Information Security: SQL Injection Attack

Name: Chandan N Bhat

PES1201701593

Section H

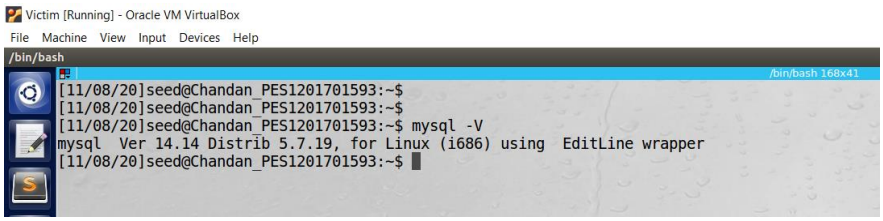
SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. This vulnerability is usually observed when user inputs are not correctly checked in web applications before being sent to backend servers.

Usually SQL queries are constructed using the user's input and when not constructed carefully SQL Injection vulnerability can occur. SQL Injection is a very common attack on web applications.

We are given a web application with address http://www.seedlabsqlinjection.com/ that is vulnerable to SQL Injection attack.

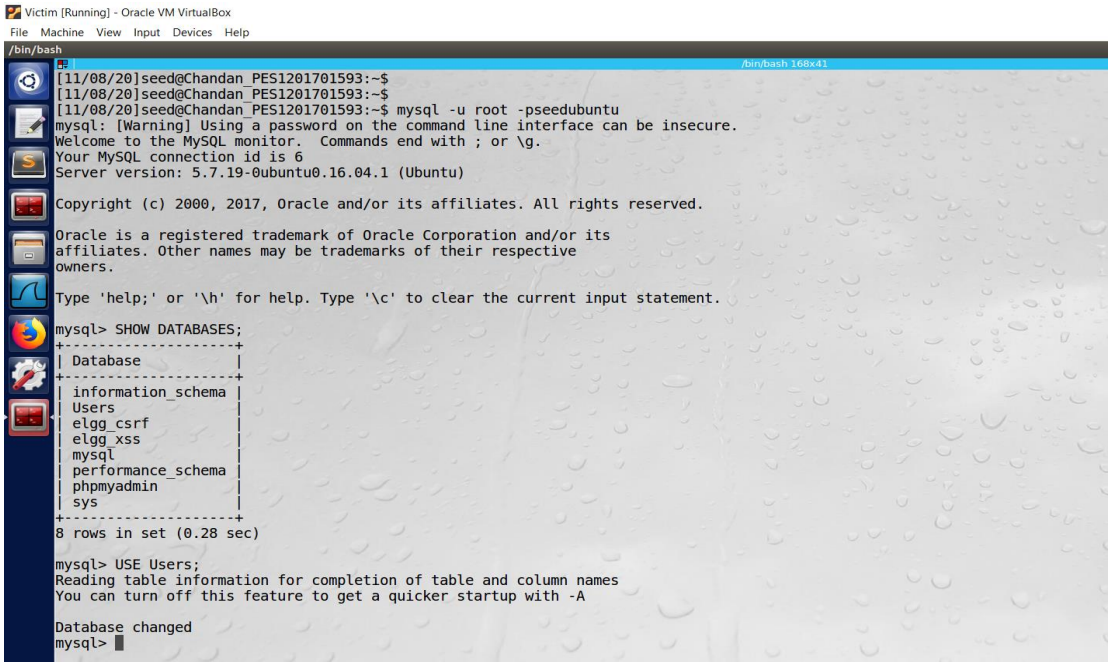
Task 1: Getting familiar with SQL Statements

First, we play around with the given database to get familiar with SQL statements. The 'Users' database contains a table called credential that stores personal information. We will be using MySQL which is already setup on the seed VM, with username 'root' and password 'seedubuntu'.



a. Login to MySQL console

We login to the MySQL console using the command mysql -u root -pseedubuntu



We use the existing database ‘Users’ and display all the tables in the database. Using the SELECT statement we display all the content of the credential table.

```
Victim [Running] - Oracle VM VirtualBox
File  Machine  View  Input  Devices  Help
/bin/bash
/bin/bash 168x41
mysql>
mysql>
mysql> USE Users;
Database changed
mysql>
mysql> SHOW TABLES;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  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 (0.11 sec)

mysql>
```

In the above table we replace the entry ‘Alice’ with ‘Chandan’ and ‘Boby’ to ‘Dhruva’ using the query:

UPDATE credential SET Name=’Chandan’ WHERE Name=’Alice’;

```
Victim [Running] - Oracle VM VirtualBox
File  Machine  View  Input  Devices  Help
/bin/bash
/bin/bash 168x41
mysql>
mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  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> UPDATE credential SET Name='Chandan' WHERE Name='Alice';
Query OK, 1 row affected (0.19 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE credential SET Name='Dhruva' WHERE Name='Boby';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM credential;
+----+---------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name    | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+---------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Chandan | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Dhruva  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan    | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  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>
```

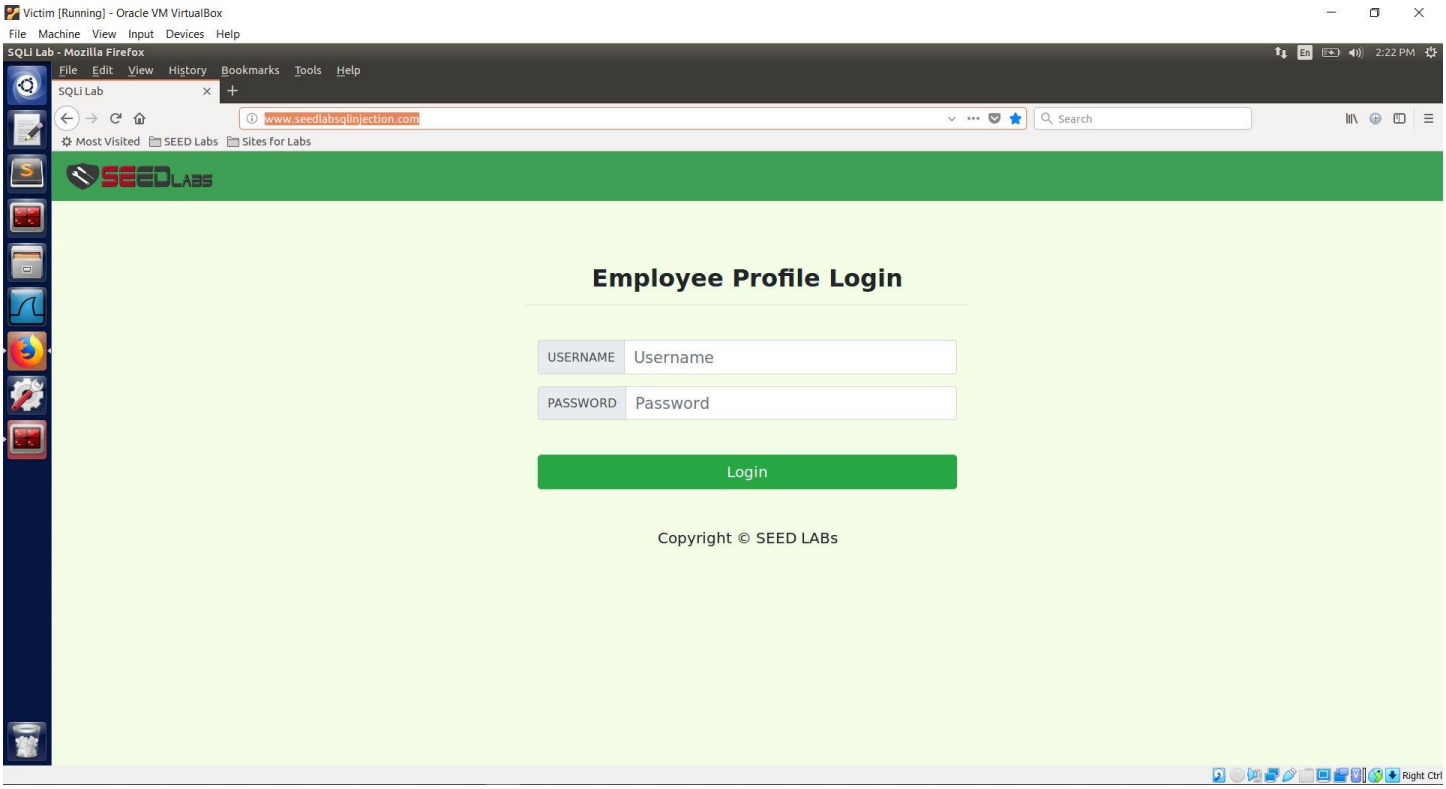
We observe that we updated the credential table with ‘Chandan’ and ‘Dhruva’ as shown above.

```
mysql> SELECT * FROM credential WHERE Name='Chandan';
+----+---------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name    | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+---------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Chandan | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+---------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Task 2: SQL Injection attack on SELECT Statement

With SQL injection attackers can execute their own malicious SQL statements generally referred as malicious payload. Through the malicious SQL statements, attackers can steal information from the victim database or even worse, they may be able to make changes to the database.

We will use the login page from http://www.seedlabsqlinjection.com/ . The application authenticates the user based on the credentials. Thus, we will try to login without the employee credential.



The php code at the backend for authentication of the credentials is shown below. We observe that SQL SELECT statement is used with input username and hashed password in the WHERE clause.

Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

unsafe_home.php [Read-Only] (/var/www/SQLInjection) - gedit

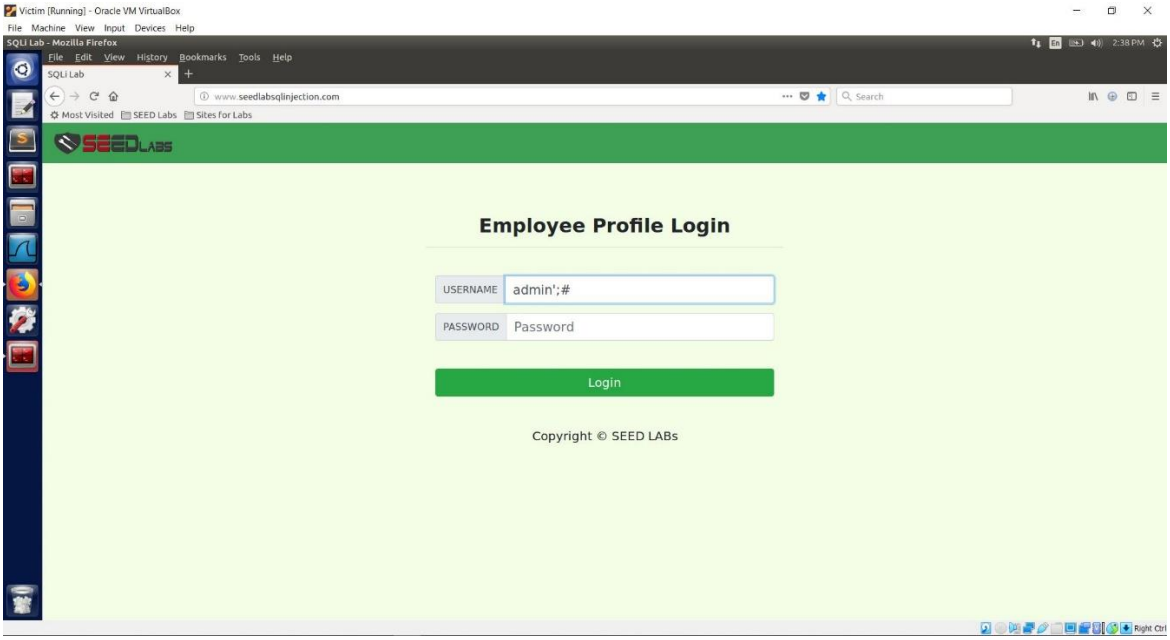
Open

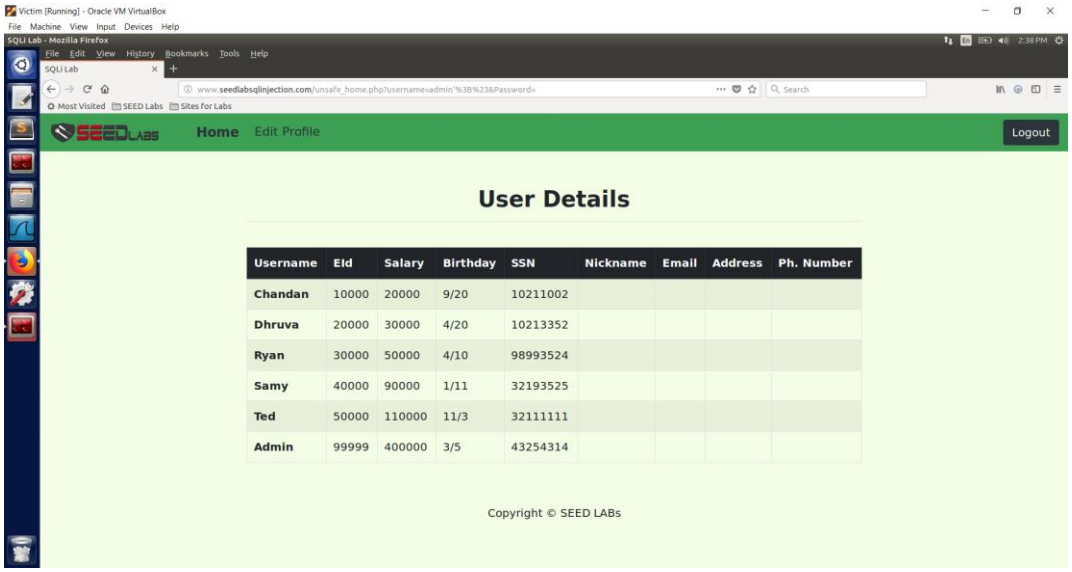
```php
    }

    // Function to create a sql connection.
    function getDB() {
      $dbhost="localhost";
      $dbuser="root";
      $dbpass="seedubuntu";
      $dbname="Users";
      // Create a DB connection
      $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
      if ($conn->connect_error) {
        echo "</div>";
        echo "</nav>";
        echo "<div class='container text-center'>";
        die("Connection failed: " . $conn->connect_error . "\n");
        echo "</div>";
      }
      return $conn;
    }

    // create a connection
    $conn = getDB();
    // Sql query to authenticate the user
    $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
    FROM credential
    WHERE name= '$input_uname' and Password='$hashed_pwd'";
    if (!$result = $conn->query($sql)) {
      echo "</div>";
      echo "</nav>";
      echo "<div class='container text-center'>";
      die('There was an error running the query [' . $conn->error . ']\n');
      echo "</div>";
    }
    /* convert the select return result into array type */
    $return_arr = array();
    while($row = $result->fetch_assoc()){
      array_push($return_arr,$row);
    }

    /* convert the array type to json format and read out*/
    $json_str = json_encode($return_arr);
    $json_a = json_decode($json_str,true);
    $id = $json_a[0]['id'];
    $name = $json_a[0]['name'];
    $eid = $json_a[0]['eid'];
    $salary = $json_a[0]['salary'];
    $birth = $json_a[0]['birth'];
    $ssn = $json_a[0]['ssn'];
    $phoneNumber = $json_a[0]['phoneNumber'];
    $address = $json_a[0]['address'];
    $email = $json_a[0]['email'];
```

a. SQL Injection attack from web page

We will try to login as a administrator whose account name is admin. We observed that the SELECT statement directly places the username without cleaning or filtering the input.

Thus we can pass “admin’;#” in the username field in the form which will comment the comment the remaining SQL statement, and returns true, eventually giving us admin access.



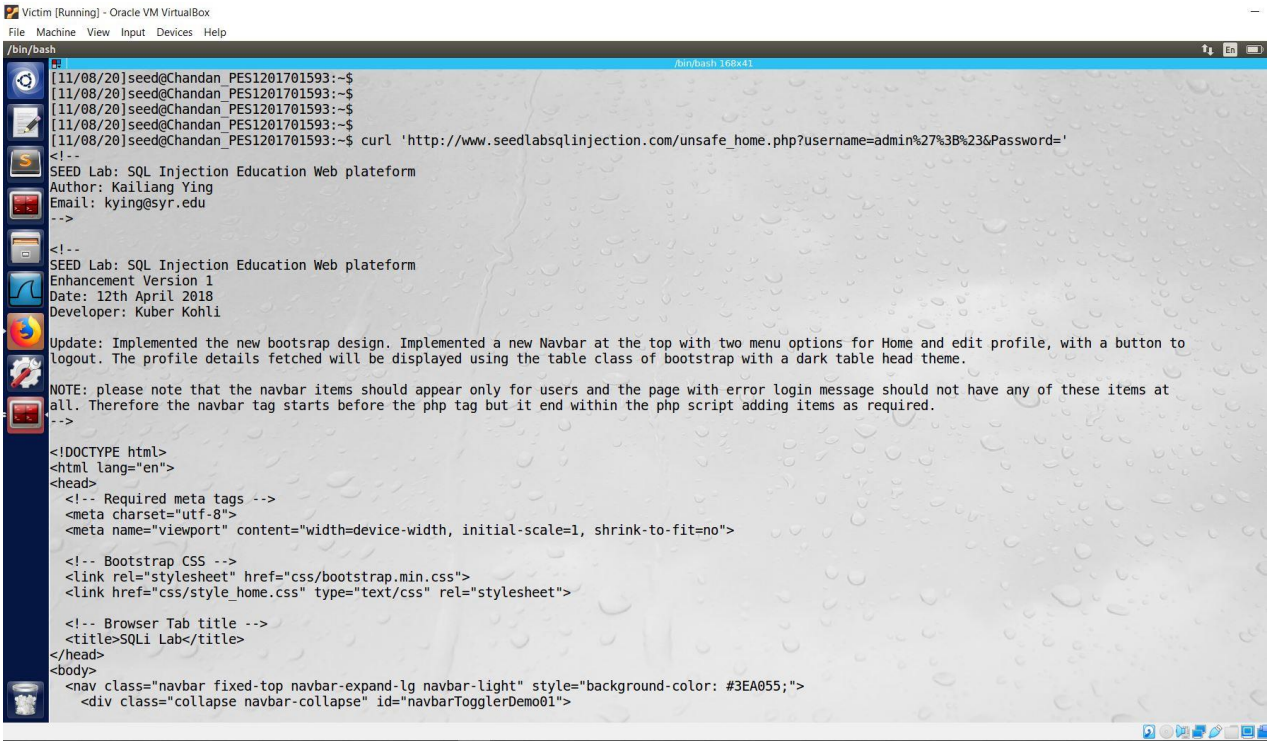


Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Chandan	10000	20000	9/20	10211002				
Dhruva	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

We observe that we are able to login as an Administrator thus we are able to access all the data of the credential table.

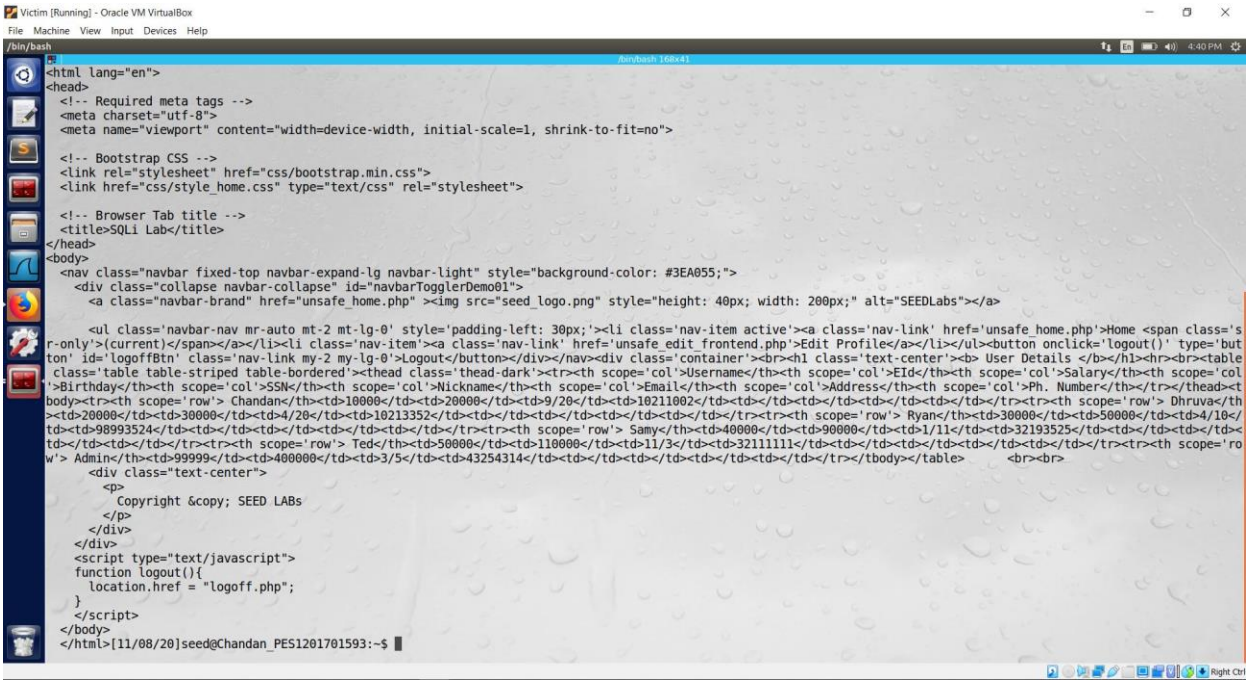
b. SQL Injection attack from command line

We will repeat the attack in Task 2(a) with command line tool curl which sends HTTP requests. Using the curl command we send a request with the username and password as a query string in a GET request.



In the curl command we pass username as “admin%27%3B%23” which is equivalent to “admin’;#”

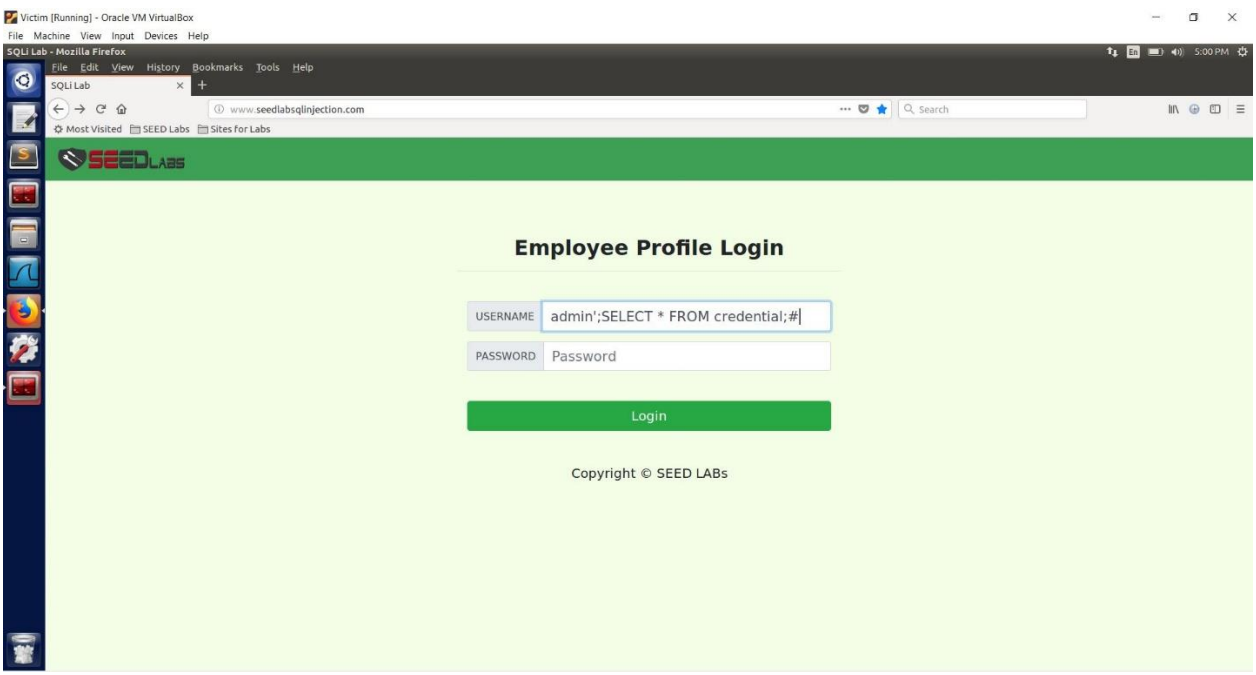
%27 represents (‘), %3B represents ‘;’ and %23 represents ‘#’.

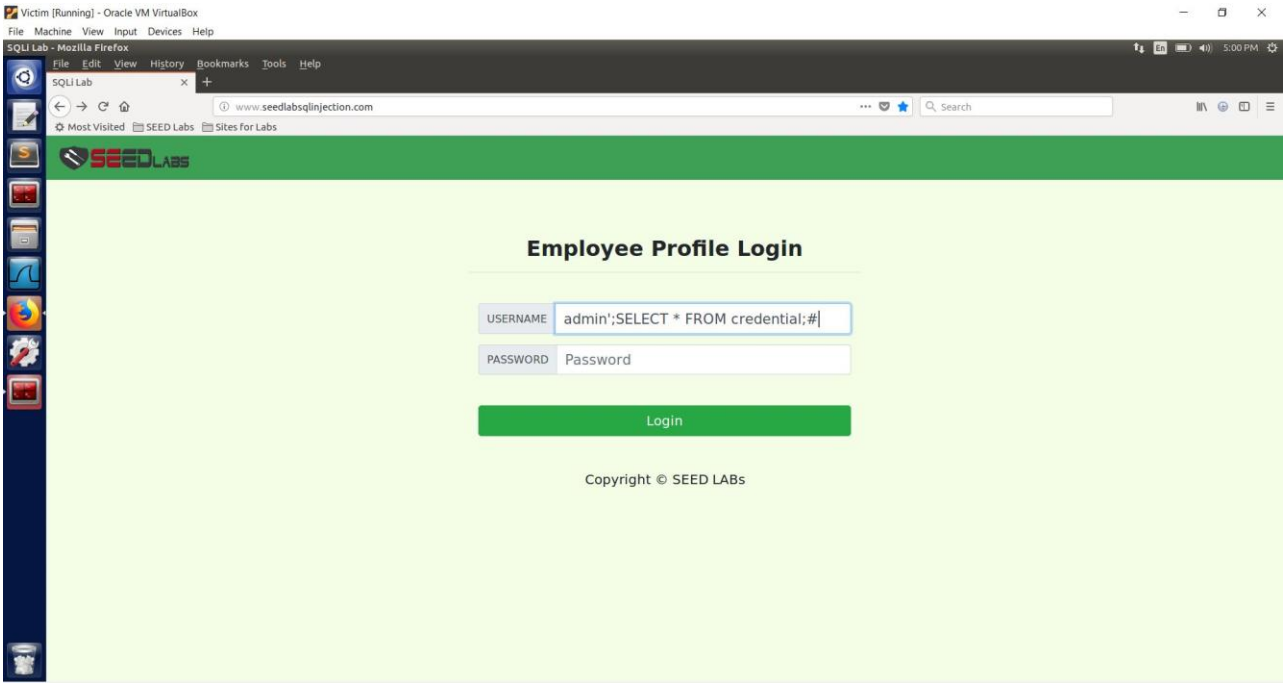


We see that the entire webpage that we got in task 2(a) is now printed as source code in the terminal.

c. Append a new SQL statement

In the previous task we had # at the end of the input string, which comments out the remaining in MySQL. Similarly, in MySQL ' ; ' marks the end of a query. Now we will try to append another SQL statement after 'admin;' to see if we can run multiple statements.

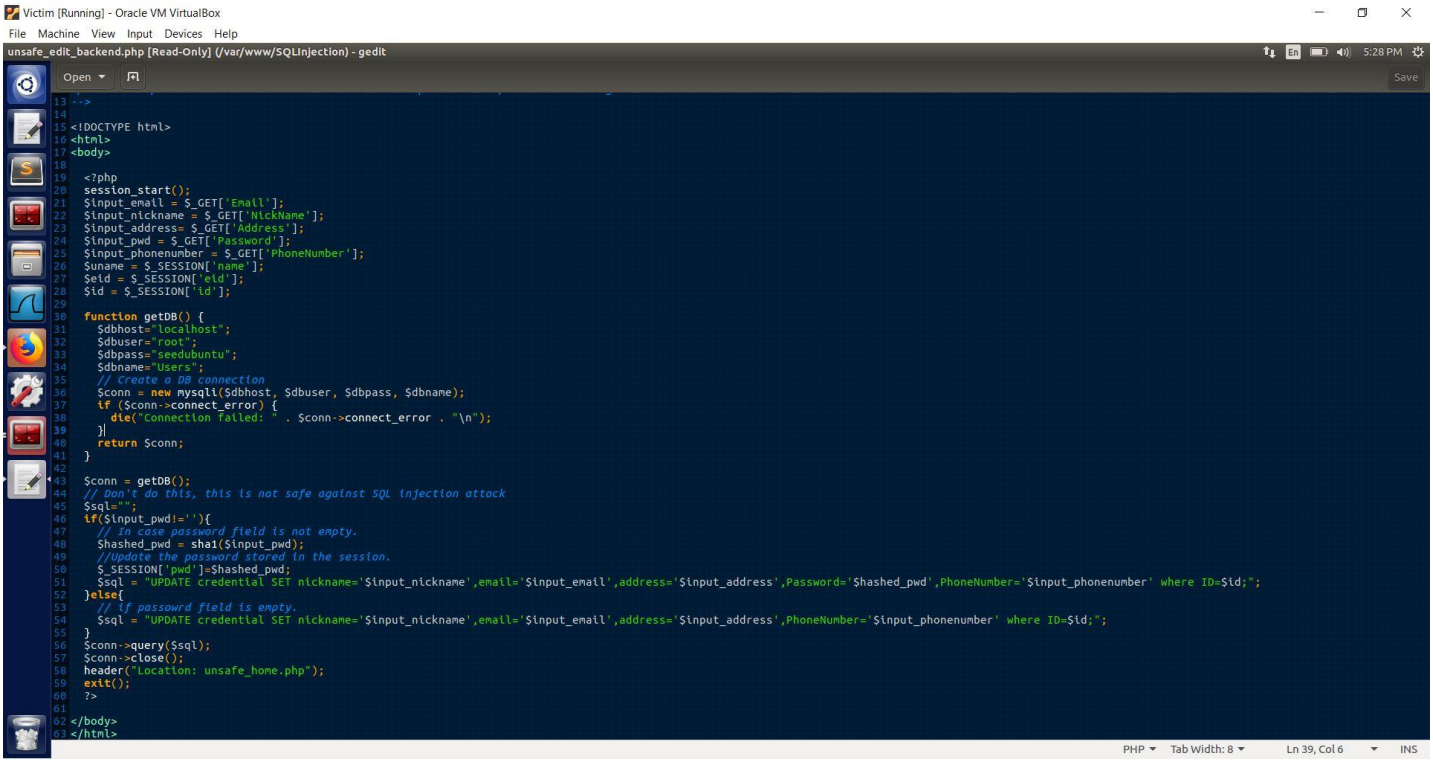




We observe that we fail in executing multiple queries and results in error at the backend.

Task 3: SQL Injection attack on Update Statement

In this task we will try an SQL injection attack on an update statement. The php code "unsafe_edit_backend" uses the UPDATE statement and passes the input without cleaning.

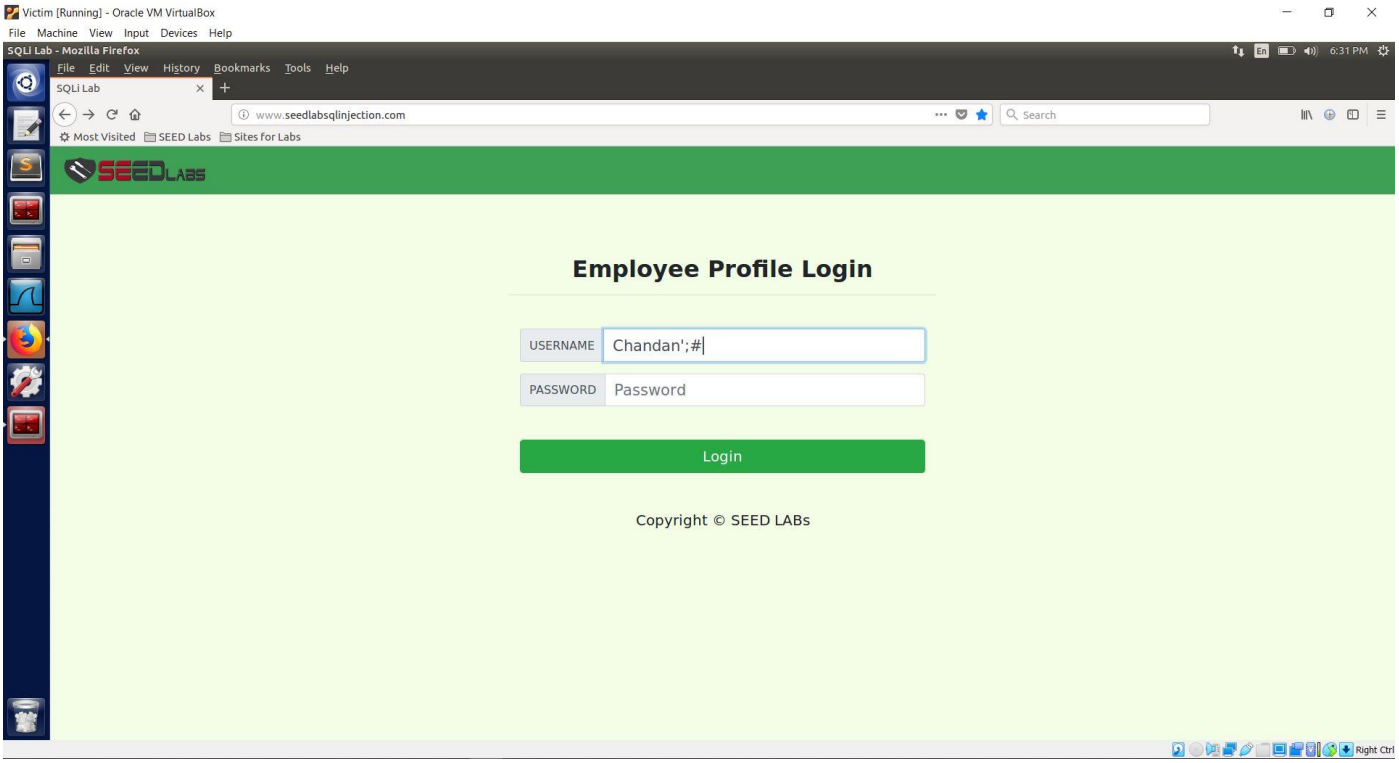


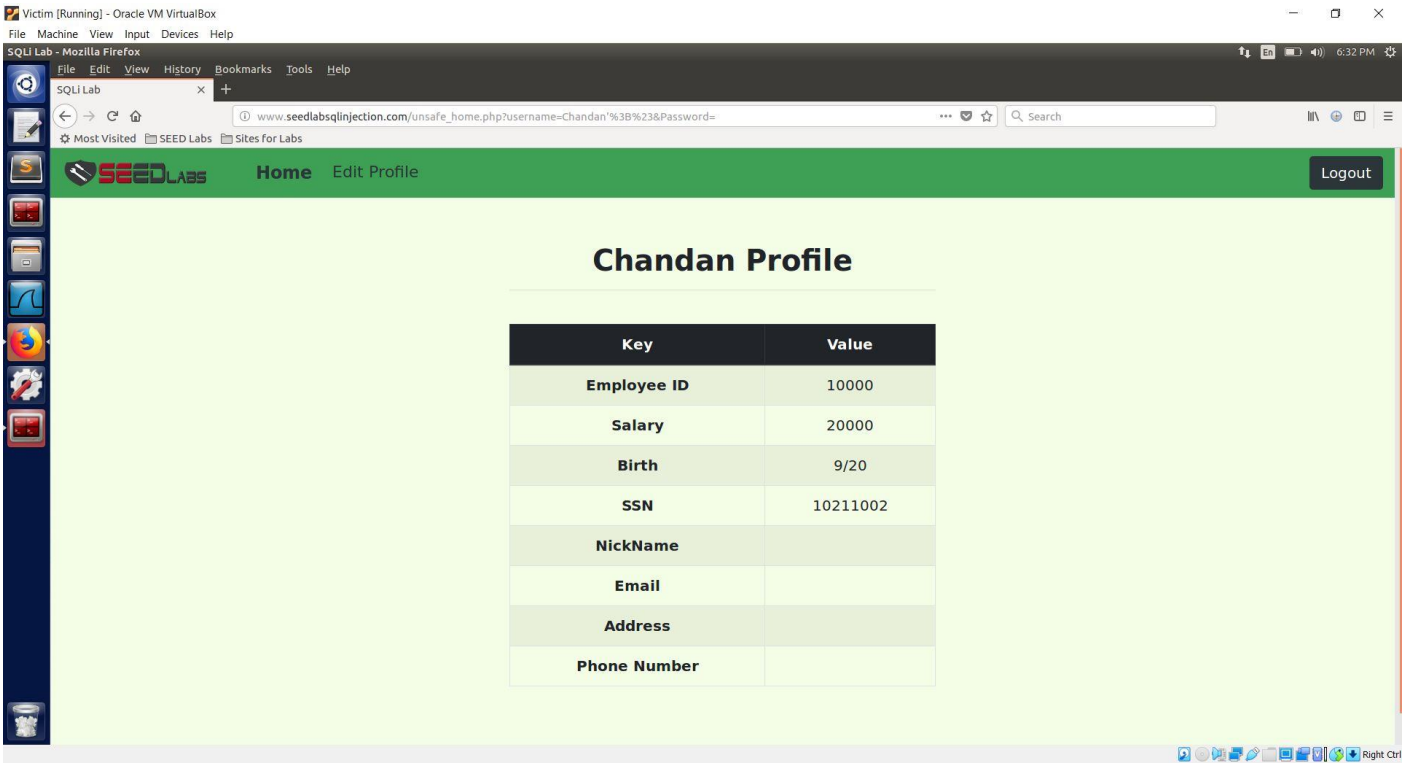
We go to the 'Edit Profile' web page to perform the attack.

a. Modify your own salary

In this subtask we will increase our own salary to a large amount.

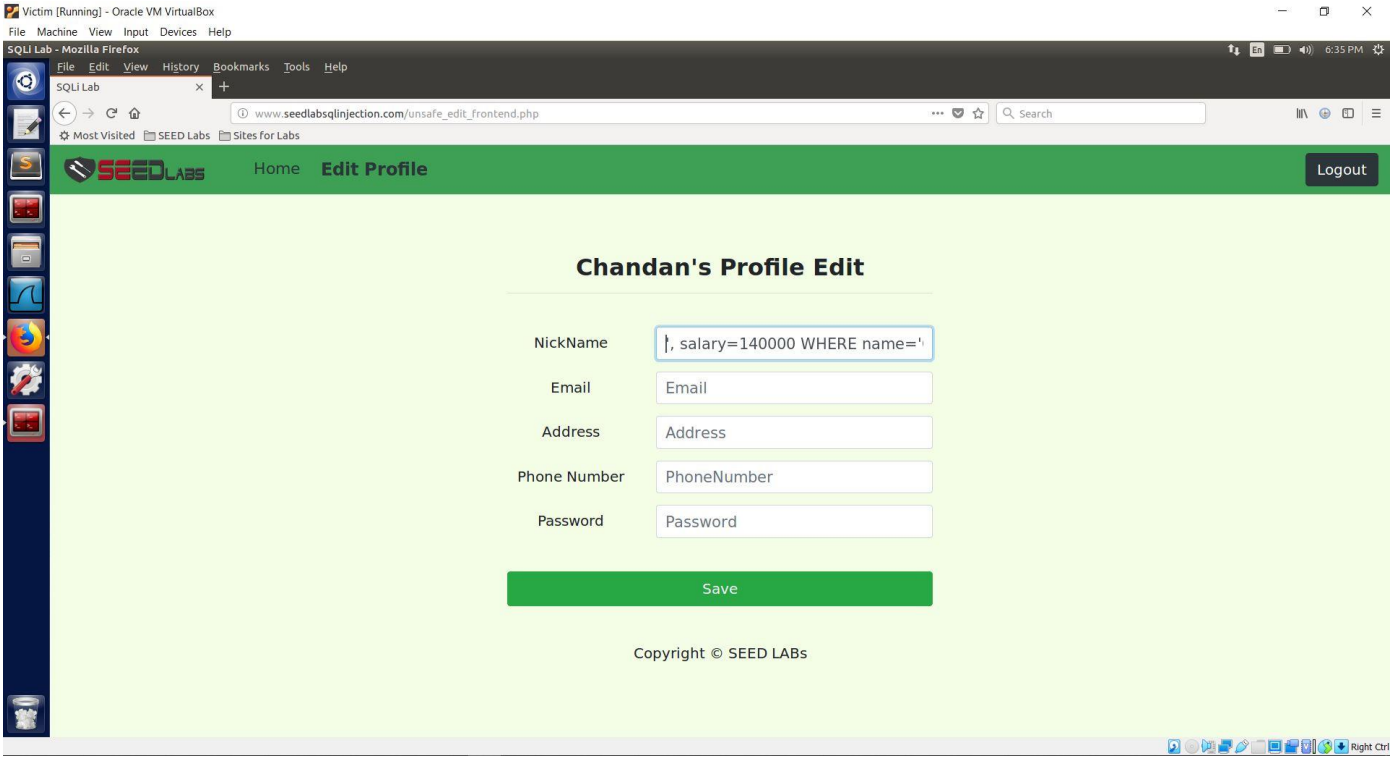
First we will login to our account by submitting in the form “Chandan’;#” to login as ‘Chandan’ without the password.



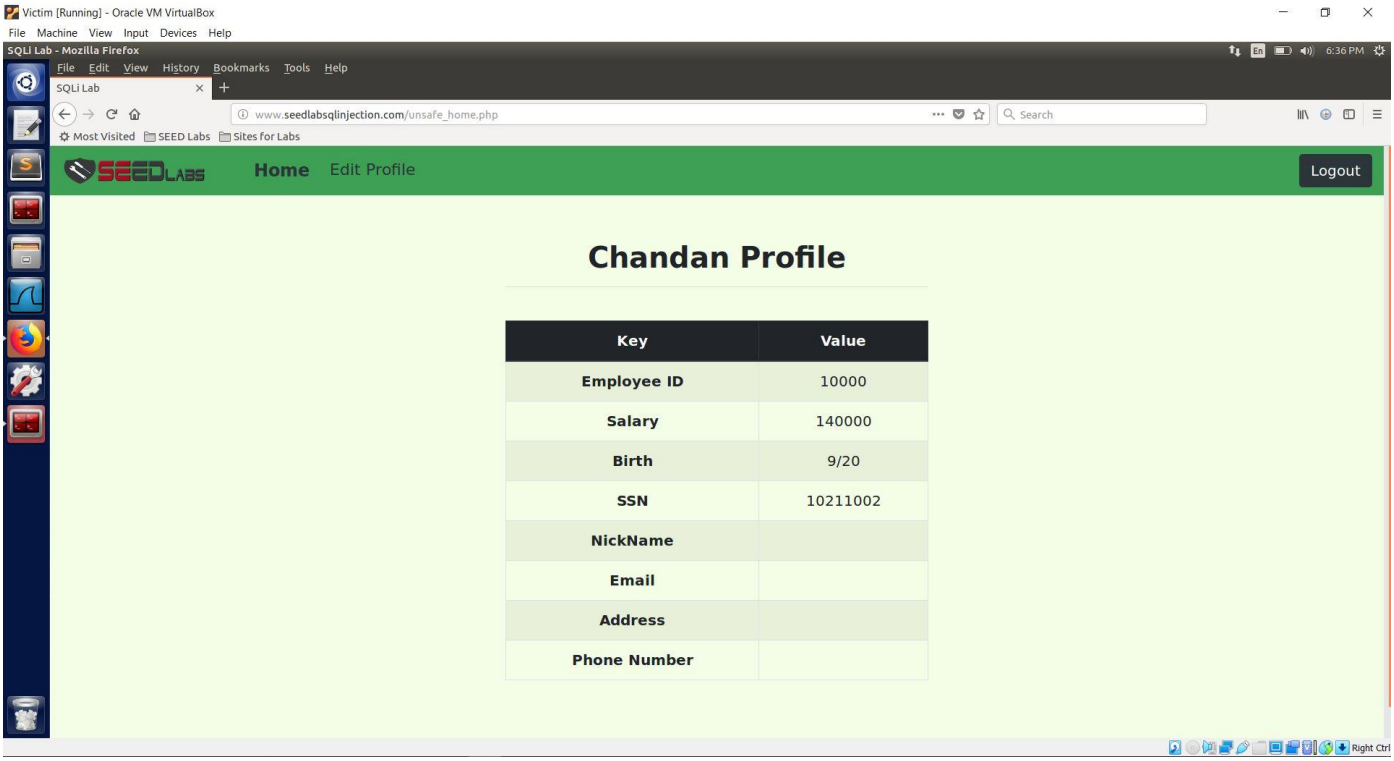


We observe that we are in Chandan’s Profile. Thus we successfully logged in as Chandan without password. We see that the salary is 20000. Now using SQL Injection we will update the salary to a large number 140000, as shown below. The input we type in the form field is:

‘, salary = 140000 WHERE Name=’Chandan’;#



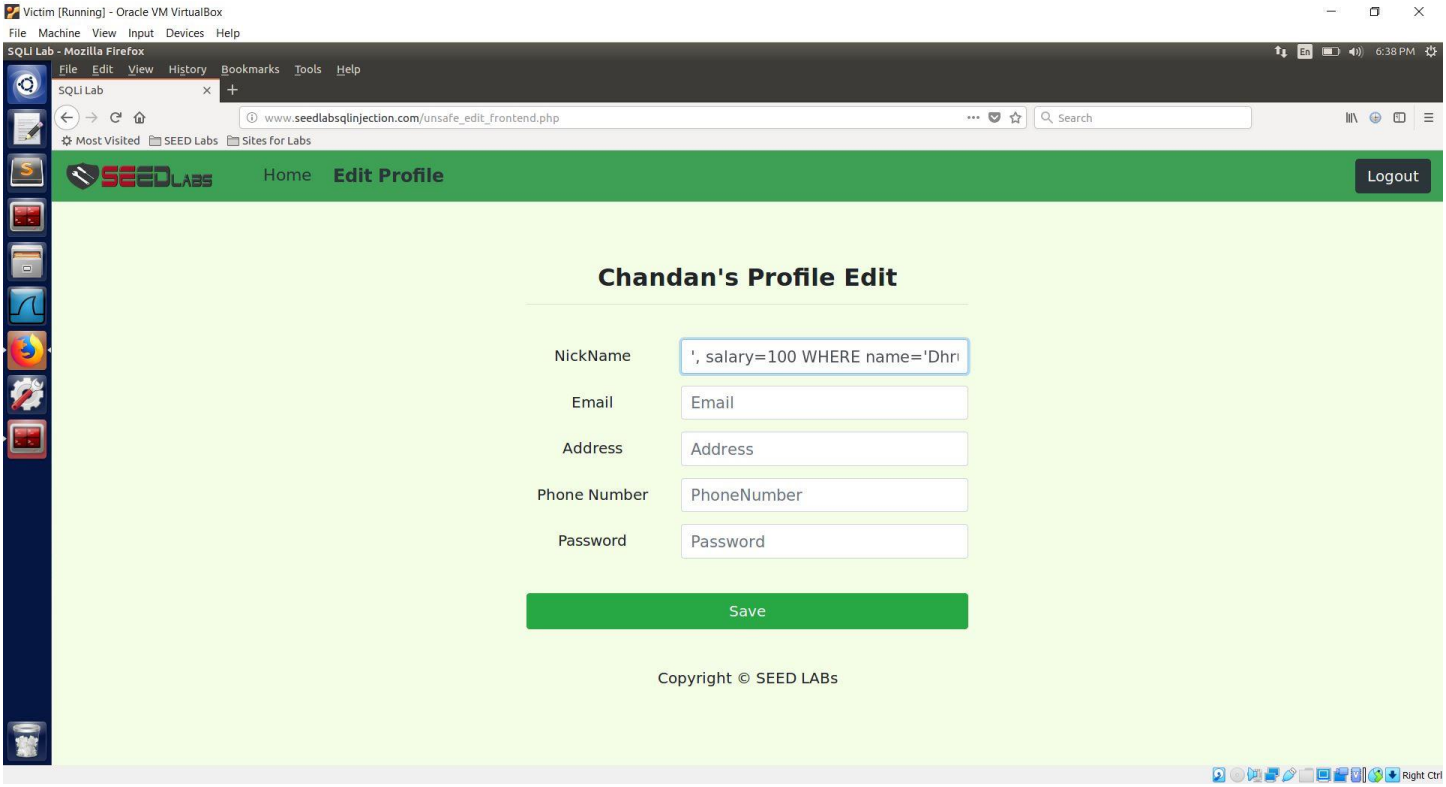
On submitting we get the below web page. We can see that the Salary field is now 140000. Thus our SQL Injection attack was successful in updating our salary.



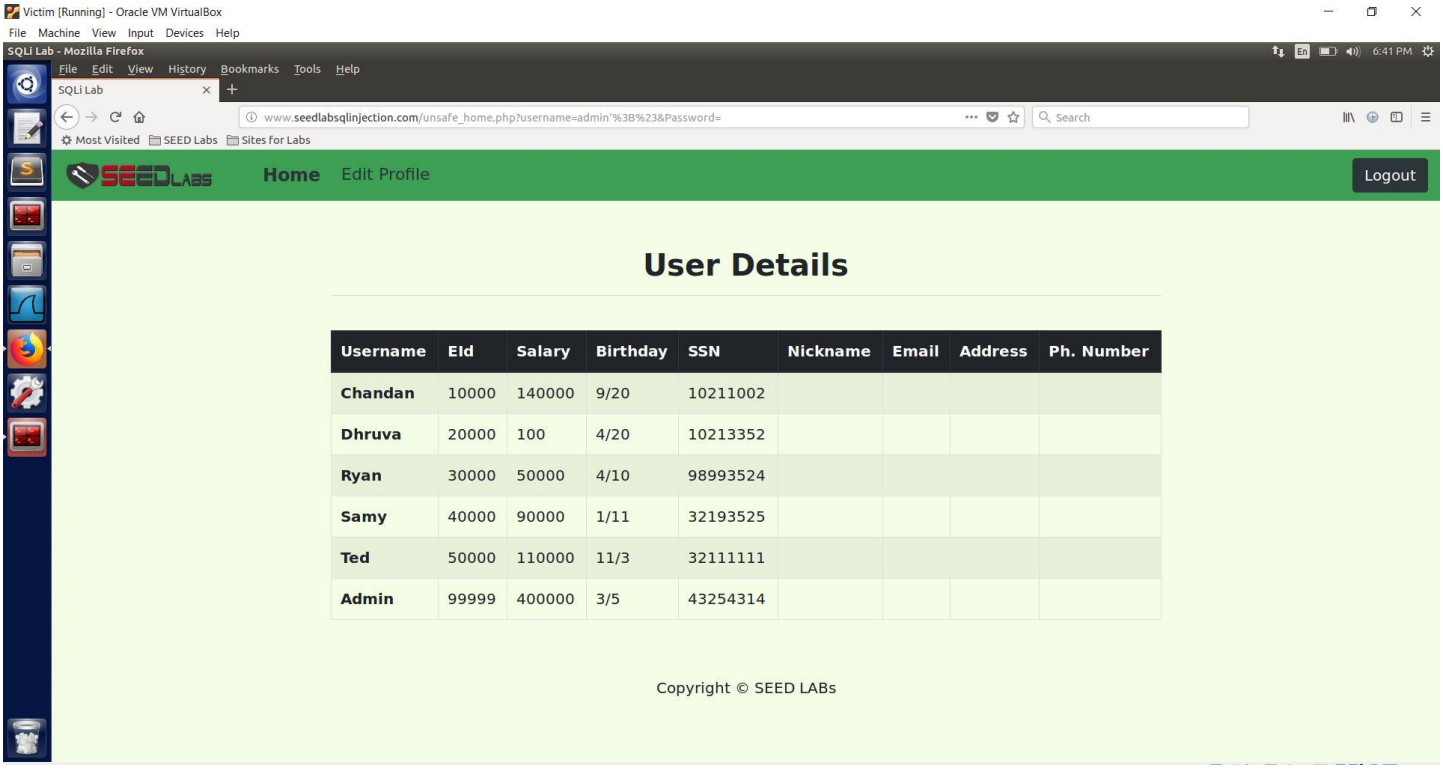
b. Modify other People's Salary

In this subtask we will change the salary of Dhruva to 100. We will first login and we observe that Dhruva's salary is 30000. Using SQL Injection we will update this to 100. The string we submit in the field input is

', salary=100 WHERE name='Dhruva';#



On submitting we get the below webpage. We observe that Dhruva's salary is now updated to 100. Thus we successfully updated other's salary using SQL Injection on UPDATE.

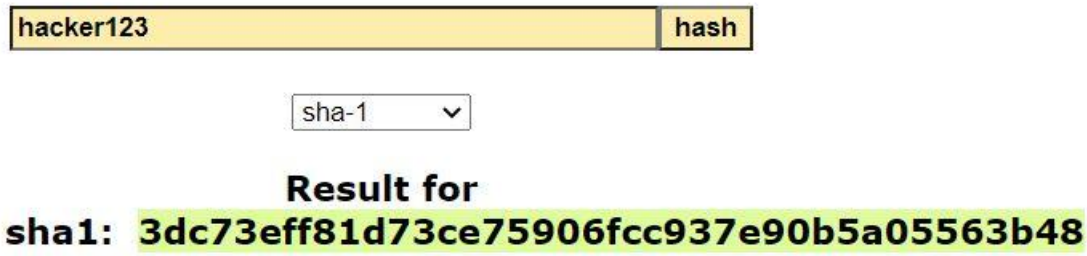


Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Chandan	10000	140000	9/20	10211002				
Dhruva	20000	100	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

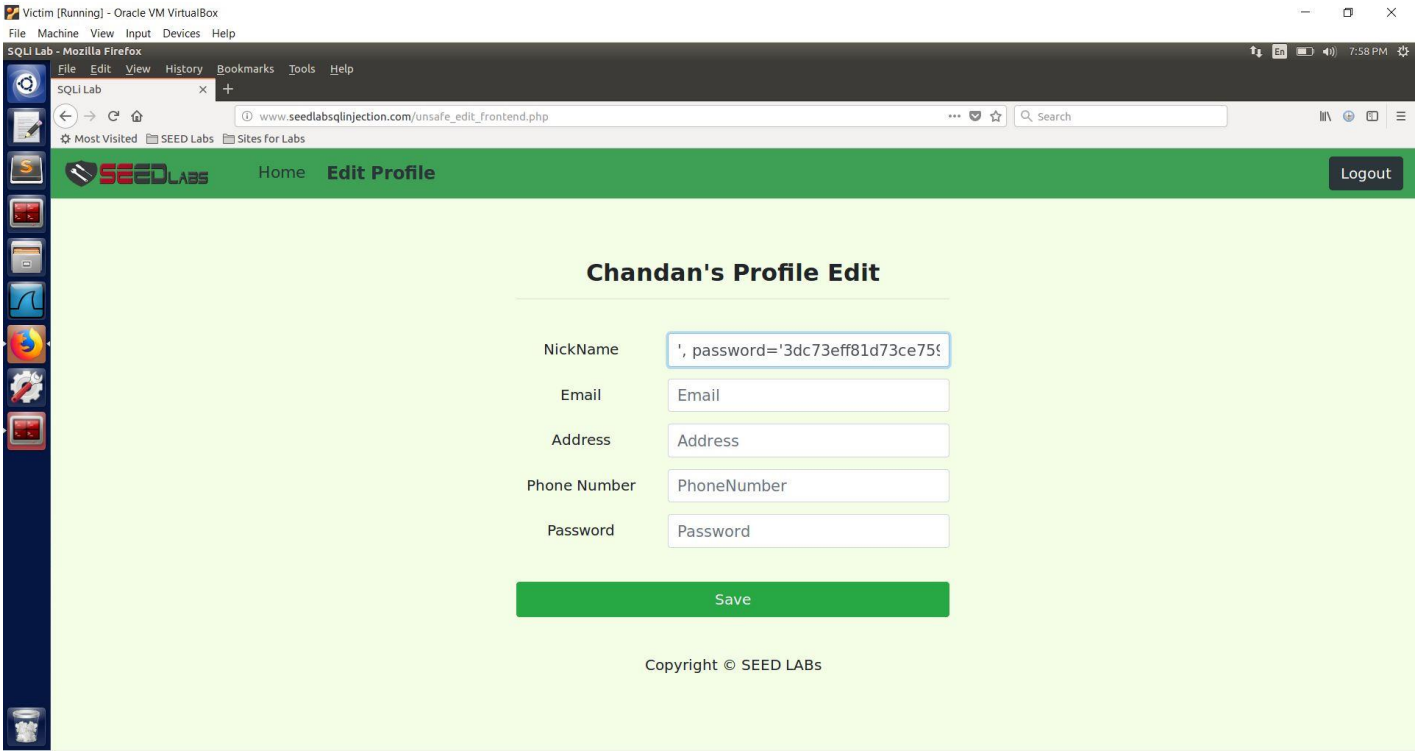
c. Modify other People's Password

In this subtask we will change Dhruva's Password so that we can prevent Dhruva's access and access it ourselves as we know the new password. We had observed that the password was first hashed using SHA1. So we need to first hash our password before updating in the database.

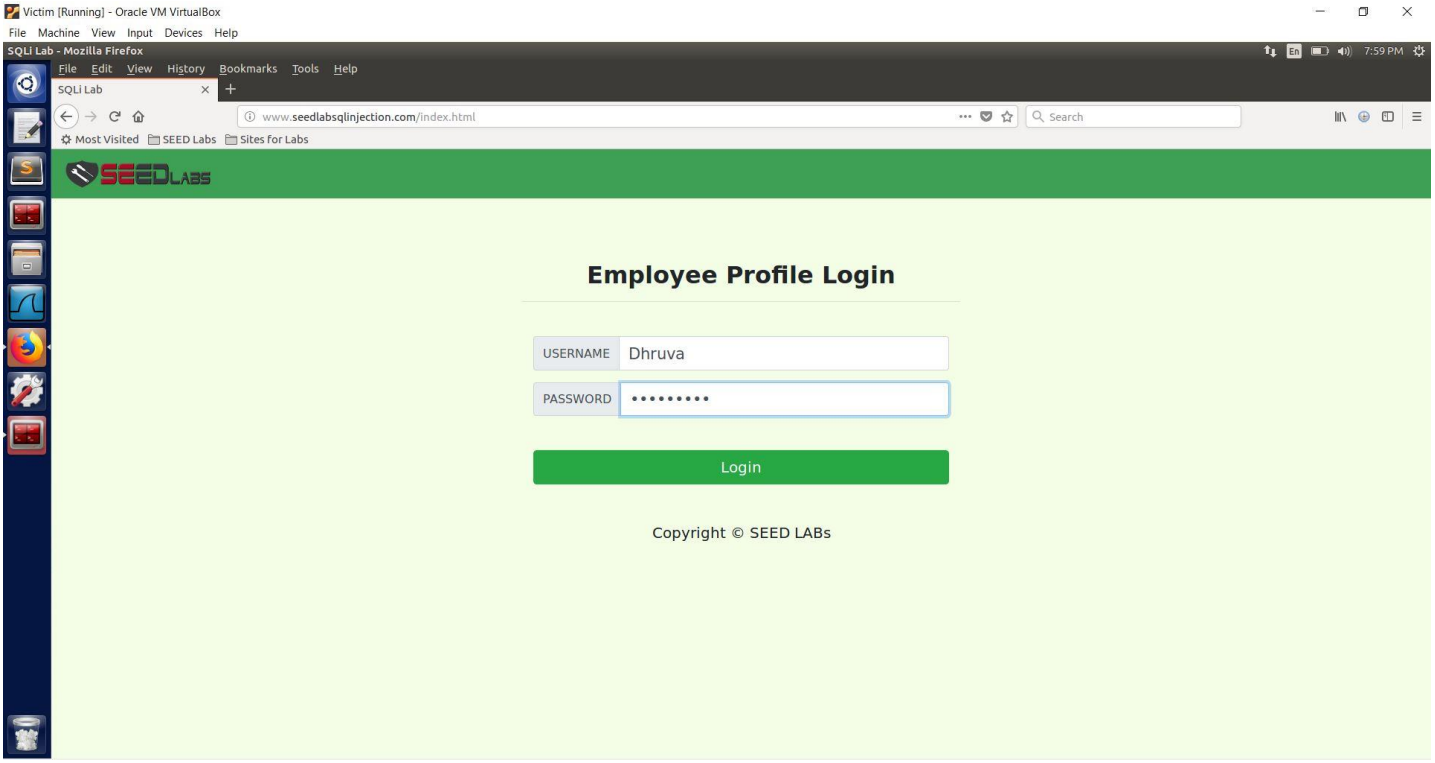
We will update Dhruva's password to 'hacker123' and its SHA1 hash is shown in the below screenshot.



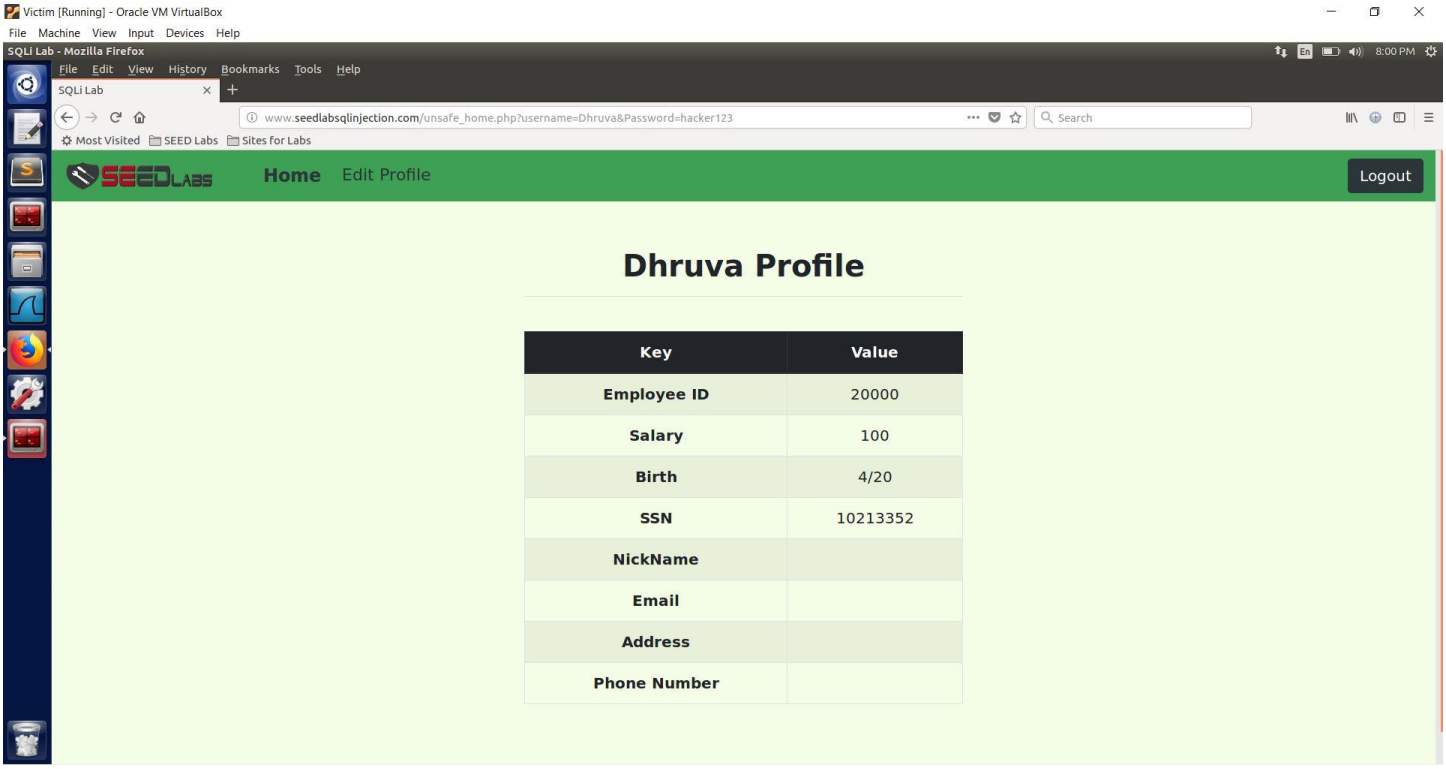
Now we use the string ', password='3dc73eff81d73ce75906fcc937e90b5a05563b48' WHERE name='Dhruva';# in the form field to perform the SQL Injection attack.



On submitting we then try to login as Dhruva with password 'hacker123' which we just updated. If our attack was successful, we should be able to login with the credentials shown below.



On submitting we observe that we successfully login as Dhruva as we are redirected to Dhruva's Profile, indicating that we were successful in updating Dhruva's Password to 'hacker123'. The password can be seen in the url query string as 'hacker123'.



Task 4: Countermeasure – Prepared Statement

We will replace ‘unsafe_home.php’ in index.html with ‘safe_home.php’ which uses prepared statement for constructing the SQL queries.

Prepared statements are used to separate code from data. Thus preventing SQL Injection attack.

We see that in index.html the action attribute of form is set to ‘unsafe_home.php’.

Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

index.html [Read-Only] (/var/www/SQLInjection) - gedit

Open

```
Developer: Kuber Kohli

Update: Implemented Bootstrap to redesign the UI of the website.
-->
<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-light" style="background-color: #3EA055;">
    <a class="navbar-brand" href="#" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
  </nav>
  <div class="container  col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    <h2><b>Employee Profile Login</b></h2><hr><br>
    <div class="container">
      <form action="unsafe_home.php" method="get">
        <div class="input-group mb-3 text-center">
          <div class="input-group-prepend">
            <span class="input-group-text" id="uname">USERNAME</span>
          </div>
          <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="uname">
        </div>
        <div class="input-group mb-3">
          <div class="input-group-prepend">
            <span class="input-group-text" id="pwd">PASSWORD </span>
          </div>
          <input type="password" class="form-control" placeholder="Password" name="Password" aria-label="Username" aria-describedby="pwd">
        </div>
        <br>
        <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
      </form>
    </div>
    <br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABs
      </p>
    </div>
  </div>
</body>
</html>
```

We will change it to ‘safe_home.php’ as shown below.

Victim [Running] - Oracle VM VirtualBox

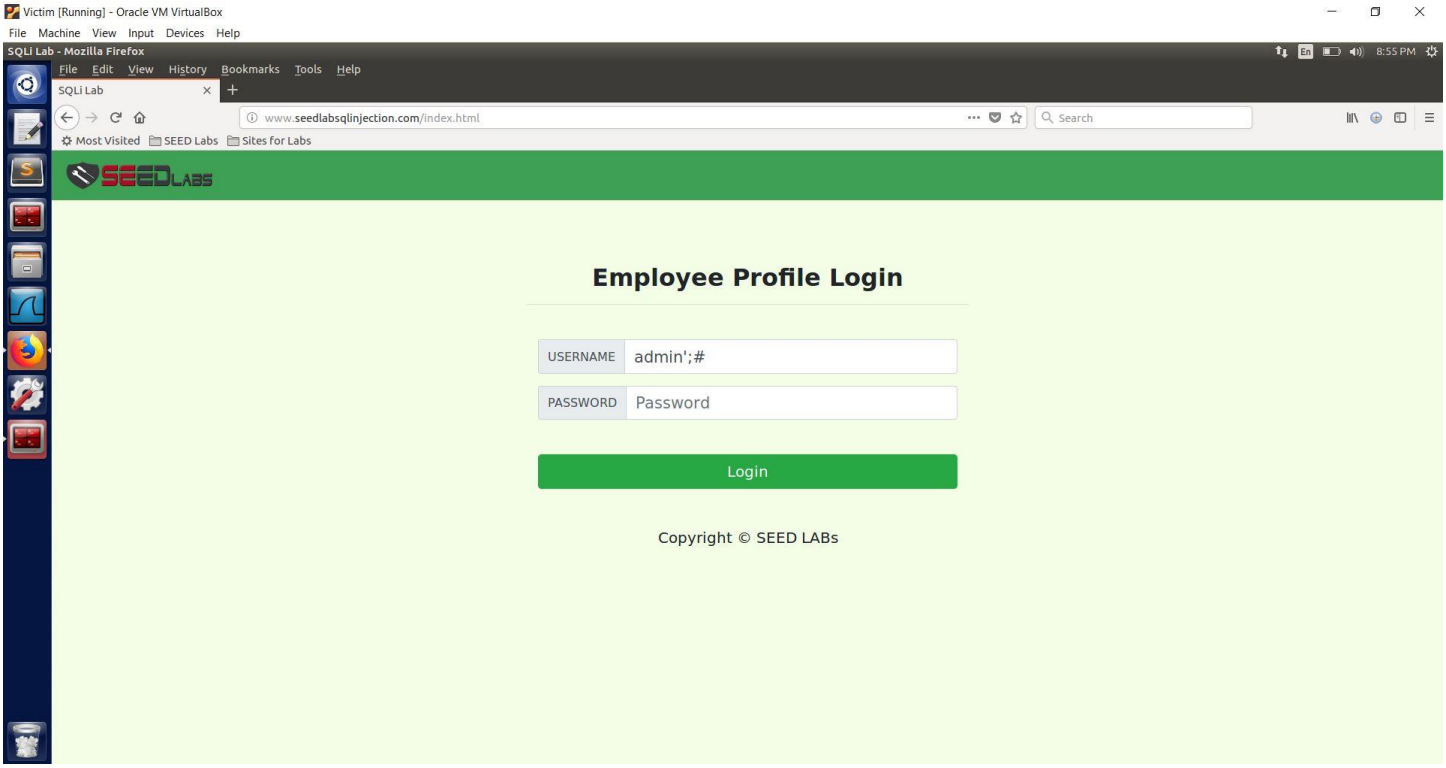
File Machine View Input Devices Help

*index.html [Read-Only] (/var/www/SQLInjection) - gedit

Open

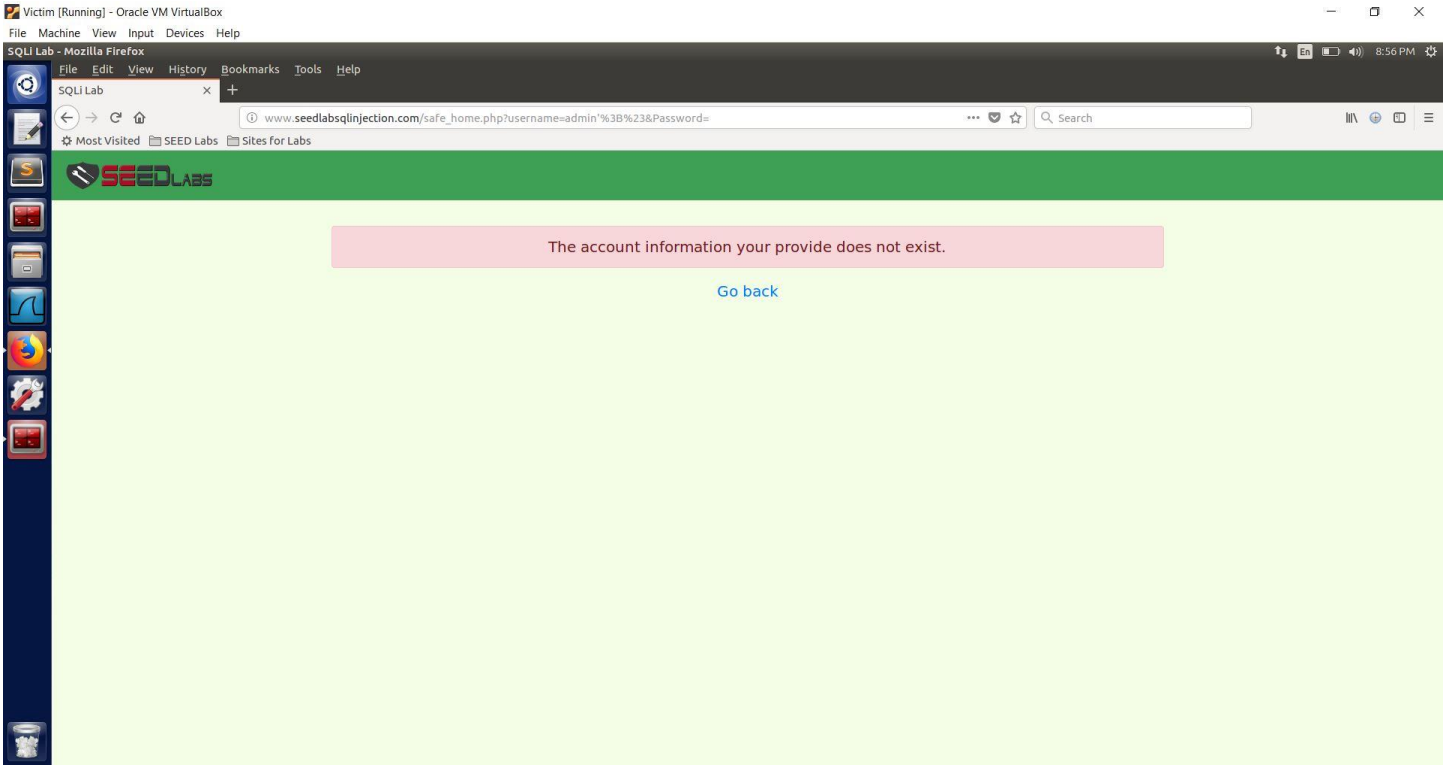
```
Developer: Kuber Kohli

Update: Implemented Bootstrap to redesign the UI of the website.
-->
<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-light" style="background-color: #3EA055;">
    <a class="navbar-brand" href="#" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
  </nav>
  <div class="container  col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    <h2><b>Employee Profile Login</b></h2><hr><br>
    <div class="container">
      <form action="safe_home.php" method="get">
        <div class="input-group mb-3 text-center">
          <div class="input-group-prepend">
            <span class="input-group-text" id="uname">USERNAME</span>
          </div>
          <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="uname">
        </div>
        <div class="input-group mb-3">
          <div class="input-group-prepend">
            <span class="input-group-text" id="pwd">PASSWORD </span>
          </div>
          <input type="password" class="form-control" placeholder="Password" name="Password" aria-label="Username" aria-describedby="pwd">
        </div>
        <br>
        <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
      </form>
    </div>
    <br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABs
      </p>
    </div>
  </div>
</body>
</html>
```

Now we will try to login as admin using “admin’;#” as shown below.



With the countermeasure, we should not be able to login using SQL Injection attack.

On submitting the form we observe as shown below.



We observe that the SQL Injection attack was not successful indicating our countermeasure worked.

THANK YOU