Lab-6 Report

Akhilesh Gowda Mandya Ramesh

***Abstract*—The Lab-6 introduced SQL Injection(mitigation) using webgoat as to how to try and avoid the website susceptible to SQL injection.**

I. Introduction

THE insecure application that provides functionalities to observe the vulnerabilities of applications, such as WebGoat can be used to exploit its vulnerabilities using SQL injection, this can also be used to study how to avoid SQL Injection. SQL injection typically happens when you request user input, such as their username or userid, and instead of receiving a name or id, the user instead provides you with a SQL statement that you will unwittingly execute on your database. This lab is more about how to avoid SQL injection.

II. Immutable Queries

The best defence for SQL injection can be formed by using immutable queries. This is because they either do not have the data that would be interpreted or they treat the data as a single entity that is bound to the column without interpretation. This can be done using static, queries, dynamic queries and stored procedures oly if it does not generate dynamic SQL.

III. Task Parameterized queries

Task 5 was to complete the incomplete code to make to not susceptible to SQL injection. The code is to retrieve the status of the user based on their name and mail address both of which are in string format. using java programming language i used getConnection which sets up a connection object can be used to create SQL statements. I later used PreparedStatement which is a pre-compiled SQL statement which can be used to write parameterized SQL queries. The next task was task-6 where we were demanded to write a code that uses JDBC to connect to a database so that it can request data from it. The code is fairly similar to the previous task but we need to make use of driver manager to connect to the database. This code has to be run in the terminal provided by webgoat in-order to complete the task. With the help of this technique, the database will be able to recognize the code and tell it apart from input data. This coding approach helps to mitigate a SQL injection attack because user input is automatically quoted and will not result in the intent changing.

IV. Input Validation

The question now is if we now need to validate our input since its no longer subjected to SQL injection. The simple answer is yes as it will help avoid other forms of threats such as stored XSS, Logic errors and information leakage. The next

Advisor: Luyi Xing, Jiale Guan.

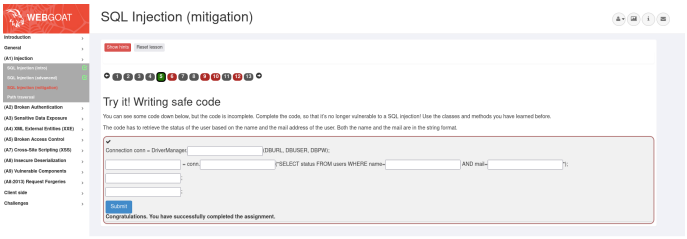

Fig. 1. Task 5 completion

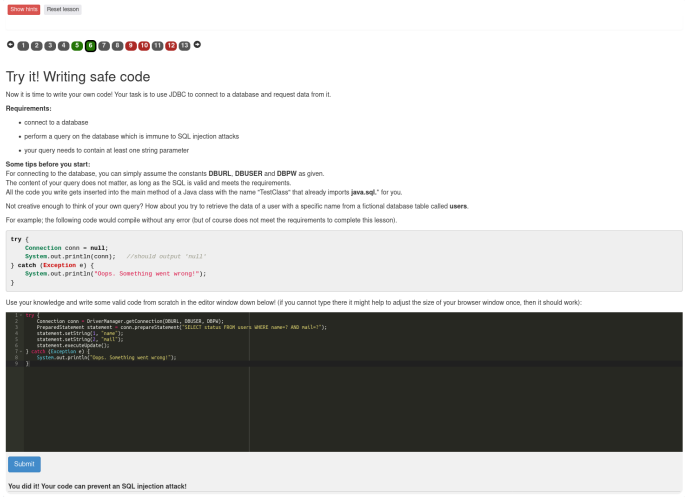

Fig. 2. task-6 Connecting to database using JDBC

task here is to use parameterized query and to validate the input which is received from the user. The challenge here was interesting as it throws an error if there is a space in the input field. Hence i replace the spaces with comments so that it does not throw an error and it worked. The next task took the input validation to a step further as it now detects white spaces, i tried several different ways to overcome this technique, but in the end was able to nest the select statements as it does not check recursively and was bale to complete the task.

V. Order By

The order expression can act as a select expression, which can be a function as well. This can be a threat as case statement anyone can ask database some questions, by substituting any kind of Boolean operation in When part the attacker can make the statement a valid query whether we use a prepared statement or not. This way order by clause can be made to have an expression. To mitigate the threats caused order by we need to provide a sorting column ans should implement a whitelist to validate the value of order by statement.

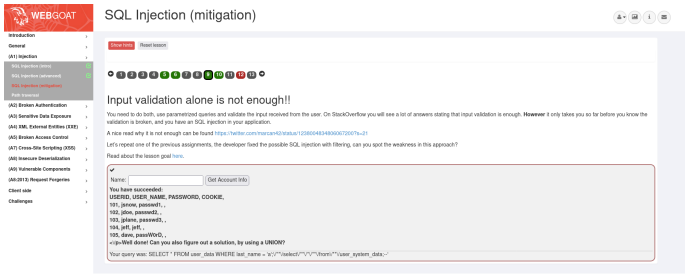

Fig. 3. Input validation Task

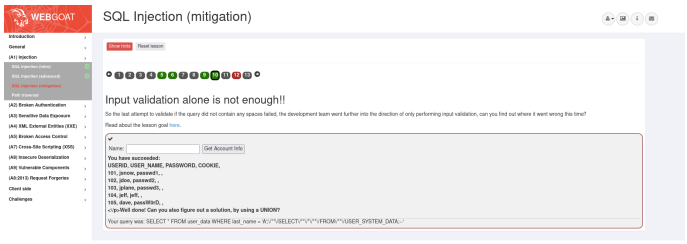

Fig. 4. input validation -2 completion

VI. Conclusion

In Summary as WebGoats vulnerabilities can be exploited to study and bolster the attacks, We can also use it to study and mitigate the attacks such caused by SQL injection as the CIA Triads can be compromised by performing SQL Injection in various was such as string SQL Injection and Numeric SQL injection and also by combining SQL statements with UNION or JOIN. This can be very harmful or costly for the organizations as they will result in a very harsh damage. Using techniques like parameterized query, Input validation and using white list for the order by clause we can mitigate SQL injection attacks. With the help of parameterized query technique, the database will be able to recognize the code and tell it apart from input data. This coding approach helps to mitigate a SQL injection attack because user input is automatically quoted and will not result in the intent changing. To mitigate the threats caused order by we need to provide a sorting column ans should implement a whitelist to validate the value of order by statement. Hence using all these techniques we can build web applications not vulnerable to SQL injection.

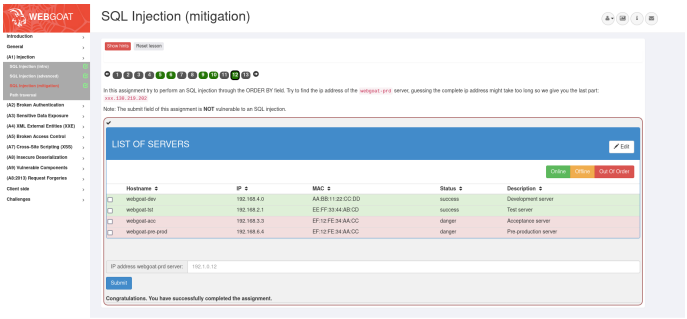

Fig. 5. order by task completion

References

[1] https://github.com/WebGoat/WebGoat
[2] https://owasp.org/
[3] https://www.zaproxy.org