SQL Injection Vulnerability Assessment Report

1. Introduction

SQL Injection (SQLi) is a critical security vulnerability that occurs when an attacker is able to manipulate a web application's database query by injecting malicious SQL code through user input fields. This report provides a comprehensive overview of the SQL Injection vulnerability identified in the application, including the exploitation process, and recommendations for mitigation.

2. Vulnerability Details

Vulnerability Type: SQL Injection

Affected Application Components:

Login Form

Search Bar

Description:

SQL Injection vulnerabilities were found in the application's login form and search bar. These vulnerabilities allow an attacker to manipulate SQL queries by injecting malicious SQL commands into user input fields, potentially exposing or modifying sensitive data in the database.

Example Vulnerable Input:

Login Form: Username field

Search Bar: Query parameter

3. Exploitation Process

3.1 Manual Testing

Testing the Login Form:

Input: Username: admin' --

Result: The application returned an SQL error message indicating that the query was malformed.

Testing the Search Bar:

Input: Search Query: test' OR '1'='1

Result: The search results displayed data from all database entries, indicating that the SQL query was manipulated.

3.2 Automated Testing with SQLMap

Setup:

Downloaded and installed SQLMap.

Scanning for SQL Injection:

Command Used: `sqlmap -u "http://example.com/page?param=value" --dbs`

Result: SQLMap identified that the parameter was vulnerable to SQL Injection.

Database Enumeration:

Command Used: `sqlmap -u "http://example.com/page?param=value" --tables`

Result: SQLMap listed tables from the affected database, confirming the vulnerability.

Screenshot:

4. Mitigation Recommendations

4.1 Sanitize and Validate Input

Input Validation: Ensure all user inputs are validated against expected formats. For example, only allow alphanumeric characters in usernames and restrict special characters in search queries.

Prepared Statements: Use parameterized queries to prevent SQL injection by separating SQL code from user inputs.

Stored Procedures: Use stored procedures to handle database interactions securely.

Escaping Inputs: Properly escape user inputs to neutralize special characters that could be interpreted as SQL code.

4.2 Regular Security Reviews and Updates

Code Reviews: Conduct regular security code reviews to identify and remediate vulnerabilities.

Update Libraries: Keep all related libraries and frameworks up-to-date to benefit from the latest security patches.

5. Conclusion

The assessment identified SQL Injection vulnerabilities in the login form and search bar of the application. These vulnerabilities pose a significant risk as they allow attackers to manipulate SQL queries and potentially access or modify sensitive database information. By implementing the recommended mitigation strategies, such as input validation, prepared statements, and regular security reviews, the risk of SQL Injection can be significantly reduced.