Indian Institute of Technology Jammu
Information Security (CSC050P1M)
Assignment 2
SQL Injection
Deadline: 11:59 PM, January 27, 2023

1 Overview

SQL injection is a code injection technique that exploits the vulnerabilities in the interface between web applications and database servers. The vulnerability is present when user's inputs are not correctly checked within the web applications before being sent to the back-end database servers.

Many web applications take inputs from users, and then use these inputs to construct SQL queries, so they can get information from the database. Web applications also use SQL queries to store information in the database. These are standard practices in the development of web applications. When SQL queries are not carefully constructed, SQL injection vulnerabilities can occur. SQL injection is one of the most common attacks on web applications.

In this lab, we have created a web application that is vulnerable to SQL injection attacks. Our web application includes the common mistakes made by many web developers. Student's goal is to find ways to exploit the SQL injection vulnerabilities, demonstrate the damage that can be achieved by the attack, and master the techniques that can help defend against such type of attacks. This lab covers the following topics:

- SQL statement: SELECT and UPDATE statements
- SQL injection

2 Lab Environment

You will need a web application and a database. We have provided docker files for both. Extract the zip file and run the following command to bring the application and DB up:

```
$ docker compose up -d
```

The link for the zip file is: Docker Zip

3 Lab Task

We have created a web application and hosted it at 10.9.0.5. This web application is a simple employee management application. Employees can view and update their personal

information in the database through this web application. There are mainly two roles in this web applica tion: Administrator is a privileged role and can read and modify each individual employee's profile information; Employee is a normal role and can view or update his/her own profile information. All employee information is described in the following table.

Name	Employee ID	Password	Salary	Birthday	SSN	Nickname	Email	Address	Phone
Admin Alice Boby Ryan Samy Ted	99999 10000 20000 30000 40000 50000	seedadmin seedalice seedboby seedryan seedsamy seedted	400000 20000 50000 90000 40000 110000	3/5 9/20 4/20 4/10 1/11 11/3	43254314 10211002 10213352 32193525 32111111 24343244				

4 Problems

1. Get Familiar with SQL Statements

   The objective of this task is to get familiar with SQL commands by playing with the provided database. We have created a database called Users, which contains a table called credential; the table stores the personal information (e.g. eid, password, salary, SSN, etc.) of every employee. In this task, you need to play with the database to get familiar with SQL queries.

   MySQL is an open-source relational database management system. We have already setup MySQL in our SEEDUbuntu VM image. The user name is root and password is see dubuntu. Please login to MySQL console using the following command:

   ```
   $ mysql -u root -p seedubuntu
   ```

   After login, you can create new database or load an existing one. As we have already created the Users database for you, you just need to load this existing database using the following command:

   ```
   mysql> use Users;
   ```

   To show what tables are there in the Users database, you can use the following command to print out all the tables of the selected database.

mysql> show tables;

After running the commands above, you need to use a SQL command to print all the profile information of the employee Alice. Please provide the screenshot of your results.

2. SQL Injection Attack on SELECT Statement

SQL injection is basically a technique through which attackers can execute their own malicious SQL statements generally referred as malicious payload. Through the malicious SQL statements, attackers can steal information from the victim database; even worse, they may be able to make changes to the database. Our employee management web application has SQL injection vulnerabilities.

We will use the login page from 10.9.0.5 for this task. It asks users to provide a username and password. The web application authenticates users based on these two pieces of data, so only employees who know their passwords are allowed to log in. Your job, as an attacker, is to log into the web application without knowing any employee's credentials. To help you start with this task, we explain how authentication is implemented in the web application. The PHP code unsafe home.php, located in the /var/www/SQLInjection directory, is used to conduct user authentication. The following code snippet shows how users are authenticated.

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email, nickname,
Password FROM credential WHERE name= '$input_uname' and
Password='$hashed_pwd'";
$result = $conn -> query($sql);
// The following is Pseudo Code
if(id != NULL) {
if(name=='admin') {
return All employees information;
} else if (name !=NULL){
return employee information;
}
} else {
Authentication Fails;
}
```

The above SQL statement selects personal employee information such as id, name,

salary, SSN, etc from the credential table. The SQL statement uses two variables input uname and hashed pwd, where input uname holds the string typed by users in the username field of the login page, while hashed pwd holds the sha1 hash of the password typed by

the user. The program checks whether any record matches with the provided username and password; if there is a match, the user is successfully authenticated and is given the corresponding employee information. If there is no match, the authentication fails.

2.1: SQL Injection Attack from webpage: Your task is to log into the web applica tion 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 ad min, 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.

2.2: SQL Injection Attack from command line: Your task is to repeat Task 2.1, but you need to do it without using the webpage. You can use command line tools, such as curl (read 'curl' man page), which can send HTTP requests. One thing that is worth mentioning is that if you want to include multiple parameters in HTTP requests, you need to put the URL and the parameters between a pair of single quotes; otherwise, the special characters used to separate parameters (such as &) will be interpreted by the shell program, changing the meaning of the command. The following example shows how to send an HTTP GET request to our web application, with two parameters (username and Password) attached:

```
$ curl 'http://10.9.0.5/index.php?username=alice&Password=111'
```

2.3: Append a new SQL statement: In the above two attacks, we can only steal information from the database; it will be better if we can modify the database using the same vulnerability in the login page. An idea is to use the SQL injection attack to turn one SQL statement into two, with the second one being the update or delete statement. In SQL, a semicolon (;) is used to separate two SQL statements. Please describe how you can use the login page to get the server run two SQL statements. Try the attack to delete a record from the database, and describe your observation.

3. SQL Injection Attack on UPDATE Statement

If a SQL injection vulnerability happens to an UPDATE statement, the damage will be more severe, because attackers can use the vulnerability to modify databases. In our Employee Management application, there is an Edit Profile page that allows employees to update their profile information, including nickname, email, address, phone number, and password. To go to this page, employees need to log in first.

When employees update their information through the Edit Profile page, the following SQL UPDATE query will be executed. The PHP code implemented in unsafe edit backend.php file is used to update employees' profile information. The PHP file is

located in the /var/www/SQLInjection directory.

```
$hashed_pwd = sha1($input_pwd);
$sql = "UPDATE credential SET
nickname='$input_nickname',
```



```
email='$input_email',
address='$input_address',
Password='$hashed_pwd',
PhoneNumber='$input_phonenumber'
WHERE ID=$id;";
$conn->query($sql);
```

3.1: Modify your own salary: As shown in the Edit Profile page, employees can only update their nicknames, emails, addresses, phone numbers, and passwords; they are not authorized to change their salaries. 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 Edit-Profile page. Please demonstrate how you can achieve that. We assume that you do know that salaries are stored in a column called 'salary'.

3.2: Modify other people's 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.

3.3: Modify other people's password: After changing Boby's salary, you are still disgruntled, so you want to change Boby's password to something that you know, and then you can log into his account and do further damage. Please demonstrate how you can achieve that. You need to demonstrate that you can successfully log into Boby's account using the new password. One thing worth mentioning here is that the database stores the hash value of passwords instead of the plaintext password string. You can again look at the unsafe edit backend.php code to see how the password is being stored. It uses the SHA1 hash function to generate the hash value of the password.

To make sure your injection string does not contain any syntax error, you can test your injection string on the MySQL console before launching the real attack on our web appli cation.

5 Submission

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. Give your own assessment, of what you have learned from this assignment. You also have to submit all the codes you have written with the necessary

comments to describe your code. Submit a single zip folder containing the report and properly commented code. The name of the zip file should be ⟨Entry No.⟩Assg2.zip.

Best wishes