Chapter 5
A Case Study on Testing for Software Security: Static Code Analysis of a File Reader Program Developed in Java

Natarajan Meghanathan
Jackson State University, USA

Alexander Roy Geoghegan
L-3 Communications, USA

ABSTRACT

The high-level contribution of this book chapter is to illustrate how to conduct static code analysis of a software program and mitigate the vulnerabilities associated with the program. The automated tools used to test for software security are the Source Code Analyzer and Audit Workbench, developed by Fortify, Inc. The first two sections of the chapter are comprised of (i) An introduction to Static Code Analysis and its usefulness in testing for Software Security and (ii) An introduction to the Source Code Analyzer and the Audit Workbench tools and how to use them to conduct static code analysis. The authors then present a detailed case study of static code analysis conducted on a File Reader program (developed in Java) using these automated tools. The specific software vulnerabilities that are discovered, analyzed, and mitigated include: (i) Denial of Service, (ii) System Information Leak, (iii) Unreleased Resource (in the context of Streams), and (iv) Path Manipulation. The authors discuss the potential risk in having each of these vulnerabilities in a software program and provide the solutions (and the Java code) to mitigate these vulnerabilities. The proposed solutions for each of these four vulnerabilities are more generic and could be used to correct such vulnerabilities in software developed in any other programming language.

DOI: 10.4018/978-1-4666-0089-8.ch005
INTRODUCTION

Static Code Analysis is the process of examining a piece of code without actually executing it (McGraw, 2006). This allows the analyst to see everything that the code does and to consider the program as a whole, rather than just as a sequence of individual lines. Static Code Analysis (also invariably referred to as ‘Source Code Analysis’) is important from a software security standpoint. There are a number of issues which must be evaluated when performing a security analysis on a piece of code. Answering these questions (a sample is given below) can prove to be time-consuming for an analyst. Providing the right answers to these questions also requires a comprehensive knowledge of possible exploits and their solutions.

- What is the basic design of this application?
- What are the different technologies involved in this application?
- Who would want to attack this application?
- What would any attacker hope to gain by attacking the application?
- What is the risk associated with a successful attack?
- (and perhaps most importantly) How are the developers protecting this application?

In addition to an analysis of these security issues, a properly-performed static code analysis must also consist of (Graff & Van Wyk, 2003):

- **Type checking**: this is usually done by the compiler and refers to the process of examining a program’s variables and parameters to ensure that a variable is not assigned to an incompatible type (Wysopal et al, 2006);

- **Style checking**: this consists of examining the format of the code, such as indentation and the use of comments, to ensure readability;

- **Program verification**: this step consists of comparing a block of code with its task specification to ensure that the code is correct and generates the correct results (Howard & Lipner, 2006);

- **Property checking**: this step seeks to ensure that particular sequences of instructions do not occur, such as accessing a memory location after it has been released or prematurely releasing all of a program’s allocated memory (Graff & Van Wyk, 2003);

- **Bug finding**: this step involves scanning the code for patterns of commands to identify where a programmer may have inadvertently left out some intended code (Graff & Van Wyk, 2003);

- and of course, the **security review**
Related Content

Requirements
(2017). Model-Based Design for Effective Control System Development (pp. 32-74).
www.igi-global.com/chapter/requirements/179498?camid=4v1a

Clustering Based on Two Layers for Abnormal Event Detection in Video Surveillance
www.igi-global.com/article/clustering-based-on-two-layers-for-abnormal-event-detection-in-video-surveillance/187168?camid=4v1a

Static Type Systems: From Specification to Implementation
www.igi-global.com/chapter/static-type-systems/30755?camid=4v1a