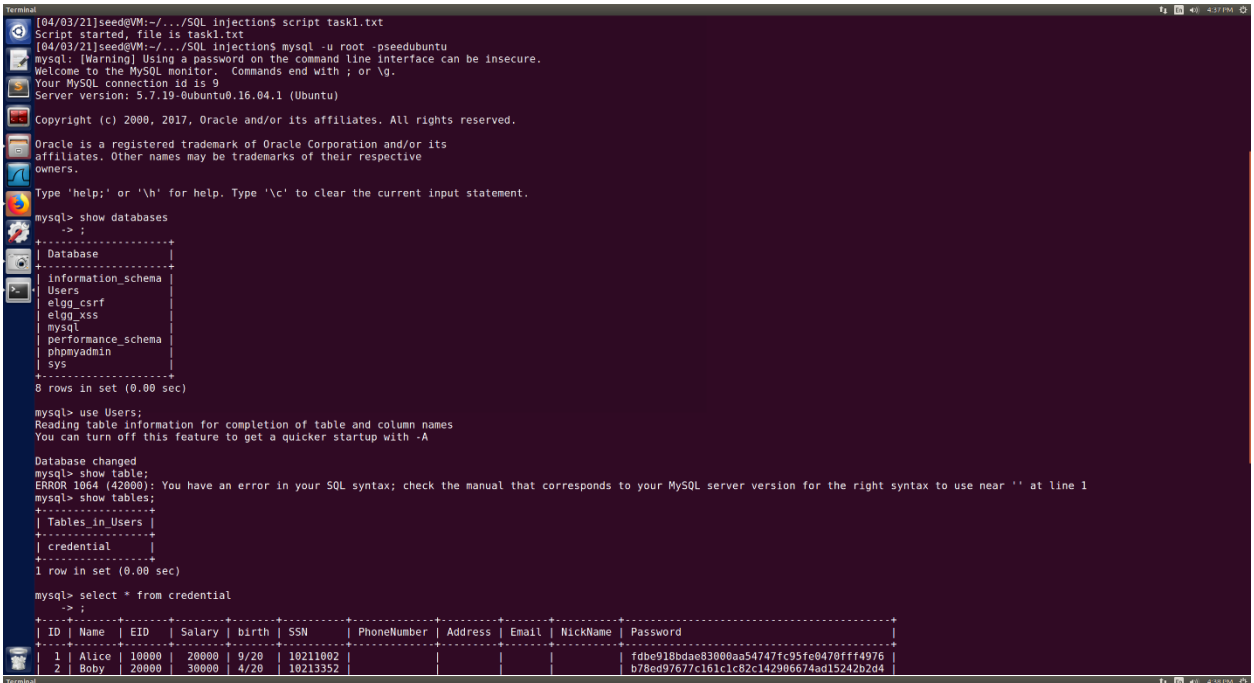
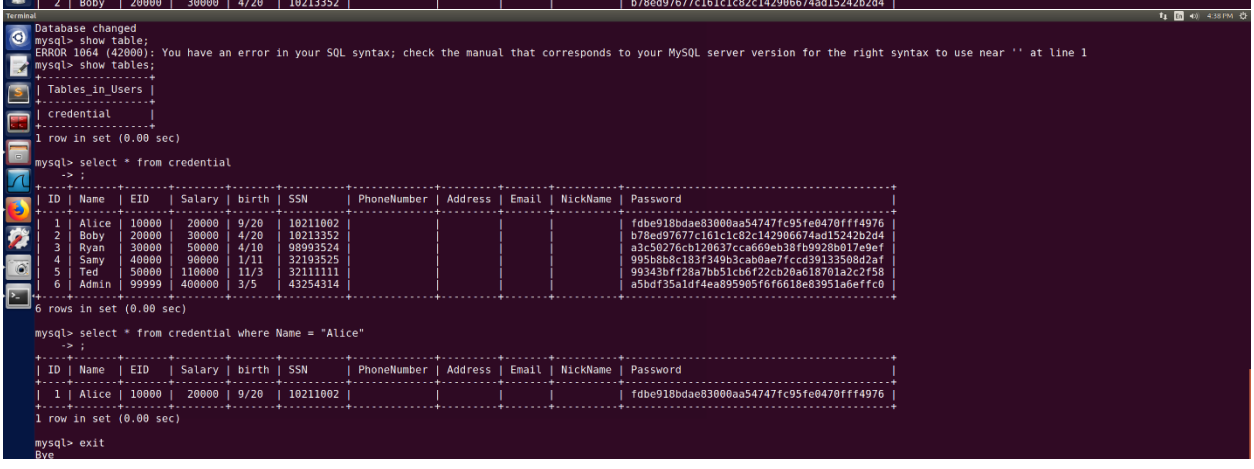
Task1:

1. Inside local mysql on the seed lab vm there are multiple database
2. We are using Users database for this lab so we load the database using command:
    a. Use Users;
3. To show what table are there inside Users database, use command:
    a. Show tables;

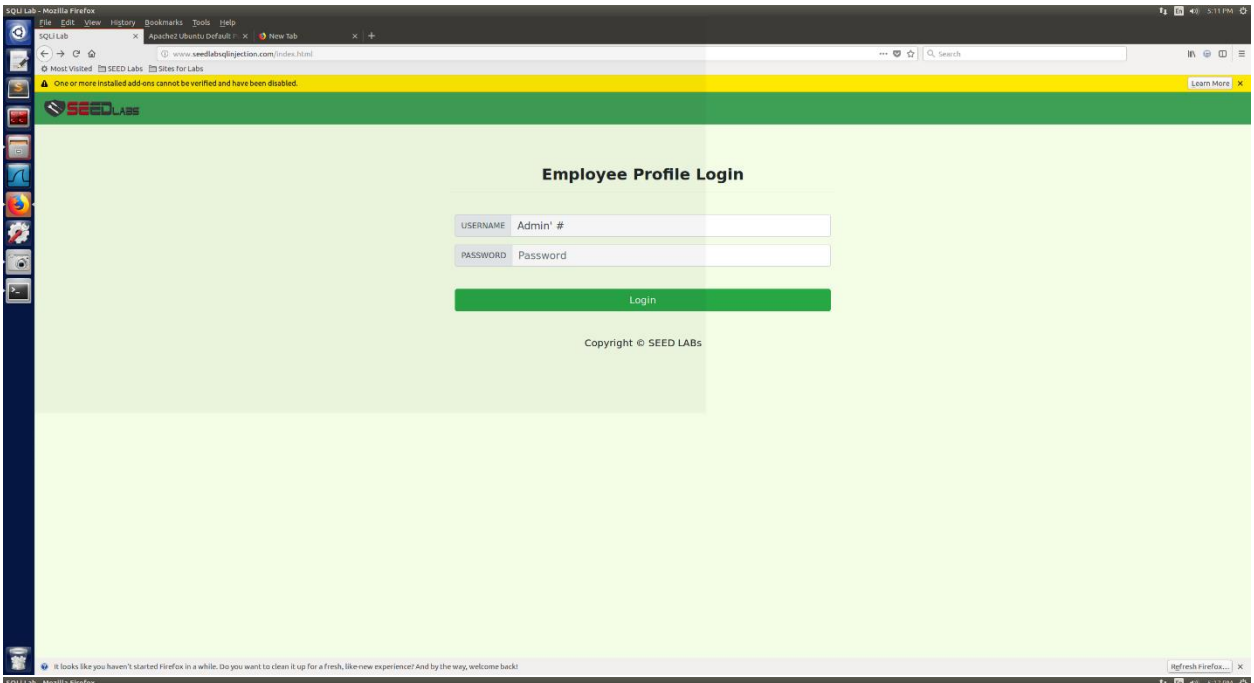
4. To print all the profile information of the employee Alice, use query:
    a. SELECT * FROM Users Where Name = ‘Alice’;

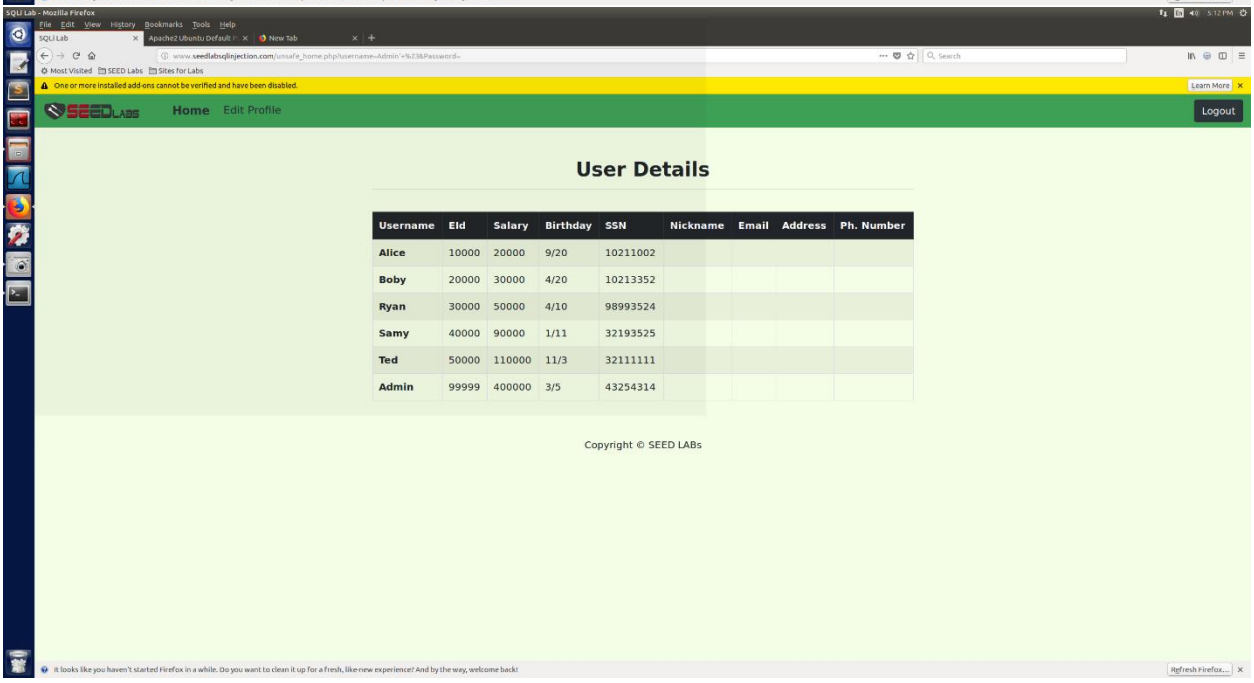




Task2.1:

1. We know to perform authentication, the server will have put our username and password input into a query looks something like this:
    a. Select something from sometable where name=’ OurNameInput ‘and password = ‘ OurPasswordInput ‘
    b. We know name = Admin but we don’t know the password for Admin so we need to put a ‘#’ sign right before password in the query so that the MySQL will treat the part behind ‘#’ as a comment and will not execute it
    c. So our query is looking like this name = ‘Admin #’ and password = ‘ ‘
    d. Notice that the query treat ‘#’ sign as part of the name input but we want the ‘#’ sign to be outside and not be evaluated as a input string
    e. So we put another single quotation mark at the end of Admin
    f. Our input now look like this name = ‘Admin’ # ‘and password = ‘ ‘
    g. The server will now execute this query: select something from Users where name = ‘Admin’
    h. And there you go, we got inside



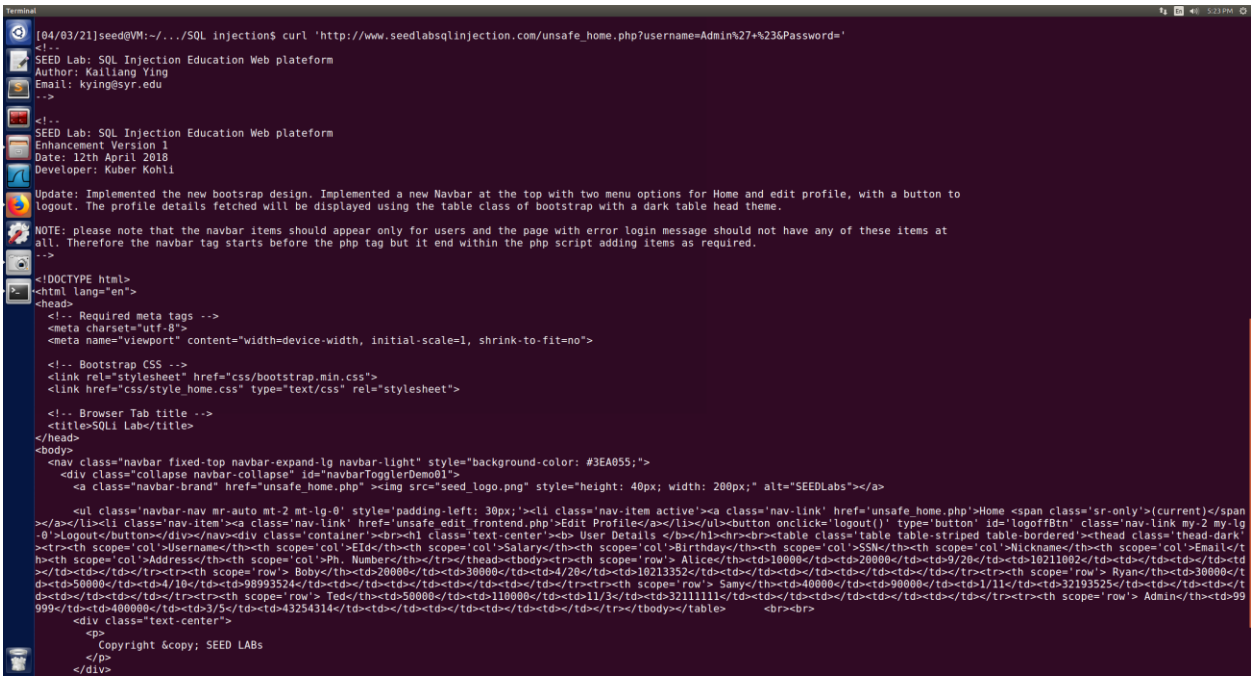


Task2.2

1. To in order to use curl we need to encode the URL containing our parameter properly, otherwise. It changes the meaning of the request. For example, if we have a space within the URL string, we need to replace it with a '+' sign.
2. Apply the idea into building our Url string for the curl request, we have:

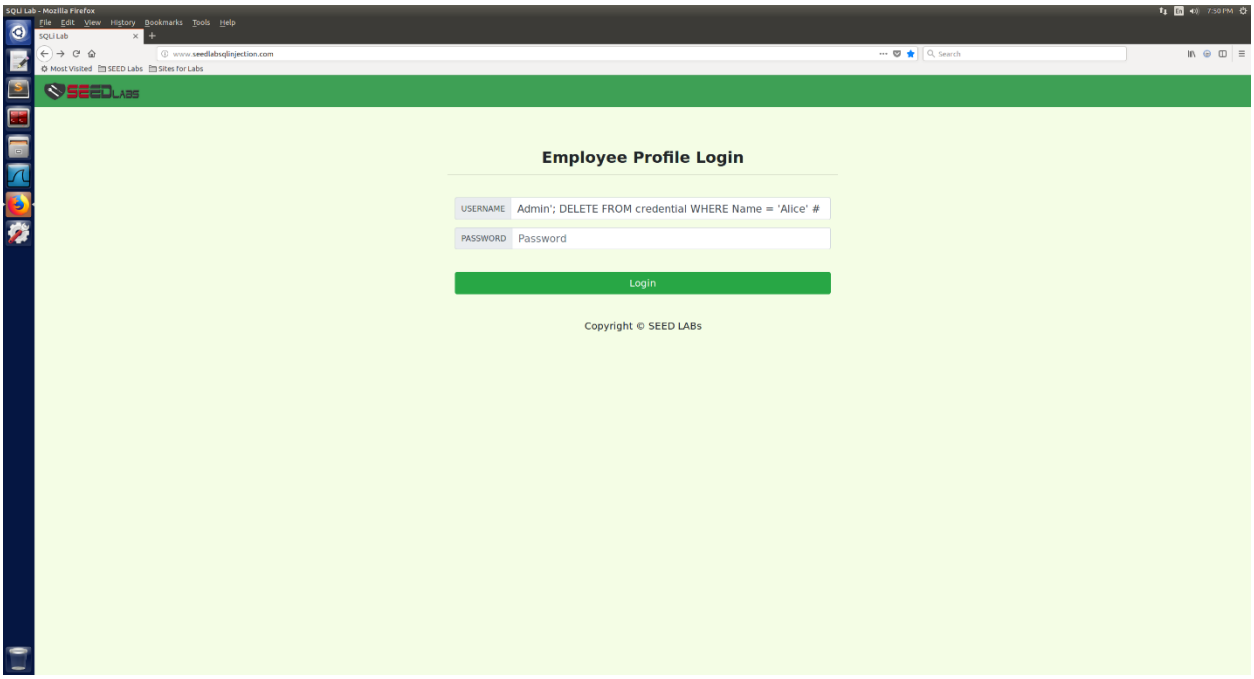
   curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27+%23&Password='

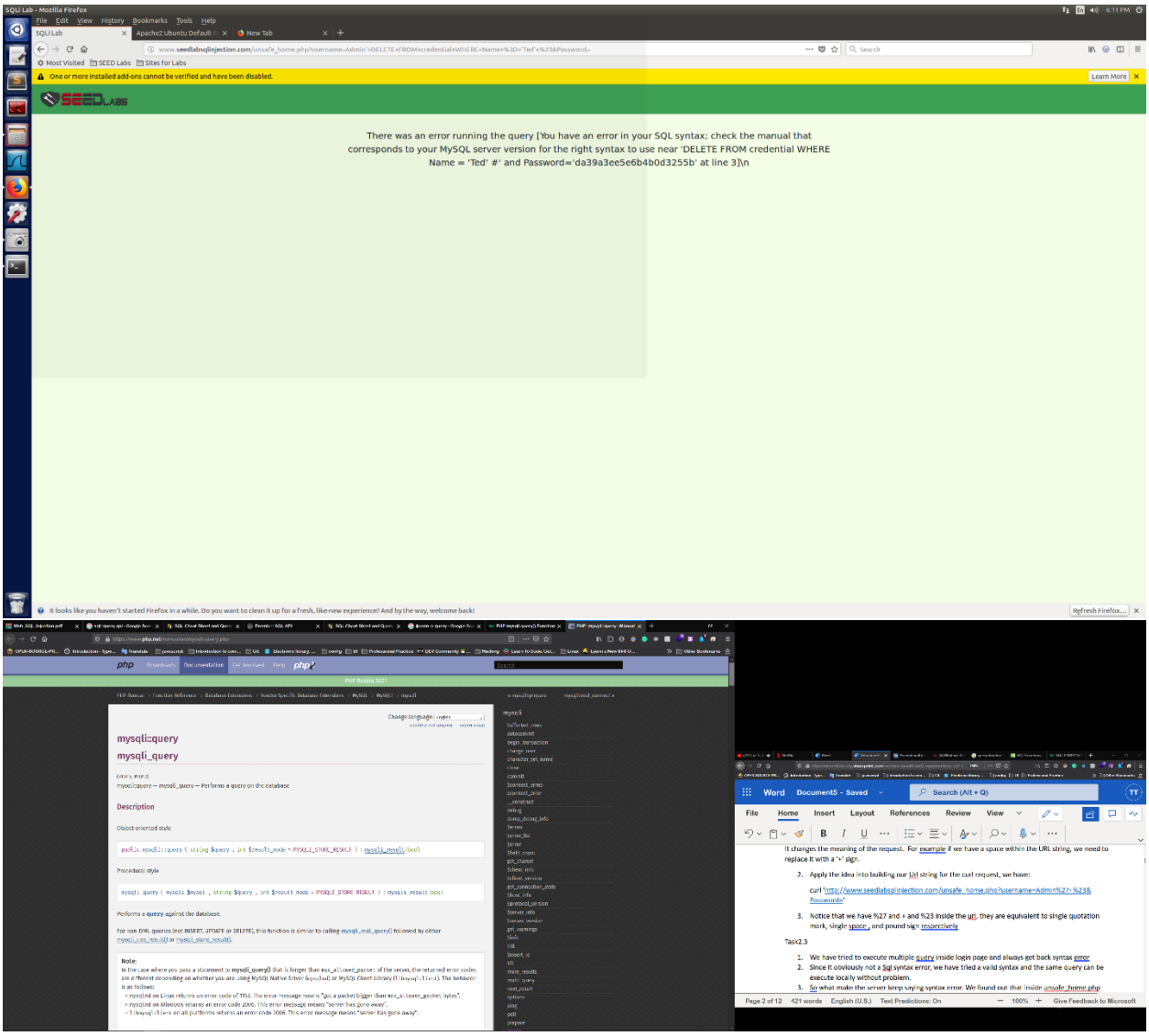
3. Notice that we have %27 and + and %23 inside the URL, they are equivalent to single quotation mark, single space, and pound sign respectively.



Task2.3

1. We have tried to execute multiple queries inside login page and always get back syntax error.
2. Since it obviously not a Sql syntax error, we have tried a valid syntax and the same query can be execute locally without problem.
3. So, what make the server keep saying syntax error. We found out that inside unsafe_home.php there is a Api call conn -> query() which have been documented in php.net that It can only perform 1 query on the database at a time compared to multi_query() which can execute multiple query on the database. Hence, the effort of execute multiple queries will always fail in this case.


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com
Most Visited SEED Labs Sites for Labs
SEEDLABS
Employee Profile Login
USERNAME
Admin'; DELETE FROM credential WHERE Name = 'Alice' #
PASSWORD
Password
Login
Copyright © SEED LABs

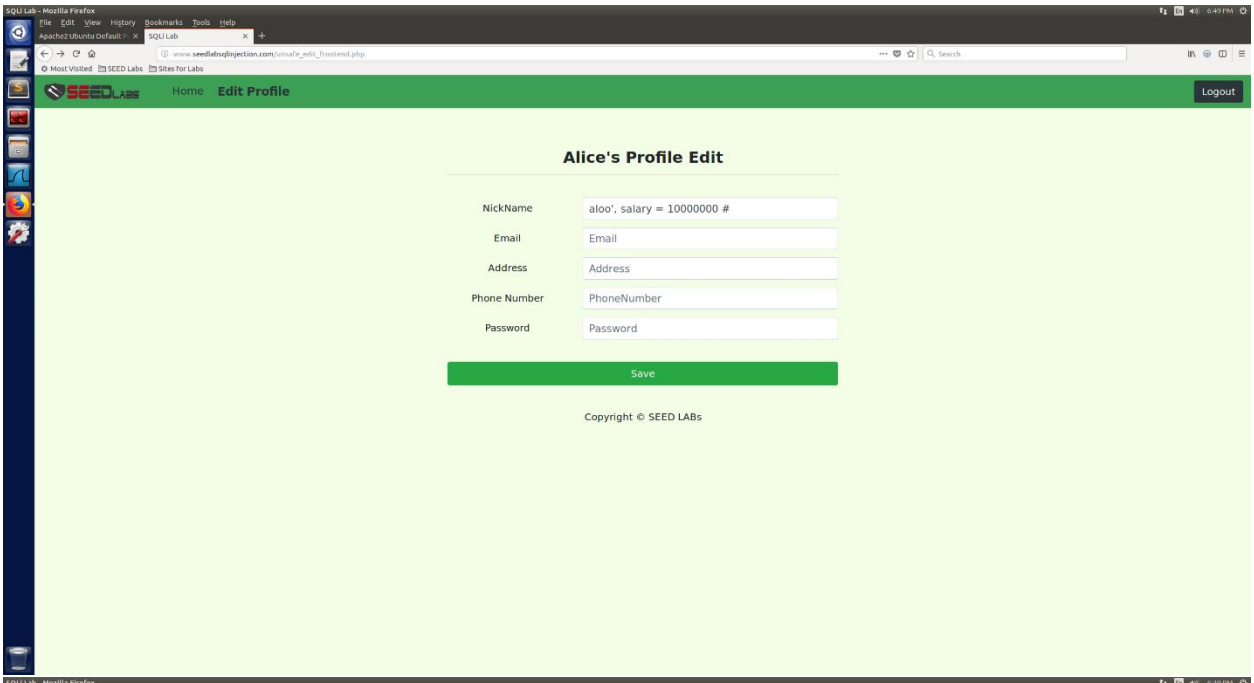


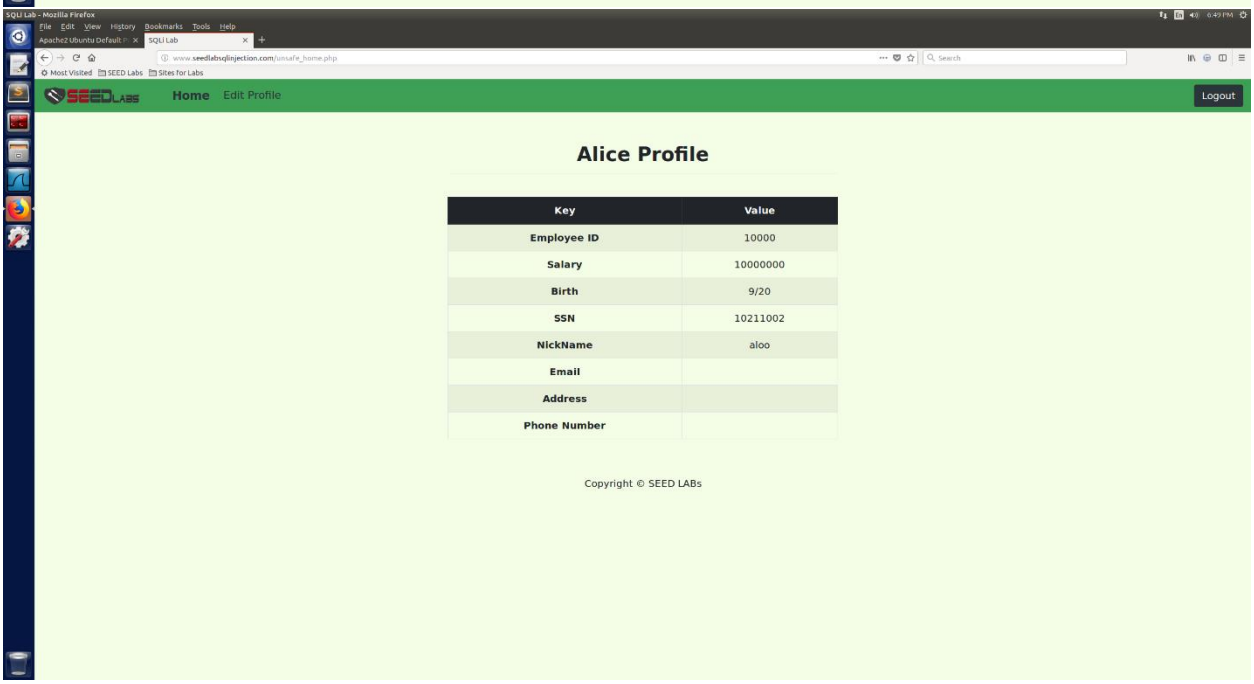
Task3.1

1. After we tried to modify the Nickname of Alice (one of the users) without using sql injection, we able to figure out how the query in the backend look like, that is:
    a. Update sometable
    b. SET Nickname = ‘our input‘
    c. WHERE Name = Alice
2. Notice that our input is between quotation marks. That why we have failed to try the inject this: ali’, salary = 100000. Thus the query became,
    a. Update Sometable SET Nickname = ‘ali’, salary=100000’ Where Name = Alice

    Which is a illegal sql syntax

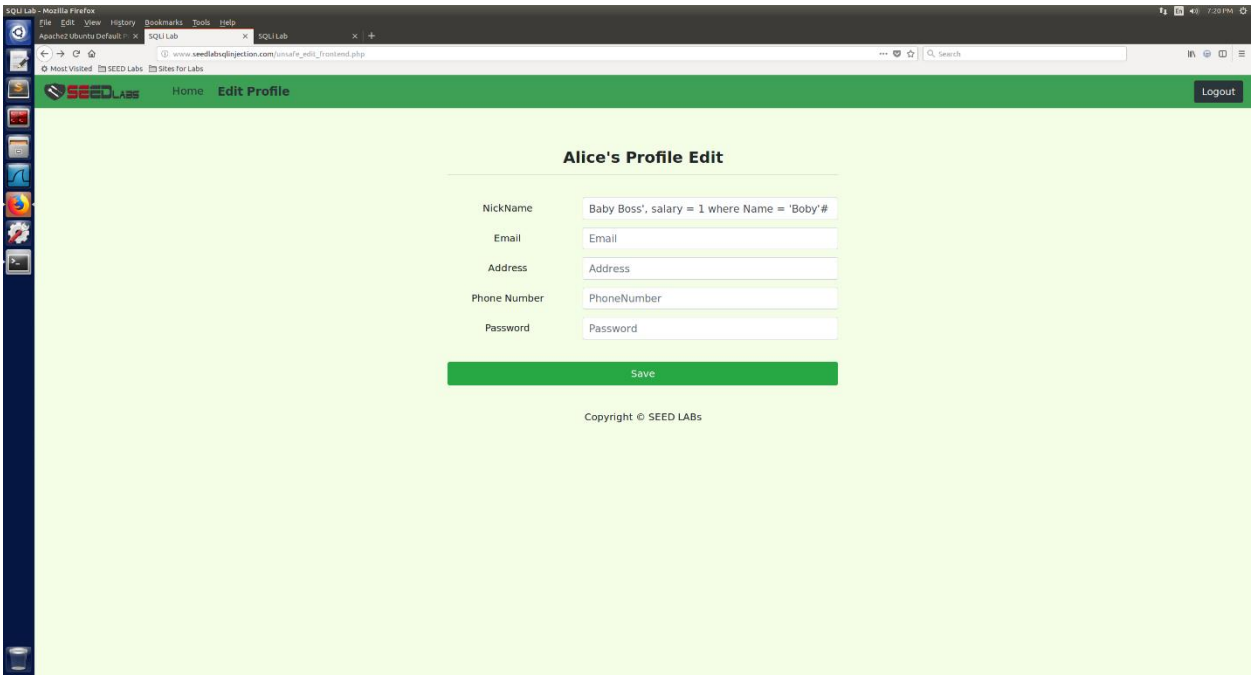
3. To remove the redundant quotation mark at the end of our input, we have put a ‘#’ mark to to commend out the rest of the query. Which mean we must regenerate where clause ourselves
4. The successful query to inject will look like this : alo, salary = 1000000 where Name = ‘Alice’ #



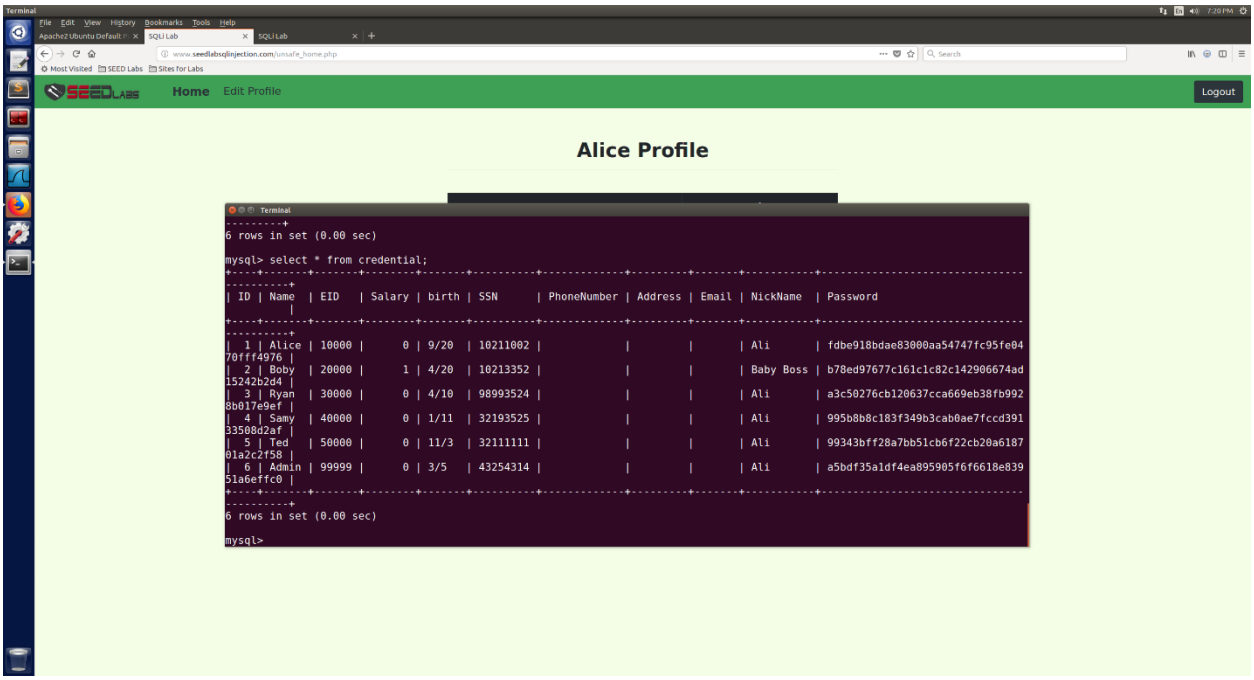


Task3.2

1. Follow the same procedure of 3.1, but in addition modify the Name inside where clause and we be able change an information of another people
2. In this case, we use the same query in 3.1 instead change the salary to 1 and Boby as Name.



3. By checked the local database, we can verify that the salary of Boby had changed to 1, as well as the his nick name had changed to ”Baby Boss”

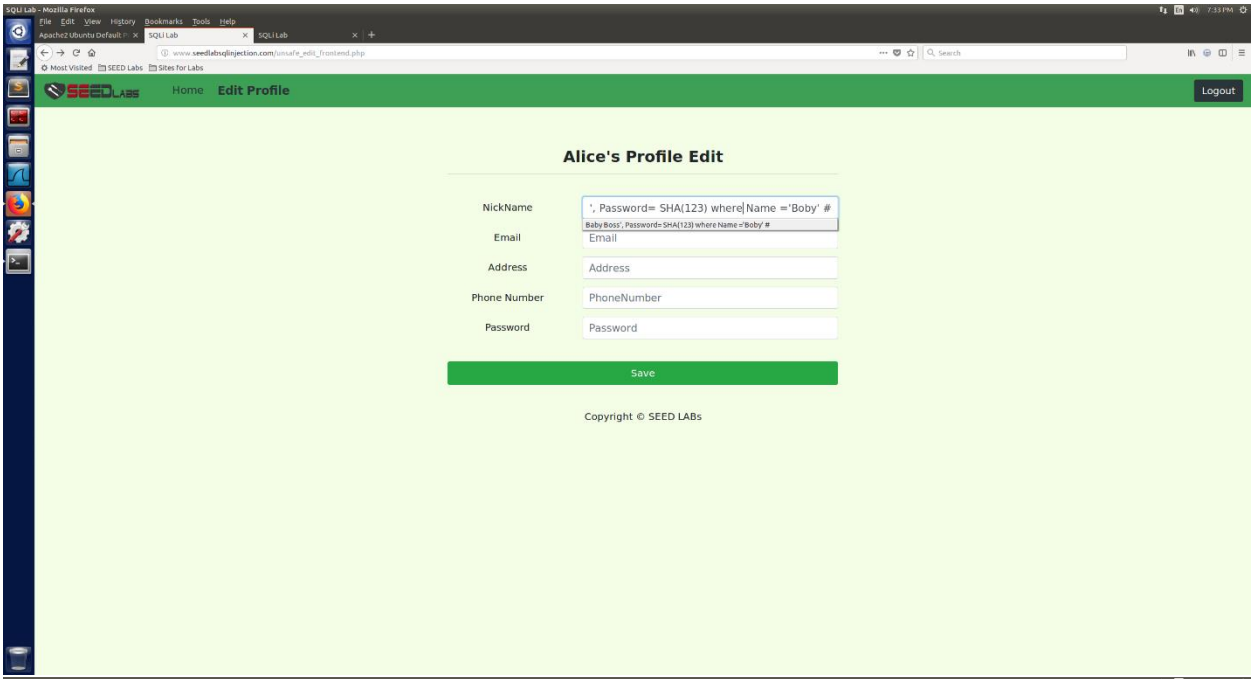


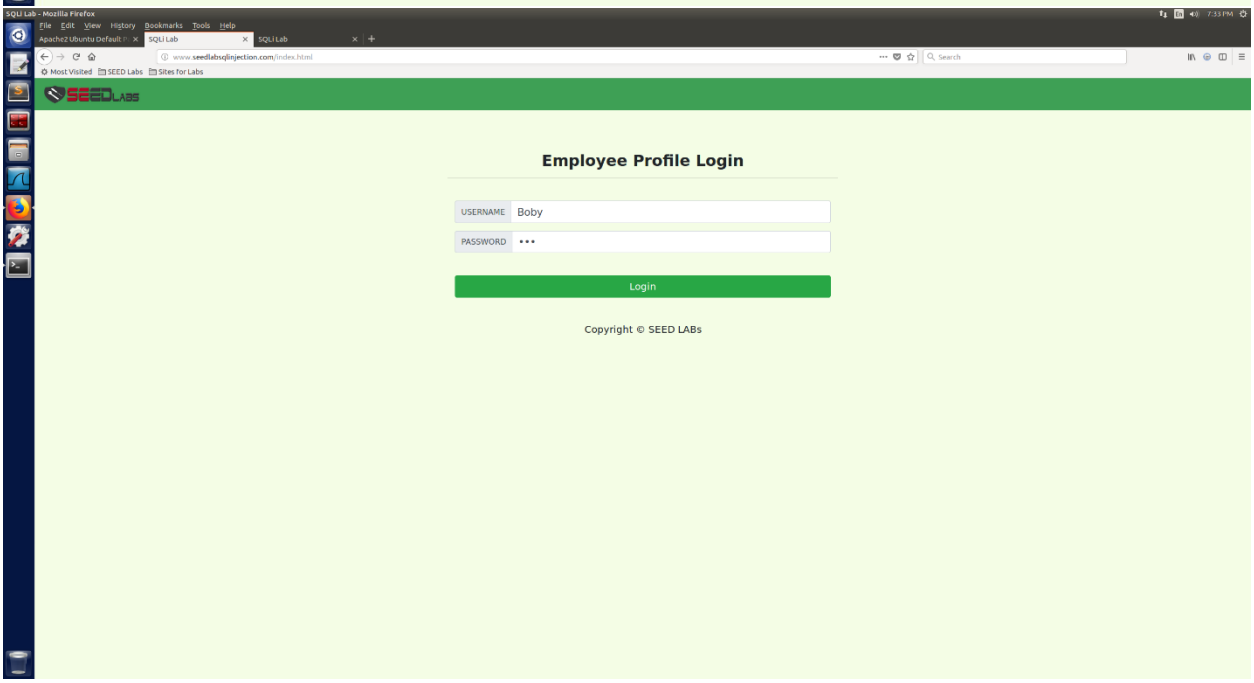
Task3.3

1. Follow the same procedure of 3.1, we imagined the query we were going to use would look like this:
    a. Hacker', Password = somepasswork #
2. This is not work since the database stores the hash value of password instead of plaintext password, that mean wherever we login with a password, the server takes the password and

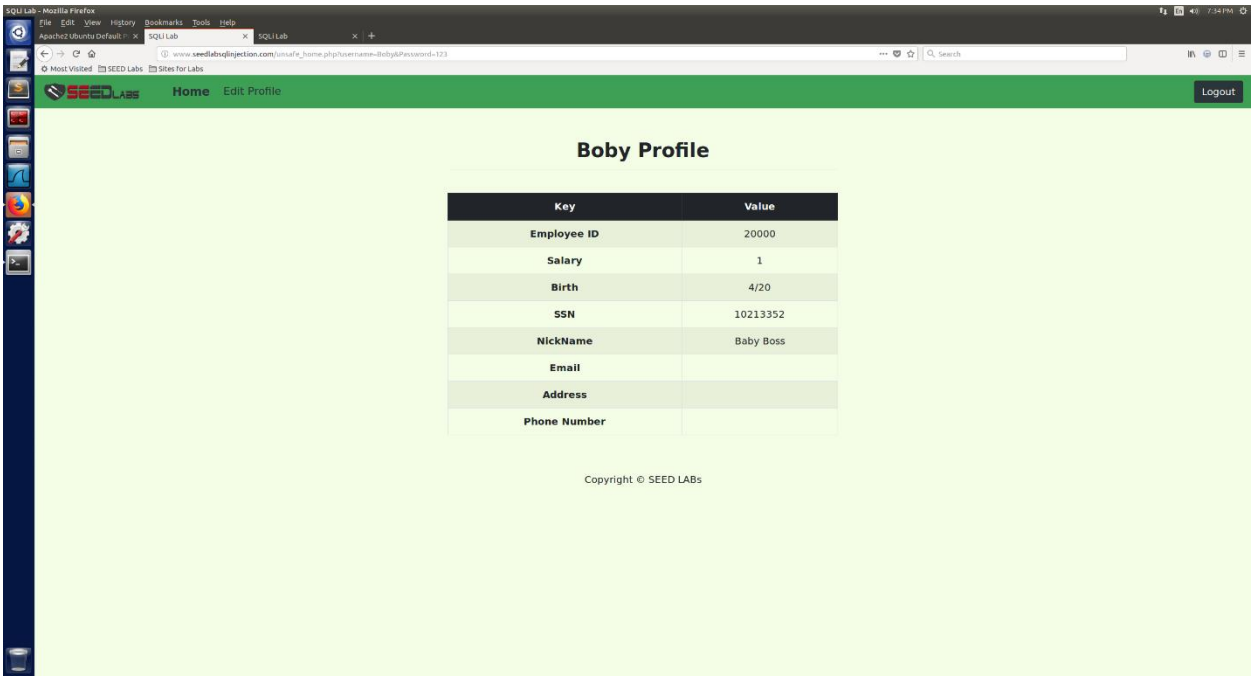
hash it then compare it with the password inside the database. If we use a plaintext password inside the query we wanted to inject, we can't login with the same password.

3. So, we need a way to hash the password before putting it in the query that we want to inject.
    a. By using built in function SHA, we can achieve that.
4. We modified Boby password to 123





By looking at the URL string we can verify that we have successfully login Boby accout using password 123


SQLi Lab – Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Boby&Password=123
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout
Boby Profile
Key
Value
Employee ID
20000
Salary
1
Birth
4/20
SSN
10213352
NickName
Baby Boss
Email
Address
Phone Number
Copyright © SEED LABs