Software security auditing using source code categorization and vulnerability ranking

Panagiotis Petsanas, George Stergiopoulous

(p3100152, geostergiop@aub.edu.gr)

Information Security & Critical Infrastructure Protection Laboratory
Dept. of Informatics, Athens University of Economics & Business (AUEB)

Software is considered vulnerable when unexpected input can cause it to behave in a way that it is not intended by its programmers. We develop a tool that assesses the danger level of source code points and creates code blocks. A code block contains information about the instructions and the actions that lead to an unwanted program behavior. Branches (mainly ifs) and suspicious instructions are flagged then fed to a Fuzzy Logic system that categorizes source code blocks and prints warning about possible vulnerabilities, much like a code auditor would audit code, based on his experience.

Tool Development

We categorize source code instructions and variables by creating code blocks for each pair of them and assessing the extracted information. Code classification according to position (Severity), type and usage (Vulnerability) helps rank all code blocks produced. The tool consists of 3 components: An AST analyzer, a bytecode analyzer and a fuzzy analyzer.

• **AST:** Static code analysis using a compiler and Abstract Syntax Trees. Provides source code information concerning every method invocation, branch statement and variable assignment found in the application under test.

• **Bytecode analysis:** A type of analysis performed on the compiled object file, which provides information about every variable usages, and analyzes points which handle the control flow of the application.

• **JAudit:** The main process of the tool. Combines all above information and creates a list of all the code blocks that are considered dangerous. It categorizes each one to one or more categories based on the type of the attacks that the software is vulnerable to. These findings are used to assess the Severity and Vulnerability levels of each code block and create warnings that are print to the user.

Source code classification

Fuzzy logic membership sets used to classify generated code blocks. The proposed fuzzy sets aim to automate reasoning based on the analysis findings, similarly to a code audit process:

(i) **Severity:** Values quantifying the impact of a logical error, with respect to how it affects the AUT’s execution flow and

(ii) **Vulnerability:** quantifying the likelihood of a logical error and how dangerous it is.

(iii) **Criticality:** The final, calculated risk value assigned for each variable. Combines Severity and Vulnerability ranks.

Criticality(x) = Severity(x) ∩ Vulnerability(x)

Aggregates two mem-ber-ship functions: ArB(x) = T (A(x), B(x))

Testing – The Juliet Test Suite

• **Juliet Test Suite:** A collection of 81,000 synthetic test cases provided by the National Institute of Standards and Technology (NIST).

• Useful for testing the effectiveness of static analyzers and other software assurance tools.

• **Six (6) categories** of instructions analyzed: (1) control flow locations, (2) input vector locations, (3) variable declarations, (4) variable assignments, (5) method declarations, and (6) method invocations.

References


