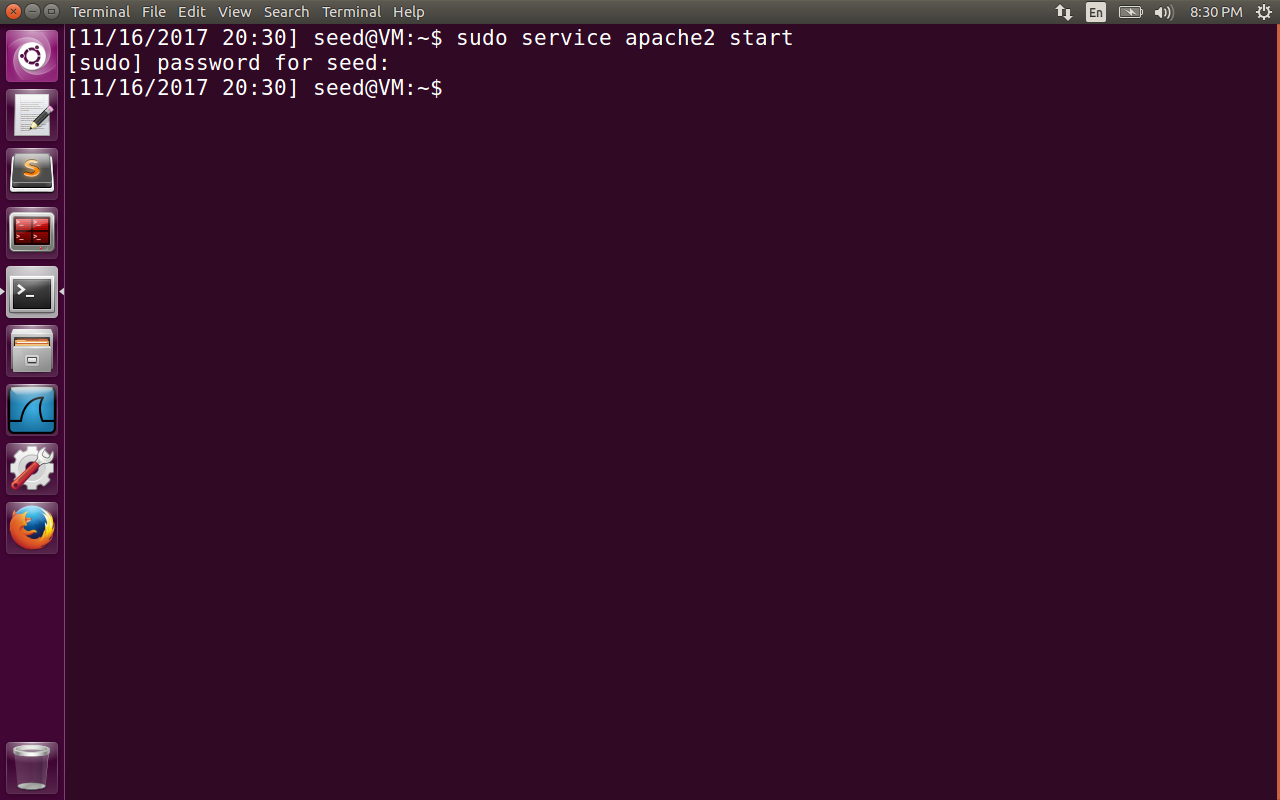
SQL Injection Attack Lab

NAME : JASHWANTH REDDY GANGULA

SUID: 646254141

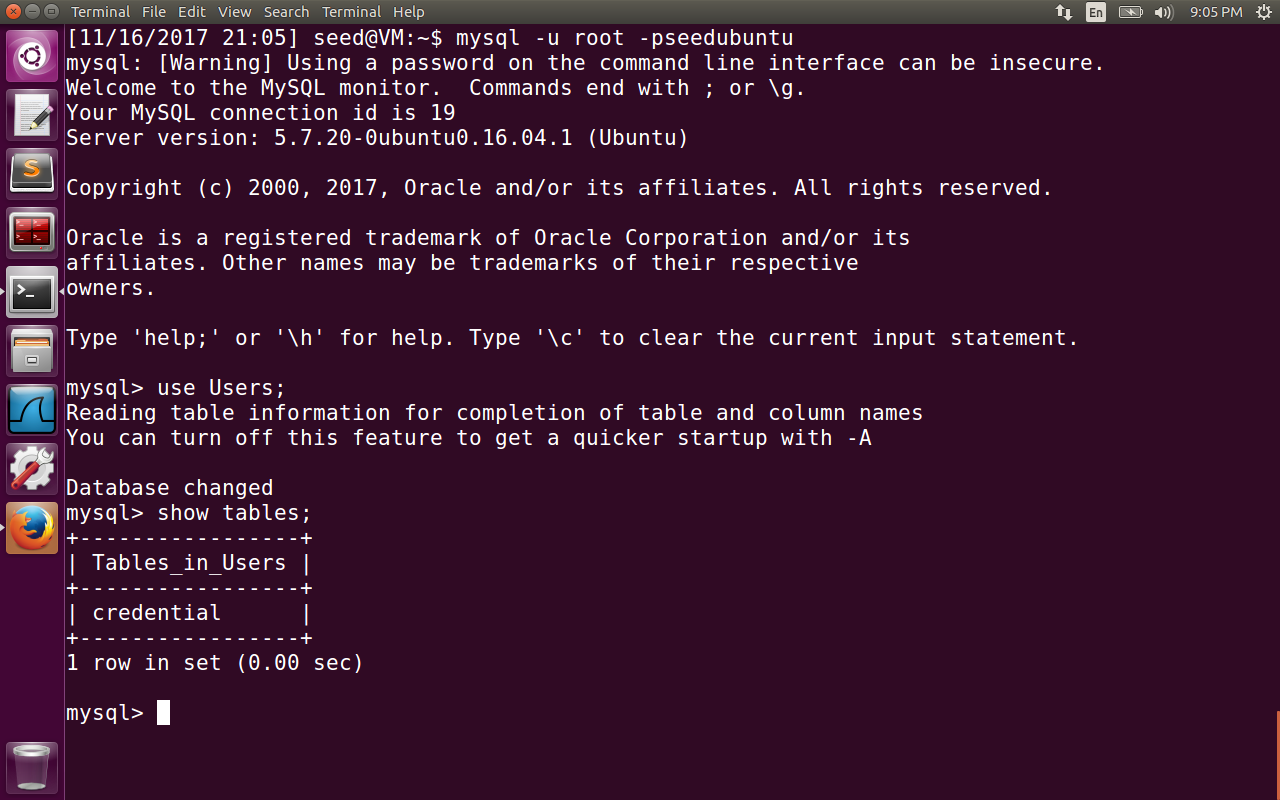
2.1 First we have apache web server started.



2.2 Turn Off the Countermeasure

SEED Ubuntu 16.04 has PHP version 7 and magic\_quotes\_gpc is deprecated here.

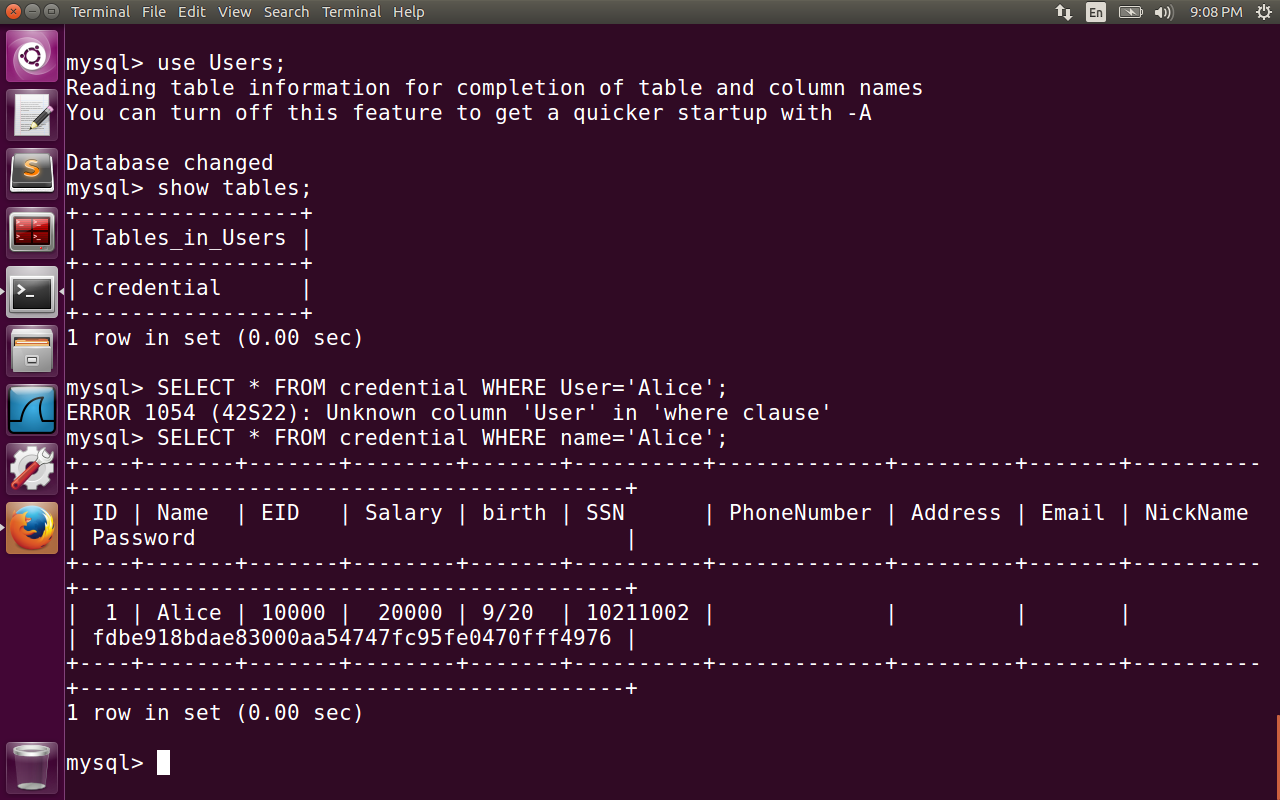
3.1 Task 1: MySQL Console  
We have created a databased called Users with a table called credential. The table stores the personal information of each employee.



We have setup MYSQL in the VM image, with username: root and password: seedubuntu

After that we load a database called Users into MYSQL. Show tables; list the table ‘credential’ under Users database.

Now we run a SQL command ‘SELECT \* FROM credential where name=’Alice’ to get the profile information of Alice as shown in the below screenshot.



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

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

Here we have to login into the web application as the administrator without knowing his password. We have to type something in the EmployeeID and password to login without knowing the credentials of the administrator.

$conn = getDB();

$sql = "SELECT id, name, eid, salary, birth, ssn, phonenumber, address, email, nickname, Password FROM credential WHERE eid= ’$input\_eid’ and password=’$input\_pwd’";

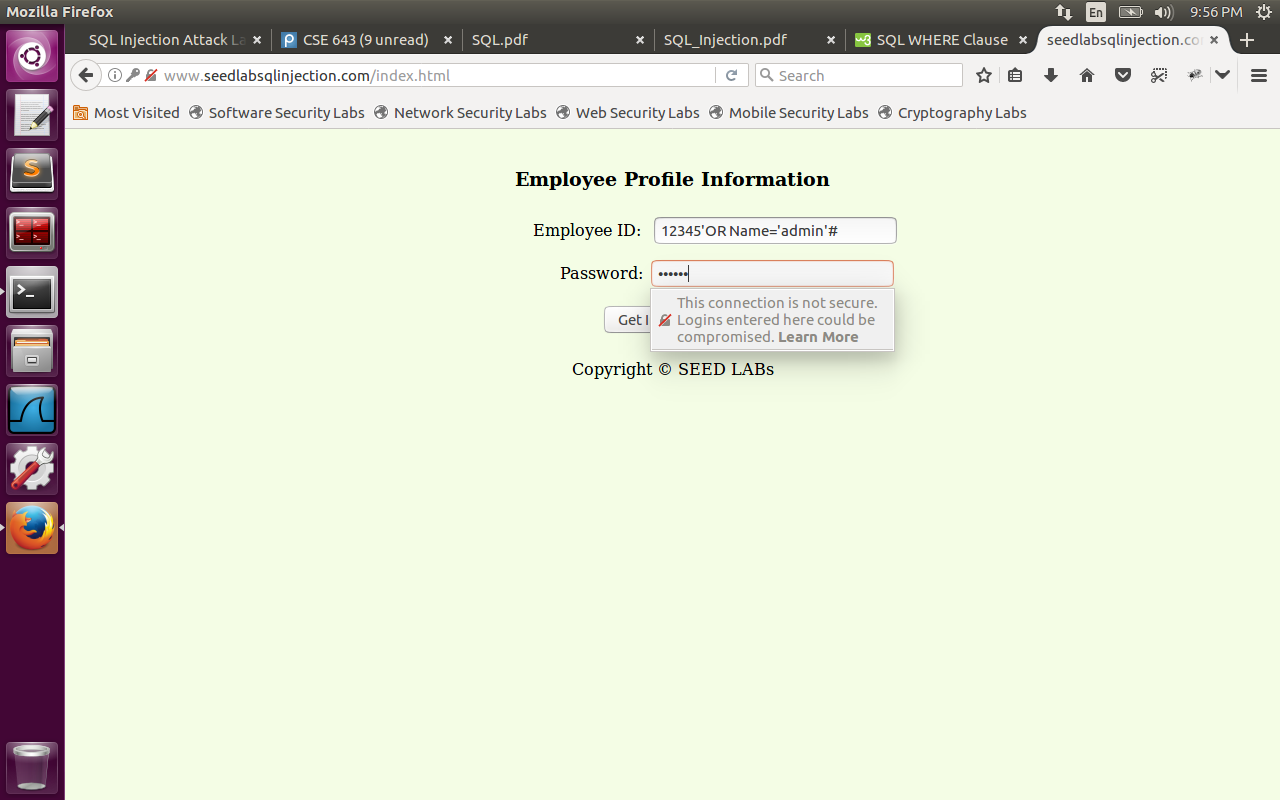
$result = $conn->query($sql));

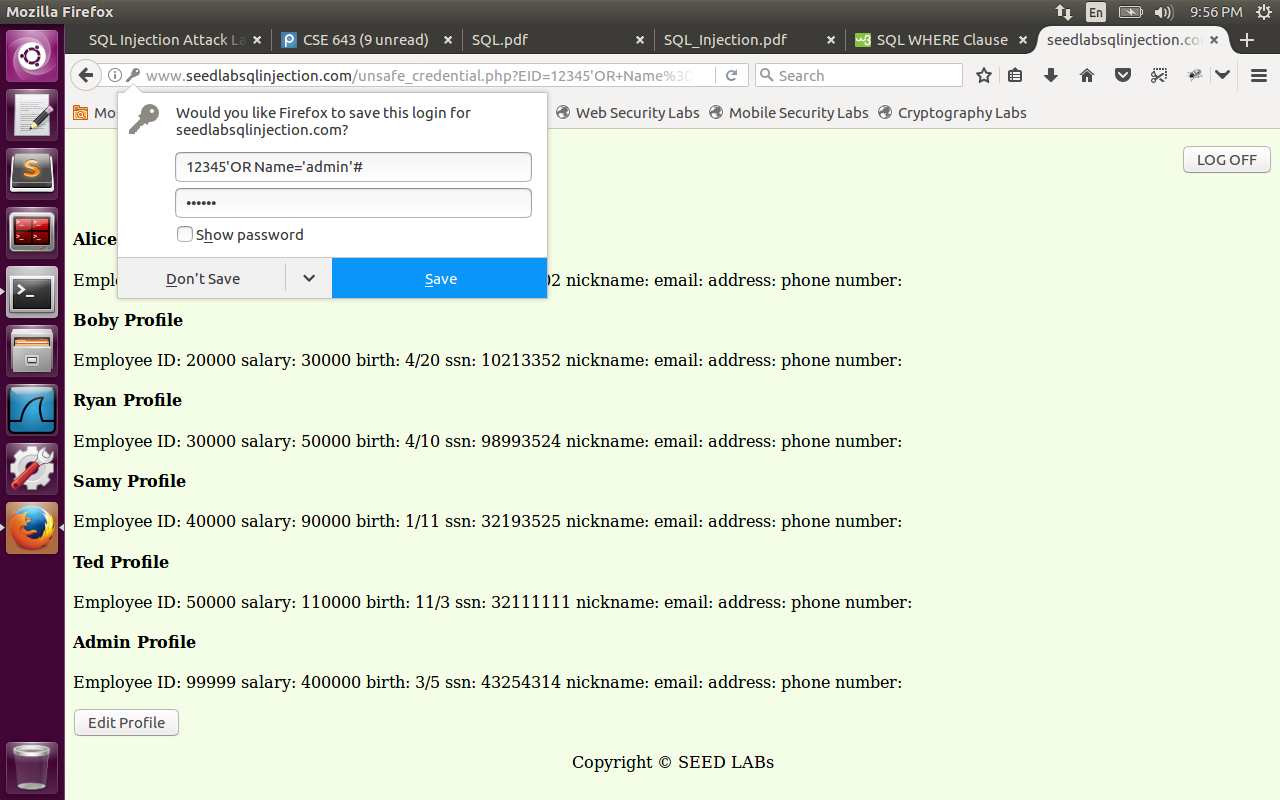
Here If we input the following string into input\_eid, we can have injection attack

The new query becomes

SELECT …. FROM credential WHERE eid=`12345`OR Name=`admin`# .

# is added so that everything after the eid commented out since we don’t know the password. Since we don’t the ID of admin, we have input some random number 12345 followed by ` to close the 1st field and OR Name=`admin’#. OR is true when one of the conditions becomes true, and we know the Name field of administrator as “admin”.





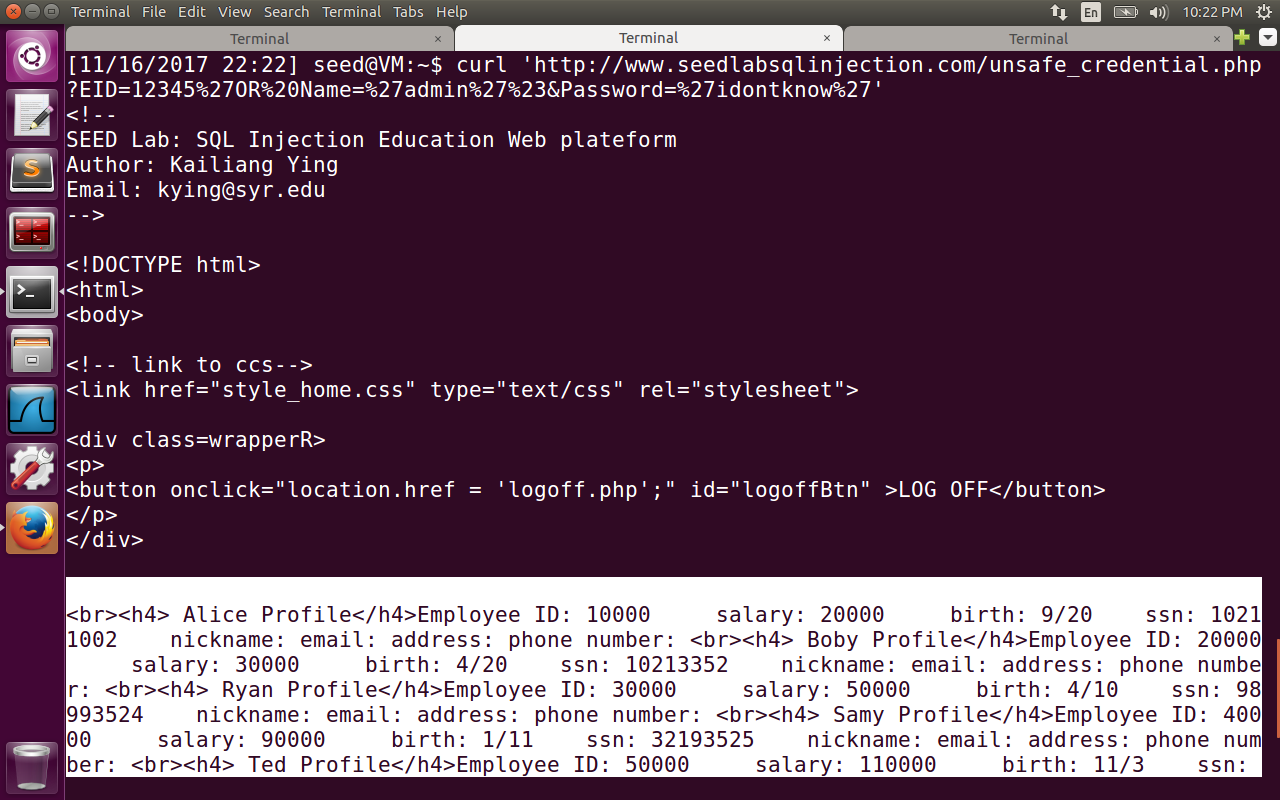
We observe that admin login is successful with all the users information displayed in the image above.

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

Space is encoded as %20, ` is encoded as %27, #(comment) is encoded as %23.

The url sent from the curl command is as follows. ‘http://www.seedlabsqlinjection.com/unsafe\_credential.php?EID=12345%27OR%20Name=%27admin%27%23&Password=%27idontknow%27’.

Please see the highlighted text to see the details of the users in the Response

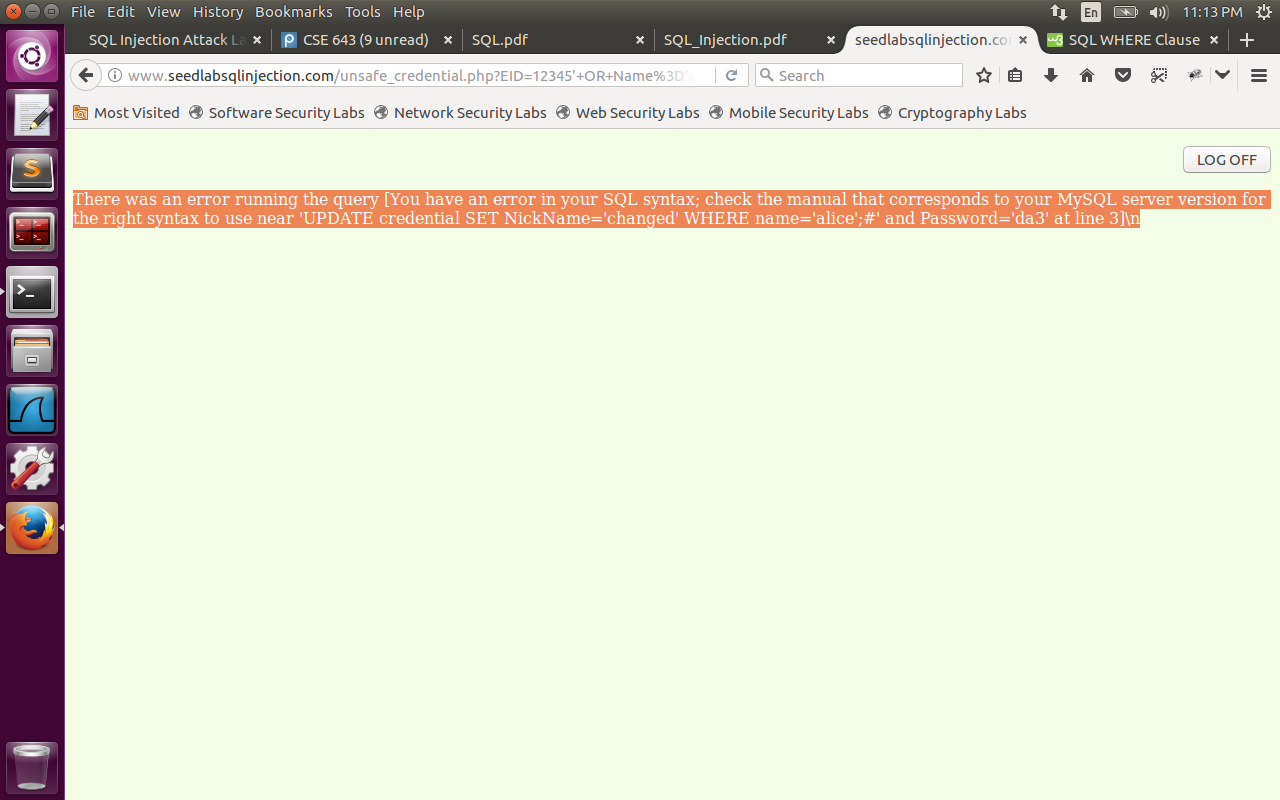


**Task 2.3: Append a new SQL statement.**

Now I have added an UPDATE SQL statement after the SELECT statement. Bu the attack was not successful. The command I have entered in the login credential is as follows:

12345` OR name=`admin`; UPDATE credential SET NickName=`changed` WHERE Name=`alice`;#

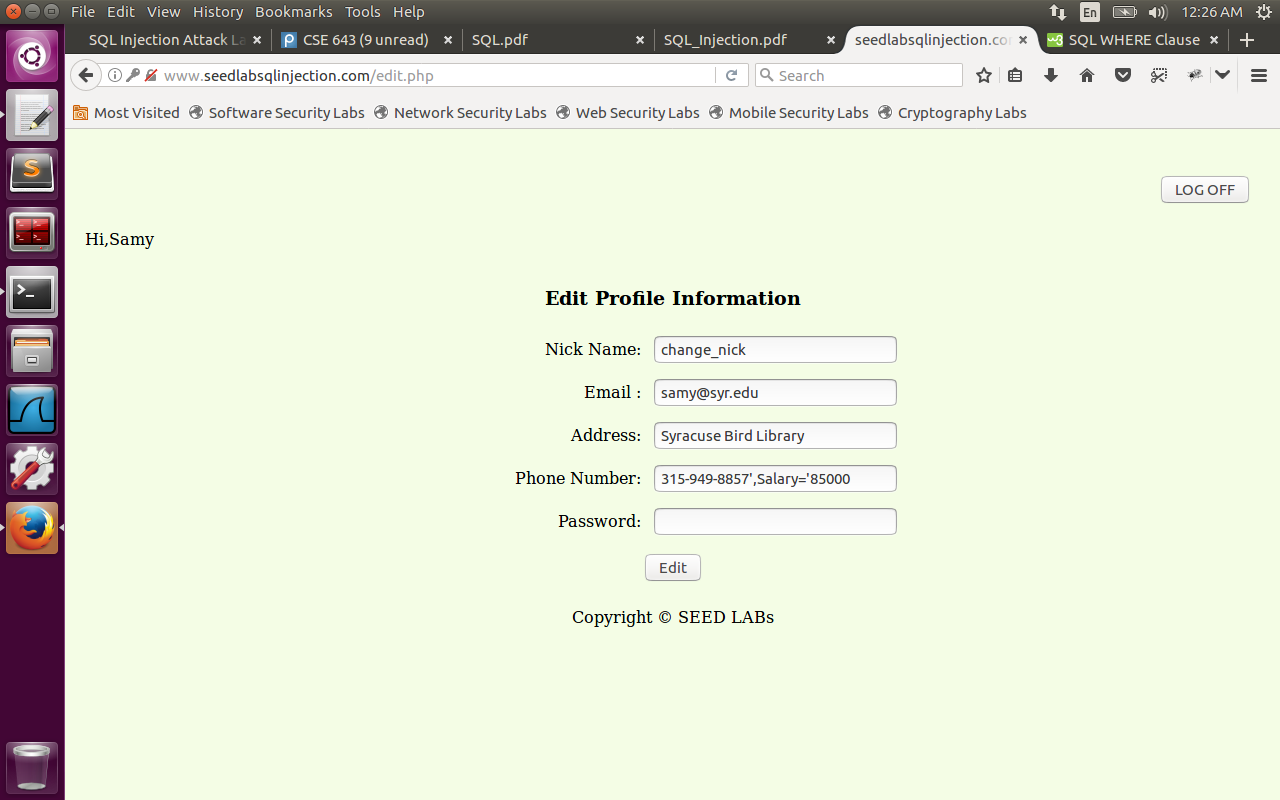
The login was successful, but the error was thrown on the second sql statement as syntax error as below.

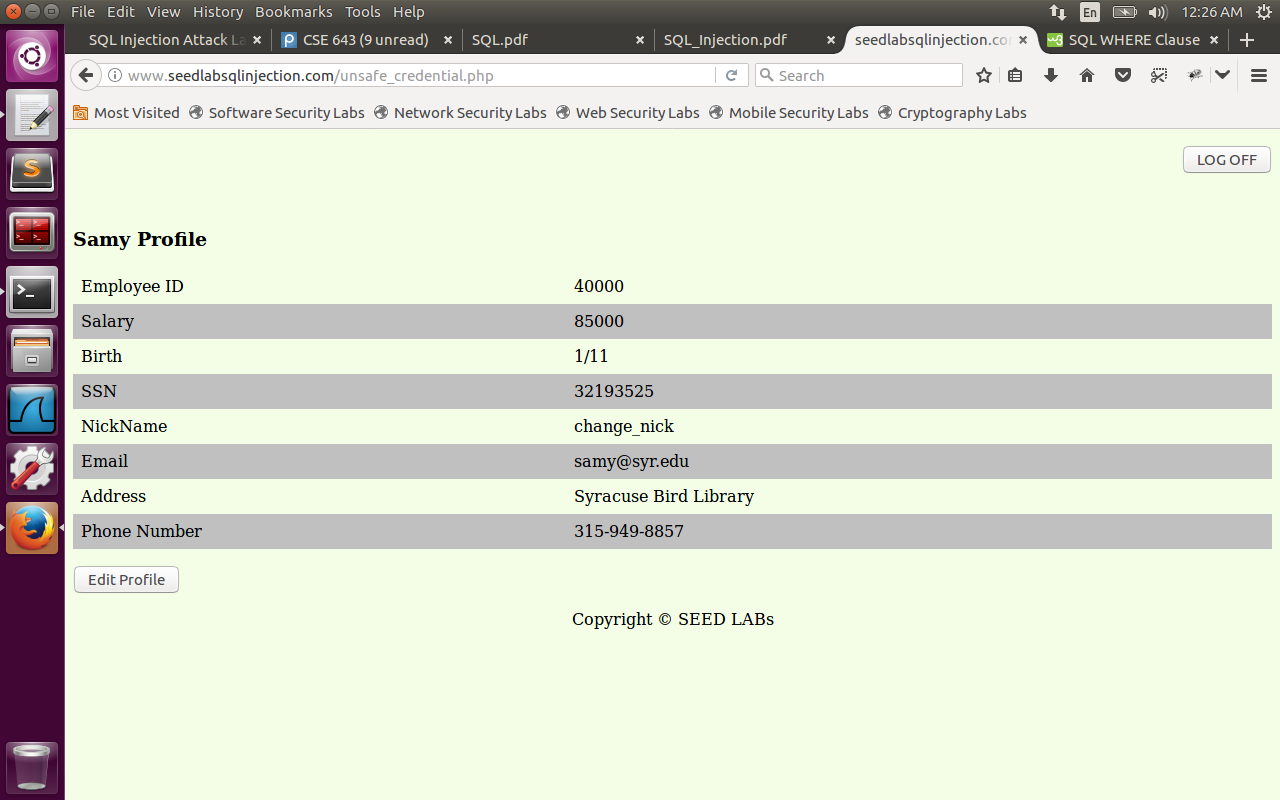


I guess MYSQL prevents from executing more than 1 command sequentially through a single query to prevent such attacks.( $conn->query($sql) prevents executing 2 statements to prevent the injection attack. )

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

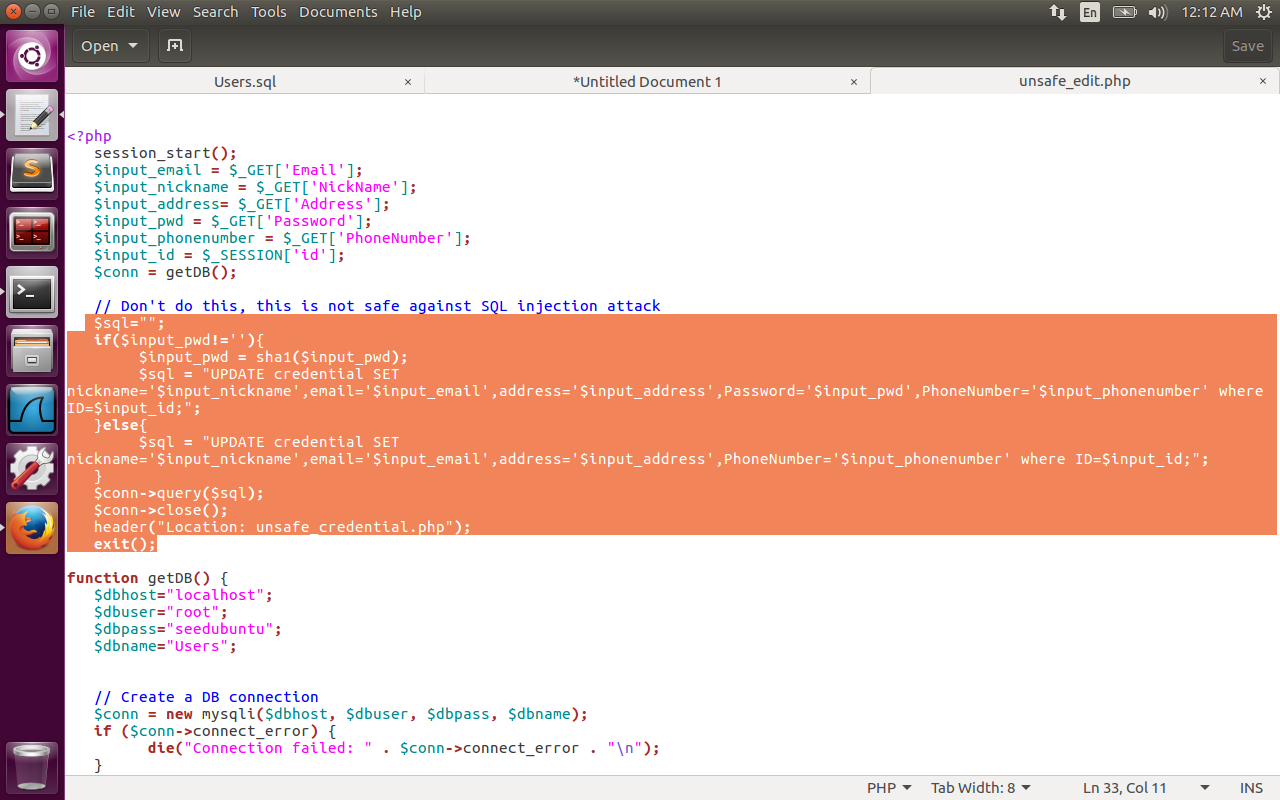
**Task 3.1: SQL Injection Attack on UPDATE Statement — modify salary**

****



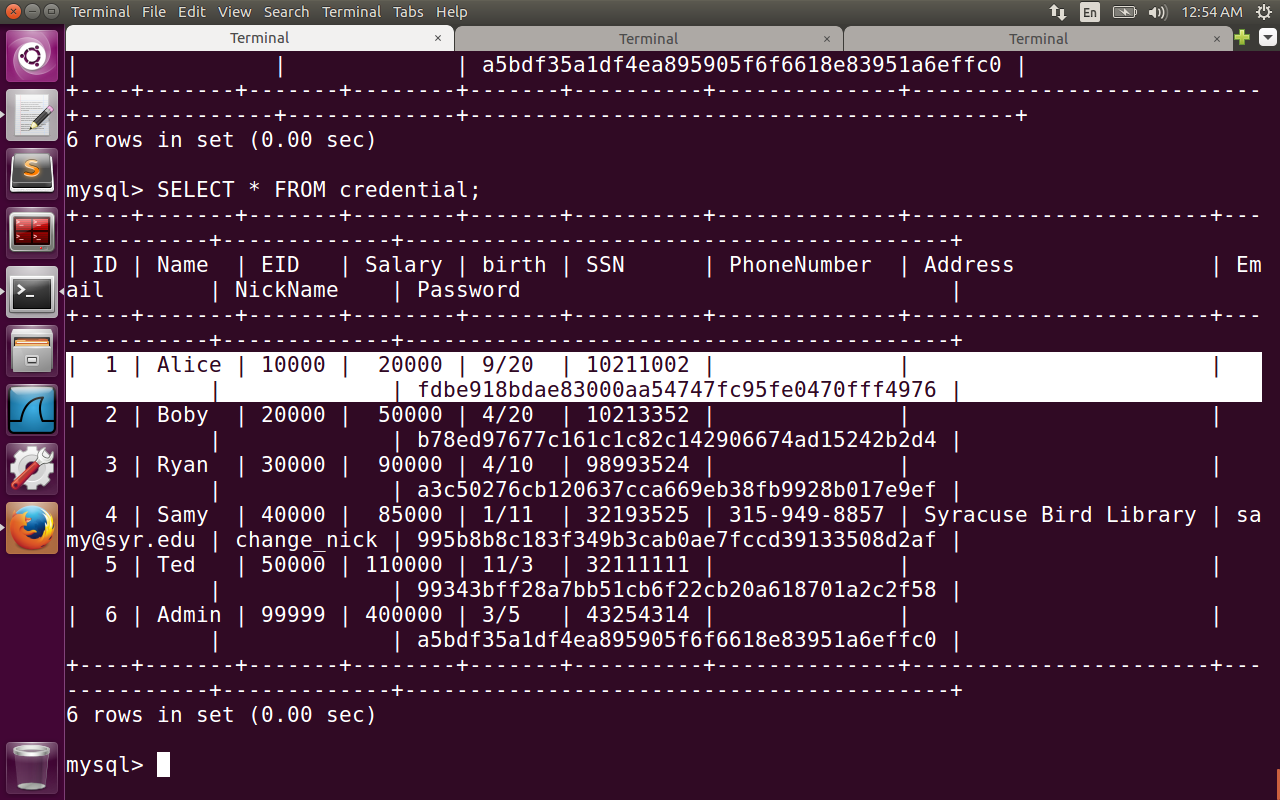
The update is done after looking into unsafe\_edit.php script as follows:

After filling the phone number we add ` to end the phone number parameter and add `,Salary=’85000 so that salary is added to the database . It doesnot matter even if we don’t enter password here because the script is taking care of both password entered and not entered cases.



**Task 3.2: SQL Injection Attack on UPDATE Statement — modify other people’ password.**

Alice account before getting hacked by Samy



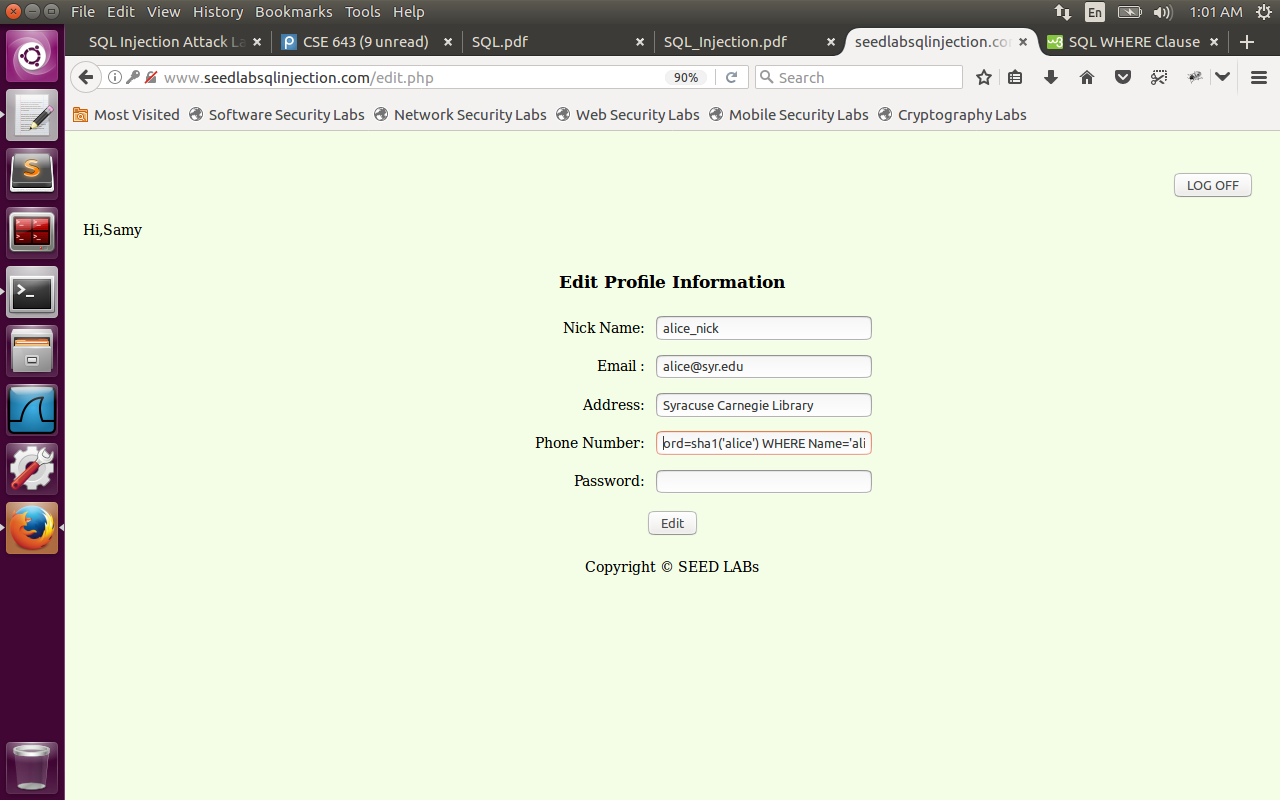
Now samy creates the following edit profile:

Nick Name: alice\_nick

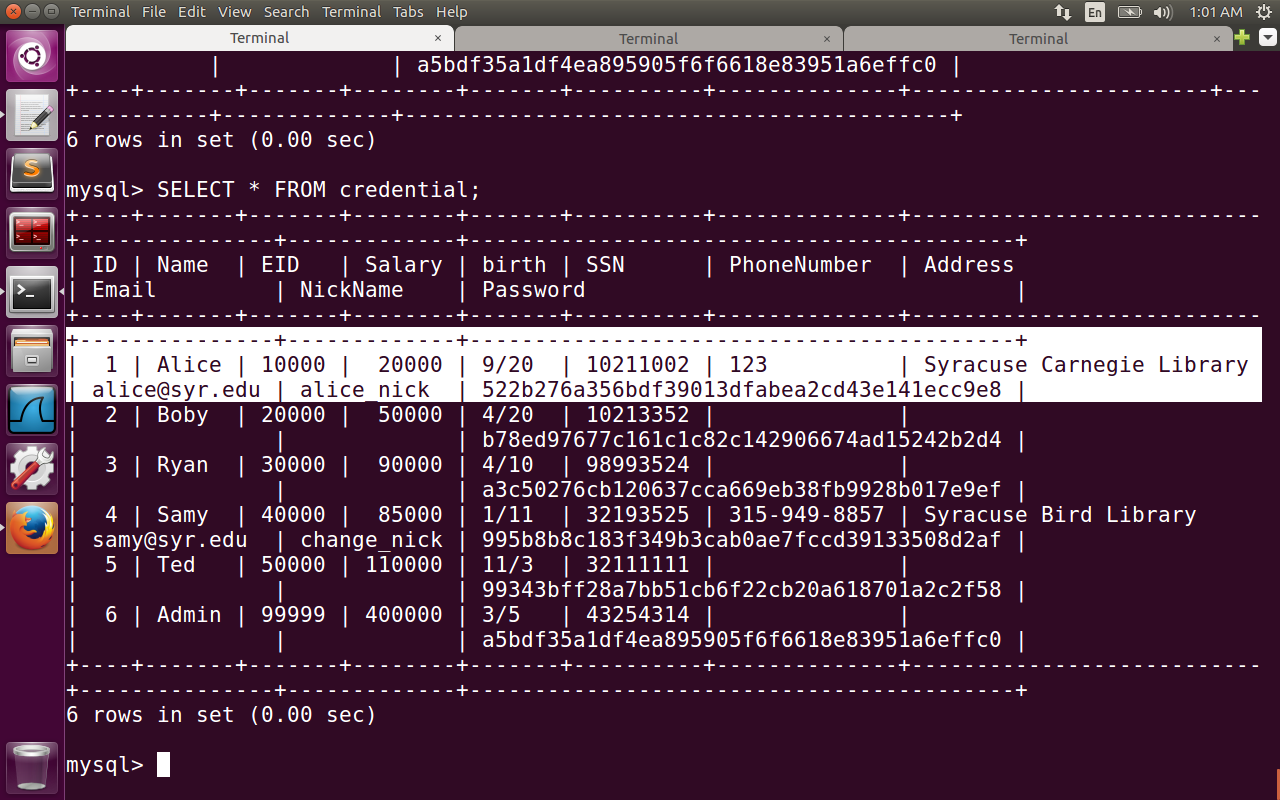
Email: [alice@syr.edu](mailto:alice@syr.edu)

Address: Syracuse Carnegie Library

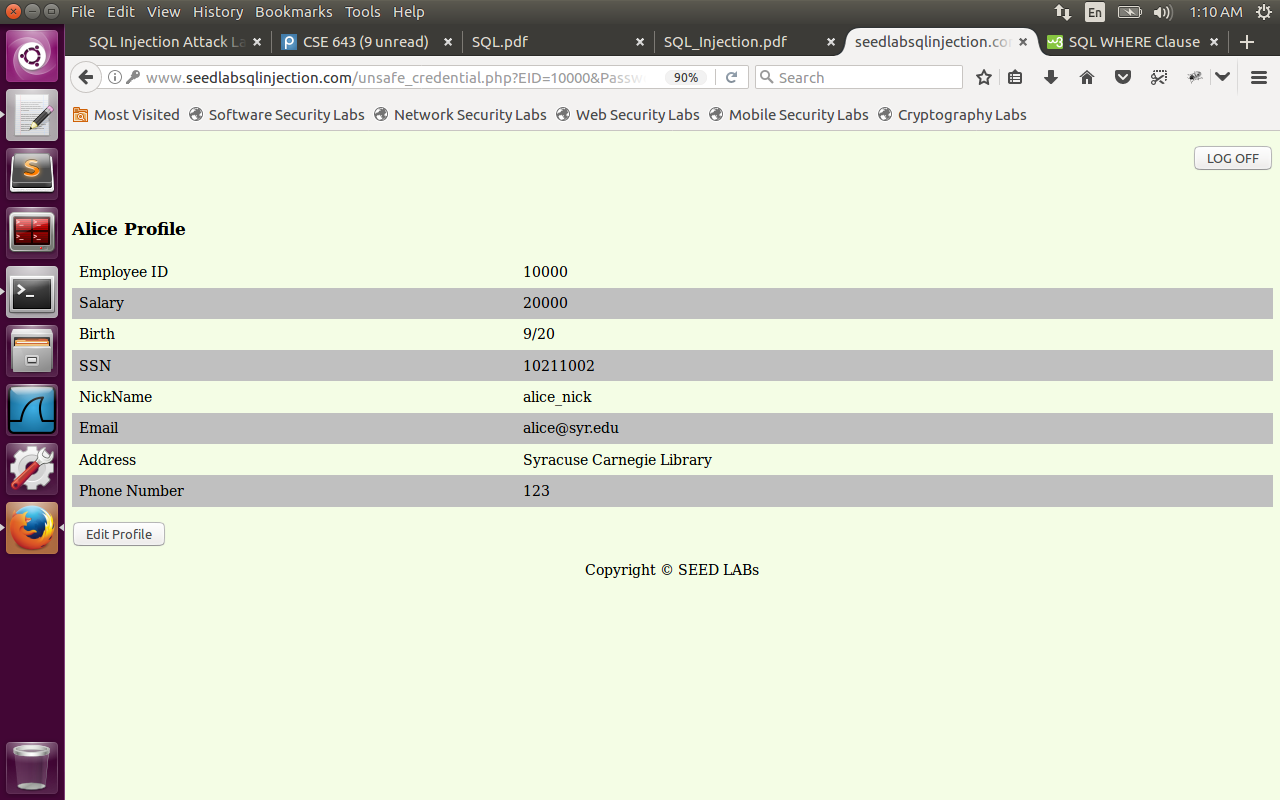
Phone Number: 123`,Password=sha1(‘alice’) WHERE Name=’alice’#



Now observe the alice fields changed as per designed by samy attack.



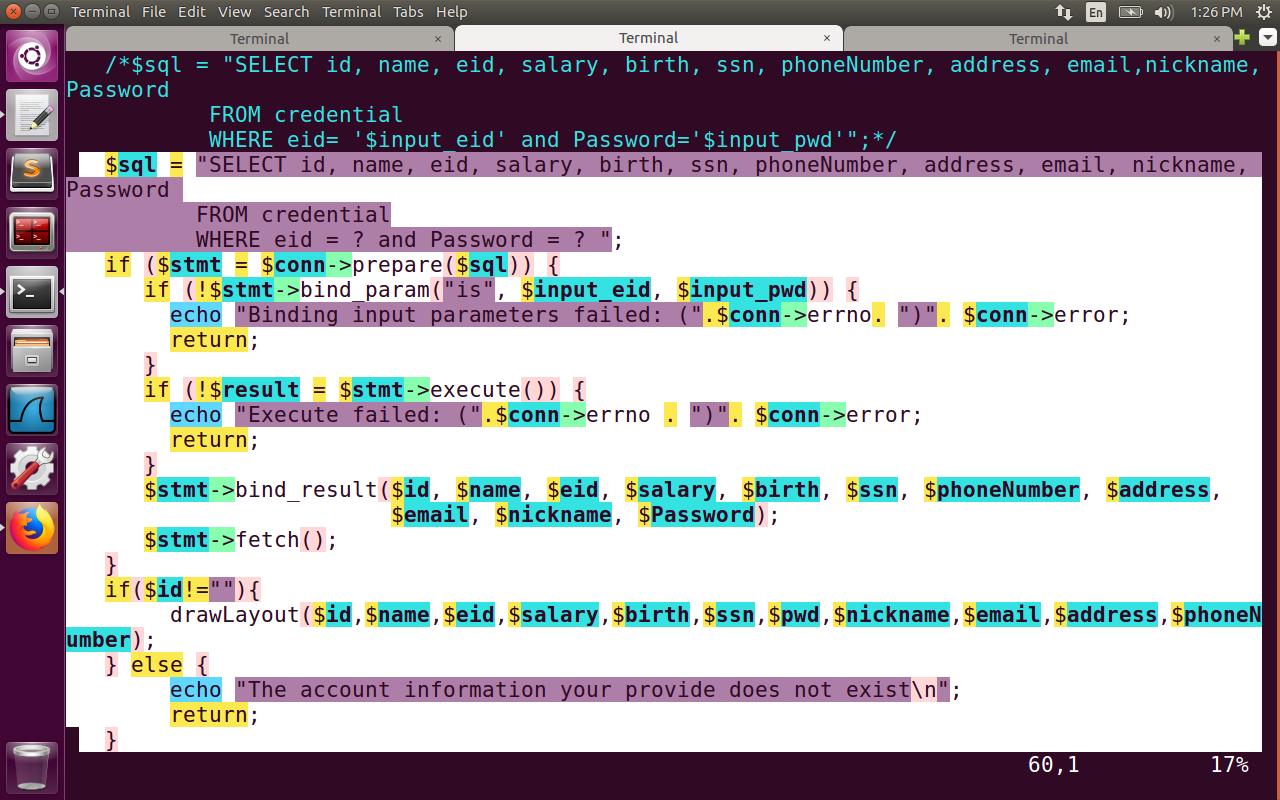
Here the string 522b276a356bdf39013dfabea2cd43e141ecc9e8 is the **sha1** value of ‘alice’ string. Password in the database is stored after hashing the string. The samy now logins with the changed password and he is successful.



**3.4 Task 4: Countermeasure — Prepared Statement**

When data is separated from the query using the prepare statement as follows:

The data is used by the Database at the execution time. The other phases like parsing, optimizing and compiling the query are done before the execution phase. Hence data is binded to the query only at the execution phase.



The attack is not successful and the entries are not displayed for the attacker trying to login as admin and by passing authentication as in the previous task.

