UNIVERSITÀ DEGLI STUDI DI SALERNO

AUTOMATIC SOFTWARE VULNERABILITY TESTING

(SQL INJECTION)

COURSE OF STUDY SOFTWARE DEPENDABILITY

PROJECT SUPERVISOR
PROFESSOR FABIO PALOMBA

PROJECT ADVISOR
DR.LANNONE
DR.GIORDANO

GROUP MEMBERS

SYED HUSSNAIN RAZA RIZVI
0522500898

MUHAMMAD WASEEM
0522500899

ELSAYED ELSAYAD
0522500913

Abstract

Vulnerabilities in web applications are the flaws or shortcomings which can impact the security and performance of the web applications which could also lead the web applications to the possibility of being harmed or attacked. Testing of web applications and predicting the possible vulnerabilities and preventing them is a significant step to resolve this problem. Unfortunately, it is not possible to investigate every vulnerability with any specific testing technique because there are several types of vulnerabilities that could be present in a web application and for each category of vulnerability there is a specific testing technique. The Open Web Application Security Project (OWASP) has recently created a guide to security test web applications, and according to that guide they suggested several testing techniques to spot single vulnerabilities. In this project, we worked on the testing technique to investigate the possible vulnerabilities caused by the SQL Injection. Moreover, we investigate automation features available in these work across a security testing process.

Introduction

Today there are large number of web applications that are widely available and hundreds of millions of people have access to those web applications worldwide. The web applications are usually developed by using implementation languages (e.g., ANSI C), library functions (e.g., ANSI C library, Java API), processors (e.g., SQL query engine, HTML parser, JavaScript engine, etc.) that often suffer from inherent vulnerabilities such as buffer overflow, SQL Injection [1], format string bug, and cross site scripting (XSS). Moreover, these applications are not always used by legitimate users in a legitimate manner. As a result, exploitations of these known vulnerabilities through successful attacks are very common.

According to the OWASP (Open Web Application Security Project), among several vulnerabilities the SQL injection is one of the most common and threatening in the web applications [2]. Normally, by utilizing the SQL injection, the databases and the web servers are attacked by the attackers and with the help of this technique the content of webpage can be manipulated and the possibility to getting stollen an important information of a person or any organization. Therefore, it's significantly important to test the web applications against the vulnerability of SQL injection and make them more dependable and reliable for the users or the organizations.

The practice of developing secure application has been established for more than a decade ago. Several widely complementary techniques of testing have been established to detect and prevent vulnerabilities. These include static analysis tools [3] to identify vulnerable code, combined static analysis and runtime monitoring approach, automatic fixing of vulnerable code, etc.

In this work, we implemented a testing technique on different java-based web applications and analyzed the vulnerabilities of SQL injection in each of those web applications.

Methodological Steps of Testing

For testing of most of the open source java-based web applications we used an IDE (integrated development environment) named as **IntelliJ** which is an essential framework for developers in terms of testing and analyzing the vulnerabilities that could be occurred in the web applications. And for testing of some of the projects we used **NetBeans** IDE.

IntelliJ: IntelliJ IDEA is an Integrated Development Environment, by JetBrains. It attempts to integrate all the development and testing tools that you might need into one single place. IntelliJ IDEA is the most-used IDE in the Java community, with 45% of respondents stating that they use IntelliJ IDEA. IntelliJ IDEA framework have many different tools and plugins that are being used in testing the vulnerabilities in web applications. In our work, we used the plugin named as SpotBugs for the testing of the open source java-based web applications.

- **SpotBugs:** IntelliJ SpotBugs plugin provides static byte code analysis to look for bugs in Java code from within IntelliJ IDEA. SpotBugs is a defect detection tool for Java that uses static analysis to look for more than 400 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks. SpotBugs can identify hundreds of serious defects in large applications (typically about 1 defect per 1000-2000 lines of non-commenting source statements).

NetBeans: NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris. We used NetBeans framework for the testing of java-based web applications and analyzes the vulnerabilities with the help of FindBugs plugin.

- **FindBugs:** FindBugs is an open source defect detection tool designed to find bugs in Java programs. FindBugs is looking for code instances that are likely to be errors called “bug patterns”. FindBugs uses static analysis to examine the code by matching bytecodes against a list of more than 200 bug patterns, such as null pointer dereferences, infinite recursive loops, bad uses of the Java libraries and deadlocks.

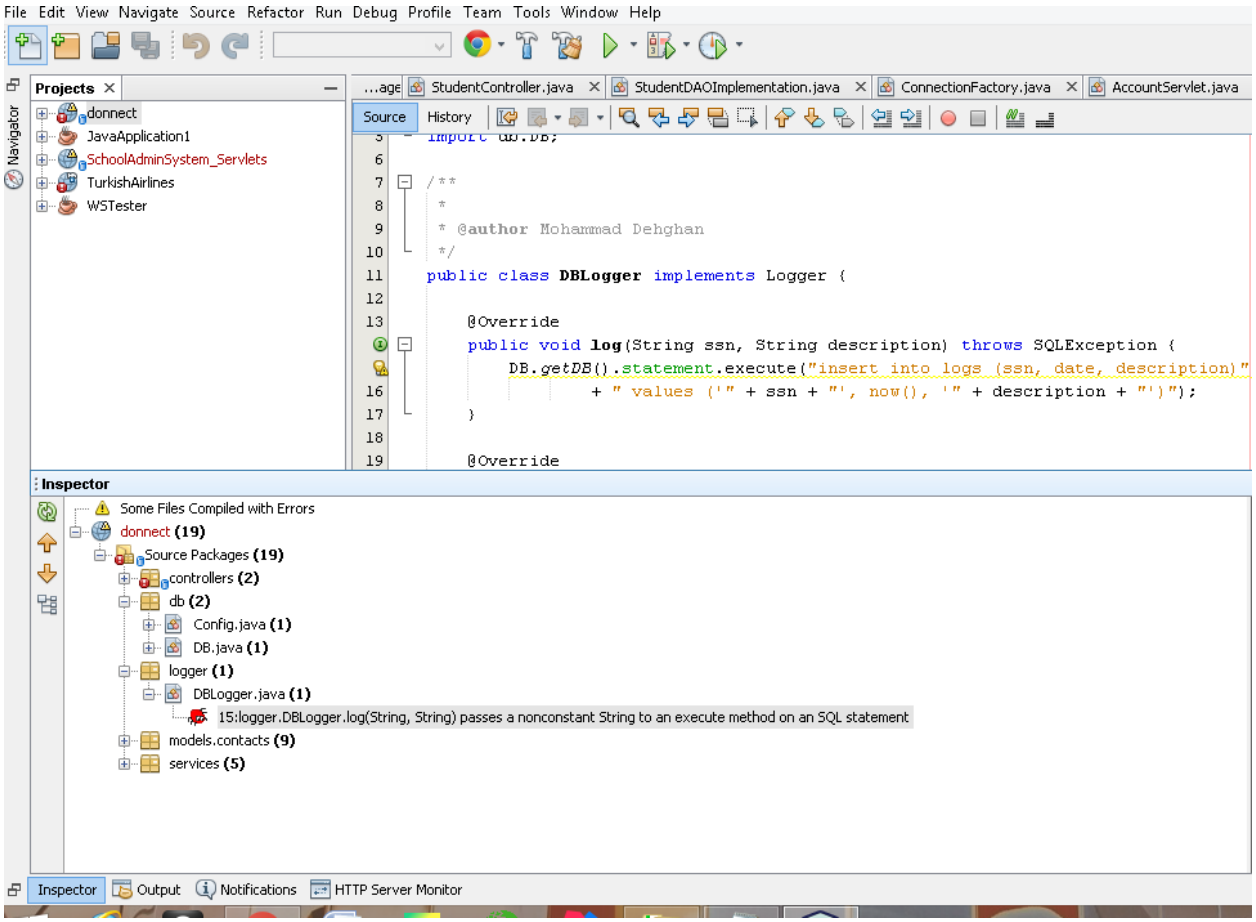
Testing of Projects

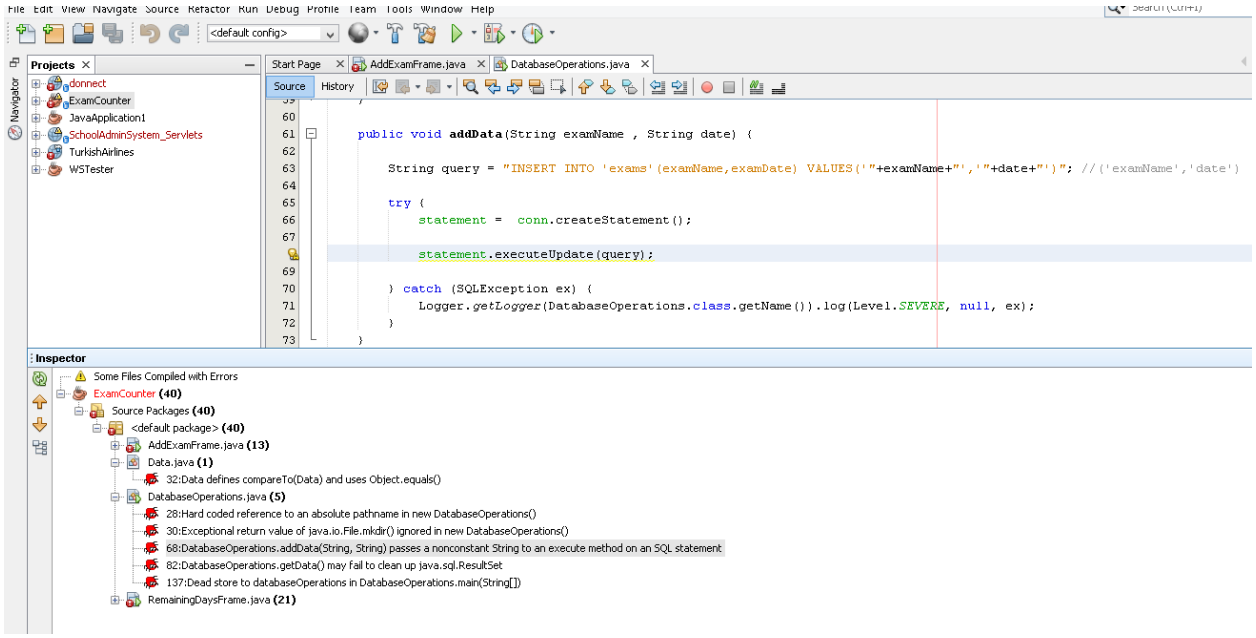
In testing phase, we worked on 15 open source java-based web applications and analyzed the vulnerabilities in them by using the IntelliJ IDEA framework and SpotBugs plugin. Following are the projects on which we have worked in testing phase of our project:

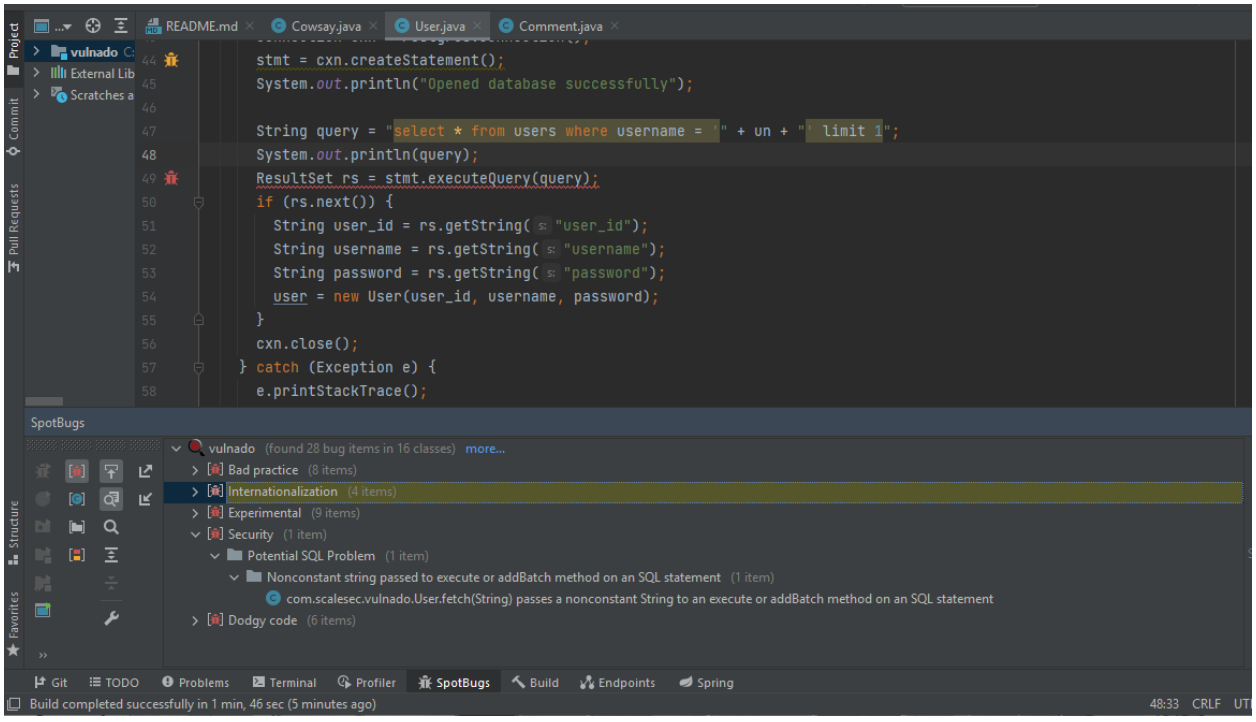
1- **Store Web App [4]**
2- **Vulnado [5]**
3- **QA [6]**
4- **Fligh Booking [7]**
5- **Donnect [8]**
6- **ExamCounter [9]**
7- **SMBMS [10]**
8- **SchoolAdminSystem Spring [11]**
9- **Health Plus [12]**
10- **Hospital Managmengt System [13]**
11- **Transport Automatios System [14]**
12- **ATM-Machine [15]**
13- **Exam Counter [16]**
14- **Hotel-Management-Project-Java [17]**
15- **New-Smart-Bus-Ticket-Management-System [18]**

Testing Results

We did testing on several projects, In which we found some vulnerabilities of SQL injection which are show below:




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
donnect
ExamCounter
JavaApplication1
SchoolAdminSystem_Servlets
TurkishAirlines
WSTester
Start Page
AddExamFrame.java
DatabaseOperations.java
Source
History
public void addData(String examName , String date) {
String query = "INSERT INTO 'exams'(examName,examDate) VALUES('"+examName+"','"+date+"')"; //('examName','date')
try {
statement = conn.createStatement();
statement.executeUpdate(query);
} catch (SQLException ex) {
Logger.getLogger(DatabaseOperations.class.getName()).log(Level.SEVERE, null, ex);
}
}
Inspector
Some Files Compiled with Errors
ExamCounter (40)
Source Packages (40)
<default package> (40)
AddExamFrame.java (13)
Data.java (1)
32:Data defines compareTo(Data) and uses Object.equals()
DatabaseOperations.java (5)
28:Hard coded reference to an absolute pathname in new DatabaseOperations()
30:Exceptional return value of java.io.File.mkdir() ignored in new DatabaseOperations()
68:DatabaseOperations.addData(String, String) passes a nonconstant String to an execute method on an SQL statement
82:DatabaseOperations.getData() may fail to clean up java.sql.ResultSet
137:Dead store to databaseOperations in DatabaseOperations.main(String[])
RemainingDaysFrame.java (21)


README.md
Cowsay.java
User.java
Comment.java
stmt = cxn.createStatement();
System.out.println("Opened database successfully");
String query = "select * from users where username = '" + un + "' limit 1";
System.out.println(query);
ResultSet rs = stmt.executeQuery(query);
if (rs.next()) {
String user_id = rs.getString(s: "user_id");
String username = rs.getString(s: "username");
String password = rs.getString(s: "password");
user = new User(user_id, username, password);
}
cxn.close();
} catch (Exception e) {
e.printStackTrace();
SpotBugs
vulnado (found 28 bug items in 16 classes) more...
Bad practice (8 items)
Internationalization (4 items)
Experimental (9 items)
Security (1 item)
Potential SQL Problem (1 item)
Nonconstant string passed to execute or addBatch method on an SQL statement (1 item)
com.scalesec.vulnado.User.fetch(String) passes a nonconstant String to an execute or addBatch method on an SQL statement
Dodgy code (6 items)
Git
TODO
Problems
Terminal
Profiler
SpotBugs
Build
Endpoints
Spring
Build completed successfully in 1 min, 46 sec (5 minutes ago)
48:33
CRLF

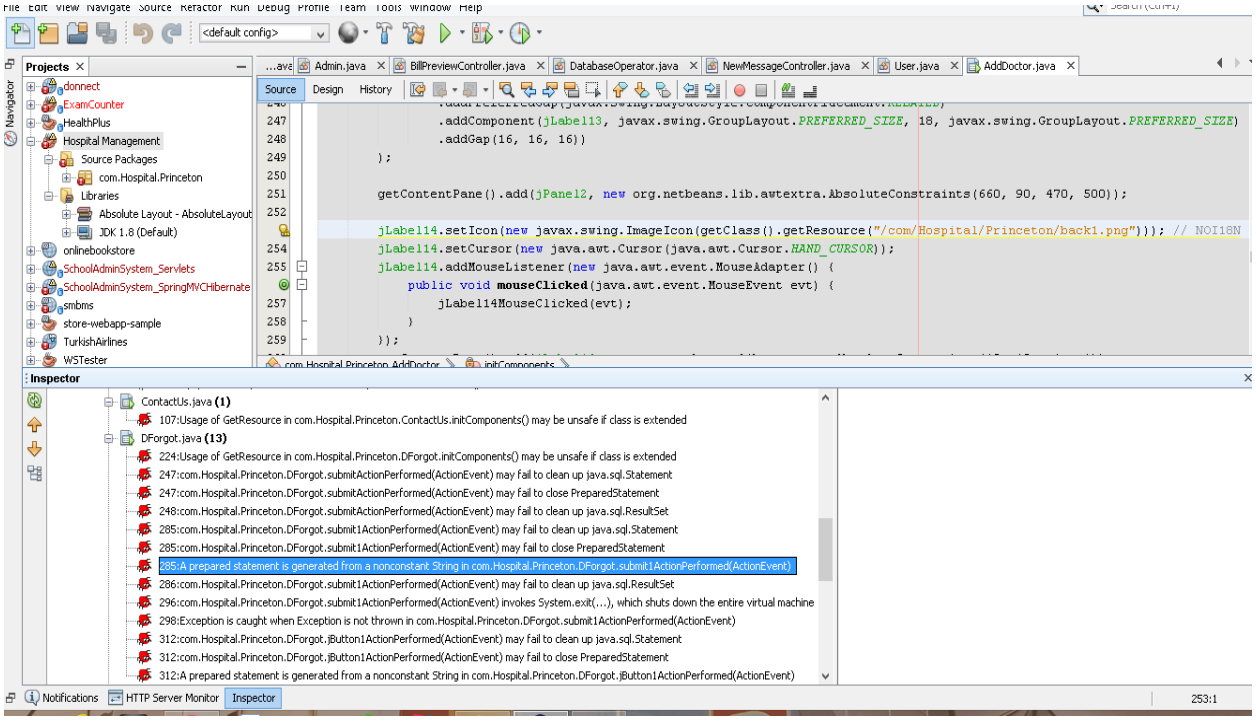
In above pictures we can see there is Some same vulnerability of SQL Injection in different projects, which is as;

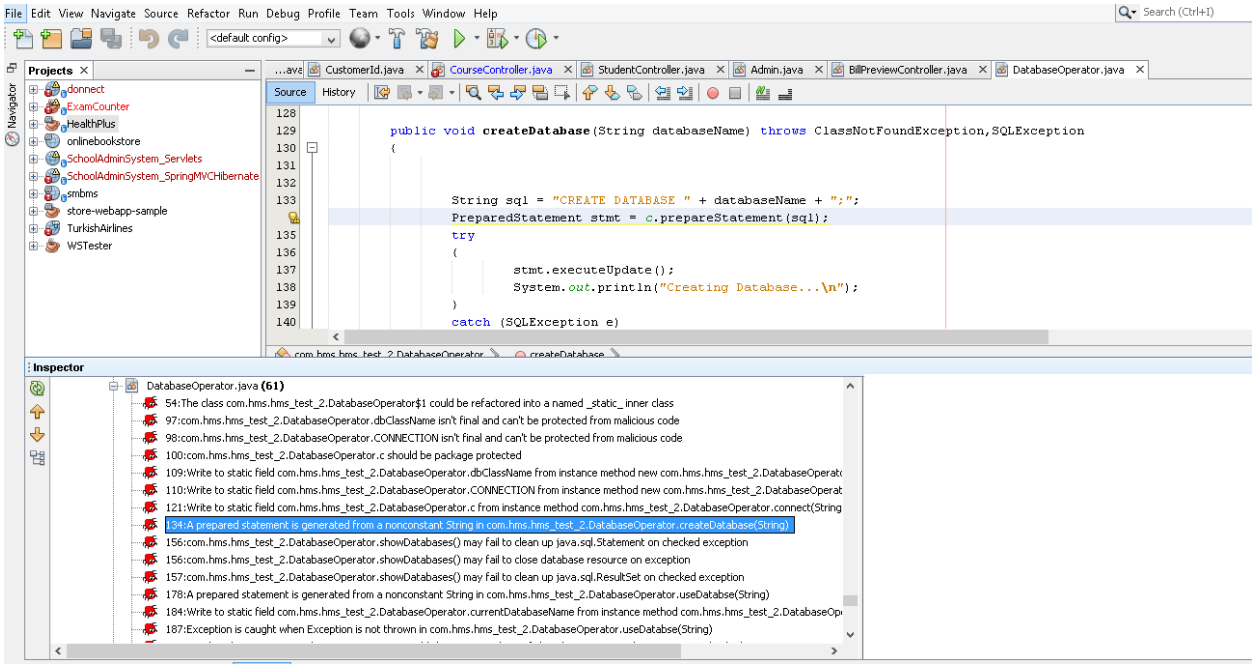
Non constant string passed to execute method on a SQL statement:

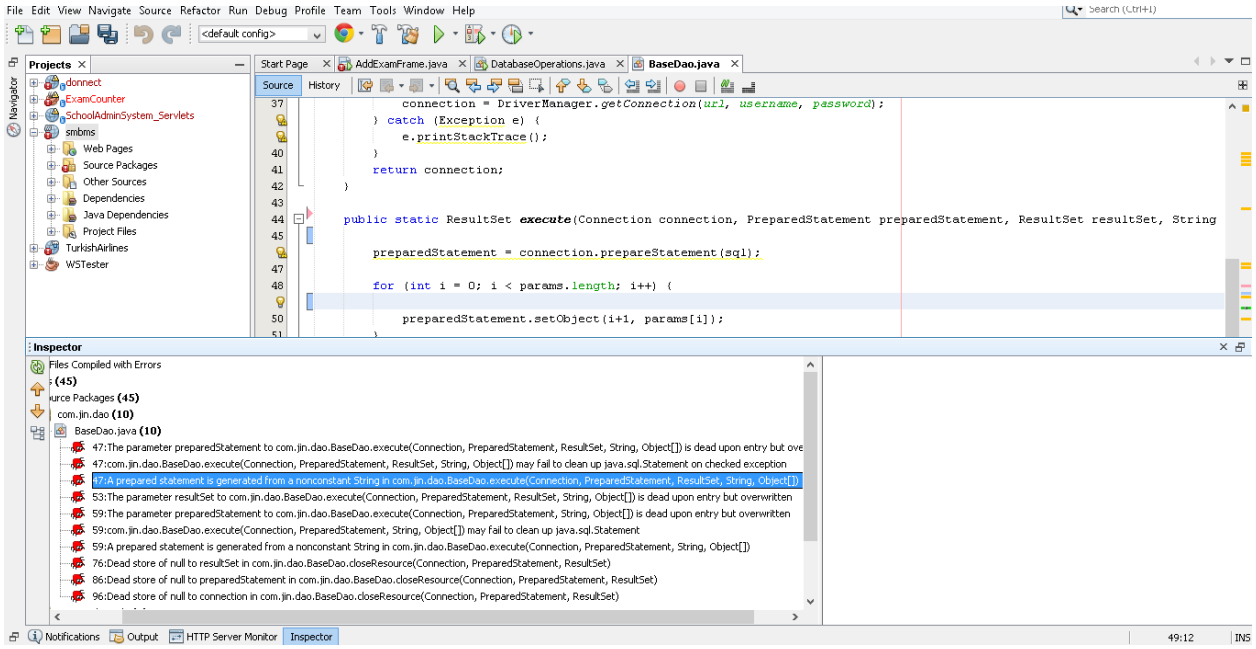
The method invokes the execute method on a SQL statement with a String that seems to be dynamically generated.

To avoid this vulnerability we can use a prepared statement instead. It is more efficient and less vulnerable to SQL injection attacks.

There are some others projects in which we found some vulnerabilities:

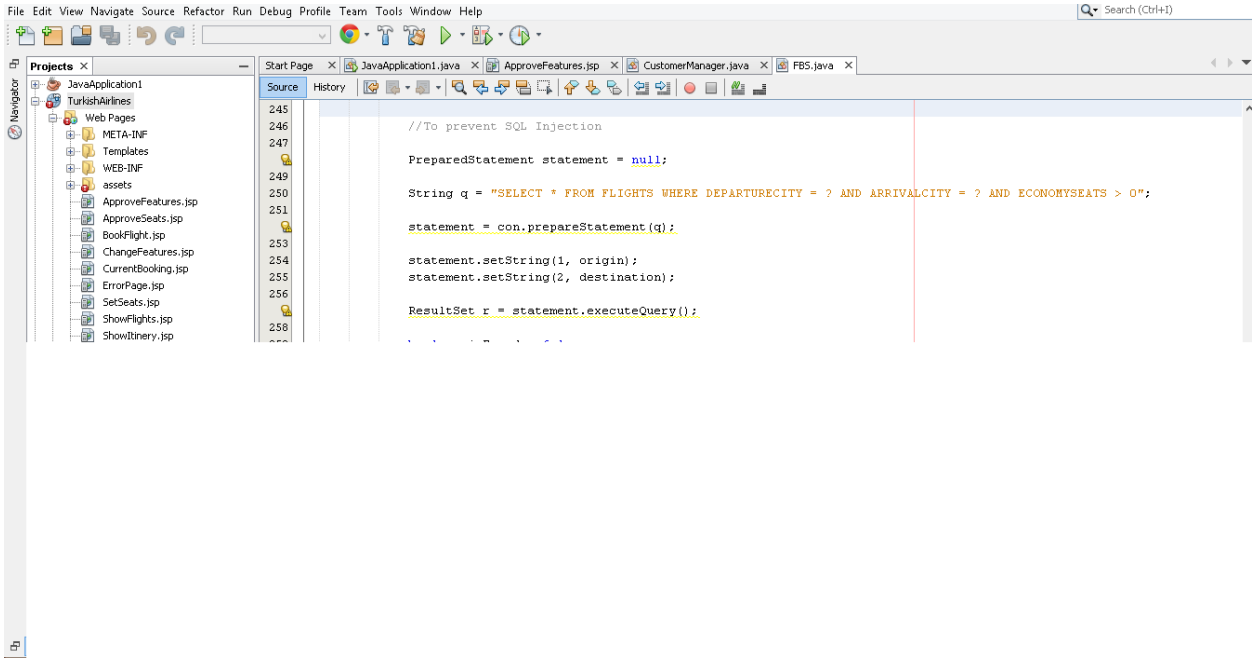



File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
donnect
ExamCounter
HealthPlus
onlinebookstore
SchoolAdminSystem_Servlets
SchoolAdminSystem_SpringMVCHibernate
smbms
store-webapp-sample
TurkishAirlines
WSTester
CustomerId.java
CourseController.java
StudentController.java
Admin.java
BillPreviewController.java
DatabaseOperator.java
Source
History
public void createDatabase(String databaseName) throws ClassNotFoundException,SQLException
{
String sql = "CREATE DATABASE " + databaseName + ";";
PreparedStatement stmt = c.prepareStatement(sql);
try
{
stmt.executeUpdate();
System.out.println("Creating Database...\n");
}
catch (SQLException e)
Inspector
DatabaseOperator.java (61)
54:The class com.hms.hms_test_2.DatabaseOperator$1 could be refactored into a named _static_ inner class
97:com.hms.hms_test_2.DatabaseOperator.dbClassName isn't final and can't be protected from malicious code
98:com.hms.hms_test_2.DatabaseOperator.CONNECTION isn't final and can't be protected from malicious code
100:com.hms.hms_test_2.DatabaseOperator.c should be package protected
109:Write to static field com.hms.hms_test_2.DatabaseOperator.dbClassName from instance method new com.hms.hms_test_2.DatabaseOperat
110:Write to static field com.hms.hms_test_2.DatabaseOperator.CONNECTION from instance method new com.hms.hms_test_2.DatabaseOperat
121:Write to static field com.hms.hms_test_2.DatabaseOperator.c from instance method com.hms.hms_test_2.DatabaseOperator.connect(String
134:A prepared statement is generated from a nonconstant String in com.hms.hms_test_2.DatabaseOperator.createDatabase(String)
156:com.hms.hms_test_2.DatabaseOperator.showDatabases() may fail to clean up java.sql.Statement on checked exception
156:com.hms.hms_test_2.DatabaseOperator.showDatabases() may fail to close database resource on exception
157:com.hms.hms_test_2.DatabaseOperator.showDatabases() may fail to clean up java.sql.ResultSet on checked exception
178:A prepared statement is generated from a nonconstant String in com.hms.hms_test_2.DatabaseOperator.useDatabse(String)
184:Write to static field com.hms.hms_test_2.DatabaseOperator.currentDatabaseName from instance method com.hms.hms_test_2.DatabaseOp
187:Exception is caught when Exception is not thrown in com.hms.hms_test_2.DatabaseOperator.useDatabse(String)


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
donnect
ExamCounter
SchoolAdminSystem_Servlets
smbms
Web Pages
Source Packages
Other Sources
Dependencies
Java Dependencies
Project Files
TurkishAirlines
WSTester
Start Page
AddExamFrame.java
DatabaseOperations.java
BaseDao.java
Source
History
connection = DriverManager.getConnection(url, username, password);
} catch (Exception e) {
e.printStackTrace();
}
return connection;
}
public static ResultSet execute(Connection connection, PreparedStatement preparedStatement, ResultSet resultSet, String
preparedStatement = connection.prepareStatement(sql);
for (int i = 0; i < params.length; i++) {
preparedStatement.setObject(i+1, params[i]);
Inspector
Files Compiled with Errors
(45)
urce Packages (45)
com.jin.dao (10)
BaseDao.java (10)
47:The parameter preparedStatement to com.jin.dao.BaseDao.execute(Connection, PreparedStatement, ResultSet, String, Object[]) is dead upon entry but ove
47:com.jin.dao.BaseDao.execute(Connection, PreparedStatement, ResultSet, String, Object[]) may fail to clean up java.sql.Statement on checked exception
47:A prepared statement is generated from a nonconstant String in com.jin.dao.BaseDao.execute(Connection, PreparedStatement, ResultSet, String, Object[])
53:The parameter resultSet to com.jin.dao.BaseDao.execute(Connection, PreparedStatement, ResultSet, String, Object[]) is dead upon entry but overwritten
59:The parameter preparedStatement to com.jin.dao.BaseDao.execute(Connection, PreparedStatement, String, Object[]) is dead upon entry but overwritten
59:com.jin.dao.BaseDao.execute(Connection, PreparedStatement, String, Object[]) may fail to clean up java.sql.Statement
59:A prepared statement is generated from a nonconstant String in com.jin.dao.BaseDao.execute(Connection, PreparedStatement, String, Object[])
76:Dead store of null to resultSet in com.jin.dao.BaseDao.closeResource(Connection, PreparedStatement, ResultSet)
86:Dead store of null to preparedStatement in com.jin.dao.BaseDao.closeResource(Connection, PreparedStatement, ResultSet)
96:Dead store of null to connection in com.jin.dao.BaseDao.closeResource(Connection, PreparedStatement, ResultSet)
Notifications
Output
HTTP Server Monitor
Inspector
49:12
INS

A Prepared Statement is generated from a non-constant string:
The code creates an SQL prepared statement from a non-constant string. If unchecked, tainted data from a user is used in building this String, SQL injection could be used to make the prepared statement do something unexpected and undesirable.

Here is an example to avoid these type of vulnerabilities:



In some of the projects we did not find any vulnerabilities and in some of projects we didn't found any vulnerability related to SQL injection.

Conclusions

Testing of web applications plays an important role to analyze the web applications against the possible vulnerabilities and is very important to prevent the damages and exploitations after the development of the application. There are number of techniques for the testing of web applications have been proposed against the major vulnerabilities like SQL injection. In this work, we have tested 15 open source java-based projects with the help of several frameworks, tools, and plugins. In the results, we have found out several vulnerabilities related to SQL injection in some of the projects and by analyzing the vulnerabilities we have proposed the solutions in order to fix the issues which can cause the damages to the users of the web applications.

References

[1] W. G. Halfond, J. Viegas, and A. Orso, “A Classification of SQL-Injection Attacks and Countermeasures”, In Proc. of the Intern. Symposium on Secure Software Engineering (ISSSE 2006), March 2006.

[2] https://www.researchgate.net/publication/316805122_Research_on_the_technology_of_detecting_the_SQL_injection_attack_and_non-intrusive_prevention_in_WEB_system

[3] D. Evans and D. Larochelle, “Improving Security Using Extensible Lightweight Static Analysis”, IEEE Software, 19(1):42–51, 2002.

[4] https://github.com/seedstack/store-webapp-sample

[5] https://github.com/ScaleSec/vulnado

[6] https://github.com/salif/QA

[7] https://github.com/harismuneer/Flight-Booking-System-JavaServlets_App

[8] https://github.com/great-coder/Donnect

[9] https://github.com/emreziplar/ExamCounter

[10] https://github.com/JinhaoZhang0/SMBMS

[11] https://github.com/JulianRendon/SchoolAdminSystem_SpringMVCHibernate

[12] https://github.com/heshanera/HealthPlus

[13] https://github.com/mittrayash/Princeton-Hospital-Management-System

[14] https://github.com/shubhamgosain/Transport-Automation

[15] https://github.com/rajyash1904/ATM-Machine

[16] https://github.com/emreziplar/ExamCounter

[17] https://github.com/shouryaj98/Hotel-Management-Project-Java

[18] https://github.com/TECHNOCHAMANS/New-Smart-Bus-Ticket-Management-System