CS253A: Software Development and Operations

Assignment 4

REPORT

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Details of the weakness as YOU understand it?

Generally, SQL databases consist of very sensitive data and it is very easy to make SQL injection attacks. The frequent problem with SQL injection attacks is the loss of confidential data. Attackers can use SQL injection to find confidential data of other users present in the database. This vulnerability also allows an attacker to change the data by inserting or deleting information present in the data. Already present information in data can also be altered using SQL injection attack. It can give attackers an access to the complete data.

The attack model under which you have built your exploit. Is there any specific requirement for the bug to be exploited (specific operating system, hardware, libraries, etc.)?

The attack model under which I have built my exploit is CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection').

The attack designed by me requires the environment to run a bash script and assumes that a MySQL database is hosted at the host's end which is the target of my attack. The data is fetched normally from the hosted database by the client using a bash script.

How to run your vulnerable system to expose the exploit.

My vulnerable system could be run only at the host end. My deliverables consist of database “Covid_Report.sql” which I exported from PHPMyAdmin and it contains the database, a bash script “script.sh” which has the vulnerable program, a bash script “corrected_script.sh” which has the mitigated program, “T.in” which contains all the input that is used to exploit my vulnerable system, an automated script “main.sh” which is used to run all the test cases and “output.txt” which contains the output of the exploitation. Simply running bash main.sh in the terminal and then pressing enter each time when asked for the password will run my program and expose the exploit.

Explain how your exploit works (in the accompanying program/system).

Firstly, I created a database (Covid_Report.sql) in phpMyadmin and then wrote a bash script that takes two inputs, Test Number and Password, and produces an output that shows Name, Test_No, Password, and Covid_Report of a person with that particular test number. Now, to make an SQL injection attack I found some vulnerable user inputs by trying different-different inputs. So, the inputs on which my exploit would work contain some value followed by an SQL query. So, these input fields are vulnerable to SQL injection because they are using SQL commands in the input in such a way that would alter the SQL statement executed by my database server. I have taken five different inputs (written in file T.in) to test this vulnerability on my program. So, now let’s explain each test case individually.

1st Input:

Test No.- 10000001 OR 1;
Password- anything

This input will return the complete database. In this case database server runs the SQL query **"SELECT * FROM Covid_Report WHERE Test_No=10000001 OR 1; AND Password=anything";** so no matter what Test No. and Password is given, the OR 1 statement is always true so this SQL query will return the entire table Covid_Report.

2nd Input:

Test No.- 10000001; DROP TABLE USERS;
Password- anything

This input will return the report of a person with Test No. 10000001 without even entering the correct password. As our database is supporting two SQL statements separated by semicolons, therefore DROP TABLE USERS; will get executed after the assignment of Test No.

3rd Input:

Test No.- 10000001; INSERT INTO Covid_Report
VALUES('Lipi',10000026,'@10000026@','NO');
Password- anything

This input will return details of the patient with Test No. 10000001 and will also insert one more row (with specified details) into the database, and the reason is same as mentioned in test case 2. The use of this type of command can alter data very easily.

4th Input:

Test No.- 10000001
Password- anything' OR 1;

This input will return the entire database. It is because of input anything' OR 1; as password. The string is ended at a single quote (') after which OR 1 command gets executed due to which the entire database gets printed.

5th Input:

Test No.- 10000001; DELETE FROM Covid_Report WHERE Test_No=10000026;
Password- anything

This input will return details of the patient with Test No. 10000001 and will also delete one row (with Test No. 10000026) from the database, and reason is the same.

Explain how to mitigate this weakness, i.e. exactly pin-point what was wrong in the program/system and what is the way the program should have been written.

The main problem with the system is that it is allowing two or more semicolon-separated SQL command to get executed when given as input. Using this, an attacker can manipulate, insert, delete or see the database which is a great threat to confidentiality and security. So to correct this problem we have to write our program in such a way that it allows input only in the required format and all the other inputs get rejected. In my code, I used two inputs, so let's look at both of them individually.

1. **1st input is Test No.** which is an eight-digit integer. So, firstly I checked whether it is an INT or not using $Number =~ ^[+-]?[0-9]+$ command and then checked whether it is eight-digit or not. If the input satisfies both of the conditions, then only all further commands will get executed otherwise the program will get terminated.
2. **2nd input is the password** which contains a string. It will give us trouble only when a single quote(‘) is present in it. So, I used for loop to iterate over the whole string and checked whether ‘ is present in the string or not. If it is present then the program gets terminated otherwise further commands get executed.

I wrote a script named corrected_script.sh which corrects my vulnerable program.

References

- https://cwe.mitre.org/data/definitions/89.html
- https://www.w3schools.com/sql/sql_injection.asp
- https://www.w3schools.com/php/php_mysql_intro.asp
- https://www.javatpoint.com/mysql-tutorial
- https://www.tutorialspoint.com/mysql/index.htm
- https://askubuntu.com/questions/1174142/shell-script-to-check-if-input-is-a-string-integer-float
- https://linuxhint.com/length_of_string_bash/