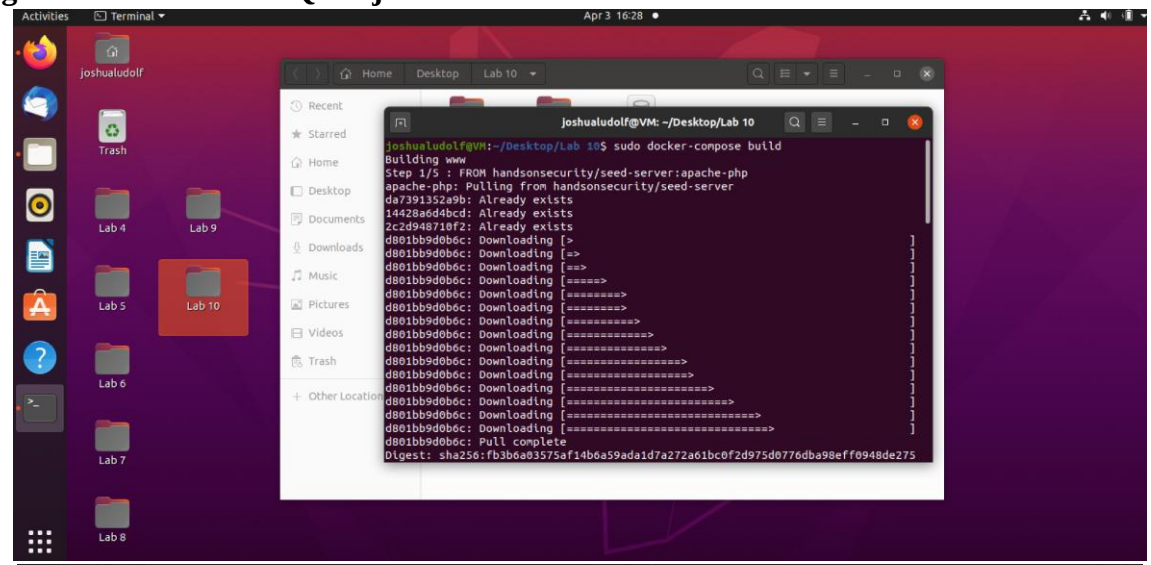
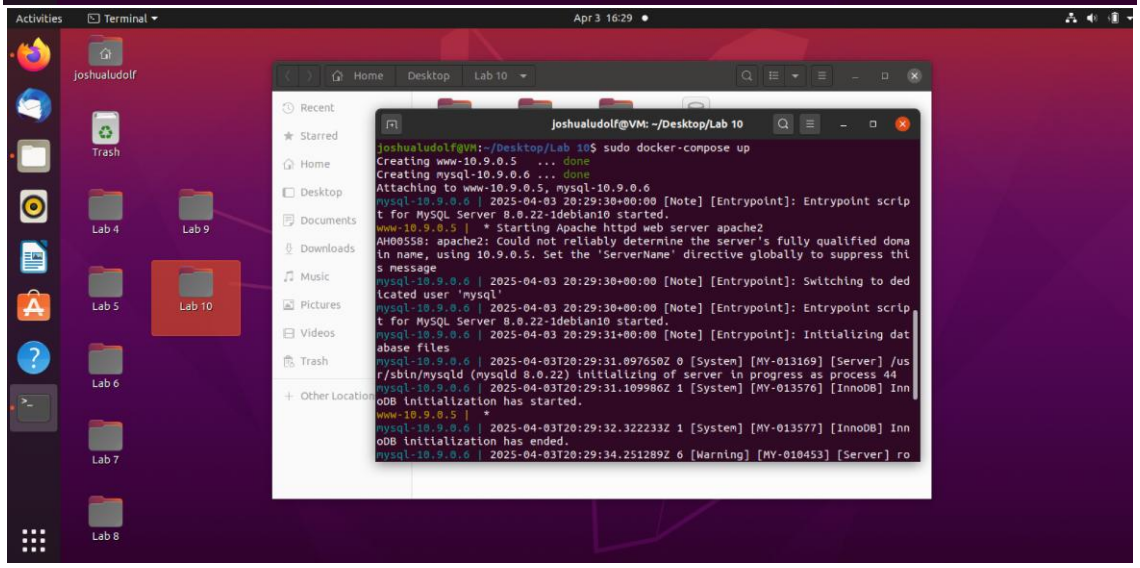
Lab 10: SQL Injection

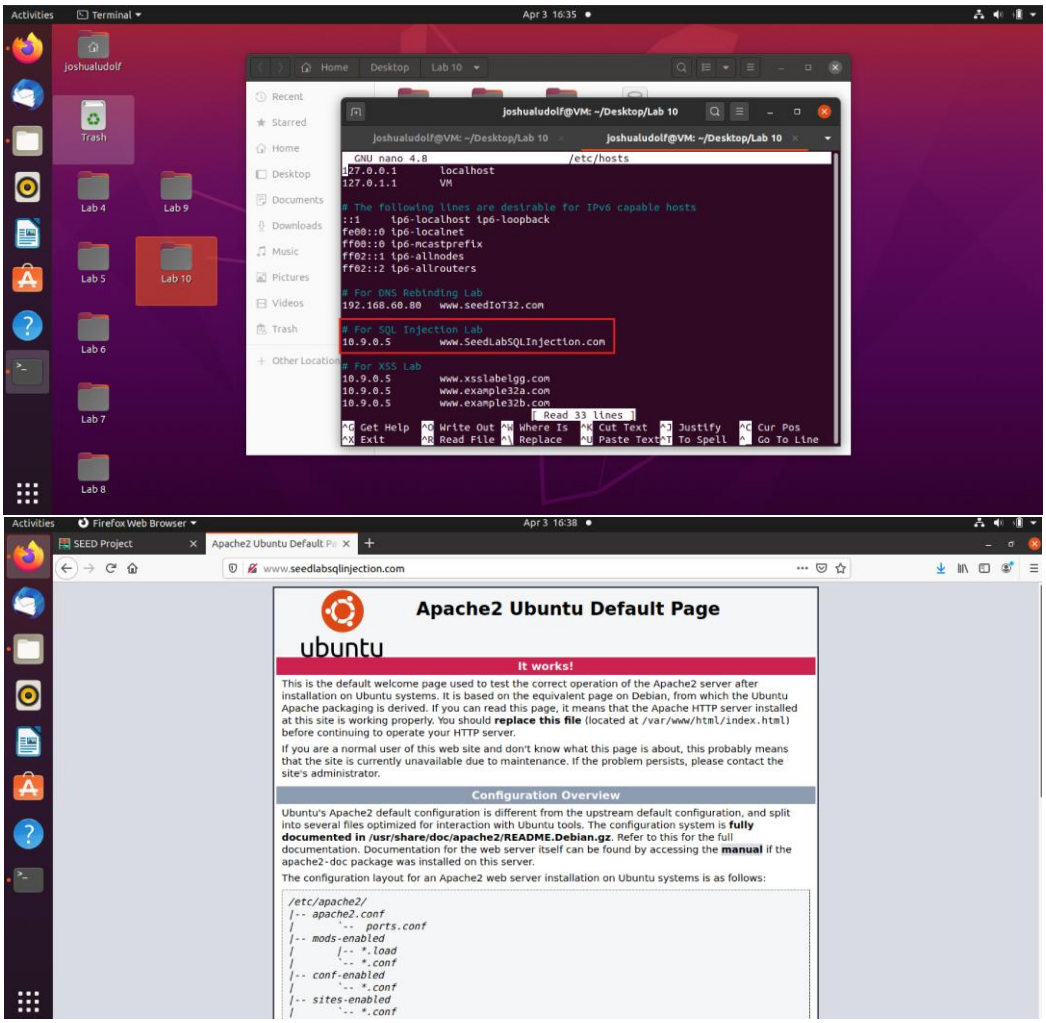
Joshua Ludolf
CSCI 4321
Computer Security

- **To get started with the SQL injection lab I had to build and activate the docker file:**

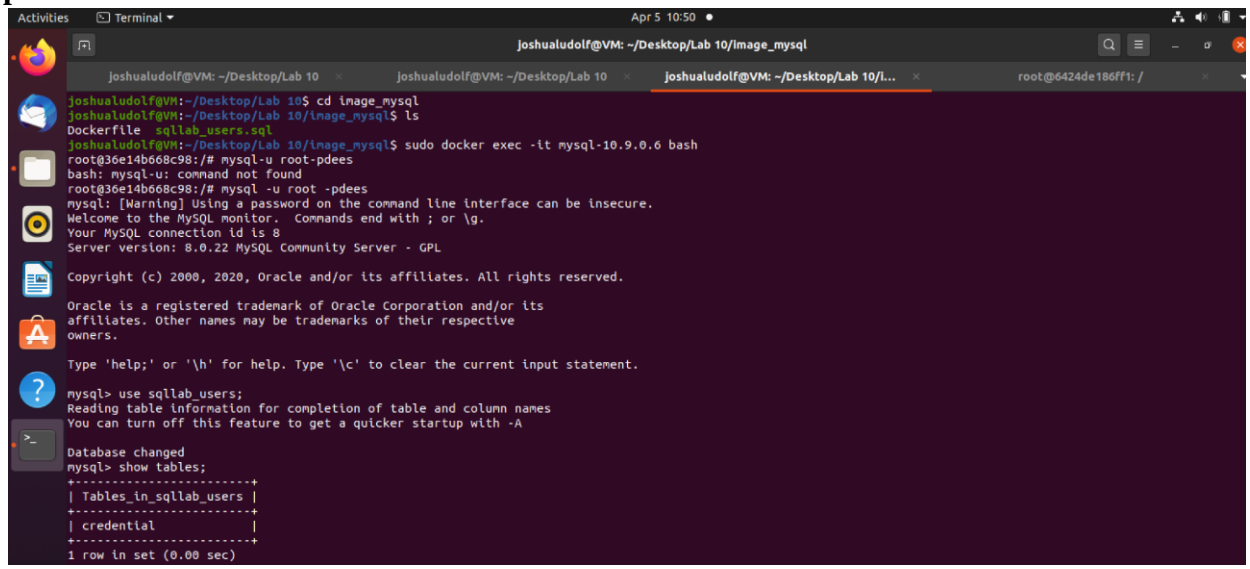


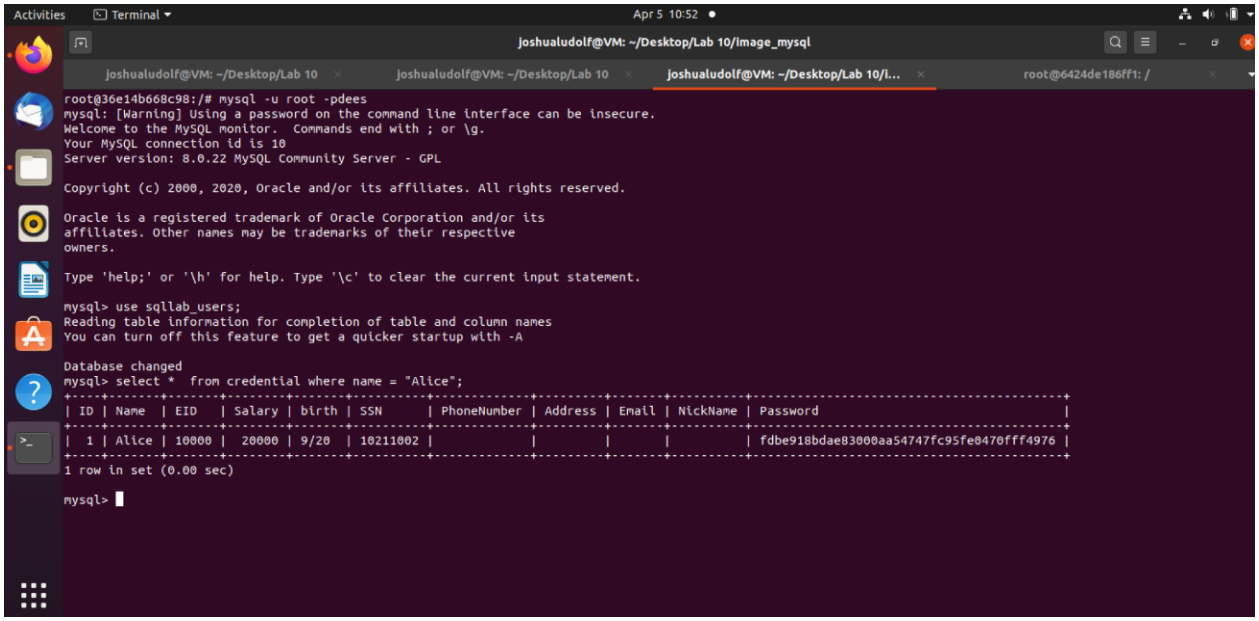


- **Additionally, I verified that the url www.SeedLabSQLInjection.com with IP address 10.9.0.5 was indeed in /etc/hosts:**

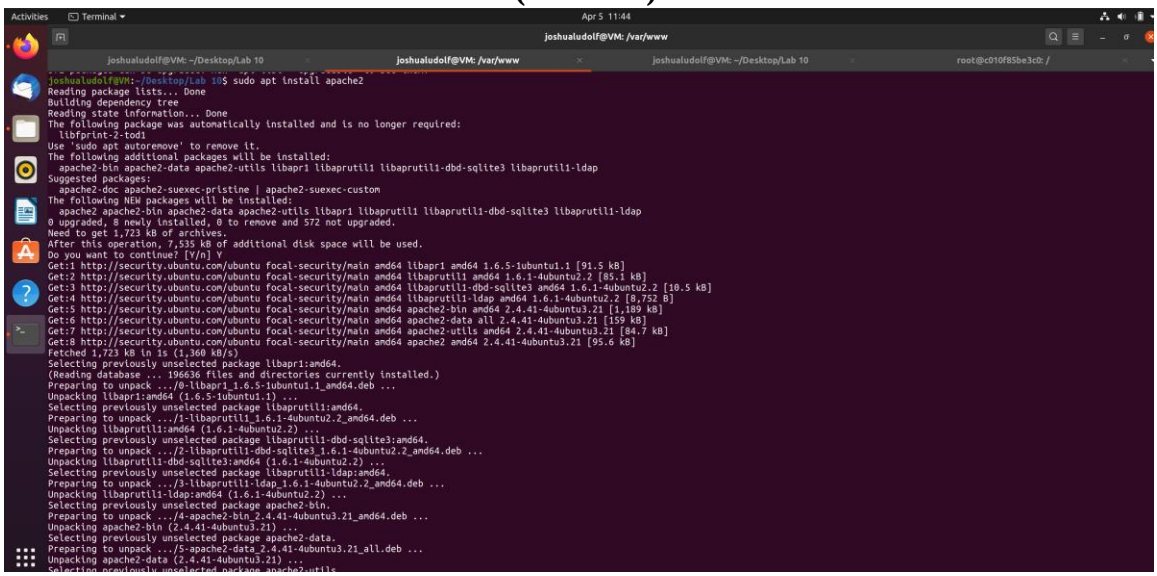


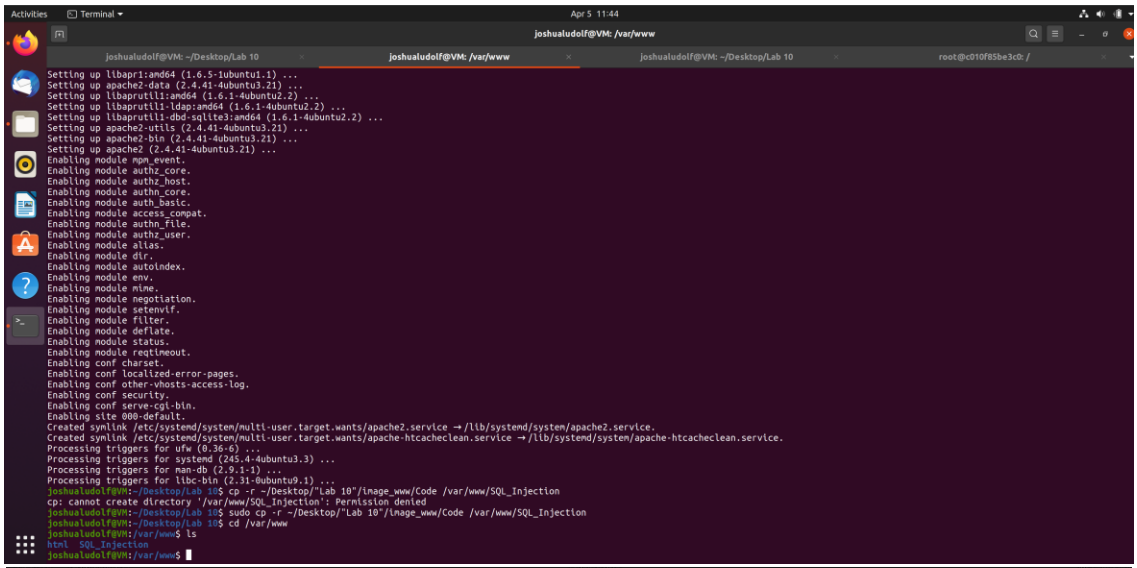
❖ **From there, I logged into the mysql container and acquired all the credentials of Alice, her password is fdbe918bdae83000aa54747fc95fe0470fff4976:**

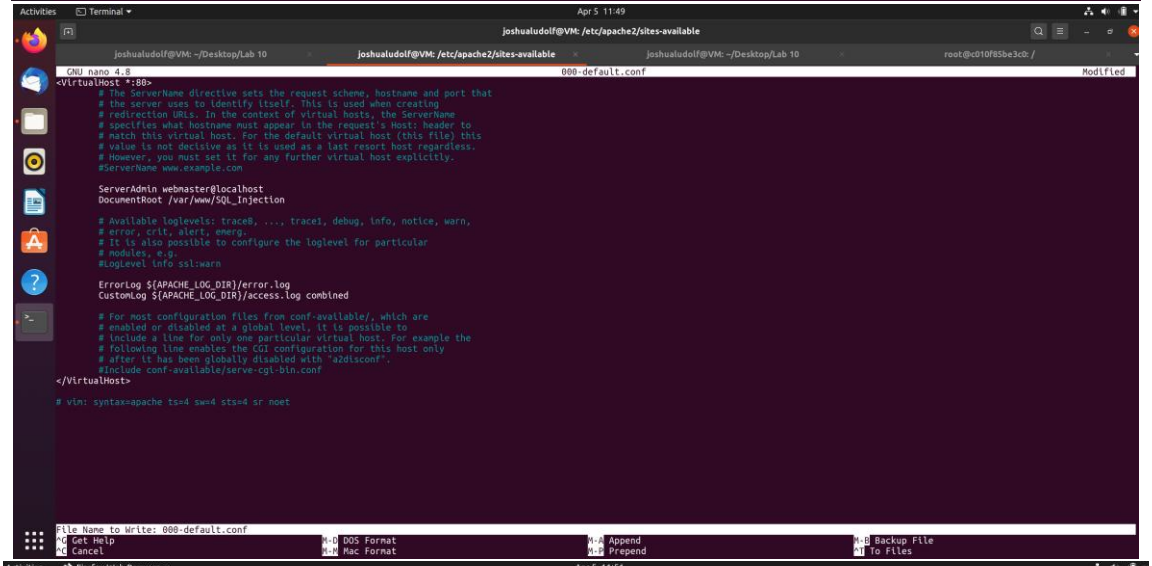


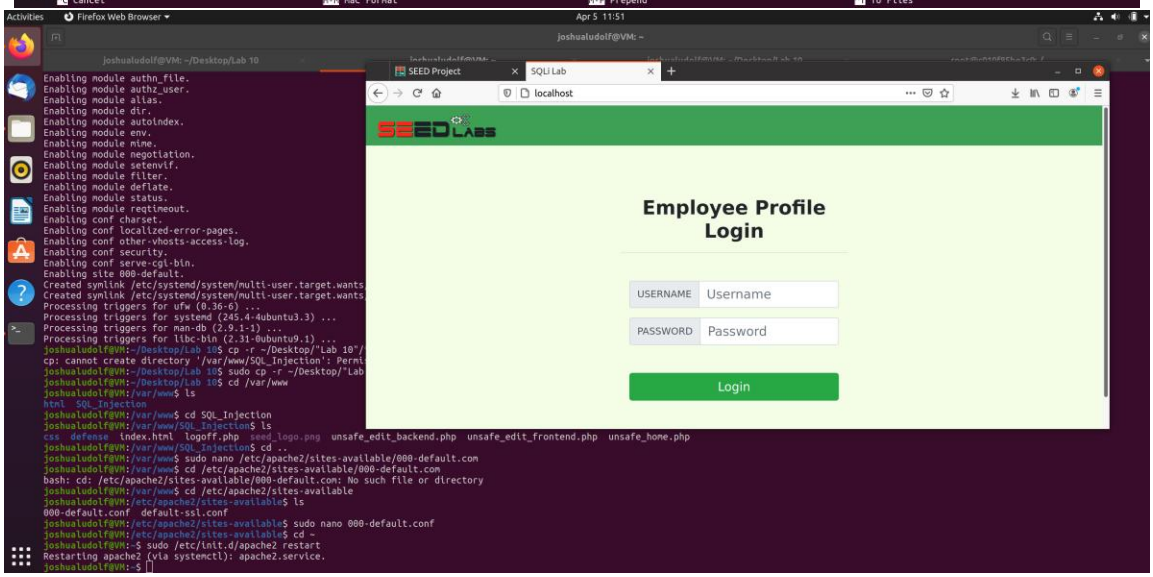


- **After that I installed Apache and copied the code folder to /var/www and was able to see the seed-server mock website in browser (localhost):**

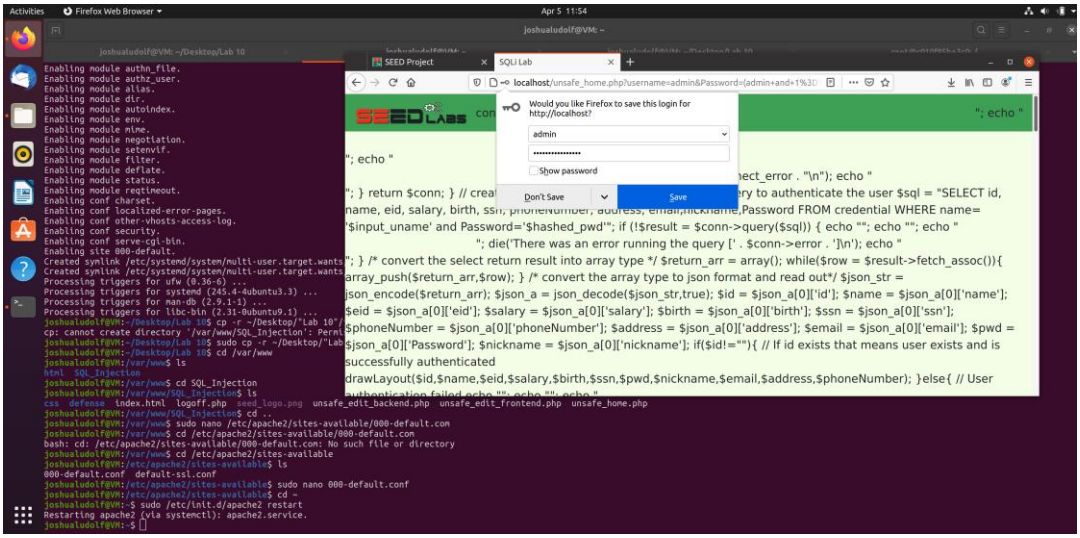



Activities
Terminal
Apr 5 11:44
joshualudolf@VM: /var/www
joshualudolf@VM: ~/Desktop/Lab 10
joshualudolf@VM: /var/www
joshualudolf@VM: ~/Desktop/Lab 10
root@c010f85be3c0: /
Setting up libapr1:amd64 (1.6.5-1ubuntu1.1) ...
Setting up apache2-data (2.4.41-4ubuntu3.21) ...
Setting up libaprutil1:amd64 (1.6.1-4ubuntu2.2) ...
Setting up libaprutil1-ldap:amd64 (1.6.1-4ubuntu2.2) ...
Setting up libaprutil1-dbd-sqlite3:amd64 (1.6.1-4ubuntu2.2) ...
Setting up apache2-utils (2.4.41-4ubuntu3.21) ...
Setting up apache2-bin (2.4.41-4ubuntu3.21) ...
Setting up apache2 (2.4.41-4ubuntu3.21) ...
Enabling module mpm_event.
Enabling module authz_core.
Enabling module authz_host.
Enabling module authn_core.
Enabling module auth_basic.
Enabling module access_compat.
Enabling module authn_file.
Enabling module authz_user.
Enabling module alias.
Enabling module dir.
Enabling module autoindex.
Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /lib/systemd/system/apache-htcacheclean.service.
Processing triggers for ufw (0.36-6) ...
Processing triggers for systemd (245.4-4ubuntu3.3) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.1) ...
joshualudolf@VM:~/Desktop/Lab 10$ cp -r ~/Desktop/"Lab 10"/image_www/Code /var/www/SQL_Injection
cp: cannot create directory '/var/www/SQL_Injection': Permission denied
joshualudolf@VM:~/Desktop/Lab 10$ sudo cp -r ~/Desktop/"Lab 10"/image_www/Code /var/www/SQL_Injection
joshualudolf@VM:~/Desktop/Lab 10$ cd /var/www
joshualudolf@VM:/var/www$ ls
html SQL_Injection
joshualudolf@VM:/var/www$

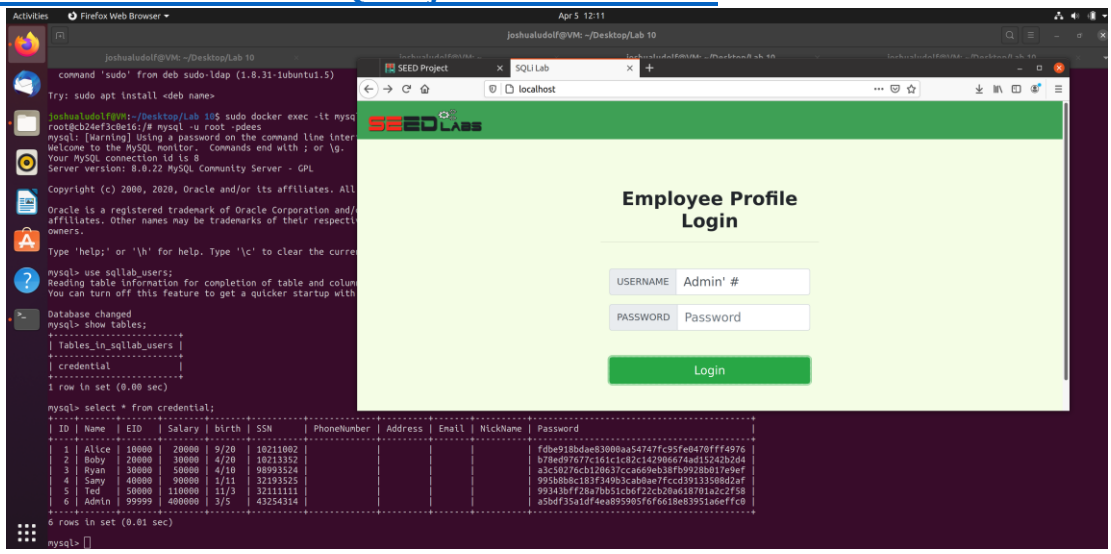

Activities
Terminal
Apr 5 11:49
joshualudolf@VM: /etc/apache2/sites-available
joshualudolf@VM: ~/Desktop/Lab 10
joshualudolf@VM: /etc/apache2/sites-available
joshualudolf@VM: ~/Desktop/Lab 10
root@c010f85be3c0: /
GNU nano 4.8
000-default.conf
Modified
<VirtualHost *:80>
The ServerName directive sets the request scheme, hostname and port that
the server uses to identify itself. This is used when creating
redirection URLs. In the context of virtual hosts, the ServerName
specifies what hostname must appear in the request's Host: header to
match this virtual host. For the default virtual host (this file) this
value is not decisive as it is used as a last resort host regardless.
However, you must set it for any further virtual host explicitly.
#ServerName www.example.com
ServerAdmin webmaster@localhost
DocumentRoot /var/www/SQL_Injection
Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
error, crit, alert, emerg.
It is also possible to configure the loglevel for particular
modules, e.g.
#LogLevel info ssl:warn
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
For most configuration files from conf-available/, which are
enabled or disabled at a global level, it is possible to
include a line for only one particular virtual host. For example the
following line enables the CGI configuration for this host only
after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
vim: syntax=apache ts=4 sw=4 sts=4 sr noet
File Name to Write: 000-default.conf
^G Get Help
^C Cancel
M-D DOS Format
M-M Mac Format
M-A Append
M-P Prepend
^B Backup File
^T To Files

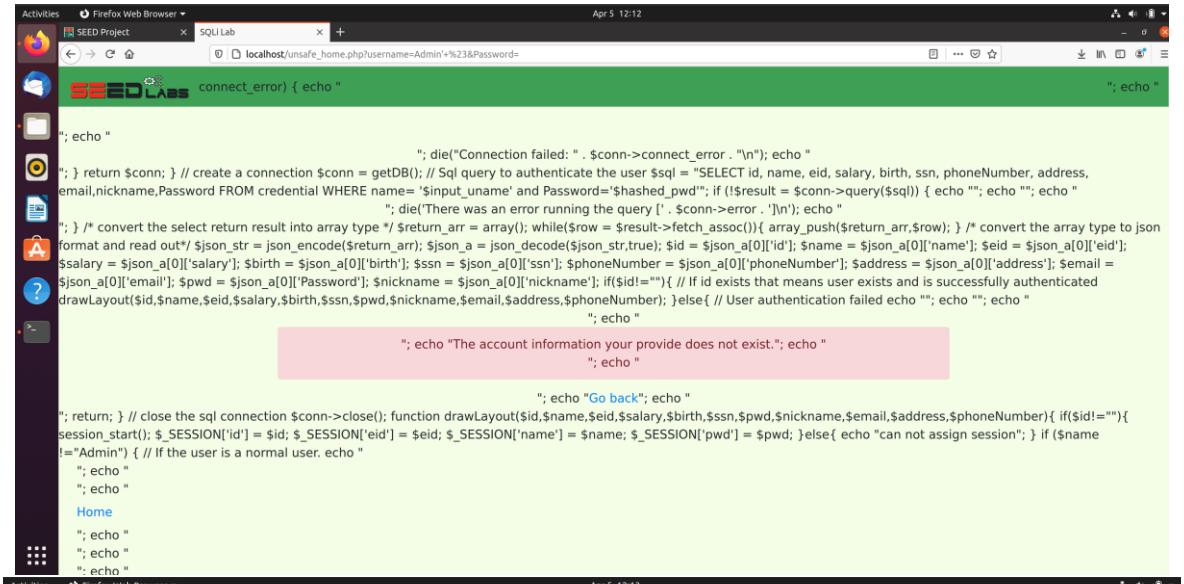

Activities
Firefox Web Browser
Apr 5 11:51
joshualudolf@VM: ~
SEED Project
SQLi Lab
localhost
SEEDLabs
Employee Profile Login
USERNAME
Username
PASSWORD
Password
Login
Enabling module authn_file.
Enabling module authz_user.
Enabling module alias.
Enabling module dir.
Enabling module autoindex.
Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Processing triggers for ufw (0.36-6) ...
Processing triggers for systemd (245.4-4ubuntu3.3) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.1) ...
joshualudolf@VM:~/Desktop/Lab 10$ sudo cp -r ~/Desktop/"Lab
joshualudolf@VM:~/Desktop/Lab 10$ cd /var/www
joshualudolf@VM:/var/www$ ls
html SQL_Injection
joshualudolf@VM:/var/www$ cd SQL_Injection
joshualudolf@VM:/var/www/SQL_Injection$ ls
css defense index.html logoff.php seed_logo.png unsafe_edit_backend.php unsafe_edit_frontend.php unsafe_home.php
joshualudolf@VM:/var/www/SQL_Injection$ cd ..
joshualudolf@VM:/var/www$ sudo nano /etc/apache2/sites-available/000-default.com
joshualudolf@VM:/var/www$ cd /etc/apache2/sites-available/000-default.com
bash: cd: /etc/apache2/sites-available/000-default.com: No such file or directory
joshualudolf@VM:/var/www$ cd /etc/apache2/sites-available
joshualudolf@VM:/etc/apache2/sites-available$ ls
000-default.conf default-ssl.conf
joshualudolf@VM:/etc/apache2/sites-available$ sudo nano 000-default.conf
joshualudolf@VM:/etc/apache2/sites-available$ cd ~
joshualudolf@VM:~$ sudo /etc/init.d/apache2 restart
Restarting apache2 (via systemctl): apache2.service.
joshualudolf@VM:~$

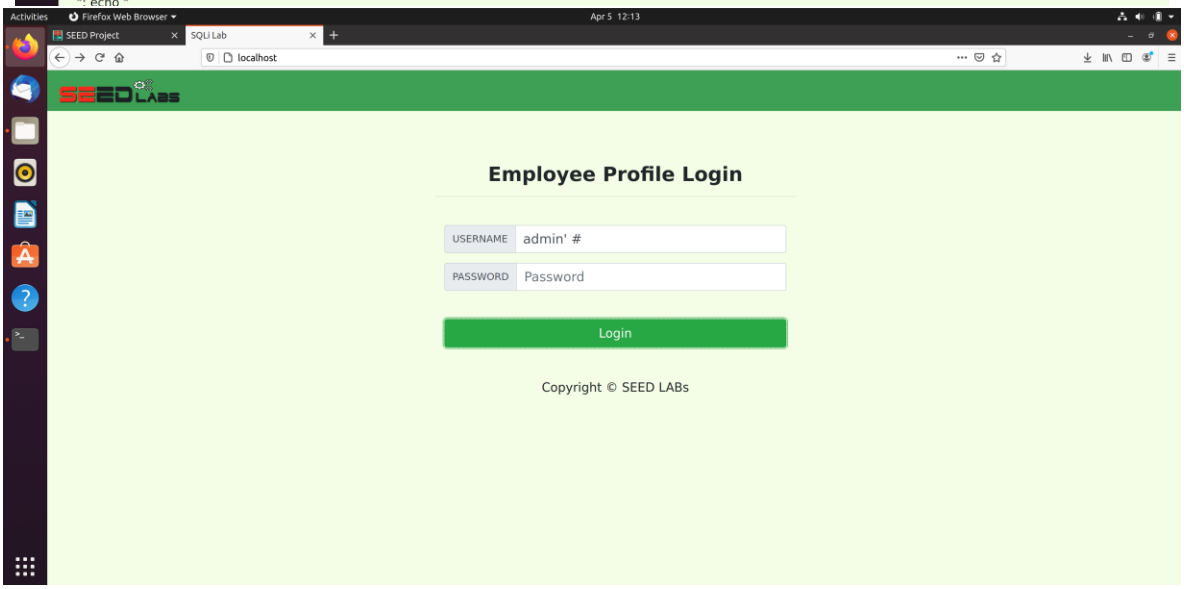
- **First I tried the famous ((admin and 1=1') which obviously didn't work as I got a bunch of html code on screen:**

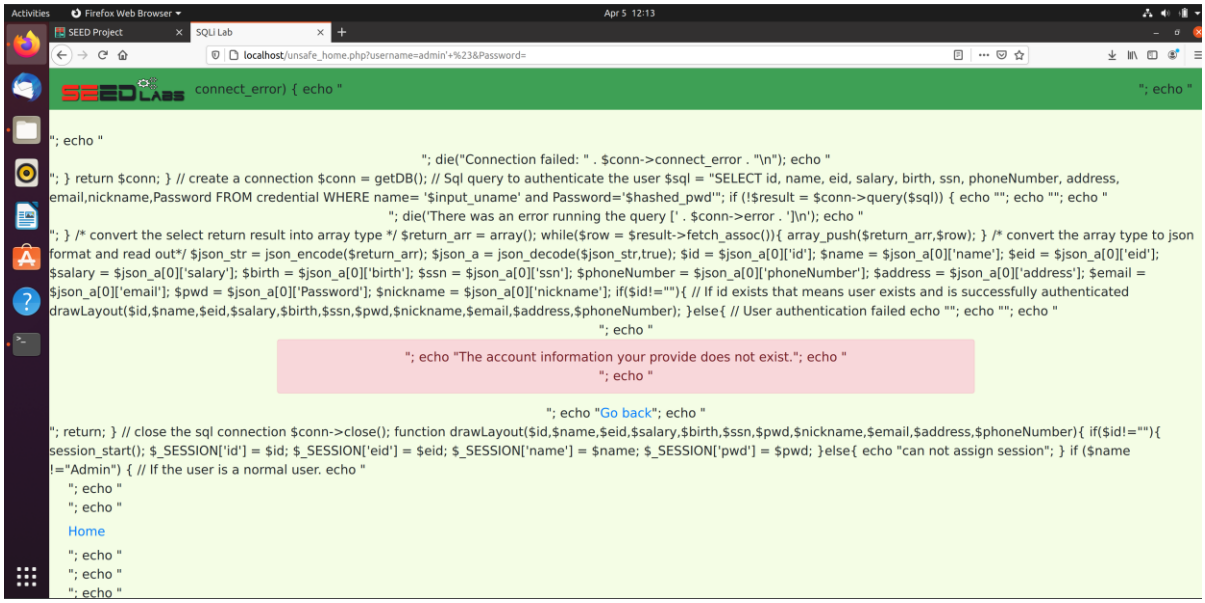


- **From there I tried the following but wasn't able to login even though it's supposed to work according to this github repo - [SEED-SQL-Injection-Lab/SQL Injection Attack Lab.pdf at main · HMIrfan2599/SEED-SQL-Injection-Lab · GitHub]:**

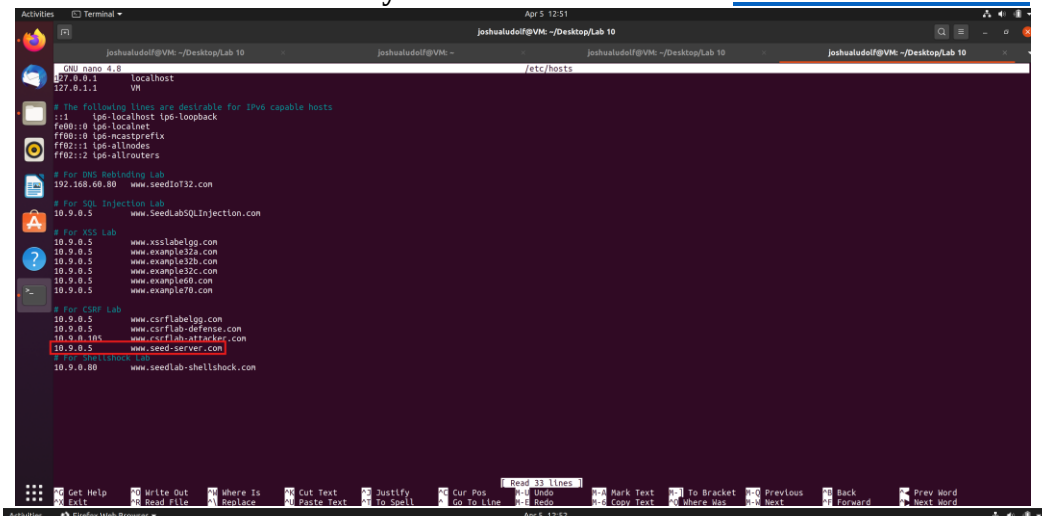


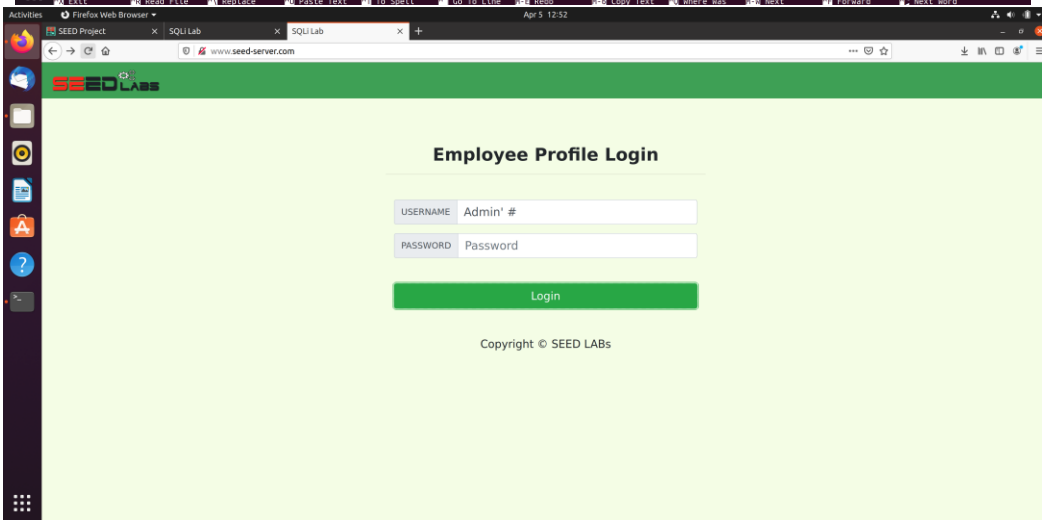

Activities
Firefox Web Browser
Apr 5 12:12
SEED Project
SQLi Lab
localhost/unsafe_home.php?username=Admin'+%23&Password=
connect_error) { echo " "; echo "
"; echo "
"; die("Connection failed: " . $conn->connect_error . "\n"); echo "
"; } return $conn; } // create a connection $conn = getDB(); // Sql 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 ""; echo ""; echo "
"; die('There was an error running the query [' . $conn->error . ']\n'); echo "
"; } /* convert the select return result into array type */ $return_arr = array(); while($row = $result->fetch_assoc()){ array_push($return_arr,$row); } /* convert the array type to json format and read out*/ $json_str = json_encode($return_arr); $json_a = json_decode($json_str,true); $id = $json_a[0]['id']; $name = $json_a[0]['name']; $eid = $json_a[0]['eid']; $salary = $json_a[0]['salary']; $birth = $json_a[0]['birth']; $ssn = $json_a[0]['ssn']; $phoneNumber = $json_a[0]['phoneNumber']; $address = $json_a[0]['address']; $email = $json_a[0]['email']; $pwd = $json_a[0]['Password']; $nickname = $json_a[0]['nickname']; if($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 ""; echo ""; echo "
"; echo "
"; echo "The account information your provide does not exist."; echo "
"; echo "
"; echo "Go back"; echo "
"; return; } // close the sql connection $conn->close(); function drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber){ if($id!=""){ session_start(); $_SESSION['id'] = $id; $_SESSION['eid'] = $eid; $_SESSION['name'] = $name; $_SESSION['pwd'] = $pwd; }else{ echo "can not assign session"; } if ($name !="Admin") { // If the user is a normal user. echo "
"; echo "
"; echo "
Home
"; echo "
"; echo "
"; echo "

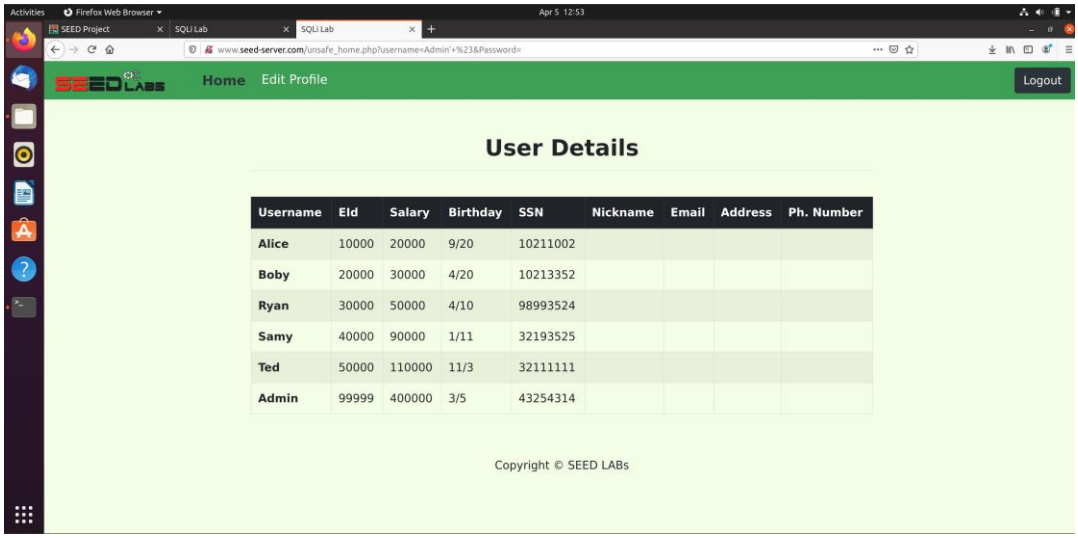

Activities
Firefox Web Browser
Apr 5 12:13
SEED Project
SQLi Lab
localhost
Employee Profile Login
USERNAME
admin' #
PASSWORD
Password
Login
Copyright © SEED LABs



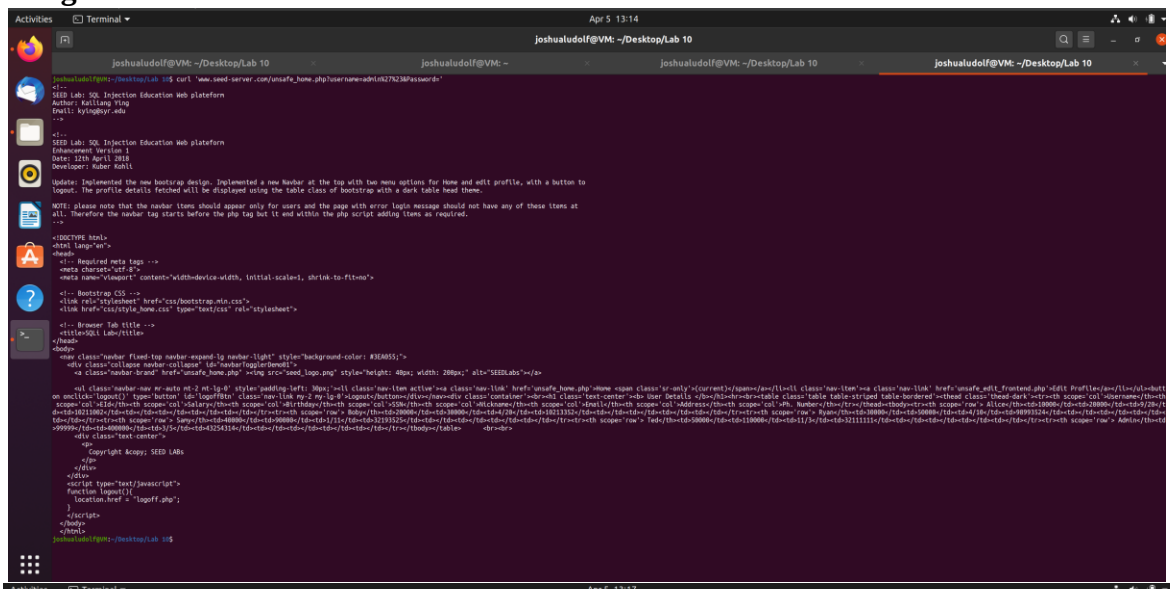
❖ **I figured out that I needed to modify the /etc/hosts to include www.seed-server.com:**

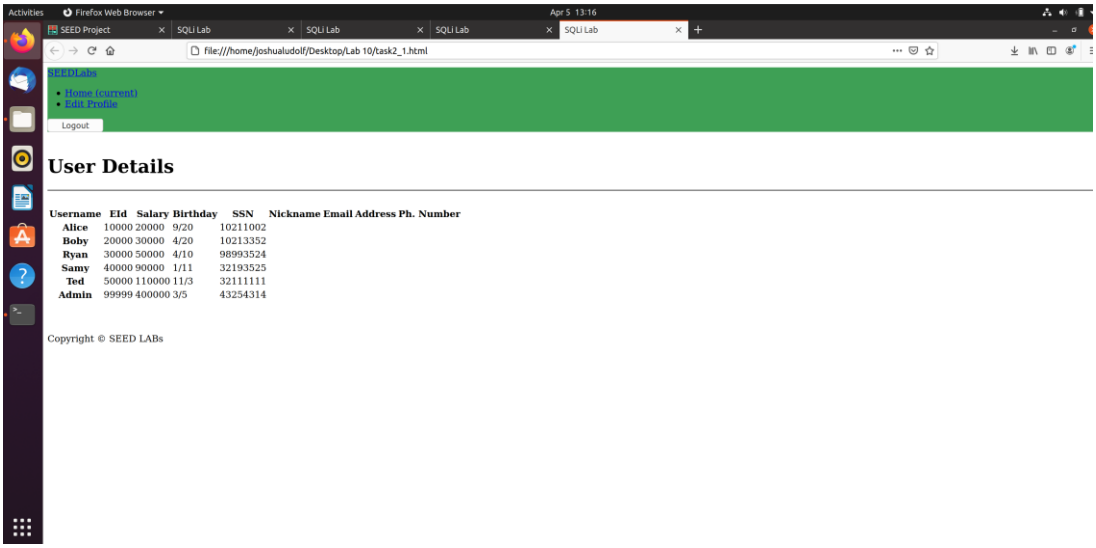




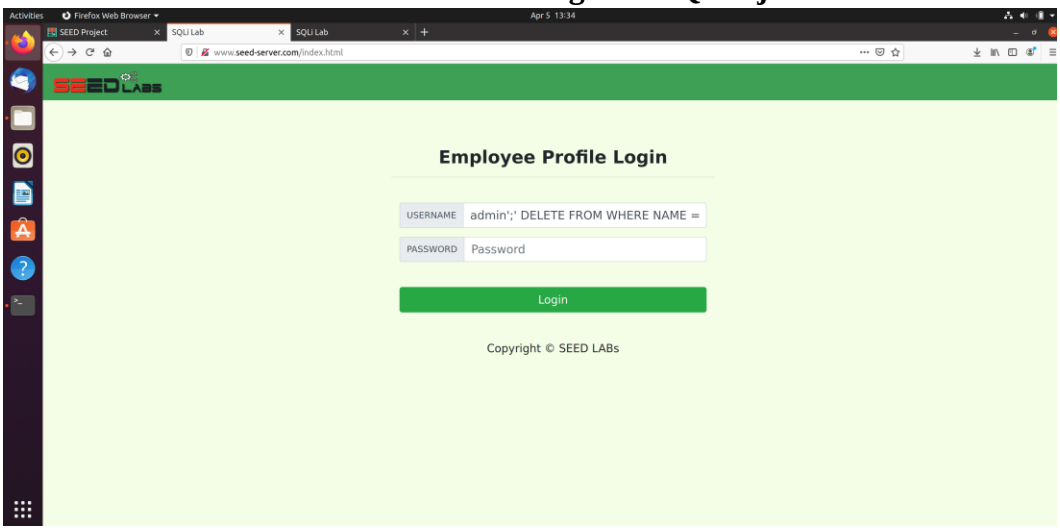


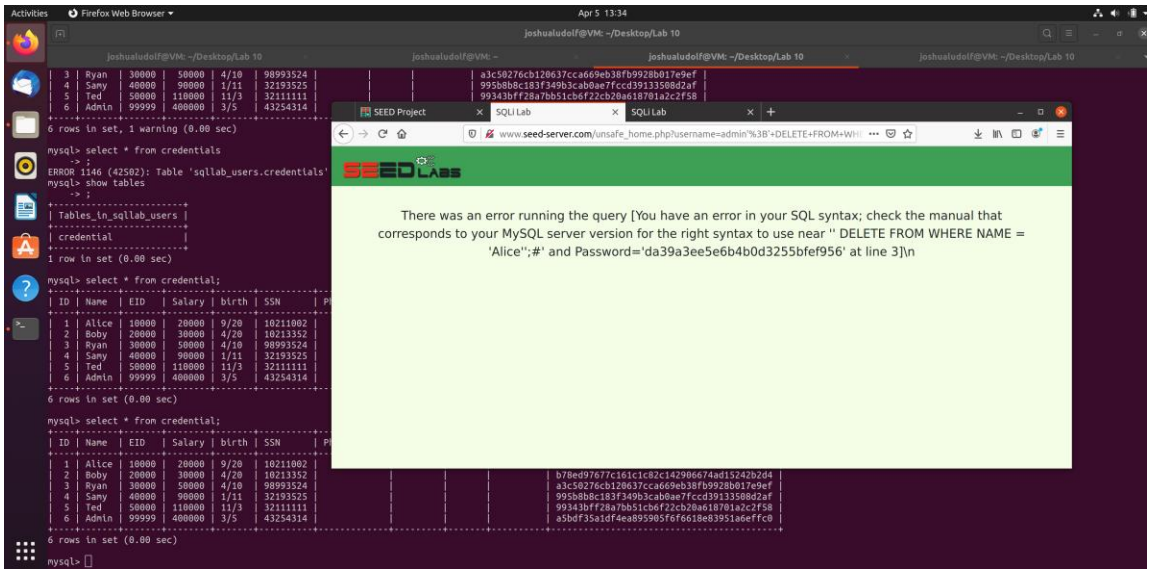
- **From there I modified the next task for doing it with admin as the Alice one wasn't working for me:**



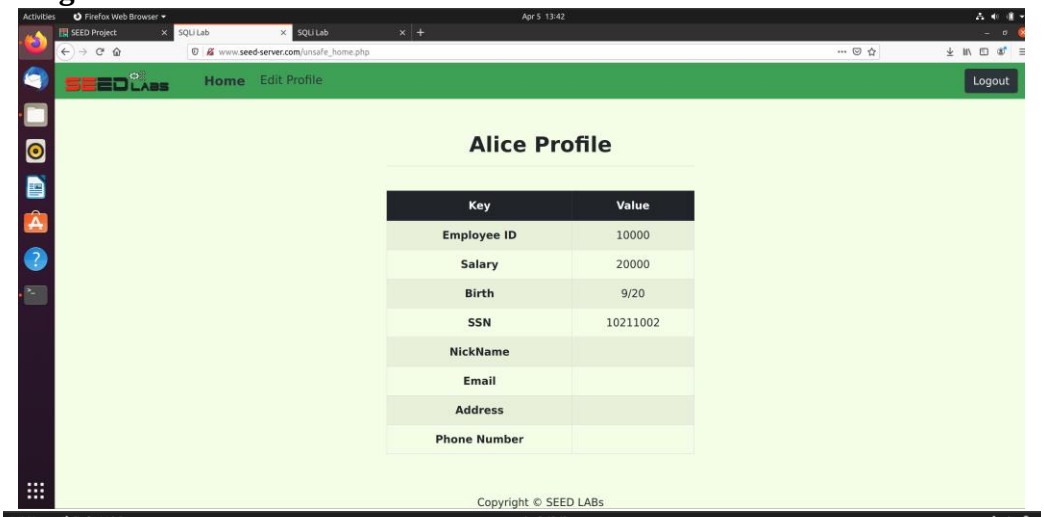


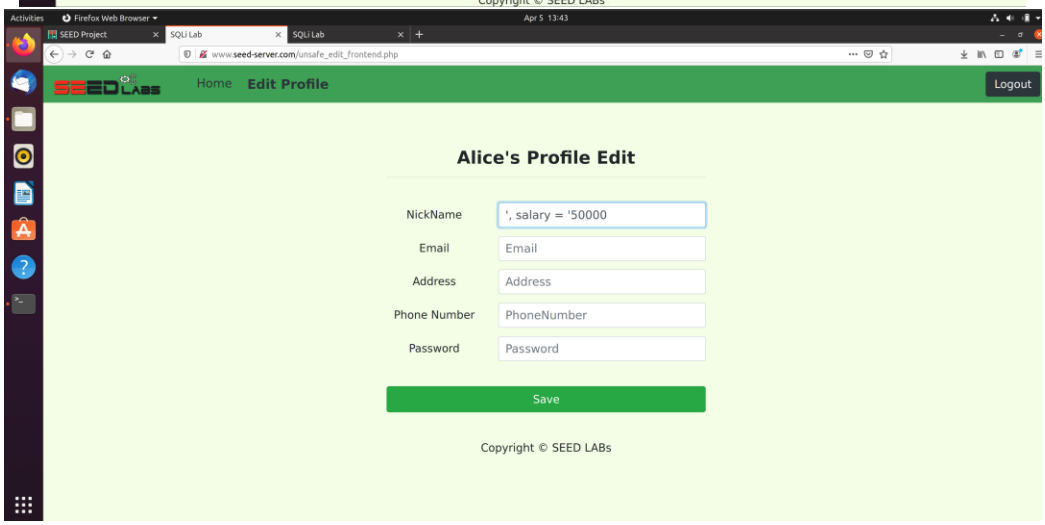
- **The next task asked us to append a new SQL Statement to the original login attack, I attempted to execute the command to delete the user Alice numerous times, only to encounter an error each time. However, after delving into online resources, I discovered that MySQL is safeguarded against such attacks due to PHP's mysqli extension. This extension, specifically the mysqli::query() API, prevents the execution of multiple queries on the database server as a defense mechanism against SQL injsection:**

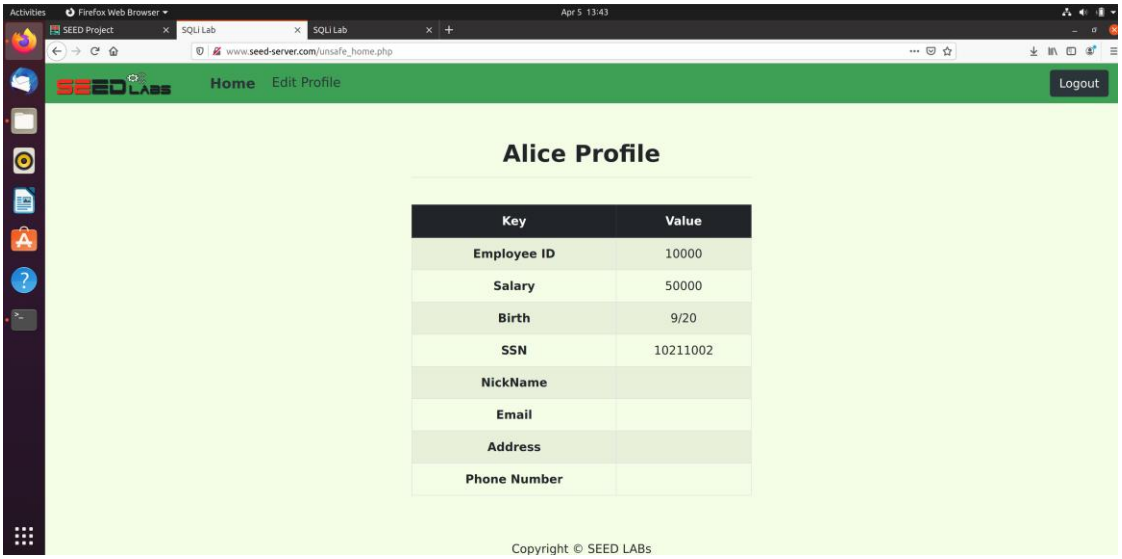




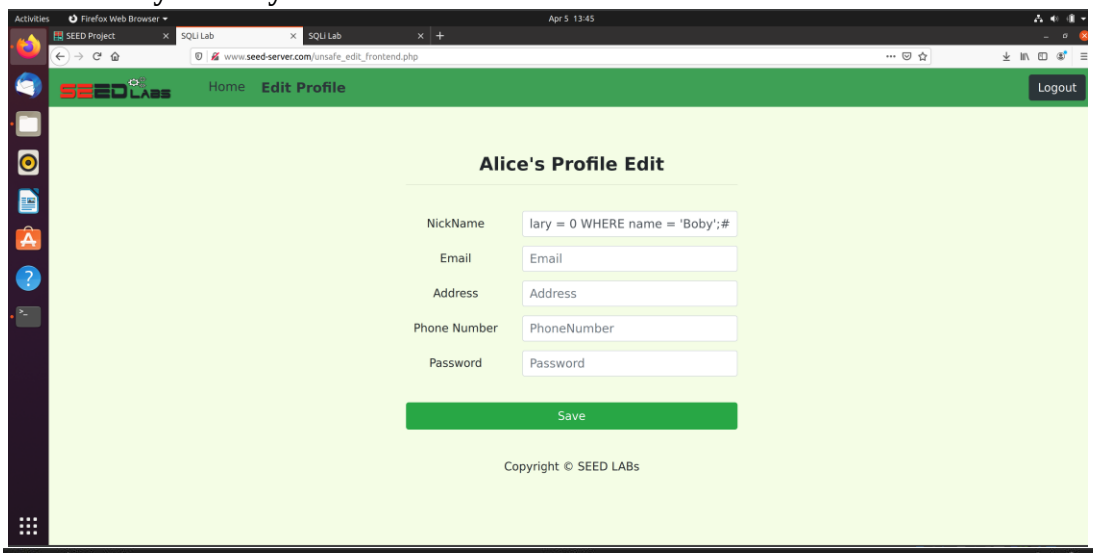
- **The next task was to change the salary, for this I used the Alice account logged in and did the following:**

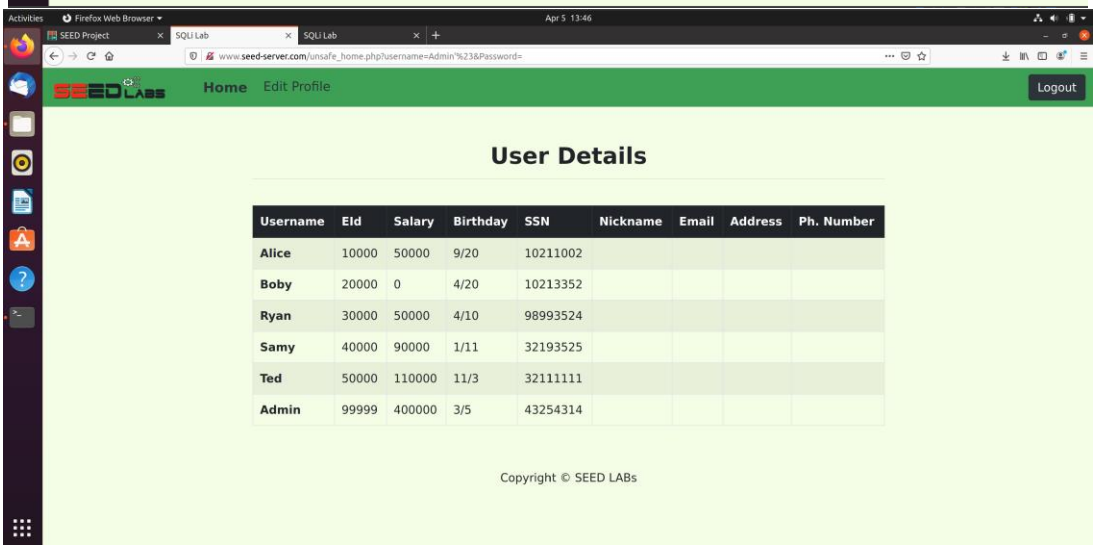




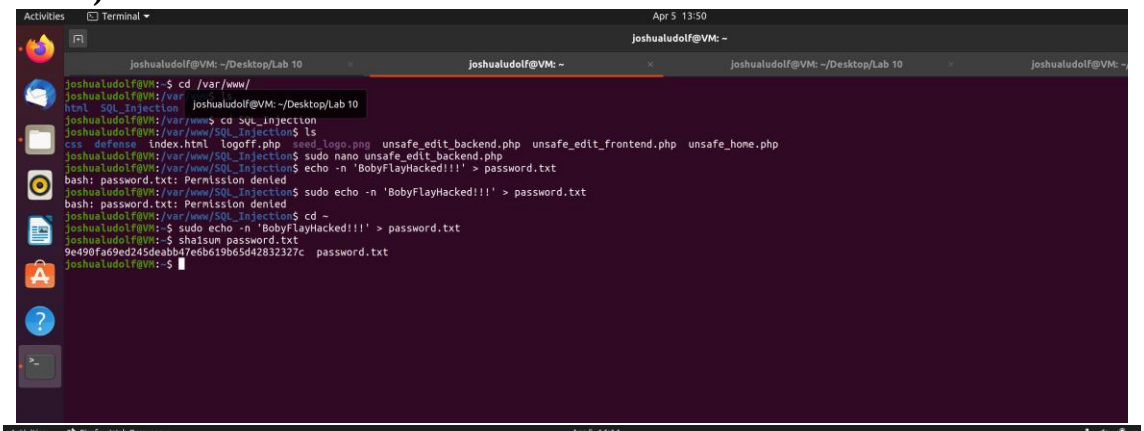


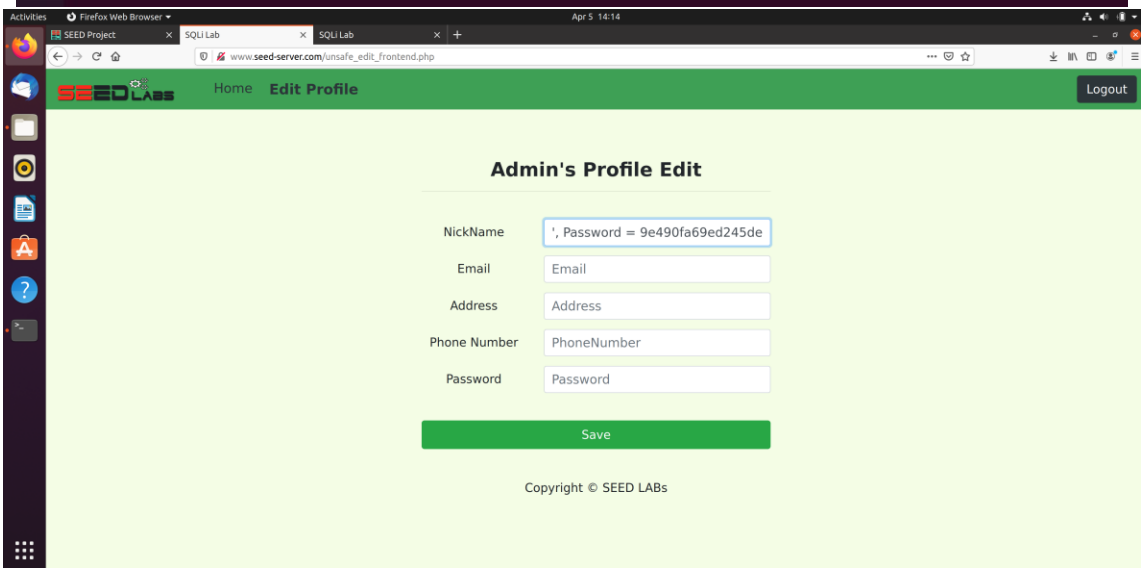
- **From that point I took interest in doing similar things but to another account, in this case I would make Boby's salary 0:**

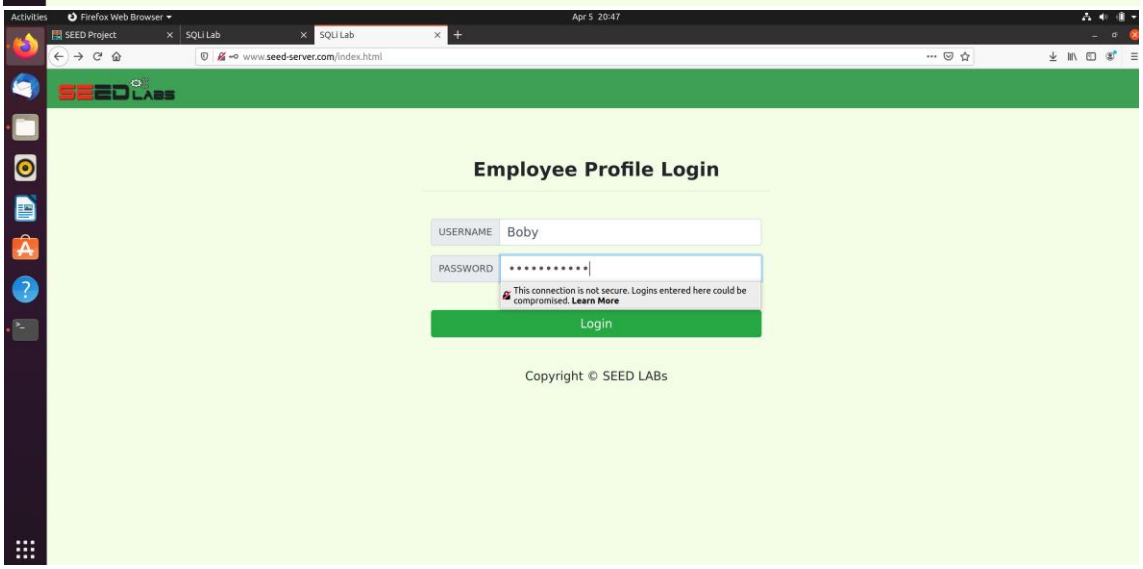


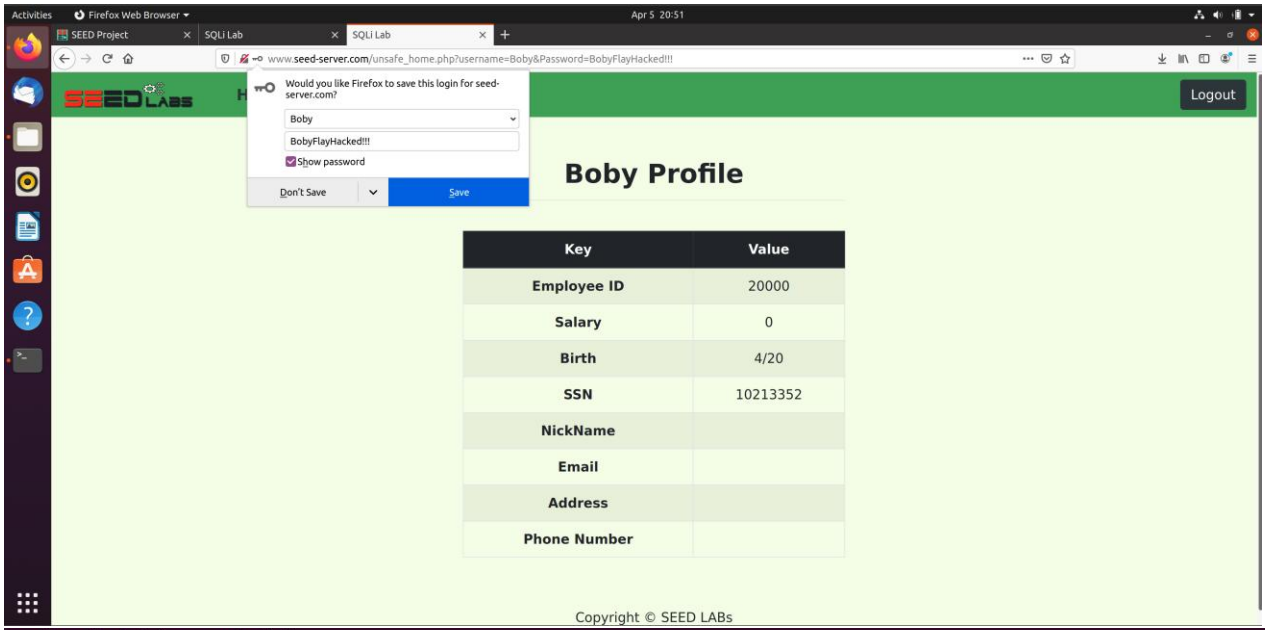


- **For the task after that, I needed to modify other peoples' password(s), in this case I chose the password BobyFlayHacked!!! (additionally can confirm as the sql database has the new hash value):**



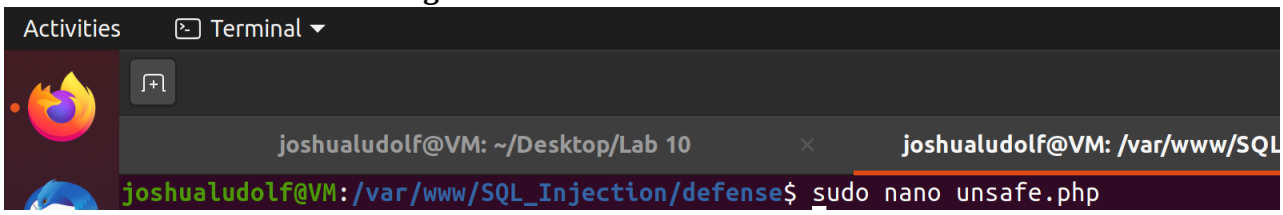


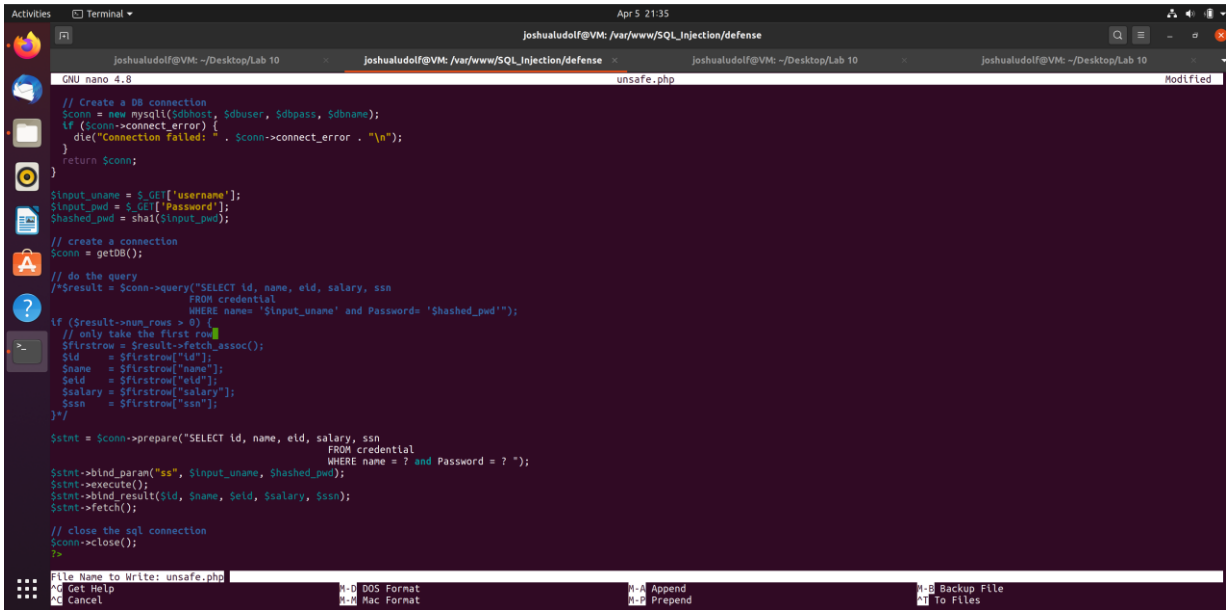


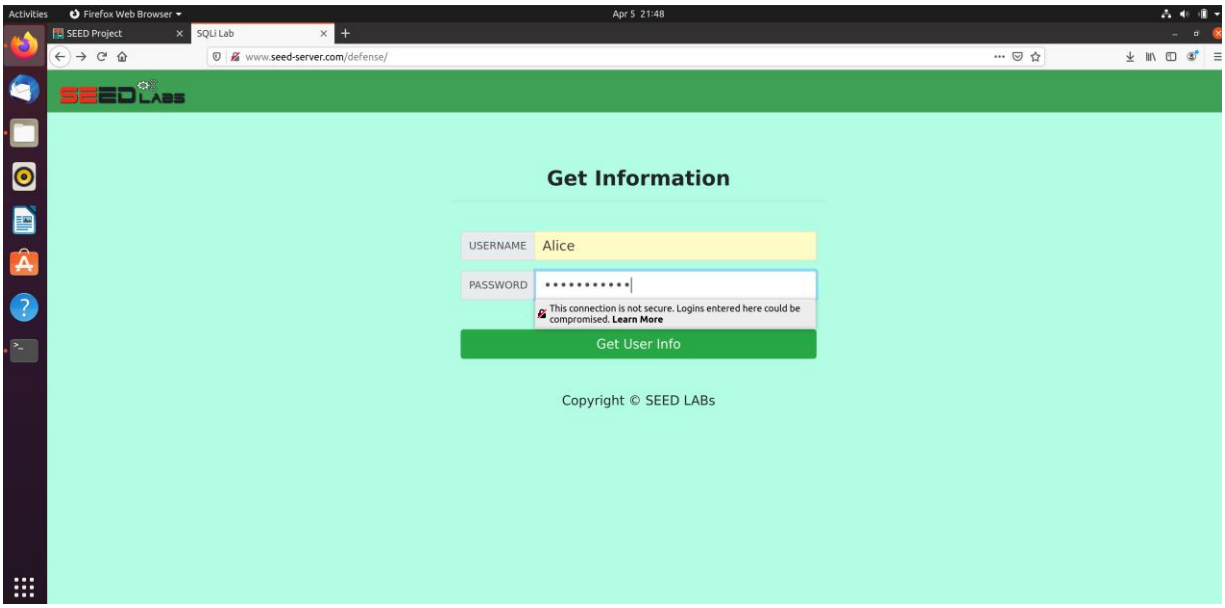


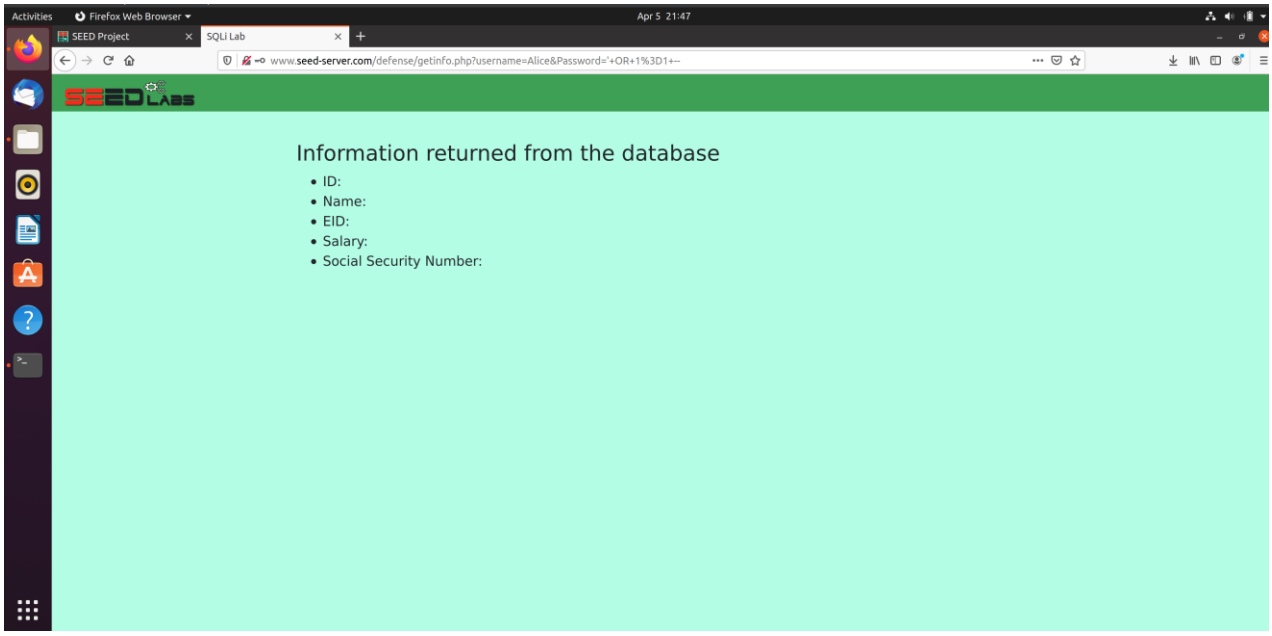

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  50000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      0 | 4/20  | 10213352 |             |         |       |          | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  50000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |      0 | 4/20  | 10213352 |             |         |       |          | 9e490fa69ed245deabb47e6b619b65d42832327c |
|  3 | Ryan  | 30000 |  50000 | 4/10  | 98993524 |             |         |       |          | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 |  90000 | 1/11  | 32193525 |             |         |       |          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |             |         |       |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |             |         |       |          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

- **Finally for task 4, I needed to implement Countermeasure, essentially only correct login information entered into the login will be able to see their information:**

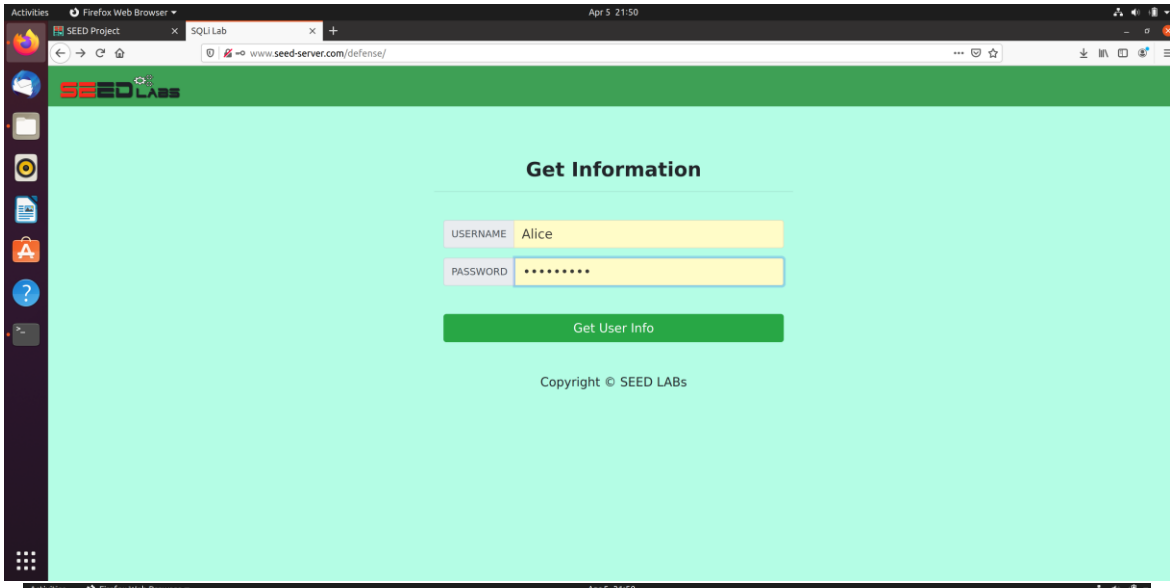


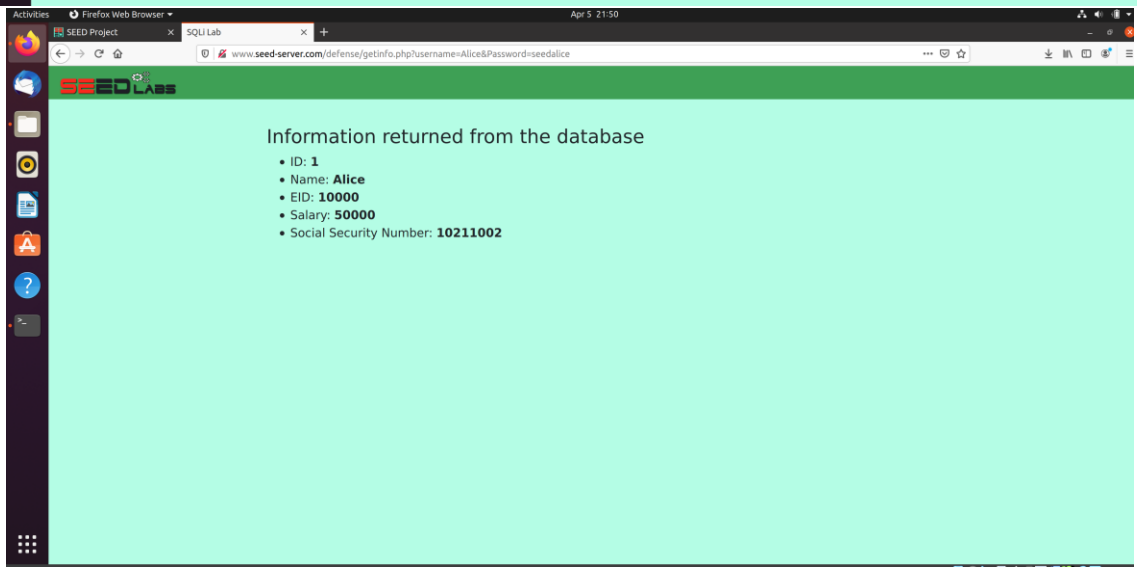

Activities
Terminal
Apr 5 21:35
joshualudolf@VM: /var/www/SQL_Injection/defense
joshualudolf@VM: ~/Desktop/Lab 10
joshualudolf@VM: /var/www/SQL_Injection/defense
joshualudolf@VM: ~/Desktop/Lab 10
joshualudolf@VM: ~/Desktop/Lab 10
GNU nano 4.8
unsafe.php
Modified
// Create a DB connection
$conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error . "\n");
}
return $conn;
}
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
// create a connection
$conn = getDB();
// do the query
/*$result = $conn->query("SELECT id, name, eid, salary, ssn
FROM credential
WHERE name= '$input_uname' and Password= '$hashed_pwd'");
if ($result->num_rows > 0) {
// only take the first row
$firstrow = $result->fetch_assoc();
$id = $firstrow["id"];
$name = $firstrow["name"];
$eid = $firstrow["eid"];
$salary = $firstrow["salary"];
$ssn = $firstrow["ssn"];
}*/
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
FROM credential
WHERE name = ? and Password = ? ");
$stmt->bind_param("ss", $input_uname, $hashed_pwd);
$stmt->execute();
$stmt->bind_result($id, $name, $eid, $salary, $ssn);
$stmt->fetch();
// close the sql connection
$conn->close();
?>
File Name to Write: unsafe.php
^G Get Help
^C Cancel
M-D DOS Format
M-M Mac Format
M-A Append
M-P Prepend
M-B Backup File
^T To Files





- **Ok and this shows that Alice still has her original information (essentially using correct password…):**





❖ What I learned from this lab:

In this lab exercise, I conducted an in-depth analysis of SQL injection vulnerabilities in a web application, as detailed in the SEED Labs document. I explored how unsanitized user inputs can be exploited to manipulate SQL queries, allowing unauthorized access and data modification. By executing attacks on both SELECT and UPDATE statements—using both web interfaces and command-line tools—I was able to bypass authentication mechanisms and alter critical information such as salaries and passwords. Additionally, I evaluated the effectiveness of prepared statements as a countermeasure, observing how they segregate code from data to prevent injection attacks. This hands-on experience not only deepened my understanding of SQL injection techniques but also reinforced the importance of implementing robust security measures in web applications.