Faces Input Validator Tester

IT Security / Advisor: Prof. Dr. Emmanuel Benoist
Expert: Dr. Igor Metz

FIVT is a code analysis plugin for the popular Java Integrated Development Environment (IDE) NetBeans. The FIVT plugin identifies input tags and their validators in Java Server Faces (JSF 2) applications and tests the validators for SQL injection and cross-site scripting (XSS) vulnerabilities. Through this analysis, web developers are able to pinpoint otherwise unknown weaknesses in their applications giving them the opportunity to provide more secure interfaces for users of their software.

Context

Despite being known problems since 1998 and since the birth of javascript, respectively, SQL injection and XSS are both still commonly overlooked vulnerabilities during the development of a web application. According to the Web Application Security Consortium's «Web Hacks Incident Database», 22.6 % of incidents reported are due to improper input handling, with 20.1 % of total incident attack methods being SQL injection related and 9.4 % being cross-site scripting related.

Thesis Objectives

The goal of this thesis was to produce a prototype of a plugin that can help JSF 2 web developers identify input validators within their applications that have SQL injection and XSS weaknesses.

Thesis Realization

As JSF 2 deals with xhtml elements rendered and used by client browsers as well as server side Java classes, the solution was divided into smaller modules performing specific tasks within the code analysis.

A parser module identifies individual JSF 2 inputs and their validators within the xhtml code of a project using predefined JSF 2 library definitions. Managed bean classes are also searched by the parser to map the variable types linked to inputs values.

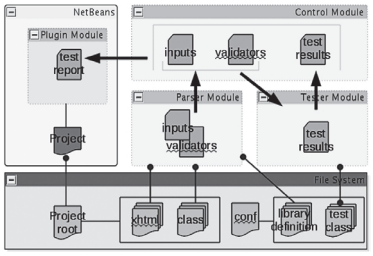
Parsed inputs and validators are then passed to a tester module which identifies unique validators, writing and compiling test classes on the fly. Each test run requires a mock JSF 2 context be created in order to execute the validation method of the validator class. Mock contexts are achieved through the MyFaces Test Framework from the Apache Foundation. Were it not possible to create the mock context a J2EE container server would have to be started in order to run the tests. In doing this the resource usage advantage of using static code analysis would have been lost. When the test cases are run they access lists containing up to date SQL injection and XSS input values recording how many of the values are rejected by each validator.

All of the information gathered by the parser and tester modules is organized by a control module and then passed on to the plugin module which handles the interaction with NetBeans and the user. Through the APIs provided by NetBeans the plugin module has access to project information such as the location of source files as well as other IDE tools. Through this the plugin module is able to provide direct linkage allowing a user to open a file containing a tested input, as well as add a FIVT SQL injection or XSS validator to an input.

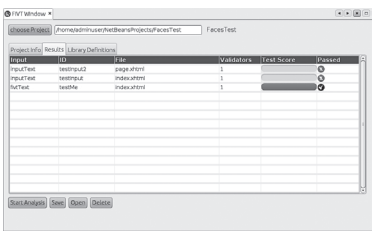
Conclusion

The realization of the thesis shows great promise for the use of static code analysis in identifying specific vulnerabilities within J2EE applications. The particular solution developed for JSF 2 provides a generalized API that could be adapted for other J2EE web frameworks. Beyond input validation testing, the thesis also uncovered the usefulness of a possible followup project involving tracking variable usage inside the Java server side code.

James Hulka



Information flow in FIVT



Analysis results list