Report

LeshanWang 201911023

1. SQL Injection

1.1 Overview

Through testing, SQL injection risk has been detected in this application. It is one of the most common web hacking techniques, by placement of malicious code in SQL statement through web page input, hackers can damage and destroy the database.

In this application, the user is required to enter their login credentials to gain access to the database and conduct the search. After getting the information as string, the system will construct a SQL command to execute. However, the original method used is vulnerable to SQL injection attacks. During the test, ‘nde’ and ‘wysiwyg0’ were supposed to be entered as valid username and password; instead, ‘’ OR 1=1 /*’ was entered as username. This resulted in SQL injection as in backend, ‘select * from user where username= ‘ ’ OR 1=1 /* and password= ‘ ’ was constructed and executed. The SQL command above is valid and will return all rows from the user table, since’ OR 1=1 /*’ is always true and will block the following query, password is no longer needed. Then, the hacker can enter a blank username and password but has direct access to the database.

1.2 Changes

In order to prevent SQL injection hacking, Prepared Statement is implemented as a parameterized and reusable query which separates the SQL command and user-provided data.

```
1 usage
private static final String SEARCH_QUERY = "select * from patient where surname like ?";
```

```
private static final String AUTH_QUERY = "select * from user where username=? and password=?";
```

```
try (PreparedStatement ps = database.prepareStatement(AUTH_QUERY)) {
    ps.setString( parameterIndex: 1, username);
    ps.setString( parameterIndex: 2, password);
    ResultSet results = ps.executeQuery();
    return results.next();
}
```

```
try (PreparedStatement stmt = database.prepareStatement(SEARCH_QUERY)) {
    stmt.setString( parameterIndex: 1, surname);
    ResultSet results = stmt.executeQuery();
```

The idea is that by binding data as variables, this method prevents adding data directly the program which might alter the program. During operation, the database engine detects the placeholders, and the query is compiled with placeholders. Then the query is stored in cache for later use. After caching, the placeholder is replaced with the user's data, since the query is pre-compiled, the final query will not be compiled again thus user data will be interpreted as string, which cannot modify the query's logic. Finally, the completed query is executed. This process actively prevents SQL injection.

2. Encryption

2.1 Overview

```
private boolean authenticated(String username, String password) throws SQLException {
```

After observing the codes, it is clear that the username and password are stored in plaintext strings, not hashed, which is dangerous if the hacker has means to break into database, for example using SQL injection if available, all users account could be accessed and stolen. In a worse scenario, many users who re-use a single password allow attackers to access other services as one being compromised. This can be solved if login credentials are hashed through hashing algorithm such as SHA, turning the plaintext into unintelligible series of numbers and letters, then stored in database.