xbrutovskyf
FIIT STU

Protection against SQL injection attacks

Table of Contents

1. Introduction

SQL injection (SQLi) is a vulnerability that allows attacker-controlled input to alter database queries. It can cause data leakage, authentication bypass, data modification and system compromise.

a. Types of SQLi

In-band SQLi (Classic SQLi) Attacker uses same channel for launching an attack and gathering results. It can be:

- **Error based** – when an attacker can learn about the database schema, names, types or SQL structure from error messages, that information turns a blind probe into a targeted exploit. Error messages leak *context* — they tell the attacker what the database expects, which column or table names exist, which SQL dialect is used, and sometimes even snippets of the offending SQL.

- **Union based** – when an attacker appends a UNION SELECT to the original SQL query. If the application outputs the results of the query to the page, the UNION lets the attacker combine a crafted SELECT (that reads attacker-chosen data) with the original query's results allowing the attacker to make the application display data from other tables.

Inferential SQLi (Blind SQLi) Attacker sending payloads and observing web responses and the resulting behaviour of the database server. It generally takes longer than in-band methods but is as dangerous as they are. For example:

- **Boolean based** – when attacker is sending an SQL query to the database which forces the application to return a different result depending on whether the query returns a true or false result.
- **Time based** – when attacker uses payloads which delay database responses when guessed condition is true. t's slow, but reliable and very useful when other techniques (UNION, error-based) are not possible.

Out-of-band SQLi Attacker forces the database server to communicate with an attacker-controlled service outside the normal request/response channel. Instead of returning data in the web page (in-band) or inferring it via boolean/time differences (blind), the DB contacts an external server (DNS, HTTP, SMB, etc.) and delivers data through that separate channel. The attacker watches that external channel for callbacks containing leaked information.

2. Tools

a. Burp suite

Burp Suite is a web-security testing platform that sits between browser and the web server. I'll use it to check for the HTTP traffic and observe it. It can be also used to send modified requests manually.

b. sqlmap

Sqlmap is opensource penetration testing tool for SQL injection. It automatise attack and tries to take control over the database. I'll use it to test integrity of unsecured webapp and then secured webapp.

3. Objectives

a. Creating the vulnerable web application to simulate the SQL injection attacks

I will build a small Flask app with at least one endpoint that accepts user input and queries a database in an insecure way so that SQL injection can be demonstrated.

b. Securing the web application so it is attack-proof

I'll apply defensive measures (parameterized queries/ORM, input validation, least-privilege DB user, proper error handling) and remove the vulnerabilities introduced in the vulnerable version.

c. Trying the same attack on the secured web application

My goal will be to reproduce the original tests and confirm that the attacks no longer succeed. Document test cases and results.

4. Timeline

Event	**Date**	**Content**
First progress report	7th week	Complete theoretical part of the project, analysis of used tools and clear plan for next part.
Second progress report	11th week	Complete experimental architecture and first test on vulnerable web application done.
Final report	12th week	Both theoretical and experimental part complete with evaluation of results and original solutions.

5. Experimental environment

a. Python Flask

Flask is lightweight python-based framework for web applications. It’s easy to use and scale so perfect for my vulnerable test app and later for secured web application.

b. PostgreSQL

Postgres is Database management system (DBMS) developed at university of California.
It can be linked to Flask based application and I will use it to build an exploitable database.

6. Used resources

- Hussein, Dina & Ibrahim, Dina & Alsalamah, Mona. (2021). A Review Study on SQL Injection Attacks, Prevention, and Detection. 13. 1-9.
- What is SQL Injection (SQLi) and How to Prevent Attacks
- Types of SQL Injection?