**SQL INJECTION LAB 2.0**

# Contents

Contents

[Contents 2](#_Toc119548396)

[SQL Injection Attack Lab 3](#_Toc119548397)

[Task 2: 5](#_Toc119548398)

[Task 2.1: 5](#_Toc119548399)

[Task 2.2: 6](#_Toc119548400)

[Task 2.3: 7](#_Toc119548401)

[Task 3: 7](#_Toc119548402)

[Task 3.1: 7](#_Toc119548403)

[Task 3.2: 8](#_Toc119548404)

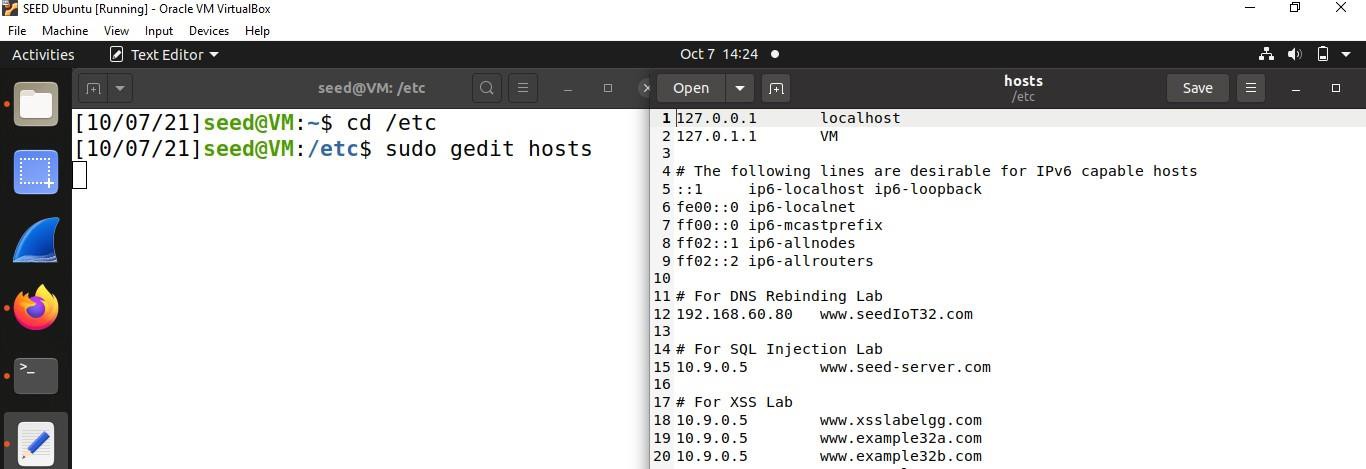
[Task 3.3: 8](#_Toc119548405)

[Task 4: 10](#_Toc119548406)

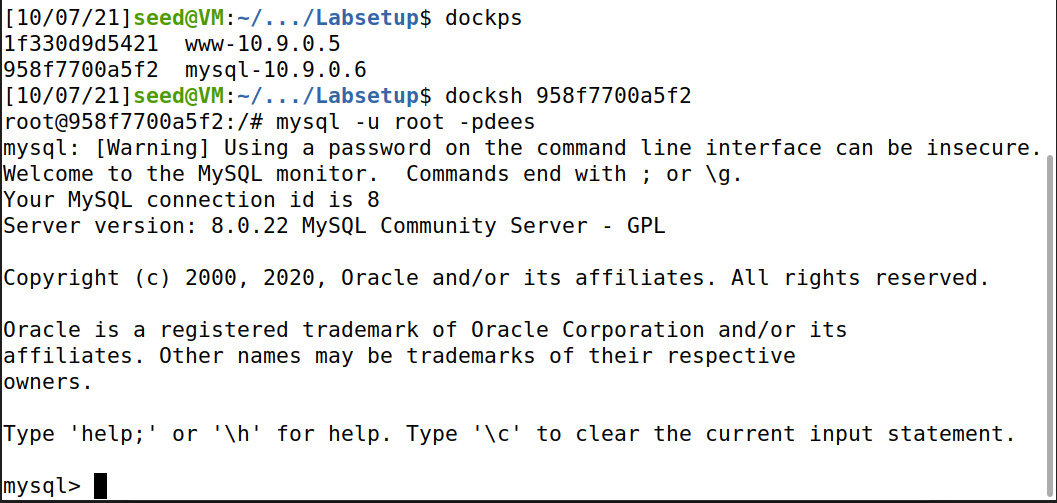
Task 1:

# SQL Injection Attack Lab

* First, we will download and launch the SEED Ubuntu VM provided to us. The SEED interface looks like this. Then, open terminal. On terminal we will add the ‘seed-server’ website for SQL Injection Attack, On line 15 in front of the IP address 10.9.0.5.

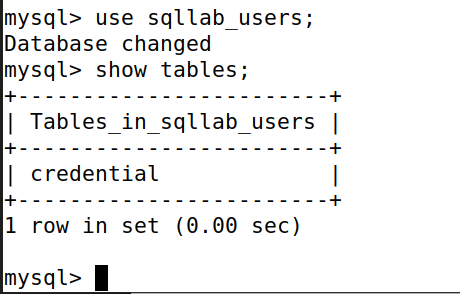


* Now, we will download the folder (Labsetup.zip) containing the docker file to set our lab environment. Open terminal from the Labsetup folder and write the command, ‘dockps’. Dockps basically shows the status of running docker containers.



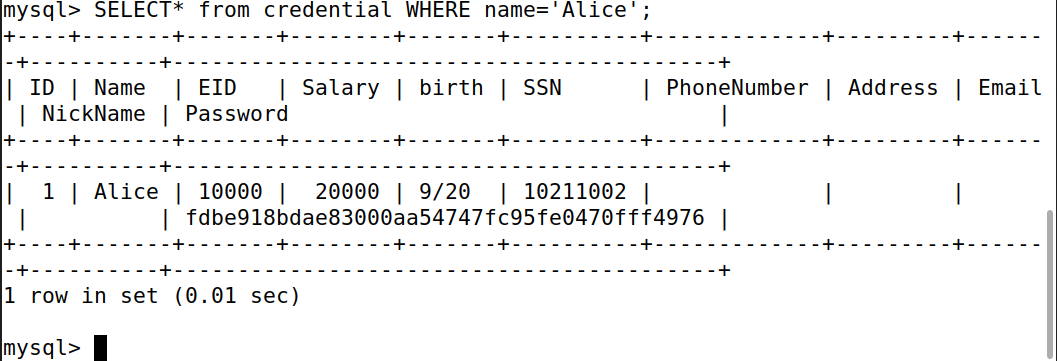
* Now, write the container ID of mysql with docksh to start a shell session inside our container so that we can interact with it using our terminal. Now, we have to specify the shell to be of mysql by writing the following command “mysql -u root -p(password of your vm)”

Now, we will use the data base provided to us, i.e ‘sqllab\_users’.



Write ‘show tables;’ to view the tables in the data base given to us. To view the information of any user (lets say, Alice) , we will write the following command, i.e.

SELECT\* from credential WHERE name=’Alice’;



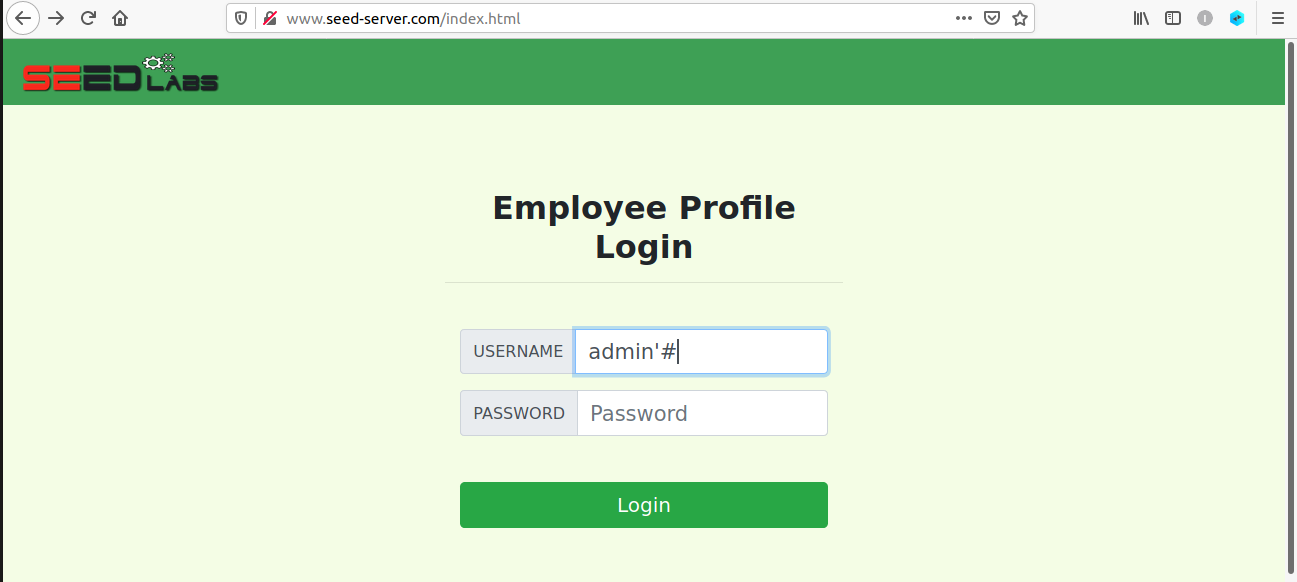
## Task 2

In this task, we will perform SQL Injection Attack from webpage and through our terminal. To open the [www.seed-server.com](http://www.seed-server.com/) we will first write the command of ‘docker-compose build’ and then ‘docker-compose up’. After using the last command, the terminal will stop at a point where we will open the seed-server on browser by writing the above mentioned link.

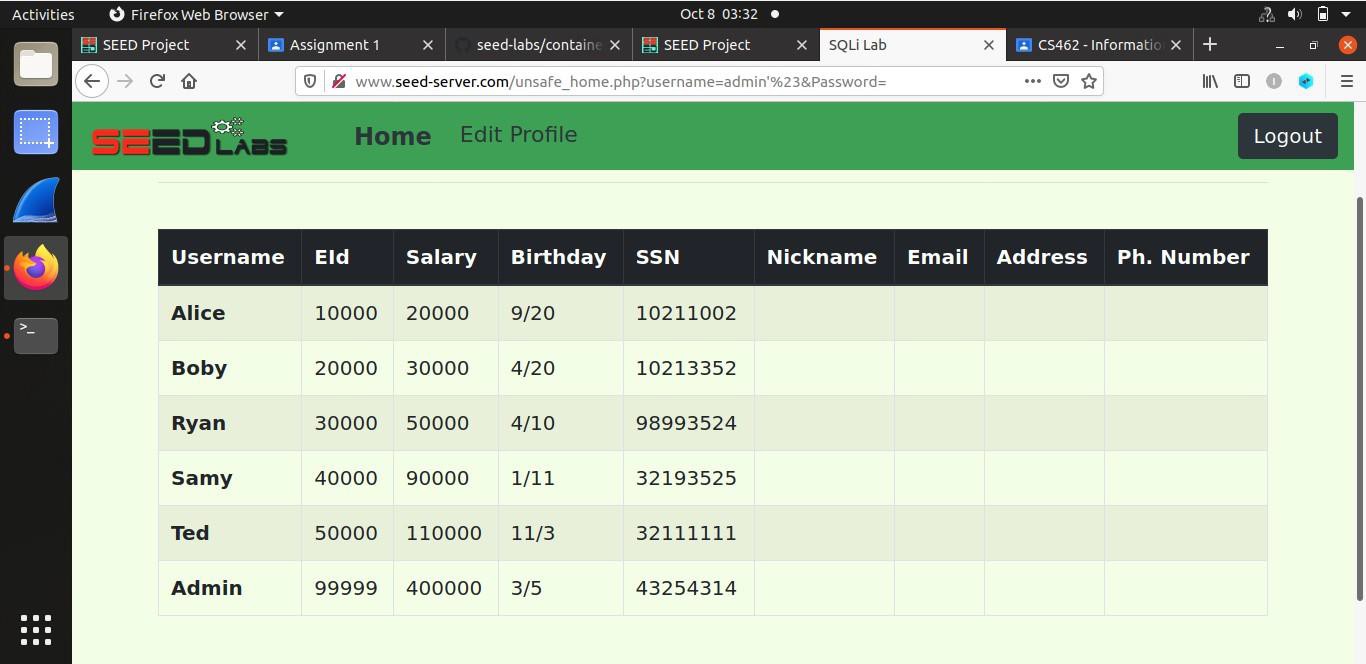
Then, we get to see the login page for employees.

### Task 2.1

We will perform SQL Injection through webpage in this task. To login we will try different special characters along with the username to get the access.



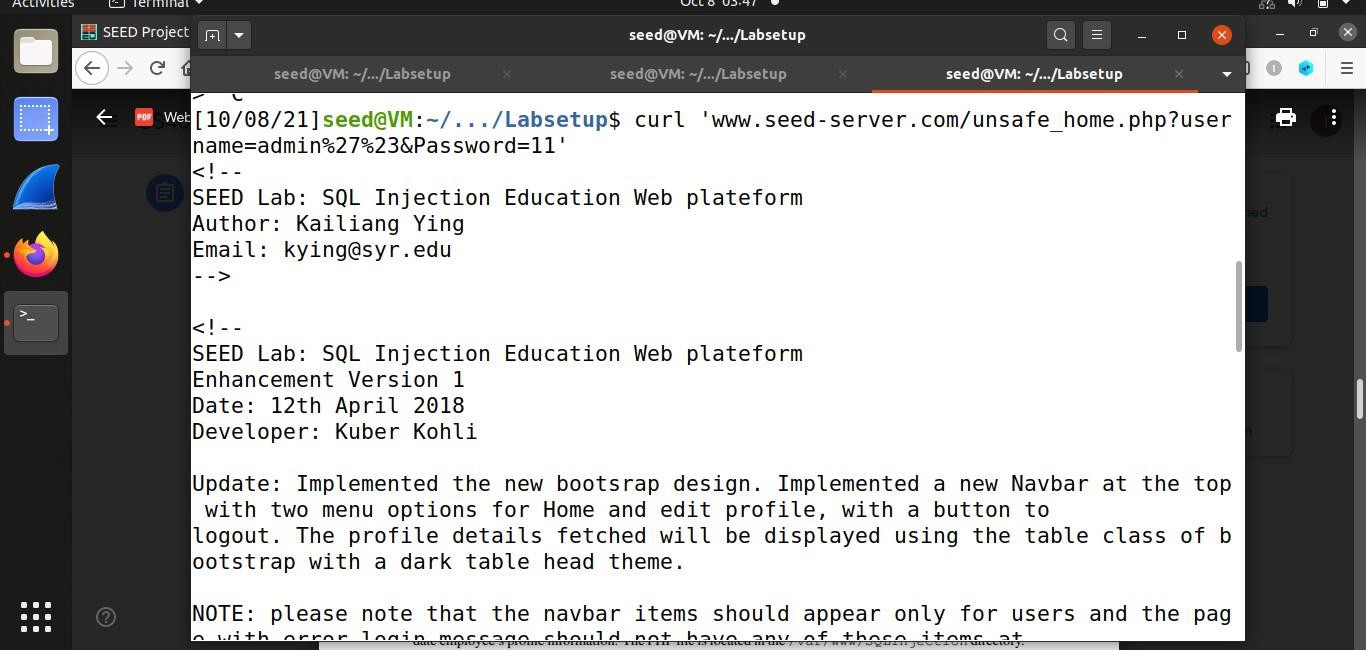
Our input ‘admin’#’ worked and gave us the access through the admin profile.



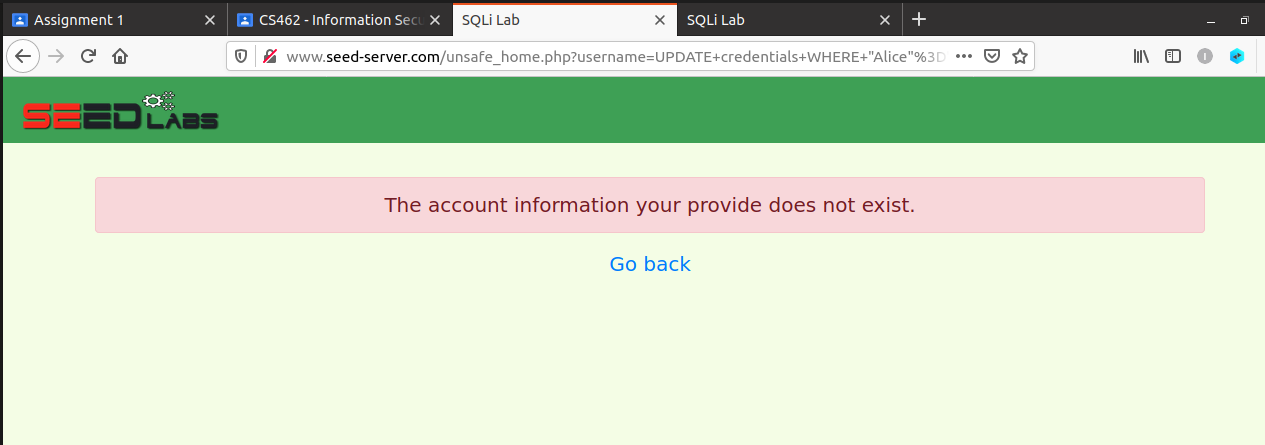
### Task 2.2

Here, we will perform SQL Injection through ‘curl’ command given to us in the manual. We will write the browser link with the curl command and write ‘%27’ to replace apostrophe and ‘%23’ to replace hash and then write a random password. The command is given below in the

Screenshot. This command gave us the whole html document of the login page and some of the data of the users.



### Task 2.3

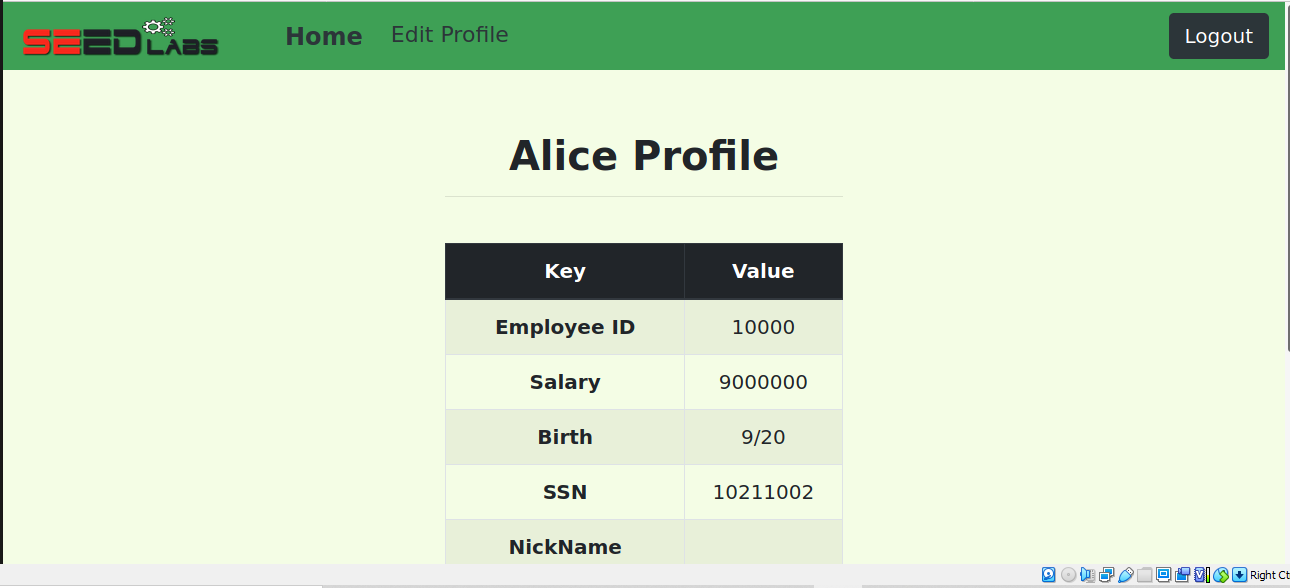
The two SQL commands we wrote on the login page, separated by ; didn’t worked. This is due to the input validation checks added, which do not accept hyphens as an input and gives us the following error.

## 

## Task 3

### Task 3.1

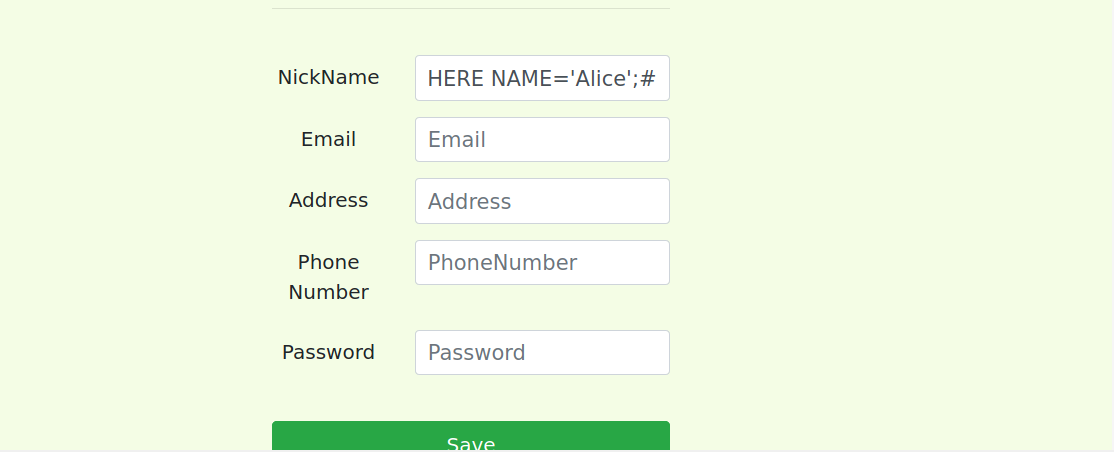
In this task, we firstly opened Alice’s profile from the information we got from Task 1.



Using Alice’s profile we increased her salary by writing the following command the Profile Edit Page..

‘,salary=’9000000’ WHERE NAME=’Alice’;#

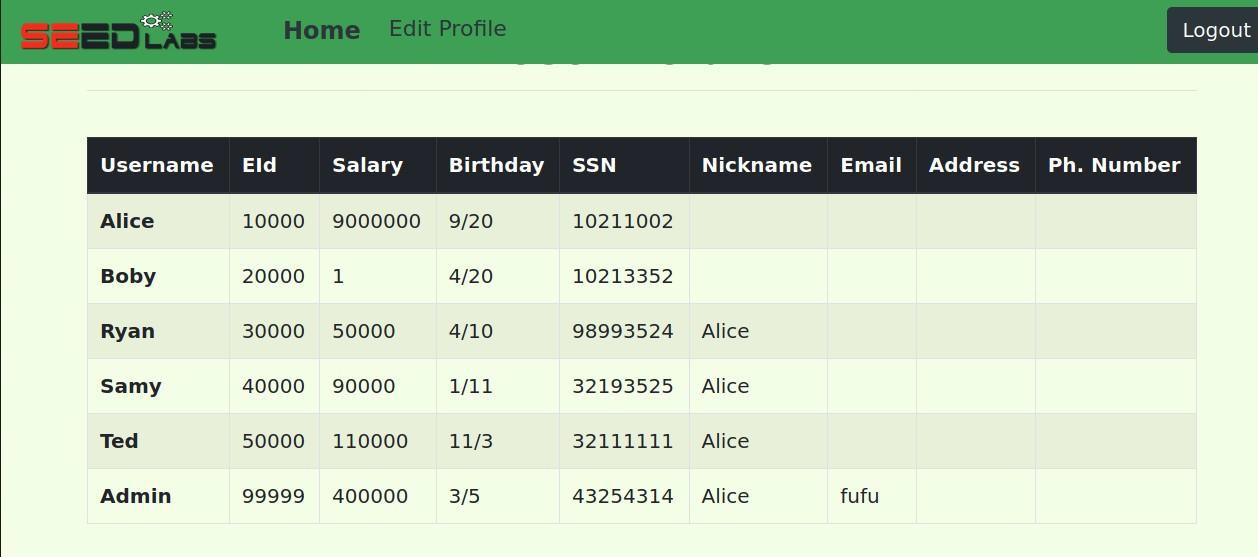
### Task 3.2



Now, to change Boby’s salary, we will write the same command on the same webpage replacing some of the values…

‘,salary=’1’ WHERE NAME=’Boby’;#

After saving the commands, we will get the desired result as shown below

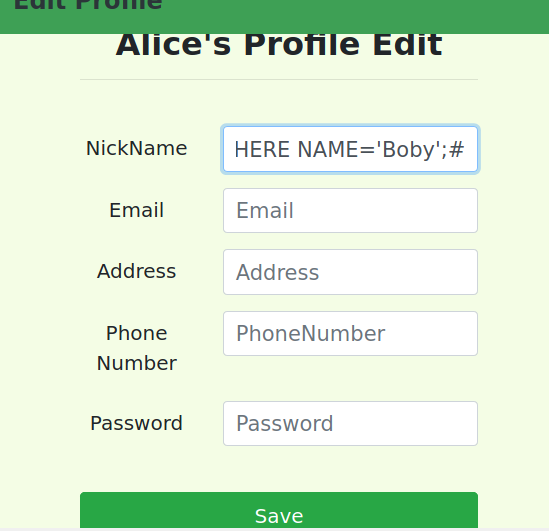


### Task 3.3

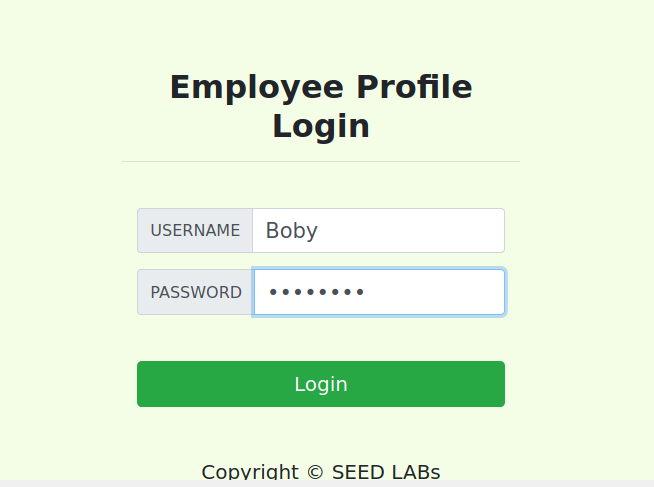
Now, to modify Boby’s password, we will first login again from Alice’s profile. In the profile edit page, we will write the following command:

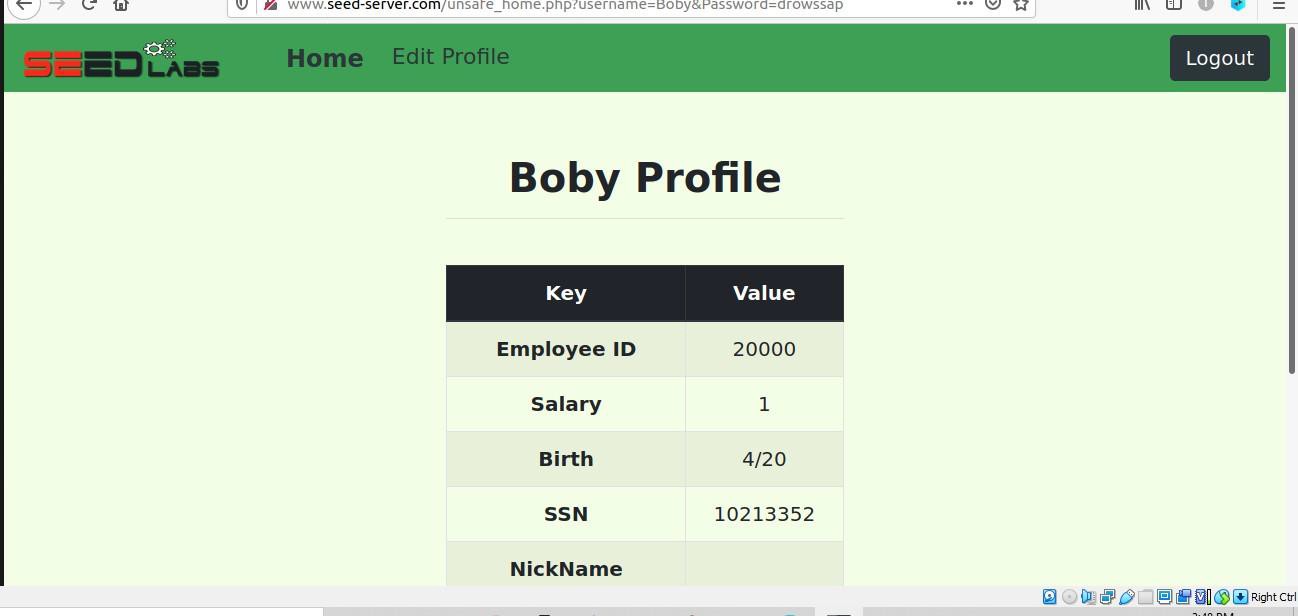
‘,Password=’ D50F3D3D525303997D705F86CD80182365F964ED’ WHERE NAME=’Boby’;#

Note, we have written the sha1 hash of our selected password, because our password is saved in the form of the hash in the database.



After saving the command, we will now try to login using our selected password through Boby’s account.

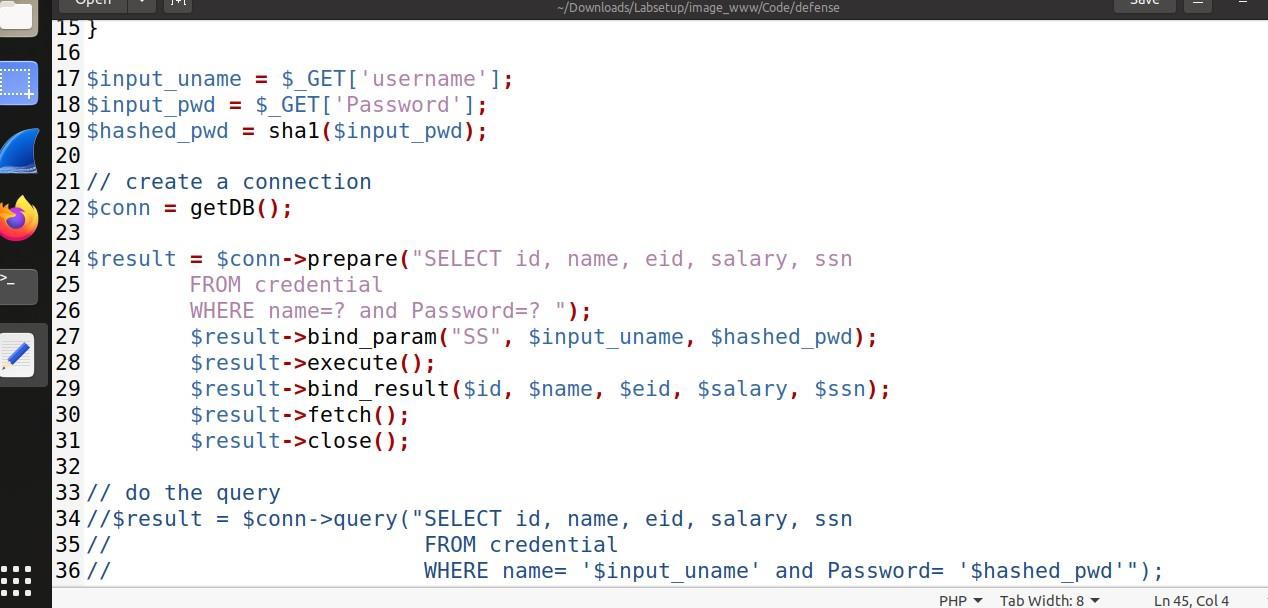




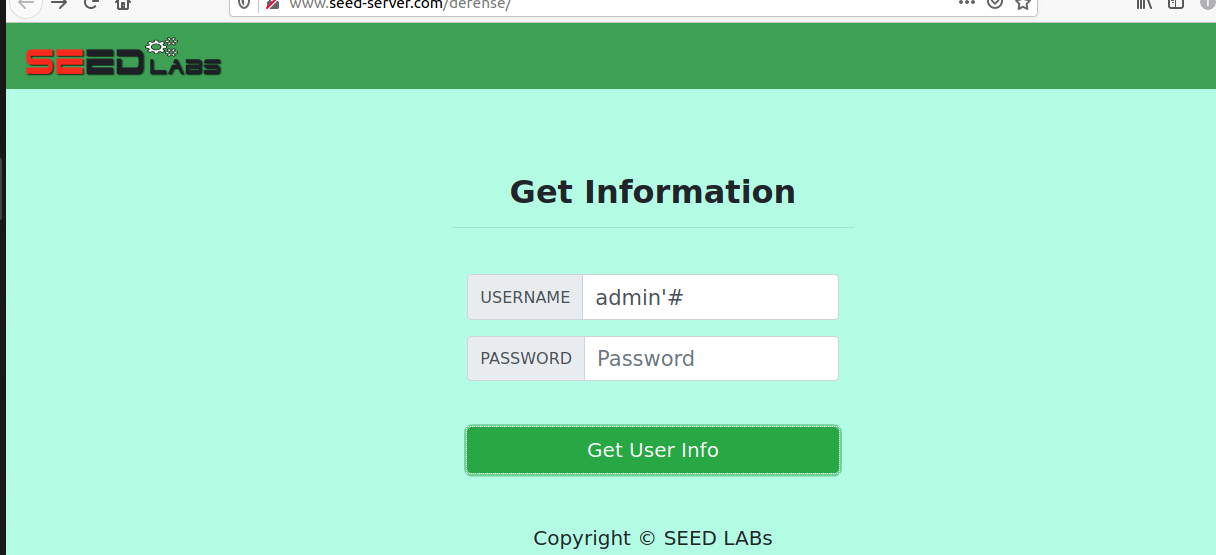
You can clearly see that we got the access to Boby’s account.

## Task 4

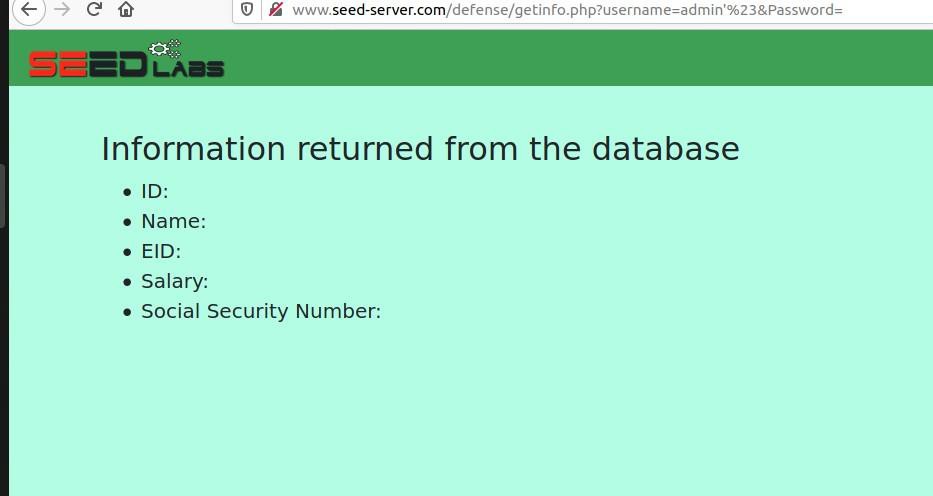
In this task, we will protect our webpage from SQL Injection by editing the code file given to us in the Labsetup folder. We will use PREPARE statement replaced from the query statement. The code we changed is given below



Now, again start docker-compose build and docker-compose up, and the open the URL given in the manual. You’ll get to see the following login page



Try SQL Injection here, and you’ll get the following result



Hence we successfully protected our website from the attack!