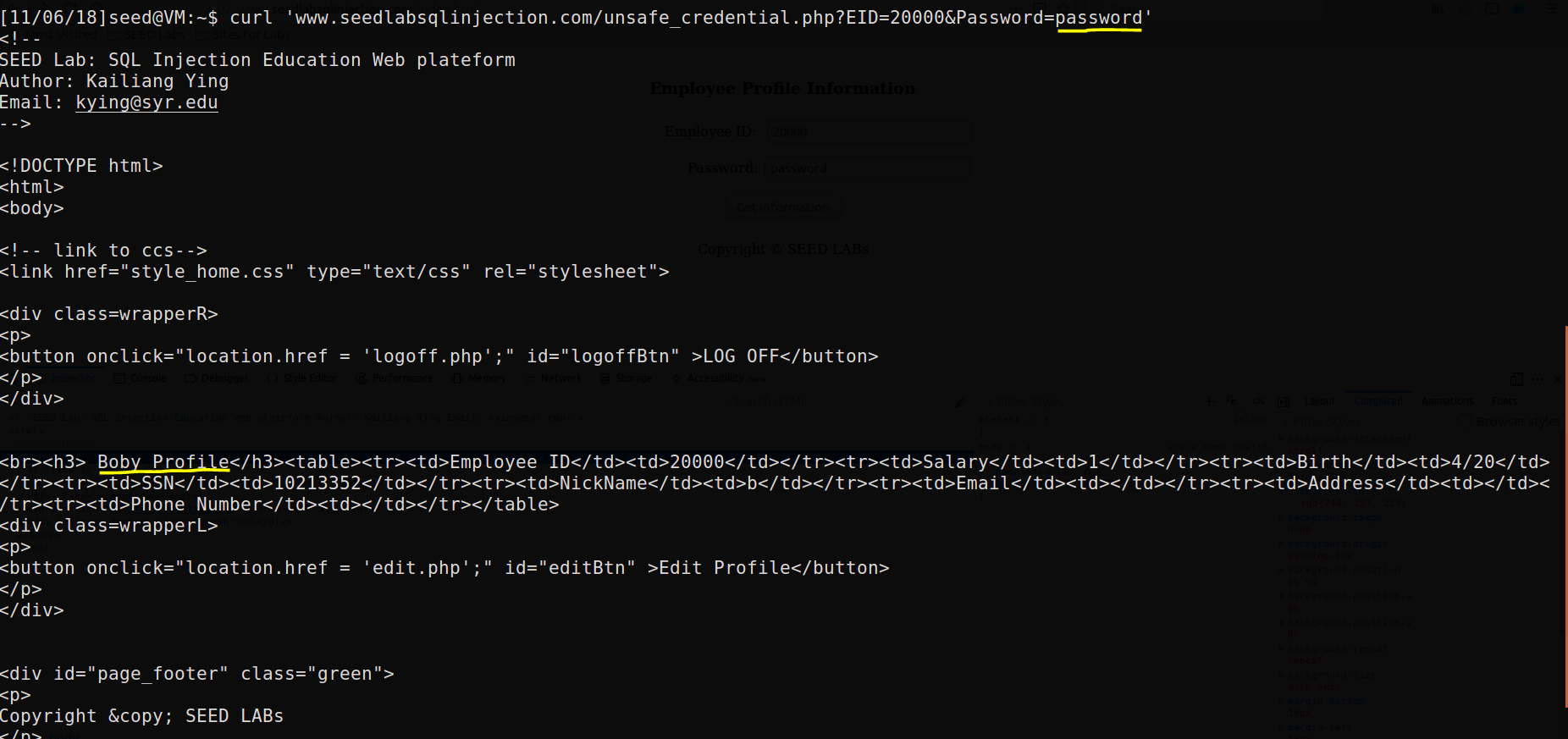
*b', password=sha1(‘password’) WHERE name ='Boby'#*



* We have to hash the password and store it. So while assigning it to password field, we hash the value and then assign it.
* Since the sha1 takes a string, we give the password in quotes.
* After the attack, we can see that the password as successfully changed:

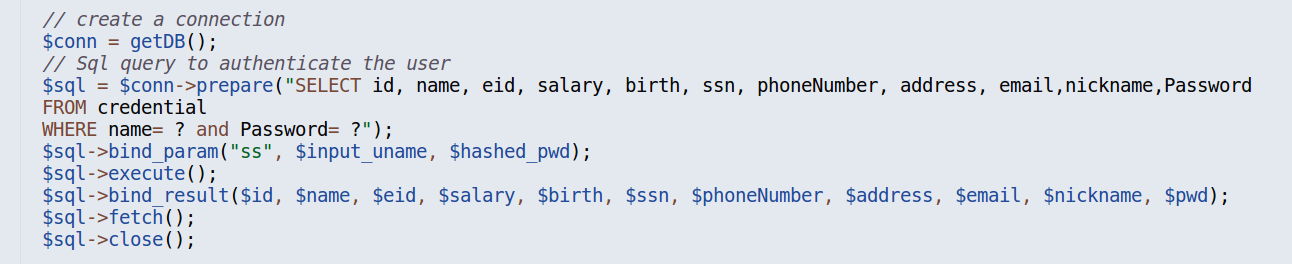


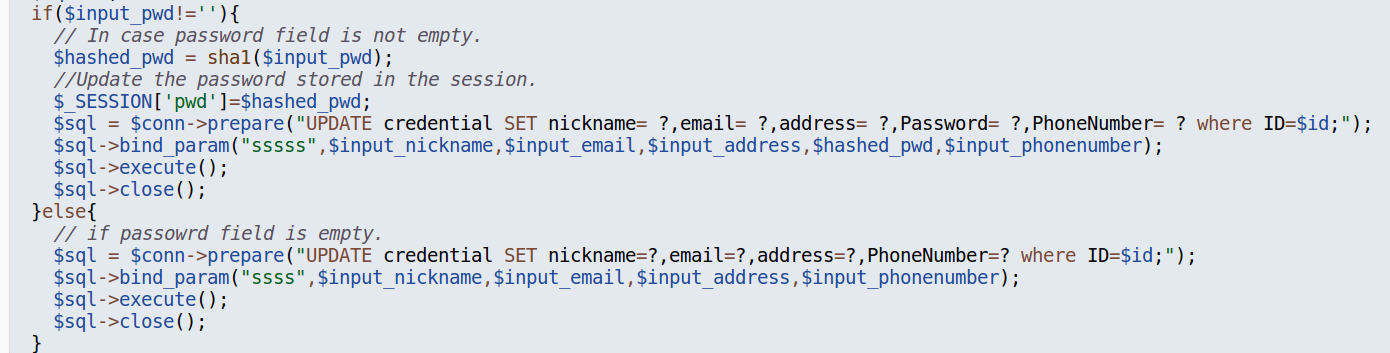
* By using the curl command to login into Boby’s profile with his EID and the new password we have given, we can see that the attack is successful:



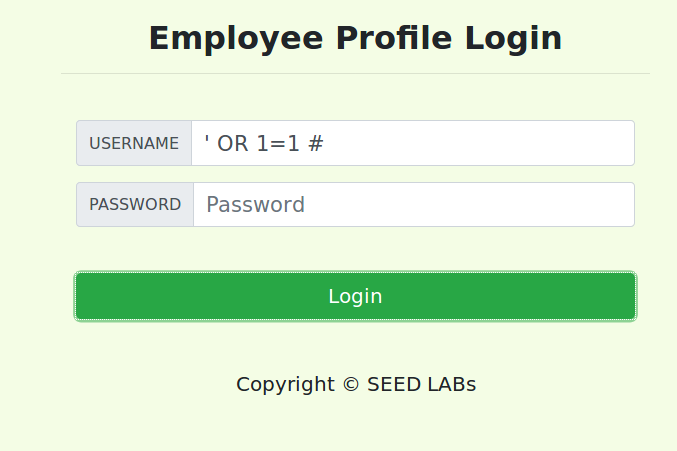
**Task 4: Countermeasure — Prepared Statement**

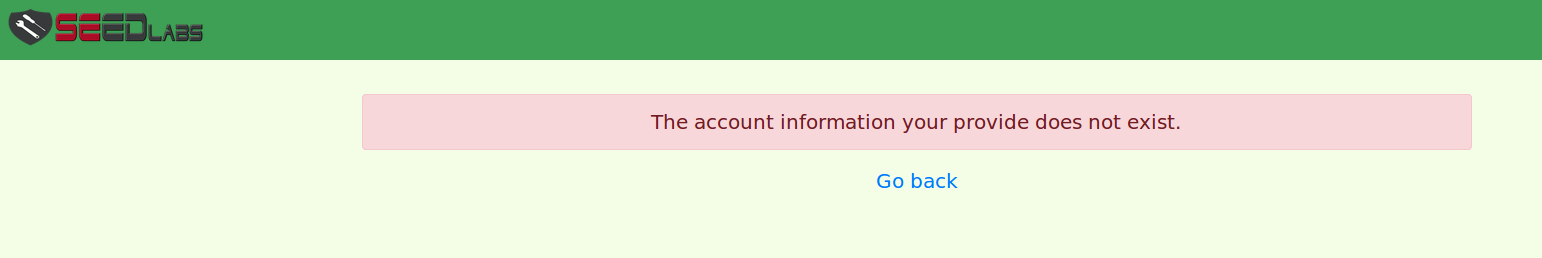
* Using prepared statement in the PHP files for the SQL query





We can see that the SQL injection done in Task 2 fails:





*Note*: In the previous tasks, I was using an older image of Ubuntu 16.04. So the login page is slightly different then used in the last task, but the SQL injection technique remains the same.