Static Analysis of Java Source Code

Laboratory for the class "Security Verification and Testing" (01TYASM/01TYAOV)
Politecnico di Torino – AY 2021/22
Prof. Riccardo Sisto

prepared by:
Riccardo Sisto (riccardo.sisto@polito.it)

v. 1.0 (01/12/2021)

Contents

Purpose of this laboratory

The purpose of this lab is to make experience with static source code analysis tools for the Java language. More specifically, the lab focuses on the SpotBugs tool, which is integrated in the Eclipse IDE as a plugin, extended with another plugin, called FindSecBugs, which is specific for finding security vulnerabilities. For the installation of the tools, please refer to the gettingStarted_v2.1.2.pdf guide. In the physical machines at Labinf and in the Labinf virtual machines the Spotbugs plugin is already installed (in the Eclipse copy named Eclipse 2020), but it may be necessary to configure it.

All the material necessary for this lab can be found in the course web pages on didattica.polito.it, Materiale Didattico 2021/22 section, 05Lab_StaticAnalysisJava folder, and the Dropbox folder.

Getting started with Spotbugs and FindSecBugs

First of all, let us configure SpotBugs to find only security-related vulnerabilities. In Eclipse, open the Preferences item from the Window menu, and select Java - SpotBugs. Here, configure the plugin by checking only the security box. Then, make sure FindSecBugs is correctly set in the plugins tab. Then, as a first test of the tool, import the eclipse project available in the lab material, in the examples folder. Once the project has been created, the maven utility inside Eclipse should download the project dependencies and compile the project automatically. When the project setup and compilation are finished, run SpotBugs on the project (right click on the project name and select SpotBugs - Find Bugs) and check that SpotBugs reports, as expected, some vulnerabilities. You can look at the details of the problems found (description of the problem, and possible ways to fix it) by opening the Spotbugs view or, in the Java view, by opening the Java files for which a number (indicating the number of problems found) is shown just after the filename.

1 Static Analysis of Java code with Spotbugs and FindSecBugs

1.1 Running SpotBugs on the OWASP Benchmark

Now, let us run the tool on the OWASP Benchmark. First of all, download the zip archive of the benchmark project and extract it to a folder which is under your home. The zip archive (Benchmark-master.zip) is available in the course Dropbox folder. Downloading it may take some time because the file is large. In Eclipse, select the 'Open projects from File System' item from the File menu and select the folder where the Benchmark is. This command will create a project named Benchmark-master. The project build process, managed by the maven inside Eclipse, should automatically start and download the necessary dependencies. When the process is finished, run SpotBugs on the project. The tool should take few minutes to complete the analysis. When the analysis is finished, open the SpotBugs perspective, which should display the results grouped by class. Analyze some examples of injection vulnerabilities to get acquainted with the information the tool reports about each warning.

Now, analyze the vulnerabilities reported by SpotBugs on the test cases 6, 93, 107, 108, and 148 (you can look for these test cases in the package org.owasp.benchmark.testcode under the source directory). Which ones of these test cases contains a real injection vulnerability? Which one des not?

Solution

Test cases with real injection vulnerabilities: 6 (command injection), 108 (sql injection) and 148 (XSS).
Test cases with false positives: 93 and 107.

1.2 Analysis and Fix of the JettyEmbedded project (https://github.com/oktadeveloper/okta-spring-boot-jetty-example)

Create an Eclipse project for the JettyEmbedded project (available for your convenience in folder Jettyembedded), by using the 'Open projects from File System' item from the File menu. The project contains a simple sample application based on Java servlets. Use SpotBugs to analyse the project code, to look for security vulnerabilities, and create a report with your findings. How many true positives (TP) and false positives (FP) did the tool report? What kind of XSS did you find?

Solution

In file HikesTodoServlet.java, the following vulnerabilities have been found:
line 35: variable hike, which comes from the servlet request, is written into the response, thus enabling an XSS attack (reflected XSS) if the response is incorporated into an HTML document. This vulnerability can be fixed by checking or sanitizing variable hike, when it is extracted from the request parameter.
lines 23, 39, 52: the servlet writes the contents of hikes into the response. These contents come from inputs from the user and they are not validated nor sanitized. For this reason, the servlet is vulnerable to (stored) XSS. This vulnerability can be fixed by checking or sanitizing variable hike, when it is extracted from the request parameter.

In summary, the tool reported 5 warnings. All of them are TP (2 of them refer to the same vulnerability). The vulnerabilities found are all XSS: one is a reflected XSS while the other ones are stored XSS.

Fix the vulnerabilities found in the project, with the aid of the suggestions given by SpotBugs, then run again the tool and observe how the results of the analysis changed.

1.3 Analysis and Fix of vulnerability CVE-2021-37573

CVE-2021-37573 is a vulnerability of the Tiny Java Web Server and Servlet Container (TJWS, http://tjws.sourceforge.net/). For your convenience, the CVE report is available in the lab material. After having had a look at it, your task is to find the vulnerability in the Java code of the application, with the help of SpotBugs, and then fix it. For your convenience, you can find a relevant portion of the code (taken from version 115, which is the latest one affected by the vulnerability) in the TJWS2 folder, with a pom.xml file that automates the download of its dependencies and its compilation. This code can be imported into Eclipse as a Maven Project.

Solution

The vulnerability (a reflected XSS) can be found in file Acme.Serve.FileServlet.java at line 183: the filename that is put into the body of the 404 response comes from the path in the request URL. The vulnerability can be fixed by sanitizing the filename or by checking it before putting it into the response.

1.4 (Optional) Analysis of a Java project of your choice

If you finished the previous jobs ahead of the end, you can choose an open-source Java project (e.g. from github) and try to analyse it with SpotBugs. A precondition is that the project can be compiled without errors.