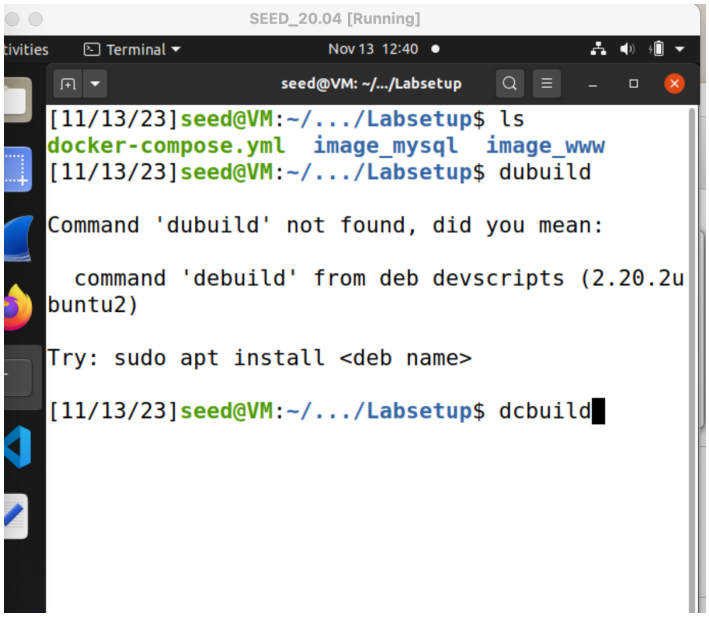
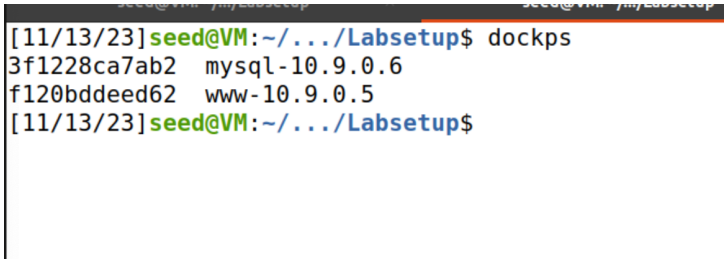
ICSI 424 Lab9

Seoyeon Choi

Type Ls the check what is in Lab Setup and type a dcbuild to setup the lab and dcup



Find id of container by typing dockps

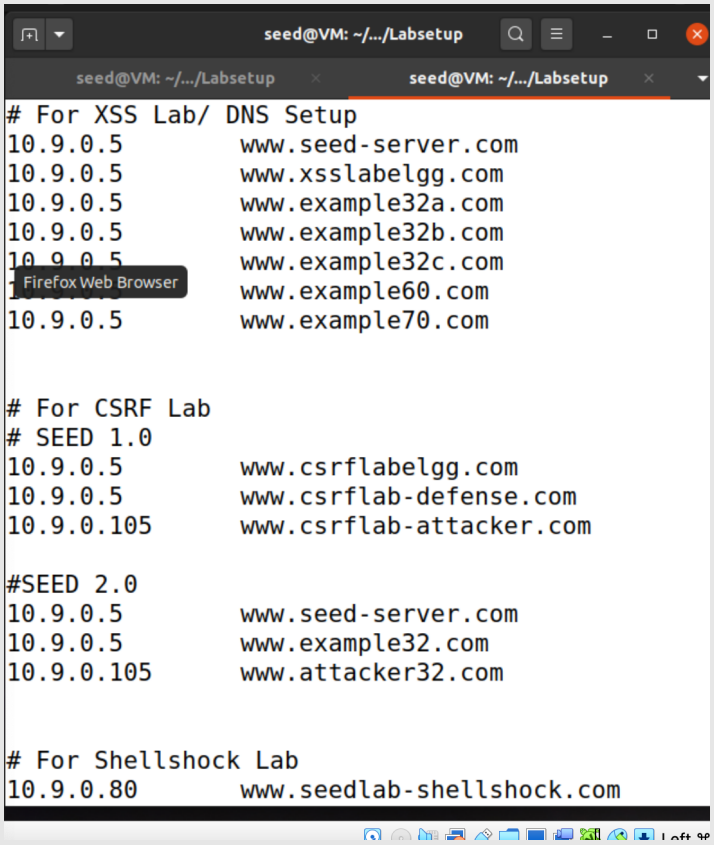


//checked

2 Lab Environment

We have developed a web application for this lab, and we use containers to set up this web application. There are two containers in the lab setup, one for hosting the web application, and the other for hosting the database for the web application. The IP address for the web application container is 10.9.0.5, and The URL for the web application is the following:

http://www.seed-server.com

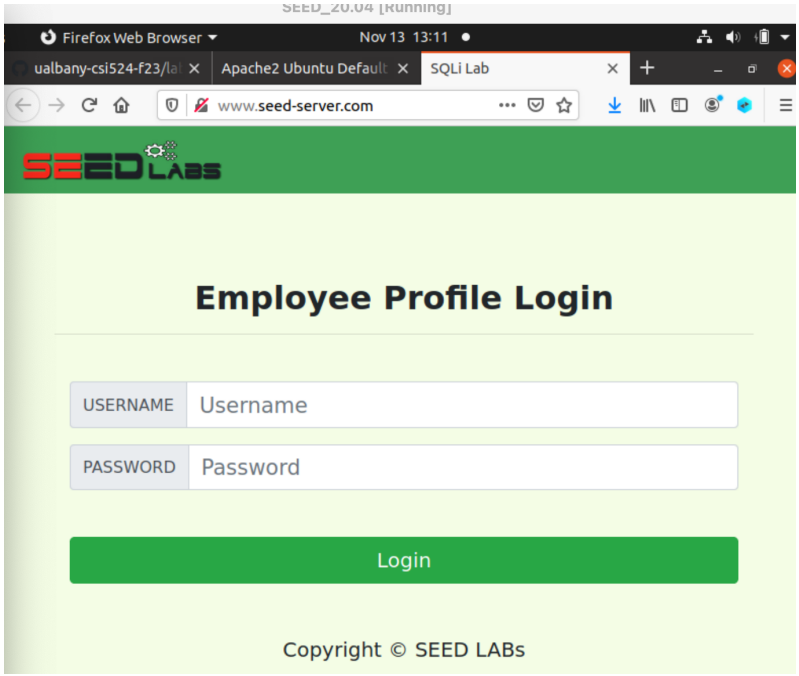


I checked the/ etc/hosts file. Above link and container ip address 10.9.0.5 was contained in the /etc/hosts file.

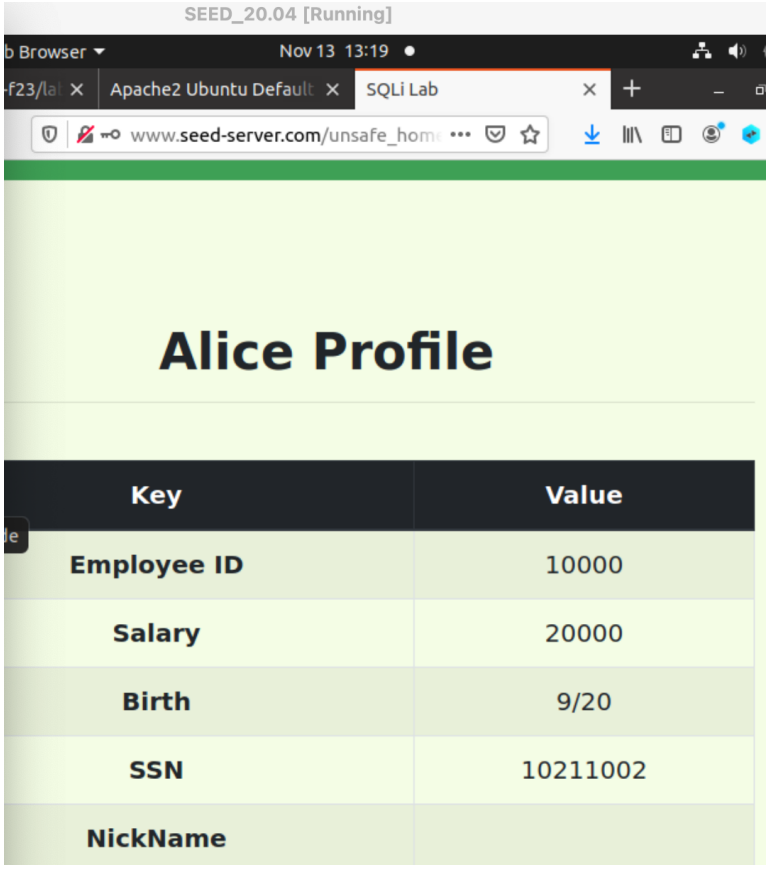
3 3.1Lab Tasks

Task 1: Get Familiar with SQL Statements

Type www.server.com and open SQLi Lab



// I visited SQLi lab website with the given address.



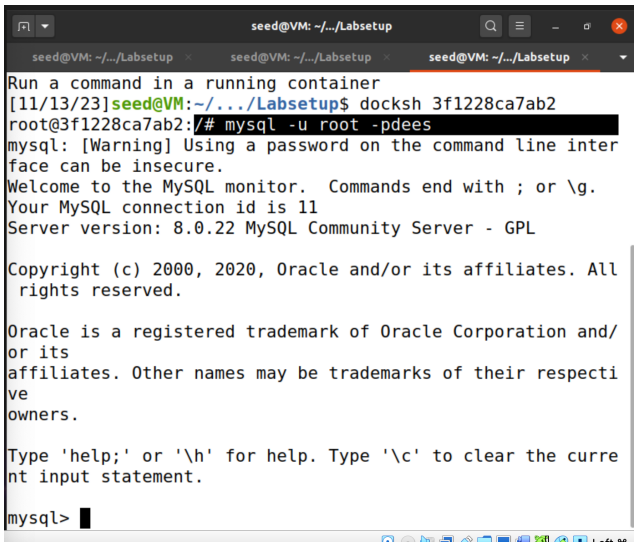
I can login as Alice with a given password and id. This shows that I can see key and value pairs related to Alice. (This is before attacking)

```
10.9.0.105      www.attacker32.com


# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com


[11/13/23]seed@VM:~/.../Labsetup$ dockps
6c5f30f3641b  www-10.9.0.5
3f1228ca7ab2  mysql-10.9.0.6
[11/13/23]seed@VM:~/.../Labsetup$
```

// I am just checking address of www and mysql, because my ubuntu got slower so I had to restart over.

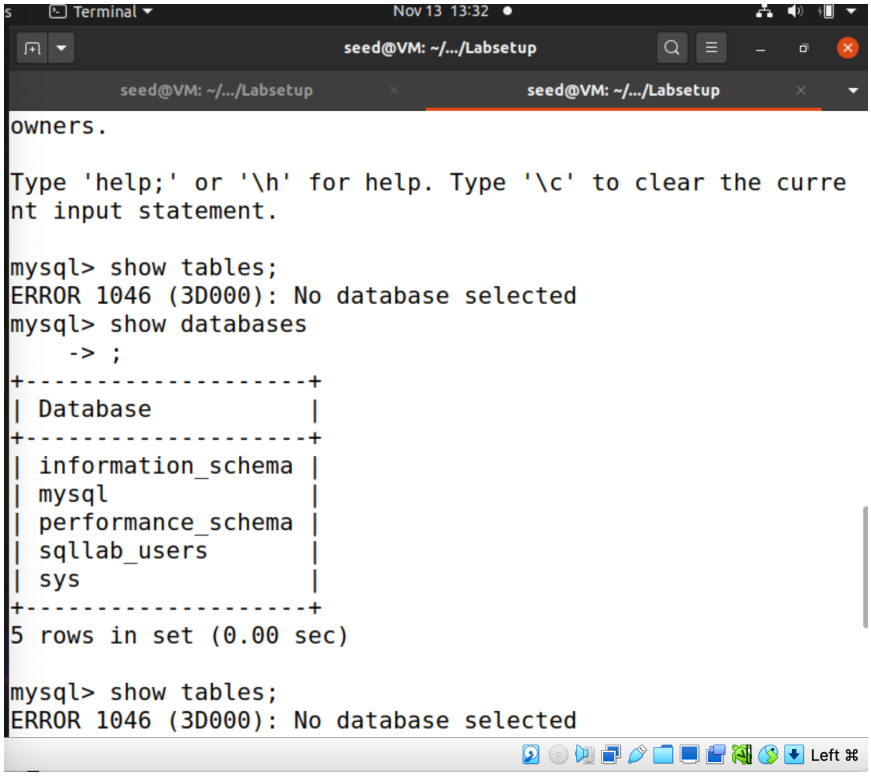
```
Run a command in a running container
[11/13/23]seed@VM:~/.../Labsetup$ docksh
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[11/13/23]seed@VM:~/.../Labsetup$ docksh 3f1228ca7ab2
root@3f1228ca7ab2:/#
```

Check shell Id using dockps and got mysql's shell Id and used docksh 3f1228ca7ab2 to go into mysql root shell.

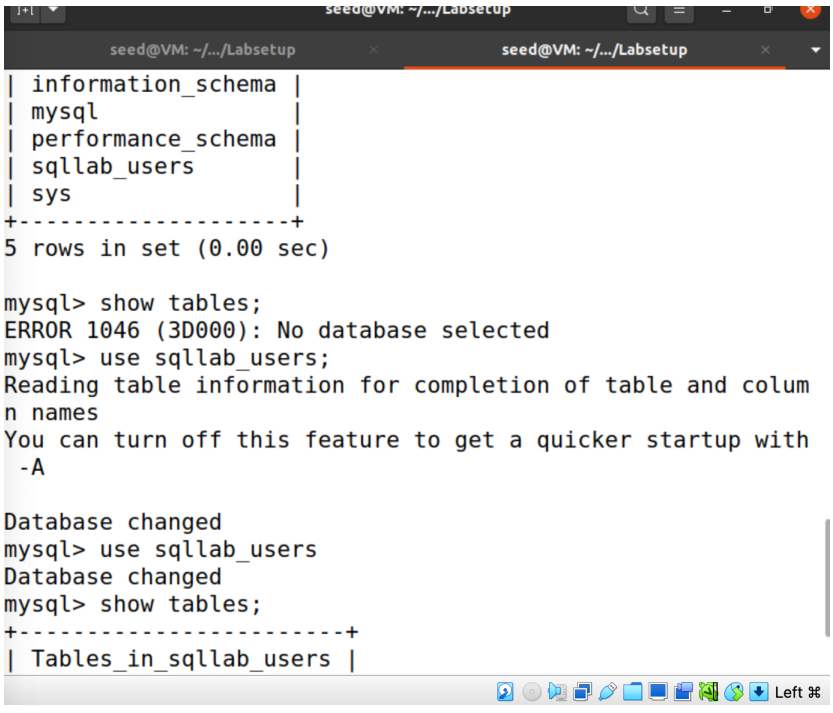


Typed given code # mysql -u root -pdees

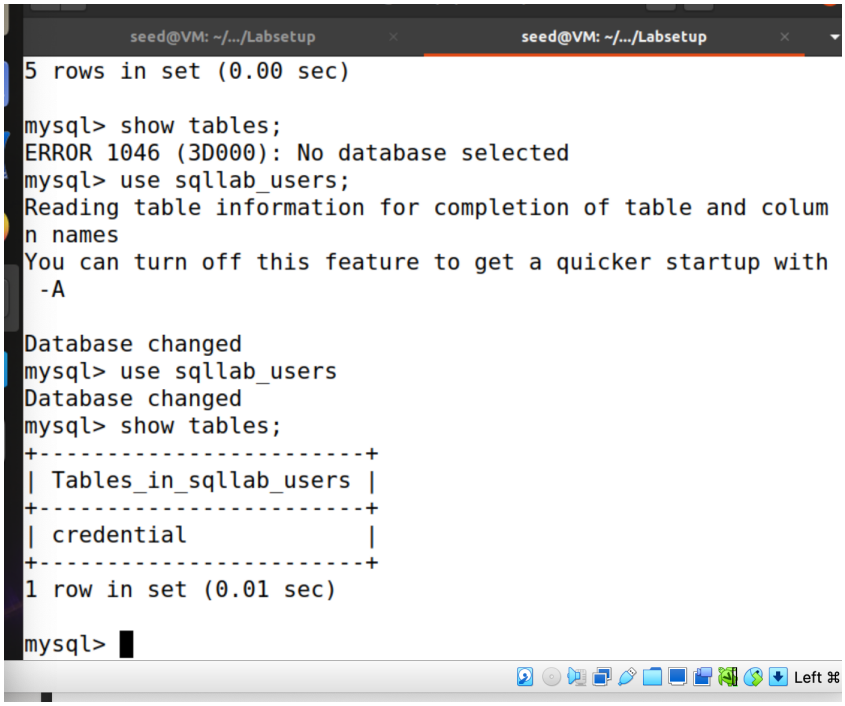
Inside the MYSQL container



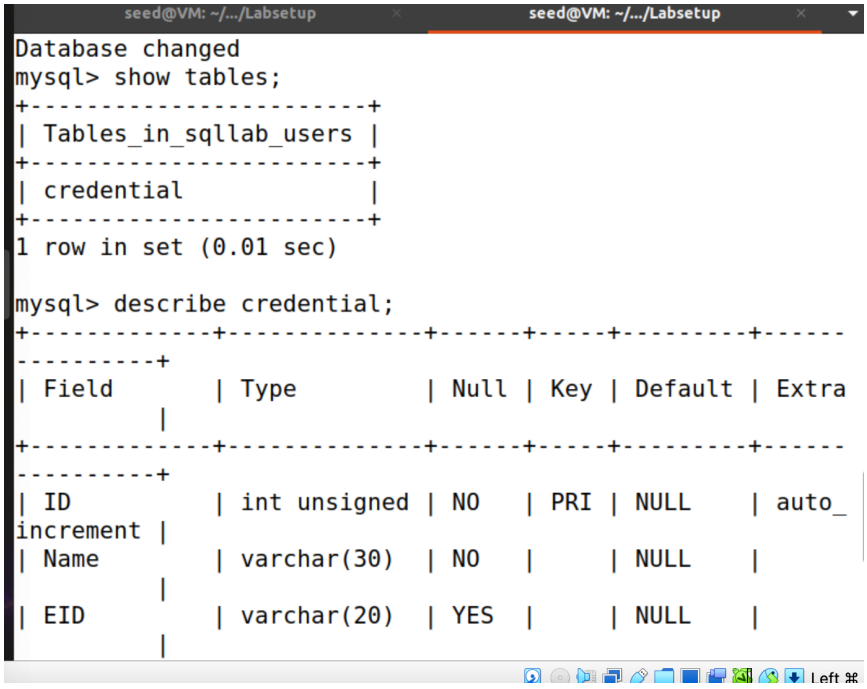
// typed show databases;
Typed use sqllab_users to select databases



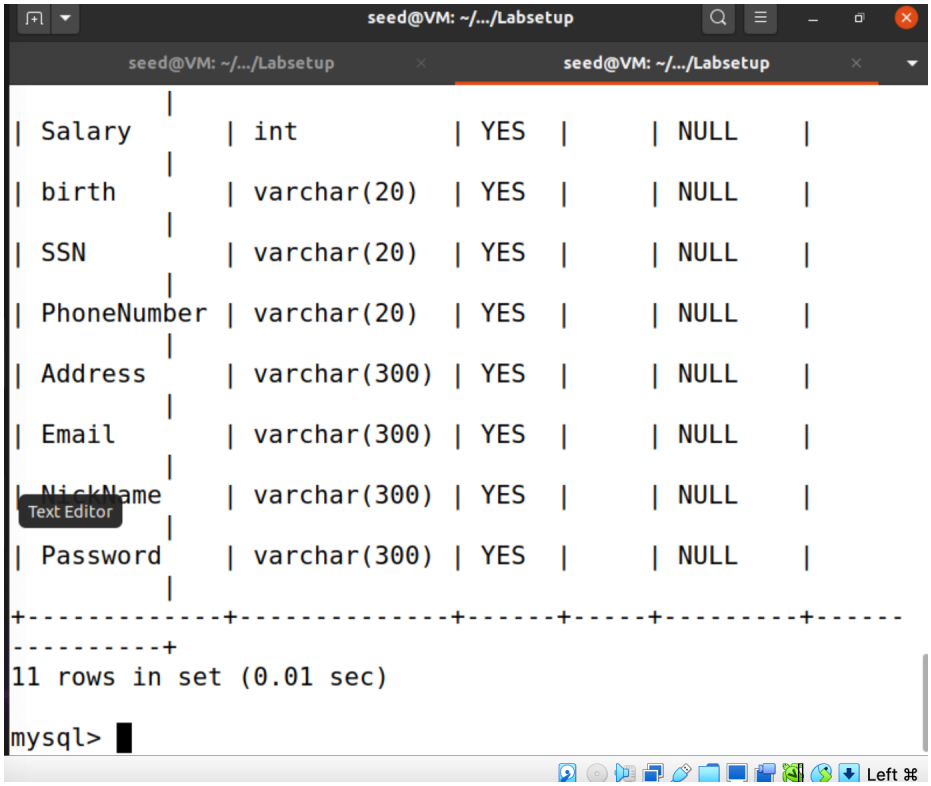
// loading existing database name sqllab_users.



Typed show tables; to see tables.

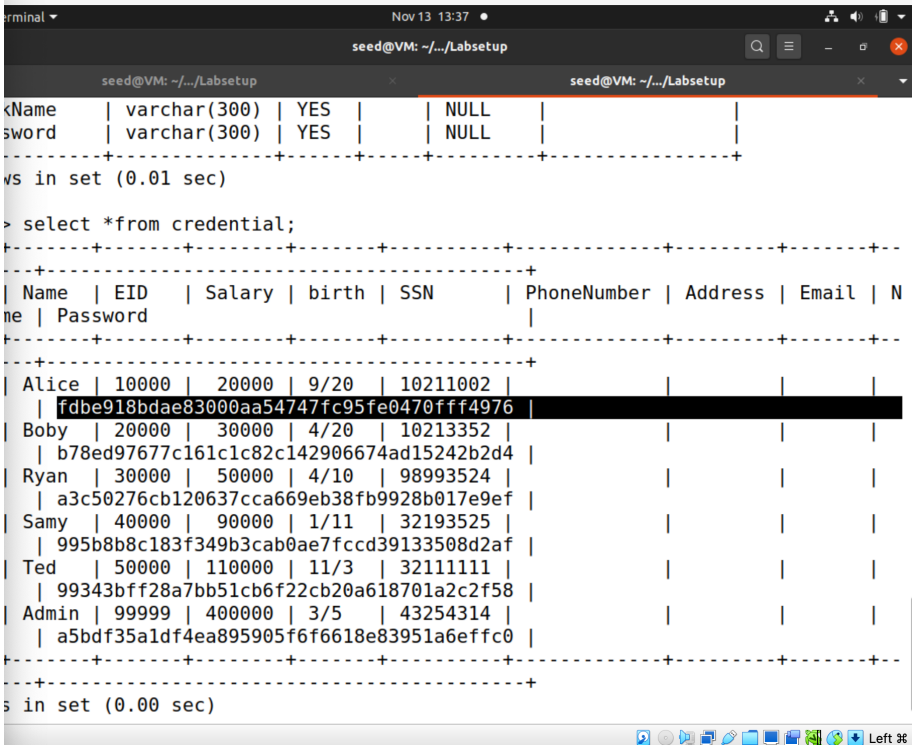


// typed show tables to see table and database name squllab_usres has one table which was credential so I typed describe credential to see the more detailed informations.



Typed describe credential -> then it will shows shemas

We can see Name, EID, salary, birth, SSN , phone number, address, email, etc,



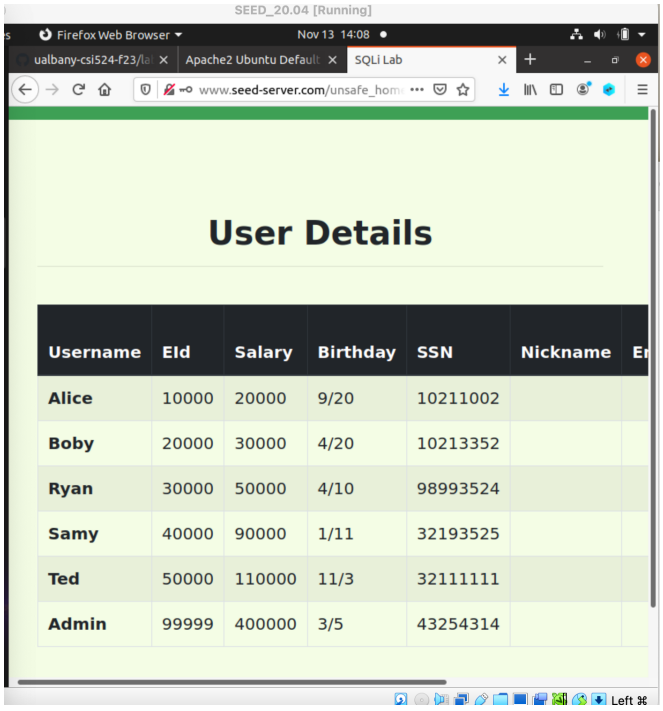
Typed select *from credential; to get hashed password ex) 99343bff28a ….

3.2 Task 2: SQL Injection Attack on SELECT Statement

In the lab setup there is a file named unsafe_home.php. If you open that file, you can see how input name and password and hashed passwords are storing. For hashed_password, it used sha1 to calculate. (detail can be seen in line 44 $hashed_pwd = sha1($input_pwd);)

hashed pwd holds the sha1 hash of the password typed by the user.

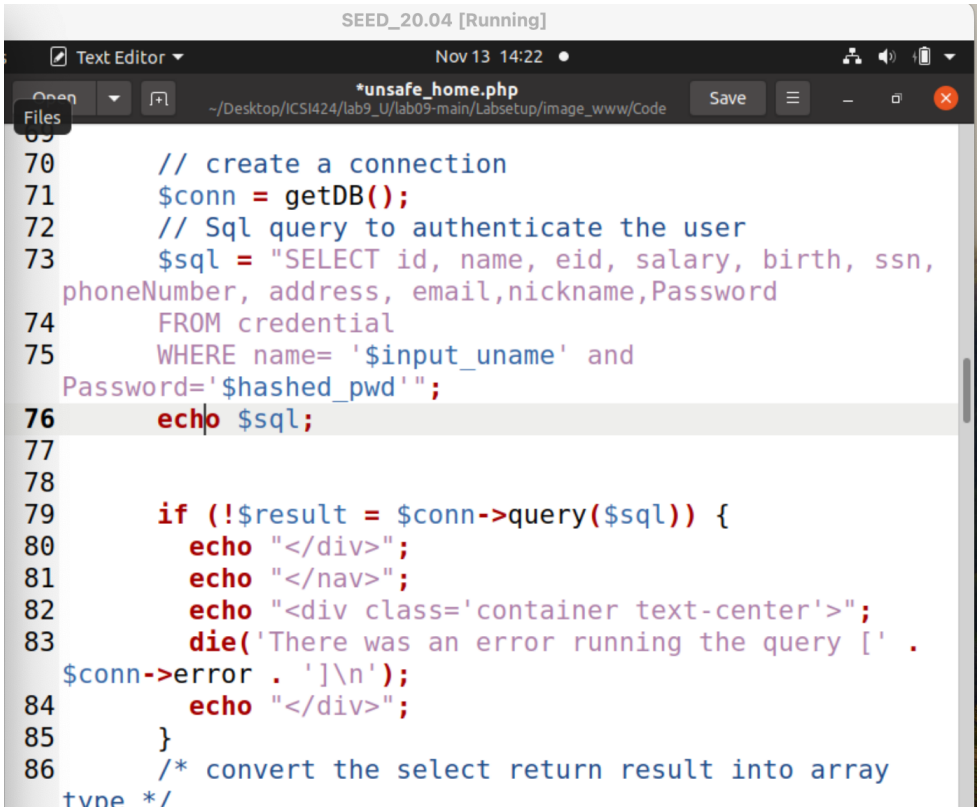
Task 2.1: SQL Injection Attack from webpage. Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator's account name which is admin, but you do not know the password. You need to decide what to type in the Username and Password fields to succeed in the attack.



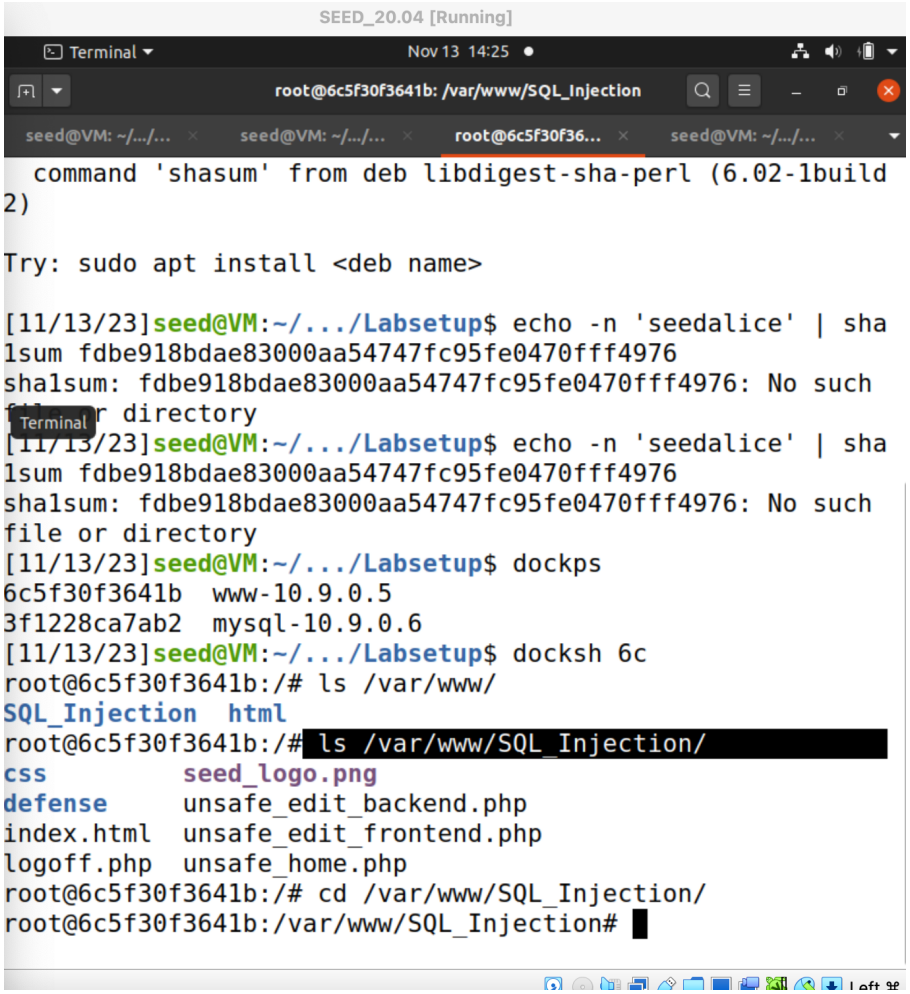
I logged in as Administer. Instead of using given admin password and name, I typed id as Admin'# and password as admin which is wrong password and name, but since I have # in id, everything after # were considered as comments and I was able to login as Admin.

I need single code to match with other single quote and this attack was worked with any kinds of password since everything after '# were considered as comments.

// below is just showing I interporated unsafe_home.php to observe what kinds of datas are stored in mysql database.

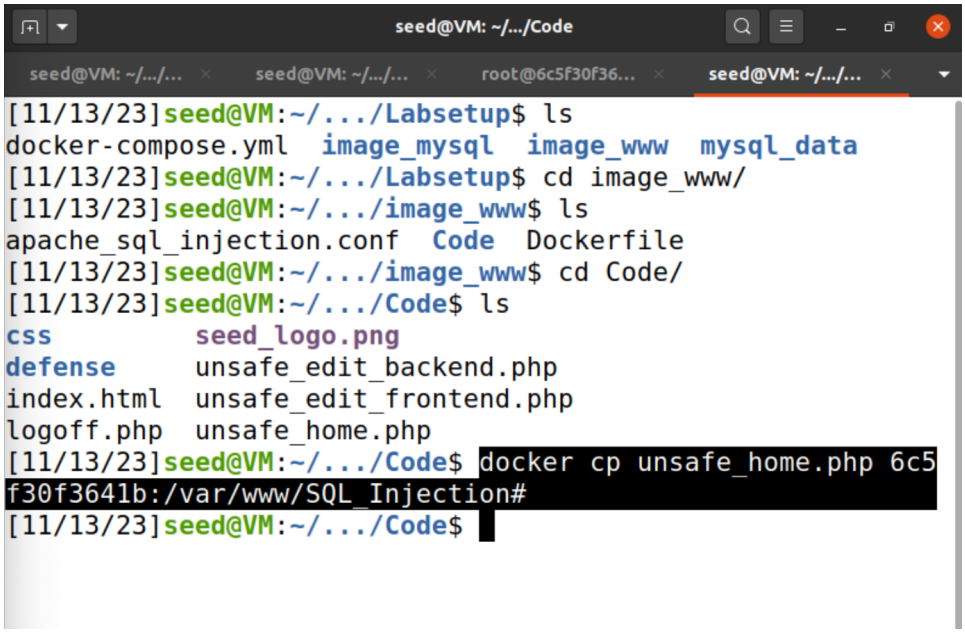


After line 75, typed echo $sql; to print out sql; I don't think this is necessary to do but when I was not succeeded in launching attack, those sql message was helpful for me to understand what is currently stored in sql and what is hashed password value.

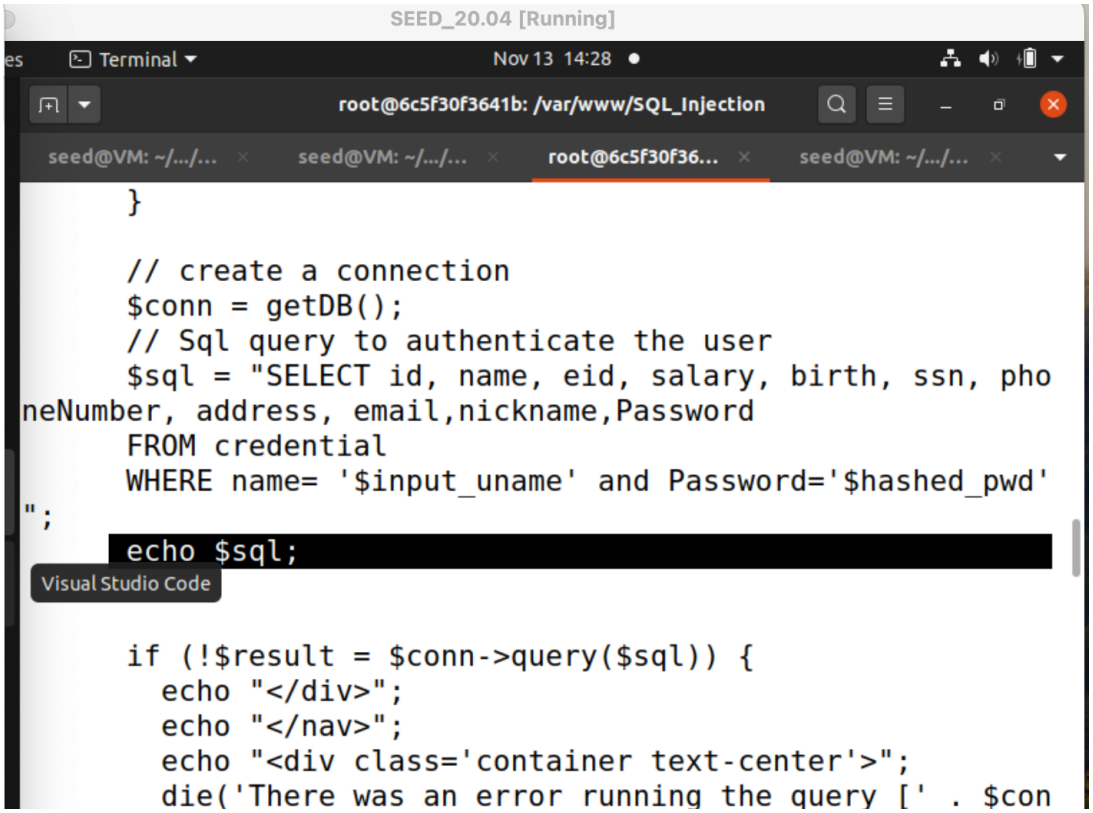


// go to ls /var/www/SQL_Injection/ folder and do cd /var/www/SQL_INjection/ to get the full path to the SQL_Injection folder. Which was 6c5f30f3641b:/var/www/SQL_Injection#

And opened new top terminal



// typed ls and went to image_ww and went to Code and copied changed unsafe_home.php into SQL_Injection path. IT is just simply overwriting modified unsafe_home.php



// I used cat unsafe_home.php to check the added line “echo $sql; “ is updated in unsafe_home.php. It was updated.

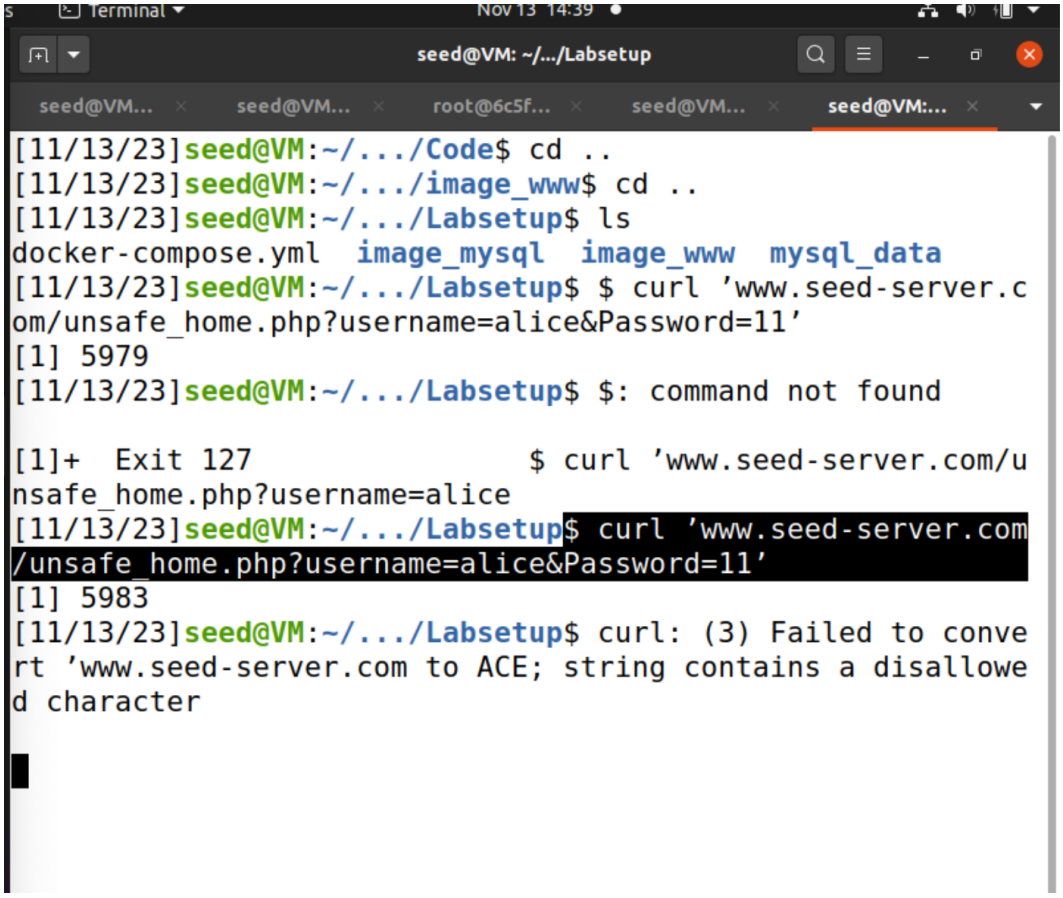
The input here for username results in the following query at the server to be executed:

SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password

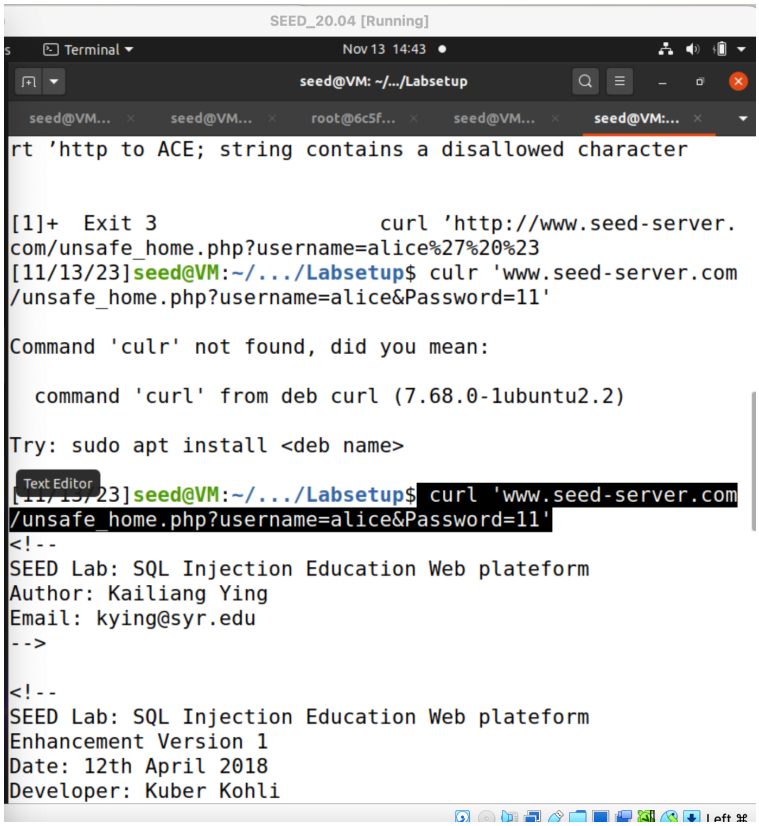
FROM credential

WHERE name= ‘admin’

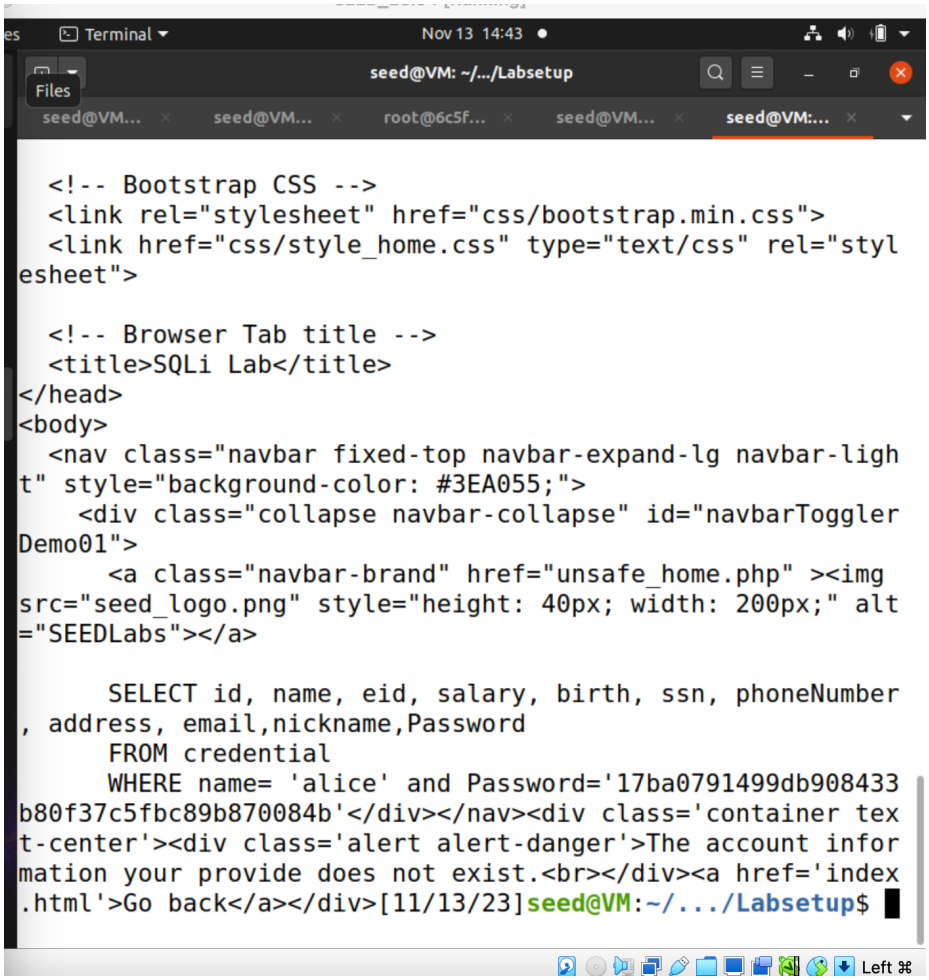
Task 2.2: SQL Injection Attack from command line.



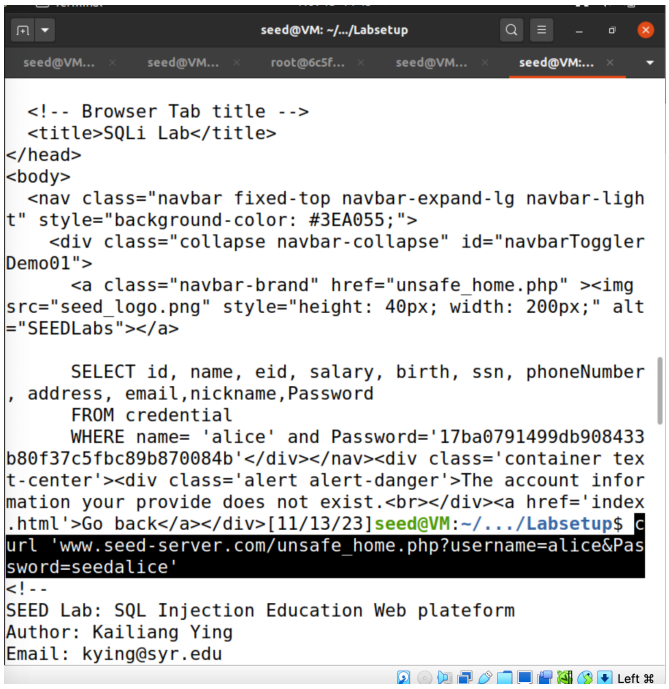
// type given curl .. command line (when I copy and pasted above line from instruction it did not work)



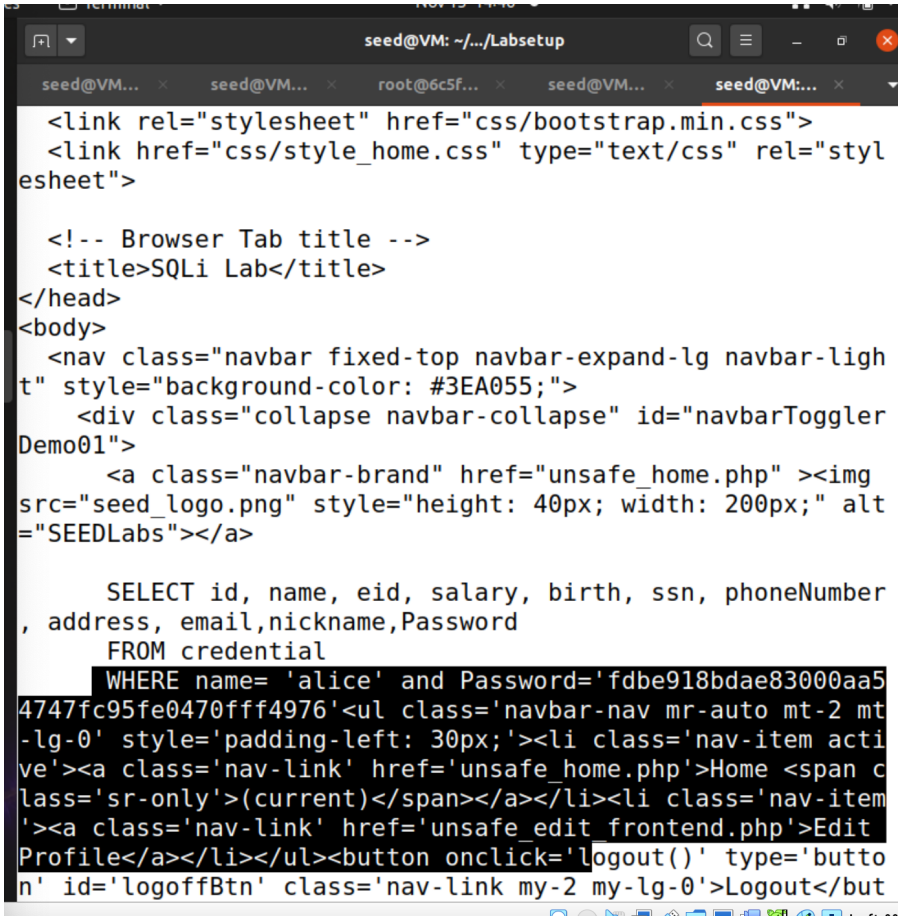
// used curl and given username and password pair. Notice Alice's password is incorrect password in this case.



Above WHERE name=’alice’ and Password=’17….’ is the wrong password for alice since we type ALice=11’ (on command line) if we type correct name and password for alice, we will get all informations of Alice.



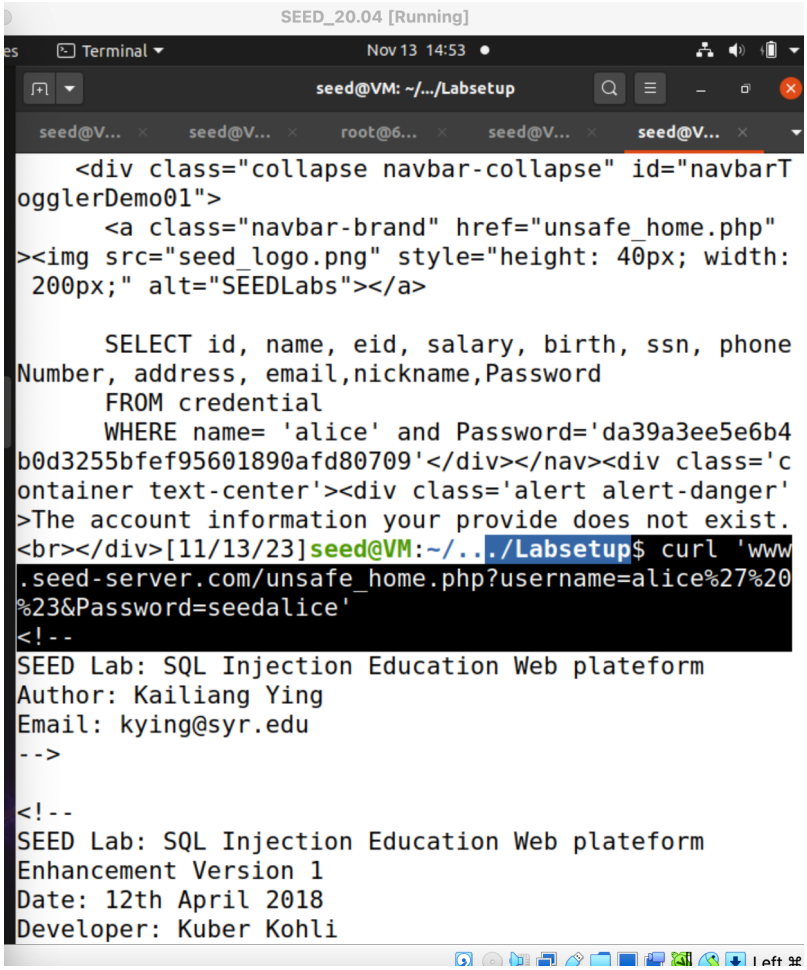
I typed curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'



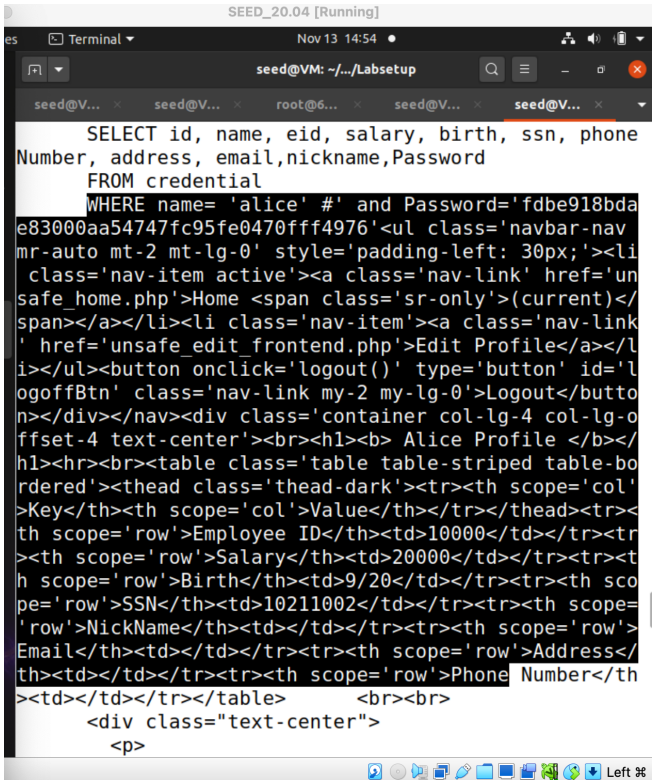
// I got Alice's real password information from credentials.

It shows her hash code password, birth and salary.

Since ‘ # is %27%20%23 if we encode (simply you can type ‘ # (make sure there is space) in encoder you get those %27%20%23 value)

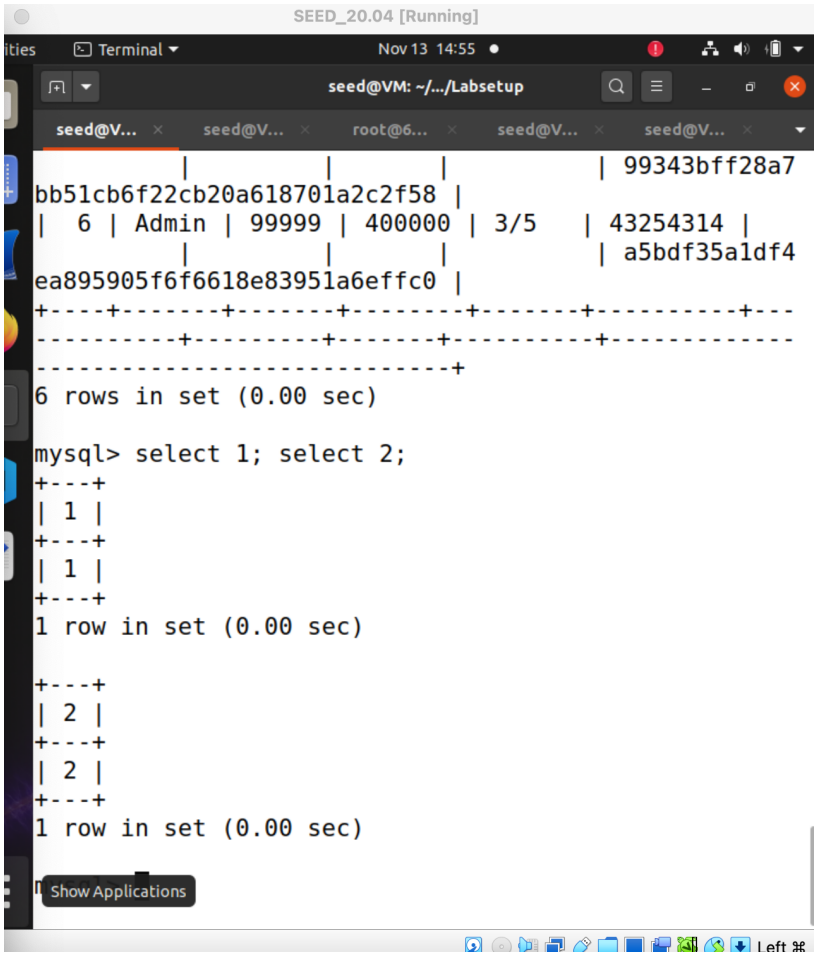


// this is how I typed values with curl. In this case I used encoded value %27%20%23 instead of using ‘ #

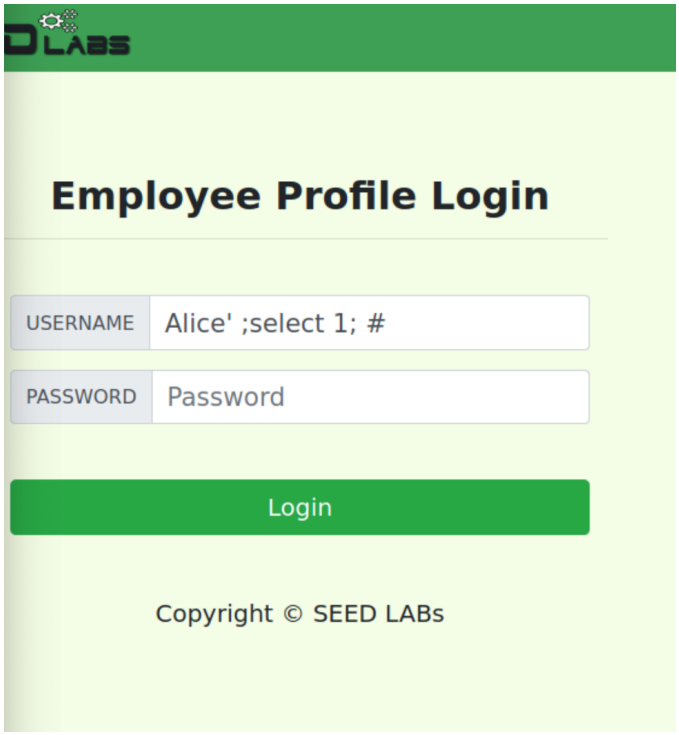


// this shows that I got Alice's credential information. I was able to get name, hashed password, her salary, birth, email, and nickname.

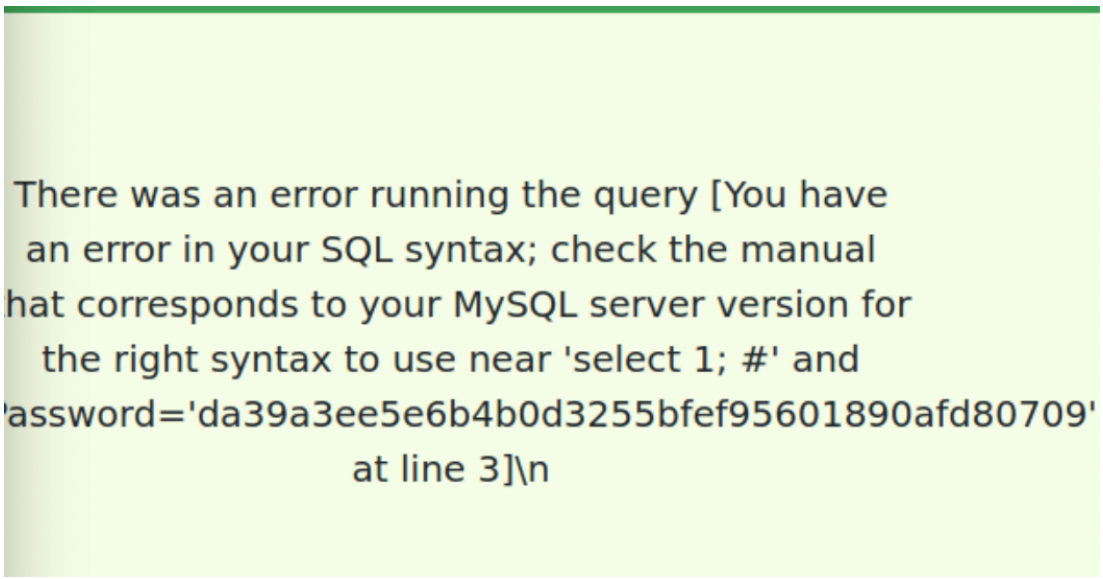
Task 2.3: Append a new SQL statement



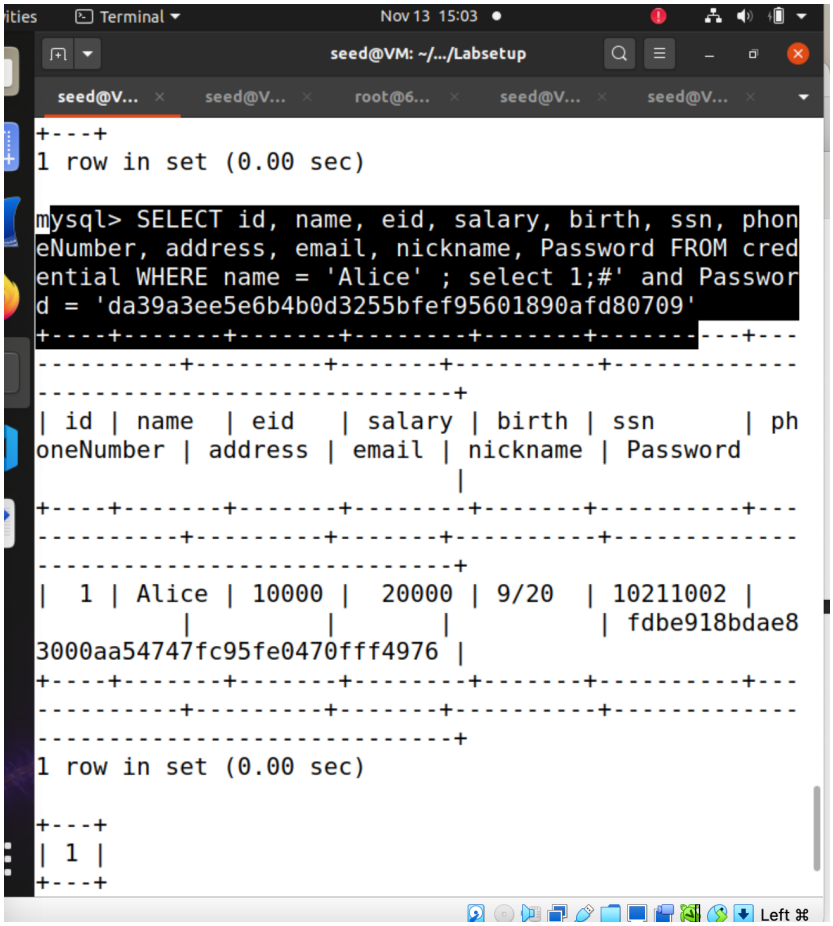
// In sql, we can select table using “select 1;” if we want to select 1 and 2 we can type them in same line by distinguish them using ;



By using ; we can separated Alice with # (use Alice' ; #)
I typed Alice' ; select; # s username and typed nothing in password.

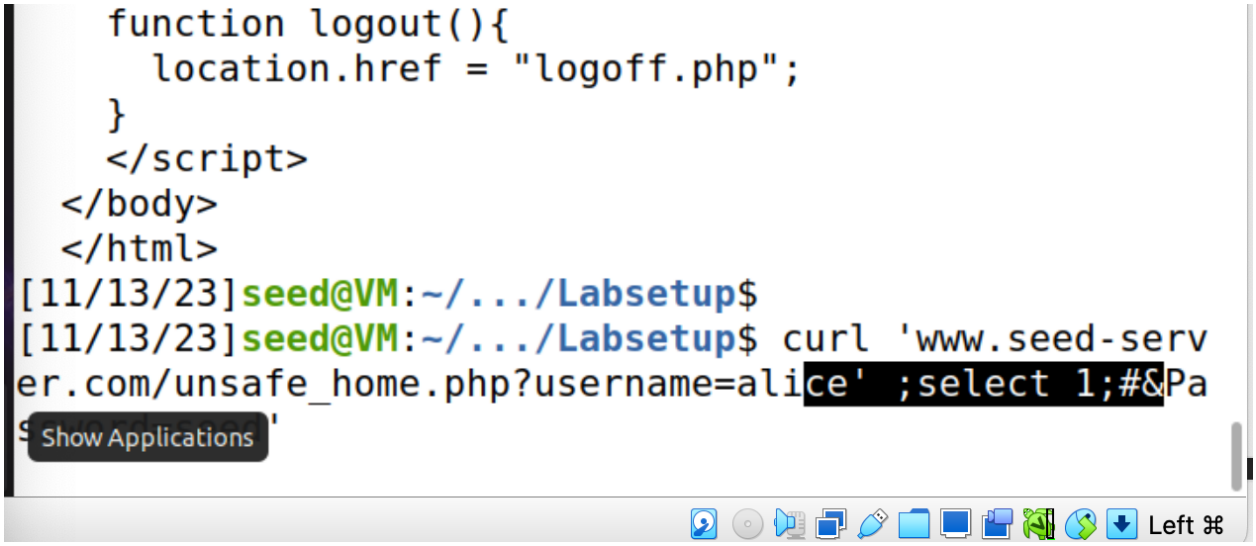


// I got this output which shows that I got error in my query and displayed hashed password was also wrong value, therefore I am going to launch attack again.

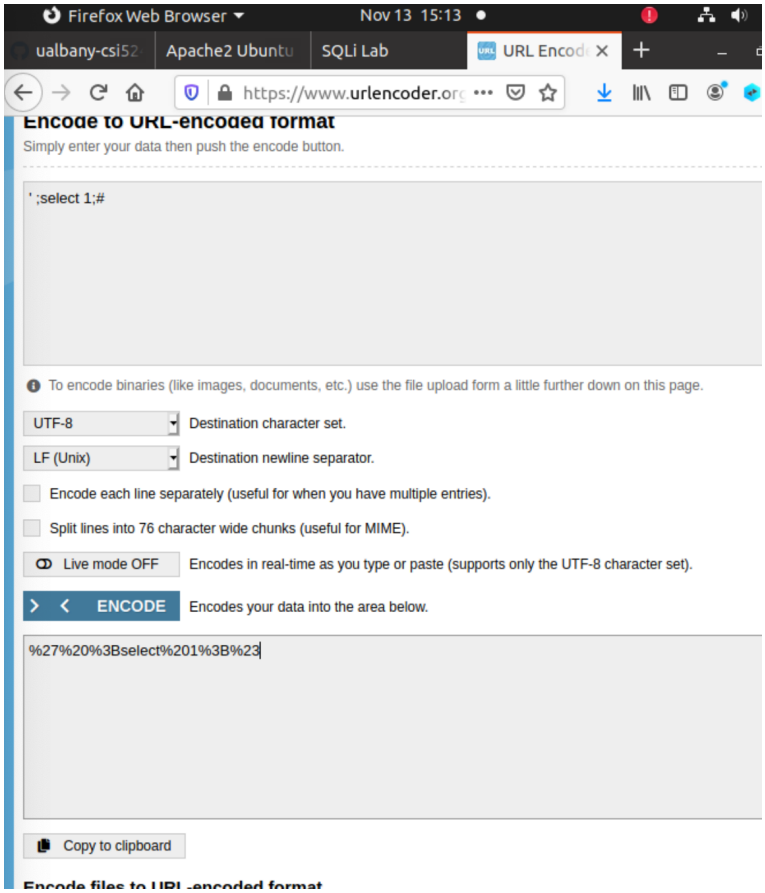


// in mysql I selected id name eid salary birth ssn and phone number, address email nickname password from credential where name should be alice and select1 and password is what seed-erver.com displayed (hashed password)

This returned credential values of Alice include her hashed password “ fdbe918… “
Since we have ;# so the password part of command line will be considered as command line.



// this time we remove encoding part %27%29%23 and typed ‘ ;select 1;# and copy that to encoder to get the encoded value with %.

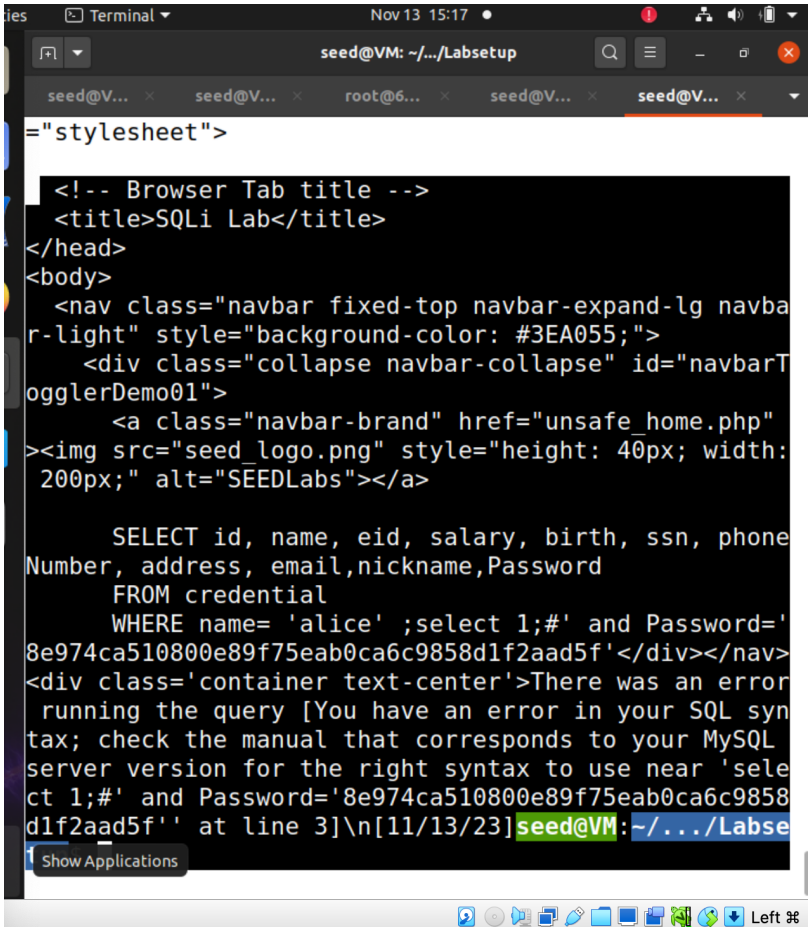


// ‘ ;select 1;# encoded value was %27%20%3Bselect%201%3B%23
(# is %23)

```
 running the query [You have an error in your SQL syn
tax; check the manual that corresponds to your MySQL
server version for the right syntax to use near 'sele
ct 1;#' and Password='92713d4709377111cf31f2a71986c41
1bd6cb5b0'' at line 3]\n[11/13/23]seed@VM:~/.../Labse
tup$ curl 'www.seed-server.com/unsafe_home.php?userna
me=alice%27%20%3Bselect%201%3B%23&Password=seedalice'
```

Left ⌘

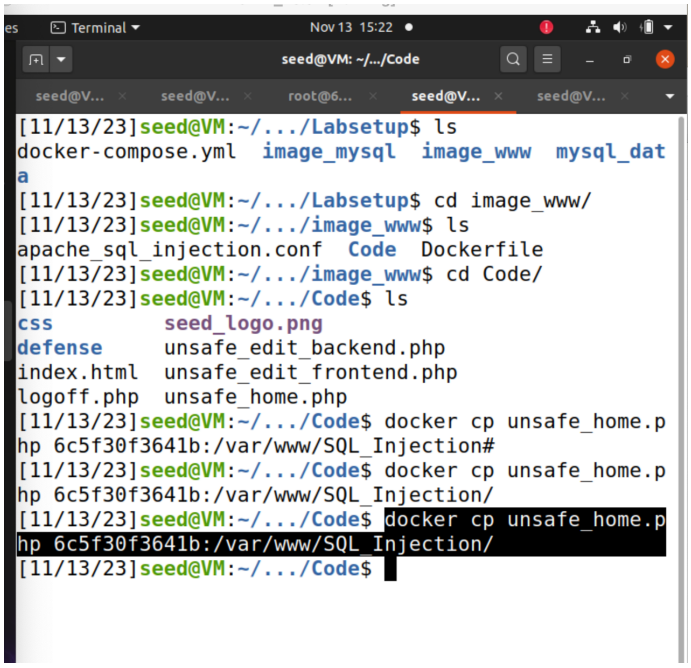
// I typed curl with encoded ‘ select1 ;# and passed valid username as alice and alice’s correct password(this was given in instruction.)



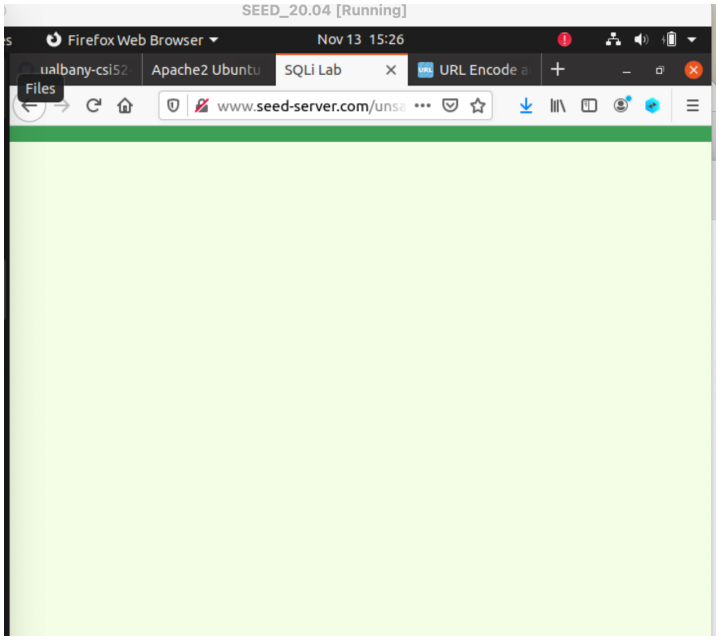
// This is the output of the command line. It seems I got an error so I will try again.

```
//
78
79        if (!$result = $conn->mlti_query($sql)) {
80            echo "</div>";
81            echo "</nav>";
82            echo "<div class='container text-
  center'>";
83            die('There was an error running the
  query [' . $conn->error . ']\n');
84            echo "</div>";
```

// IN unsafe_home.php, I changed query($sql) into mlti_quer($sql) so we can take multiple queries because when I typed ‘ select1; # , it displayed I have some error (you can see above output).



// also make sure to updated changed unsafe_home.php by using “ docker cp unsafe_home.php // some path “ -> I can check update using cat unsafe_home.php



// I was successful to launching attack, since error message is not displaying on top.

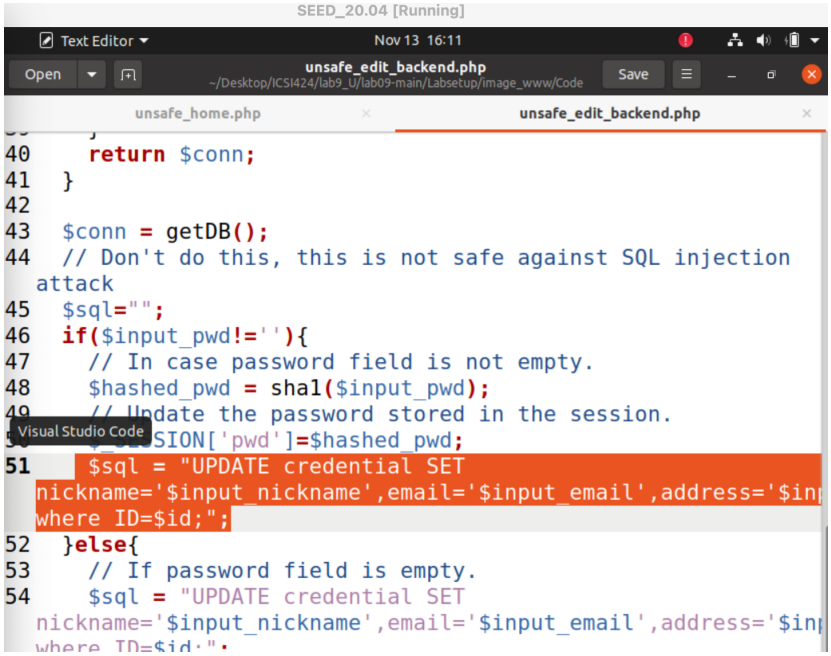
```
76        echo $sql;
77
78
79        if (!$result = $conn->mlti_query($sql)) {
80          echo "</div>";
81          echo "</nav>";
82          echo "<div class='container text-
Text Editor er'>";
83          die('There was an error running the
    query [' . $conn->error . ']\n');
84          echo "</div>";
85        }
86        /* convert the select return result into
    array type */
87        $return_arr = array();
88        while($row = $result->fetch_assoc()){
89          array_push($return_arr,$row);
90        }
91
92        /* convert the array type to json format
```

// I logged in again in the lab and it didn't show an error message so the attack succeeded.
If the attack failed this message should show up " die('There was an error running the query [' . $conn->error . ']\n');
// so I was able to append new attack using ;

3.3 Task 3: SQL Injection Attack on UPDATE Statement

Task 3.1: Modify your own salary.

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



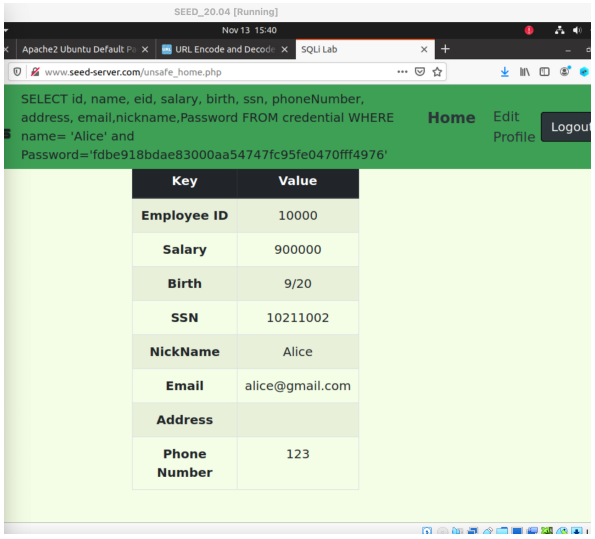
THis line 51 and 52 in unsafe_edit_backend.php file shows how they update sql values for credential Set, This code shows that they take nickname, email, address, input email, hashed_pwd and phoneNumber. Notice there is no password. (it is good for security)

In line 57 in unsafe_edit-backend.php I add a line echo 'SQL : '.$sql; to see

The SQL values. And saved it. And update the changes in unsafe_edit_backend.php using docker cp command line.

```
f3641b:/var/www/SQL_Injection/
[11/13/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 6c5f30
f3641b:/var/www/SQL_Injection/
[11/13/23]seed@VM:~/.../Code$ docker cp unsafe_edit_backend.ph
p 6c5f30f3641b:/var/www/SQL_Injection/
[11/13/23]seed@VM:~/.../Code$
```

// simply I am updating the changed unsafe_edit_backend.php file in SQL_Injection folder.



Logged in as Alice and go to Alice's profile Edit and changed information and in phone number section I typed 1234', salary=90000 WHERE name='Alice' #

So if user name is Alice her salary will be increased as 90000 and since we have # at the end the rest of the portion (password) will be considered as password.

Aftering saving changes Alice's salary was updated.

This is because query on web server changed to UPDATE credential SET

nickname=’ALice’,

email=’alice@gmail.com’,

address=’’

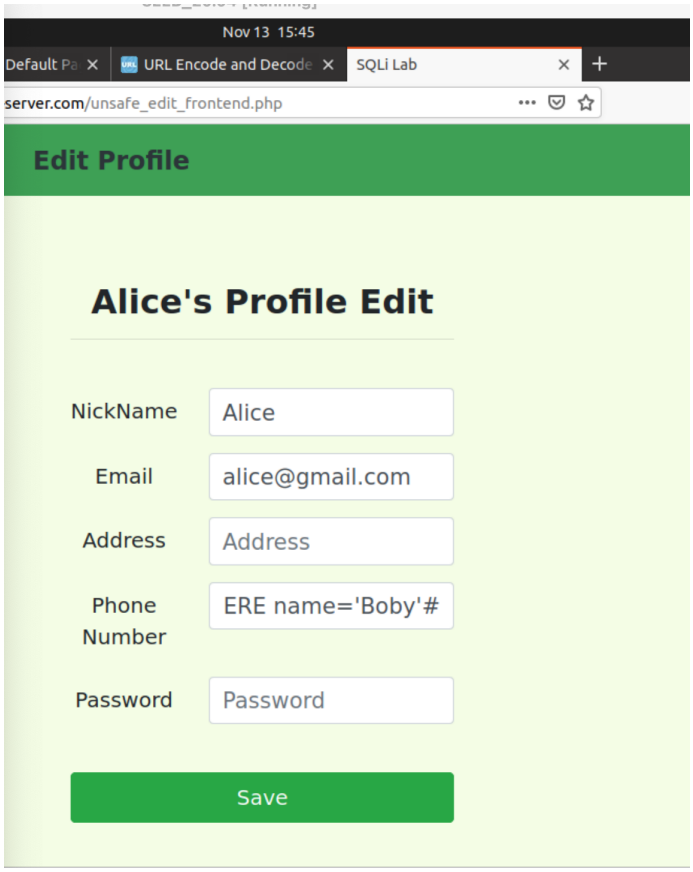
Password=’’,

PhoneNumber’1234’,salary=90000 WHERE name= ‘Alice’

(Important part is that when I add those lines there were no salary section so I typed informations to change salary in PhoneNumber updating section.)

Task 3.2: Modify other people’ salary. After increasing your own salary, you decide to punish your boss

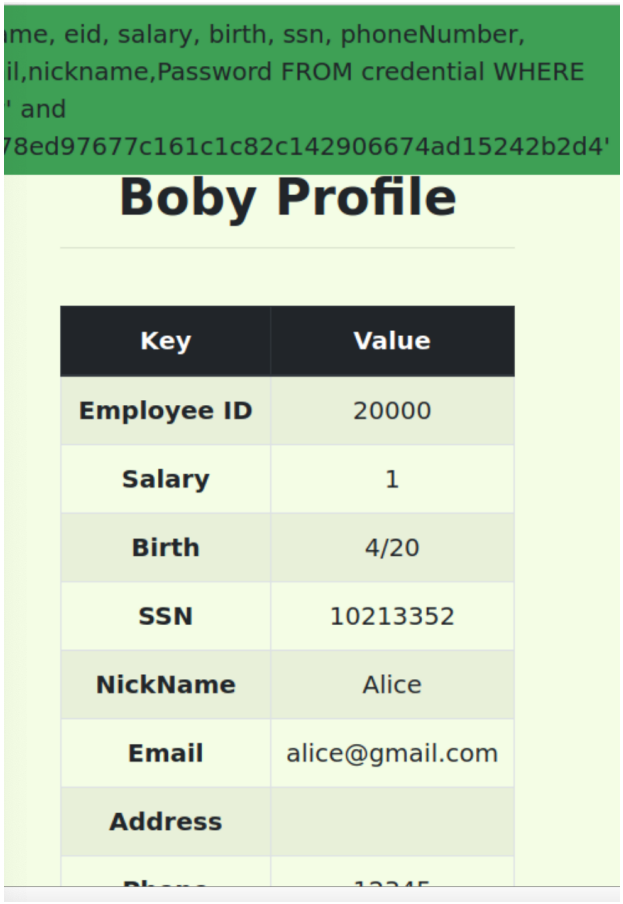
Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.



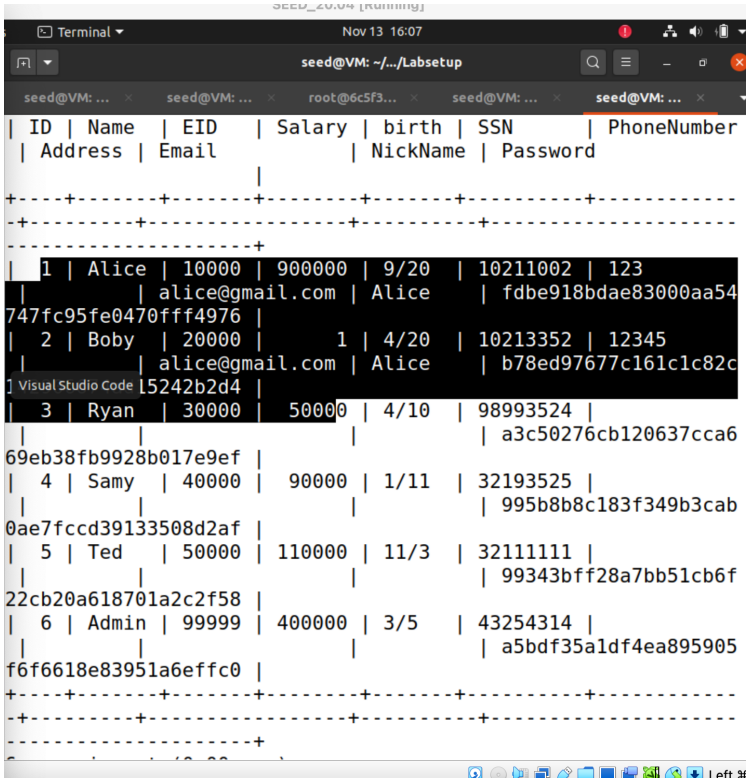
I can change BOb’s salary in ALice’s profile Edit since all hash passwords and this SQL query will be saved based on name and its credential information.

So in alice’s profile edit session, I tyed 1234’ ,salary=1 WHERE name=’Boby’#

So first 1234 is just random number for phoneNumber and I set salary as 1 if the name value is matches with Boby and I putted # so everything after Boby is considered as comment line, so even though Alice does not know Boby’s password she can modify his salary.

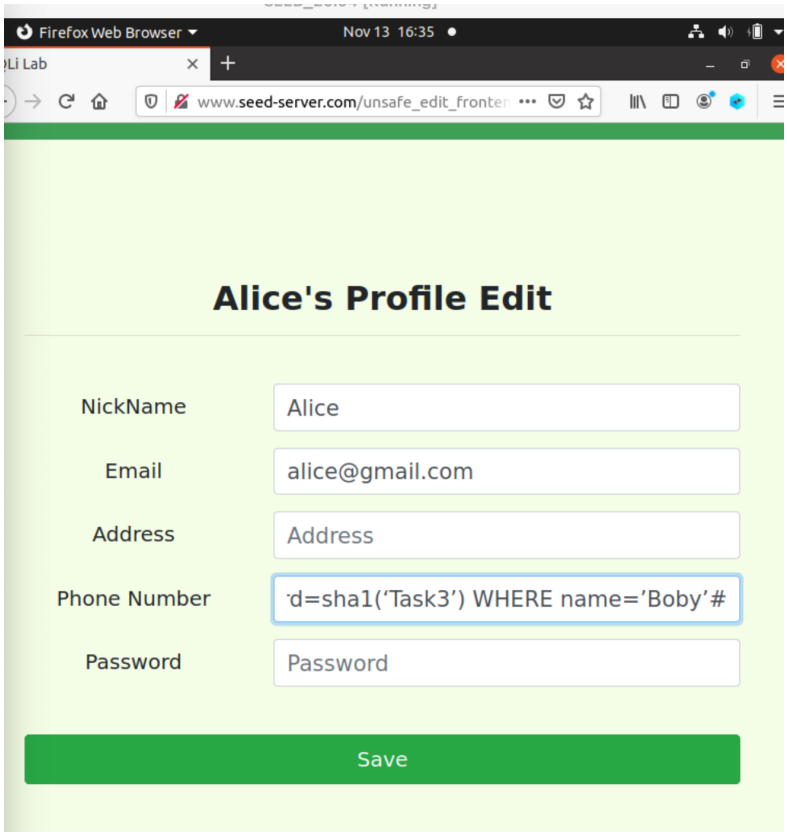


After saving changes , I checked BOby's salary. We can see that Boby's salary changed to 1 and has Alice Nickname and email address. This show that we can change Boby's salary without knowing his credential information such as password.

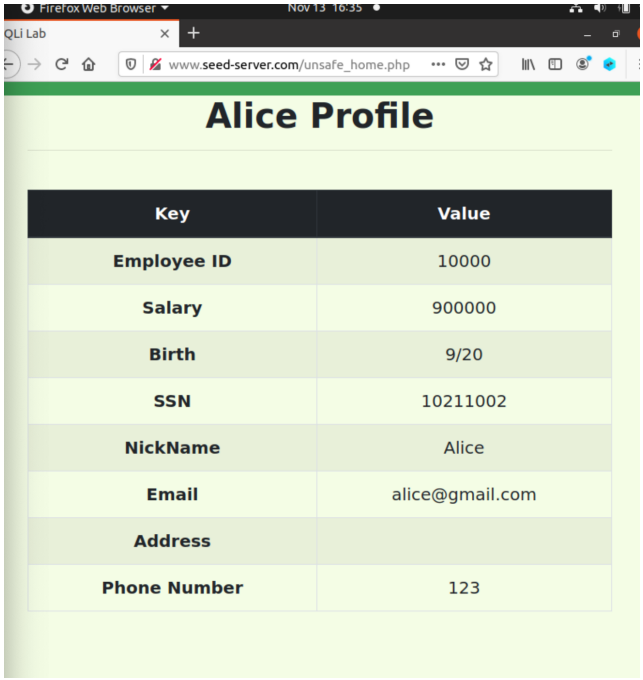


For upgrading Alice's salary and decrease Bob's salary we can see that those changed salary data values are stored in MYSQL

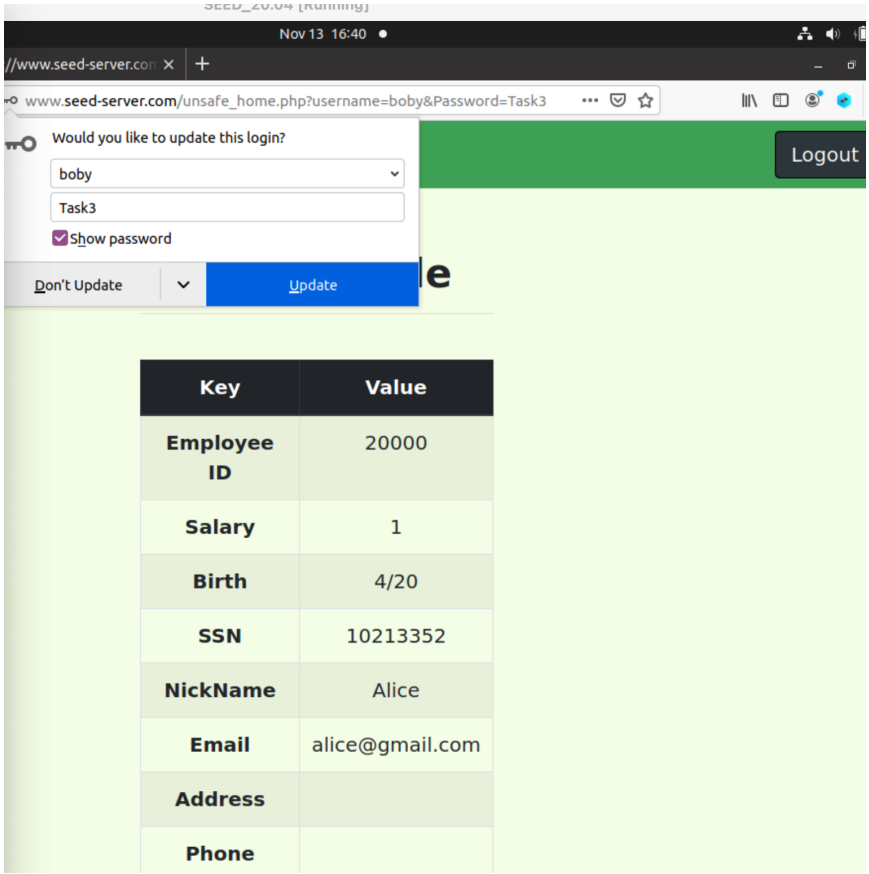
Task 3.3: Modify other people' password.



I loge in as Alice and go to alice's profile edit
Type following in phoneNumber section: 1234', Password=sha1('Task3') WHERE name='Boby'#



Alice profile after saving we can only see phone number 123. Even though I typed password values in phone number value section, only phone number digits are displayed and rest of informations were stored in database so, it will eventually affect to Boby's password.

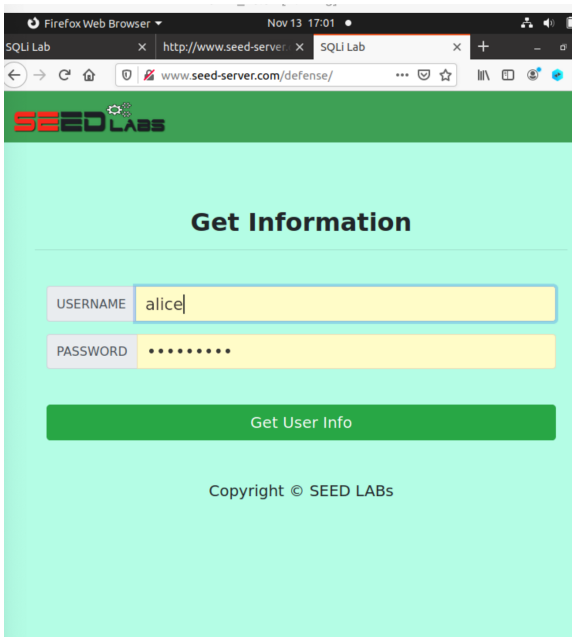


// and I used Boby as username and used Task3 as password (which I changed using sha1() in Alice's profile, since Alice does not know boby's seedboby password) THis shows that Alice was successfully able to modify Boby's password and able to access his profile.

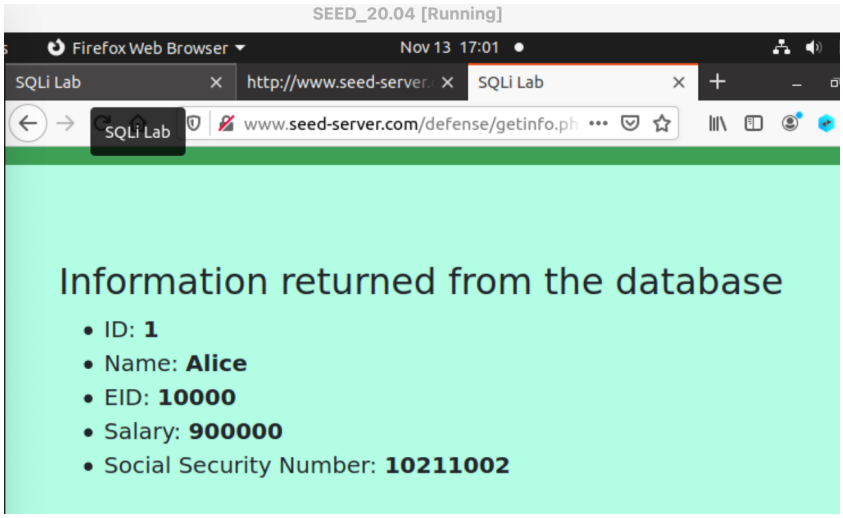
3.4 Task 4: Countermeasure — Prepared Statement

This is part of the instructions of lab9 but I think it is really important and helpful for me to study so I just added it here.

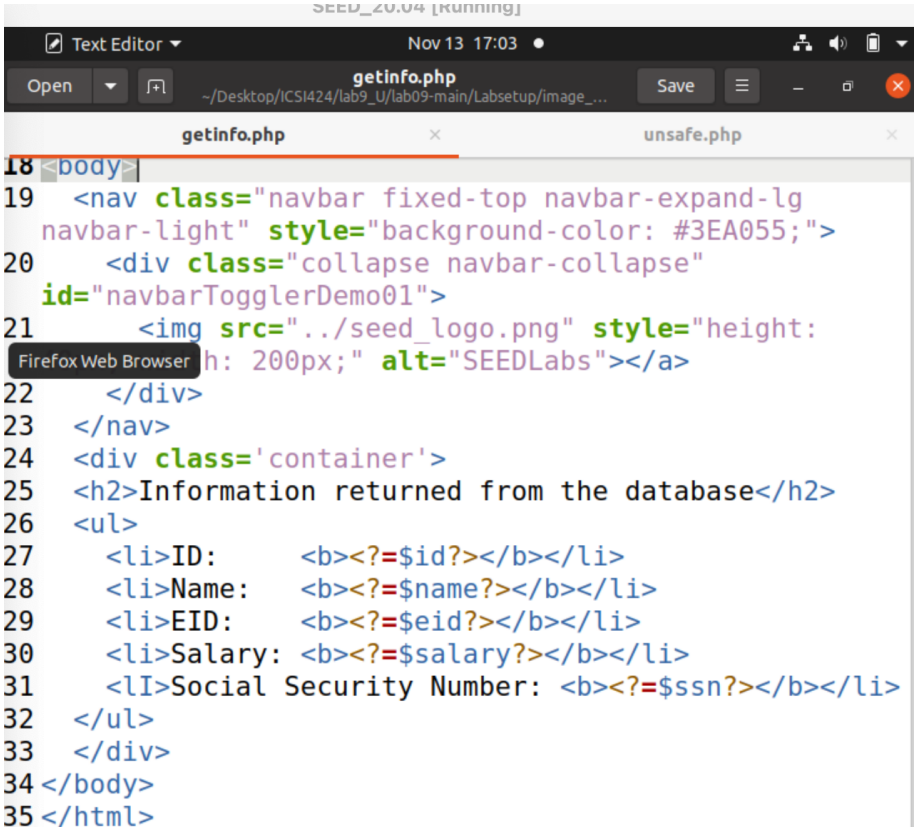
Prepared statement comes into the picture after the compilation but before the execution step. A pre- pared statement will go through the compilation step, and be turned into a pre-compiled query with empty placeholders for data. To run this pre-compiled query, data needs to be provided, but this data will not go through the compilation step; instead, they are plugged directly into the pre-compiled query, and are 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 a prepared statement prevents SQL injection attacks.



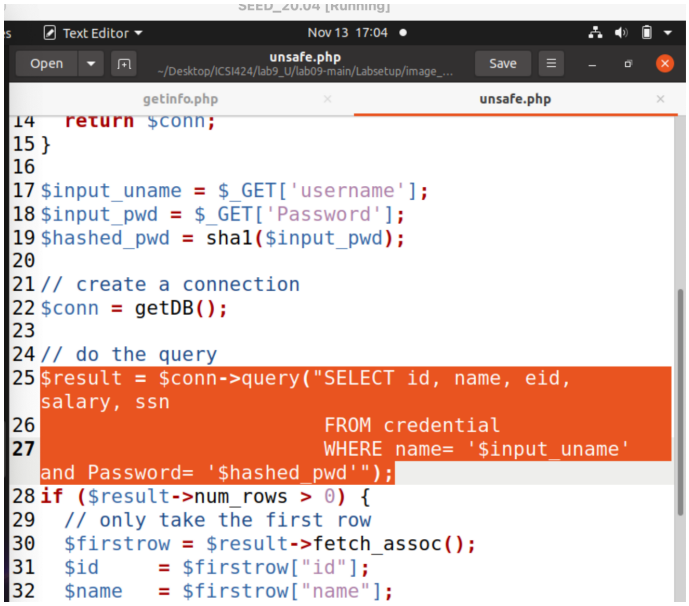
// I go to defense url and login as Alice



// We can see Alice's database. This is before I change unsafe.php to create countermeasures. You can see that Alice's salary is as high as I created. Goal is we don't want users to modify their inputs in edit profiles and change sql databases.

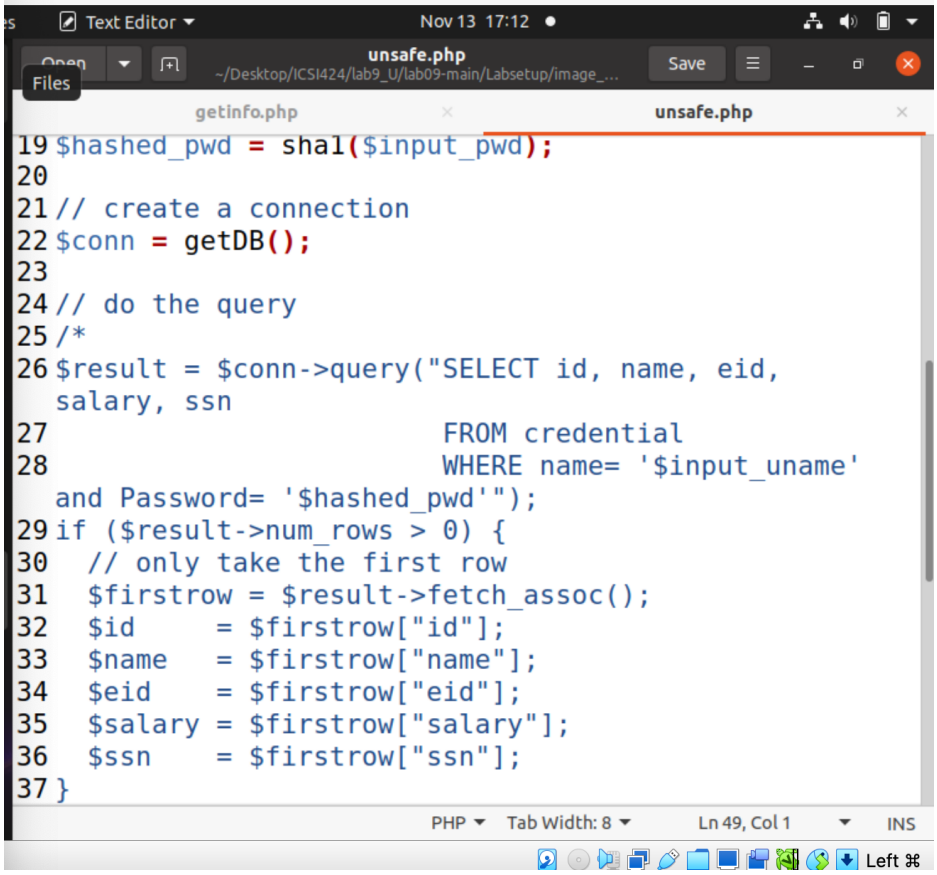


// open labsetup/image_www/code/defense folder and open getinfo.php and unsafe.php This is how getinfo.php looks like.

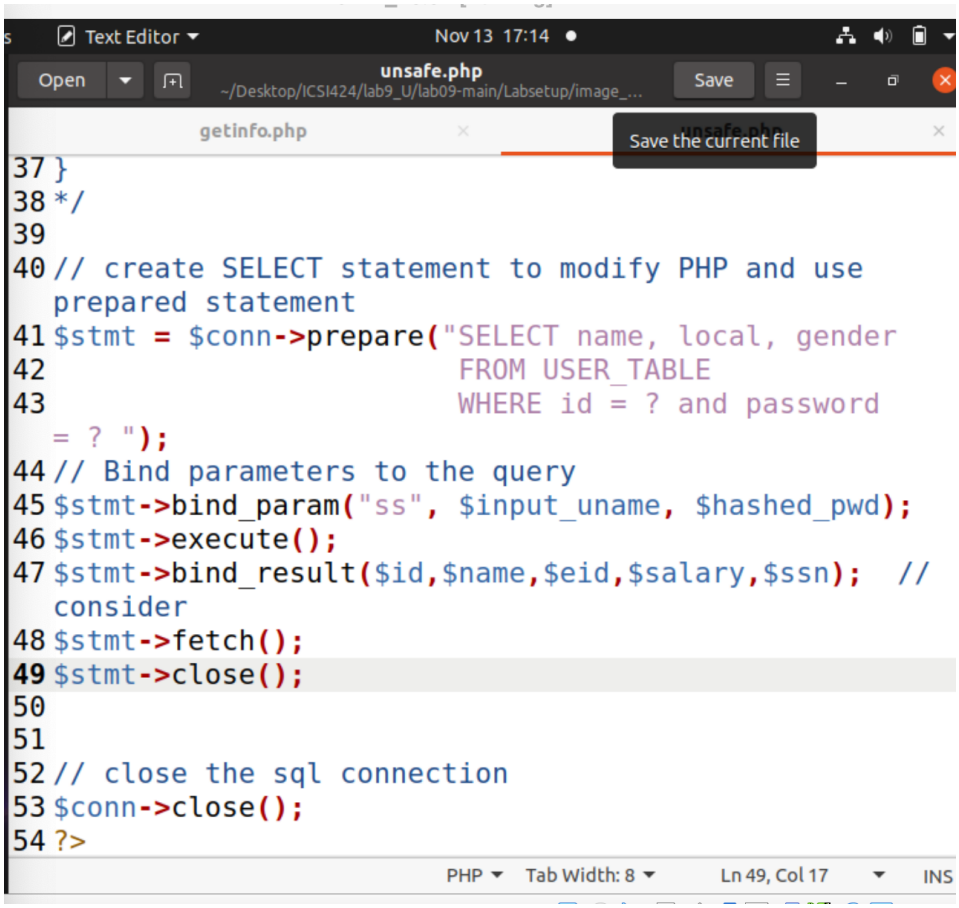


This is how unsafe.php looks like. Highlighted part has a vulnerability so we have to change that part with a preferred statement so that we can ensure code and data is not going to mix and create vulnerability.

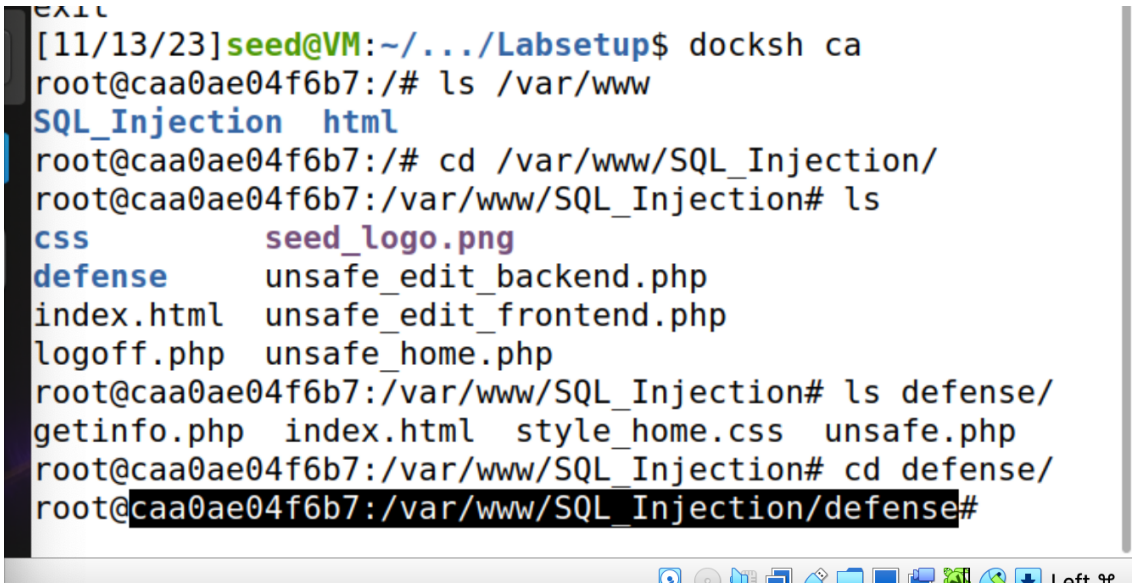
And if we use a SELECT statement to modify PHP, we can not launch attacks that we did above and use prepared statements that pass ? value instead of actual data and use bind_param and its parameter to send data to the database therefore we don't have to worry about cases where data and code might mixed.



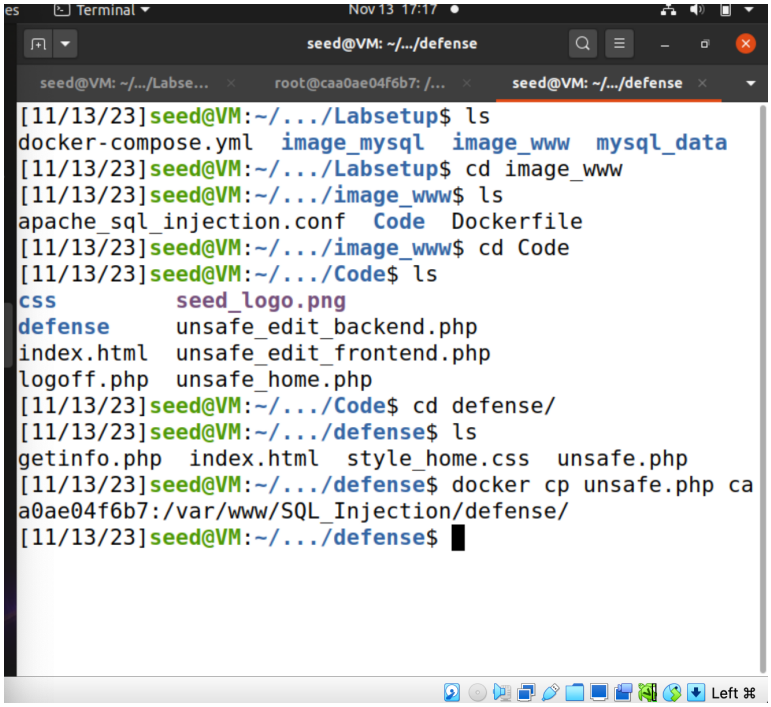
In unsafe.php I commented out part that creates vulnerability which was line 26 to line 36



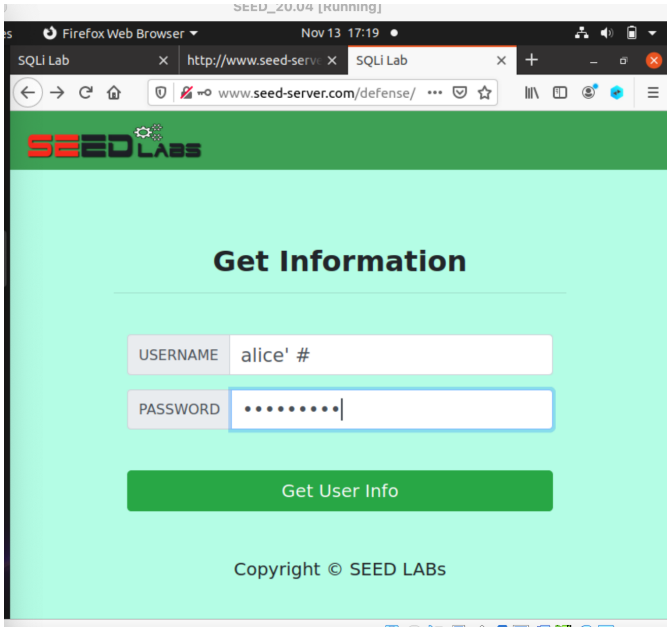
// and I added a prepared statement. Bind_param takes input_uname and hashed_pwd so I used “ss” indicating I am taking two string type values.
And I added execute() and bind_result() because we have id name eid salary and ssn values and then we fetch() and close() it.



// inorder to update the unsafe.php file in the defense folder. I found the path where the defense folder is stored.



// and then in labsetup folder > image_www > Code > defense folder and I copied changed unsafe.php file into defense folder path from the root.



// I typed alice's username with # so that password part will be considered as a comment and I can login to alice to any kinds of password. I tried but I wasn't able to log in as Alice so my attack failed and countermeasure was successful.