Tool-Assisted Code Review of a Web Application

Feb 11, 2009

1 Overview

In this exercise, students will analyze a web-based content management system to identify security risks and vulnerabilities, then fix the identified vulnerabilities where possible. Students will work in teams to complete the analysis and source code modifications. The Fortify Source Code Analyzer (SCA) static analysis tool will be used to assist students in detect security vulnerabilities. Note that the tool will produce both false positive and false negative results, so the output of the tool cannot be accepted without further analysis.

2 Resources Required

To complete this assignment, students will need a copy of the Fortify Source Code Analyzer (SCA) and Audit Workbench tools. They will also need a copy of the NKUCMS web application that we are auditing for this assignment.

3 High Level View

Study the application as a whole by reading the code and using Audit Workbench, in order to answer the following questions.

1. How large is the application (lines of code, number of files)?

2. What entry points (write as relative URLs) could someone use to enter the application? Which URL is the expected entry point?

3. What assets does this application have? What are the threats to those assets?

4. What security controls does the application use? Describe each control, noting the file and function where it is located.

5. Which inputs can be controlled by an attacker? What internal data structures store those inputs? Supply filenames in addition to line numbers and variable names to help locate these data structures.

4 Auditing Vulnerabilities

For each vulnerability that SCA detects, write a description including:

1. The type of vulnerability discovered (SQL injection, XSS, etc.)
2. The filename and line number where the vulnerability was detected.
3. A one paragraph analysis explaining whether the vulnerability is a false positive or not. These descriptions may be identical for similar vulnerabilities, but note that differences in control or data flow can change where a line of code is vulnerable or not.

5 Fixing Vulnerabilities

For each of the vulnerabilities discovered above that are not false positives, modify the source code to remove the vulnerability. Some fixes will be simple single line changes, such as replacing a dangerous function with a safe one, but other mitigations will require input validation or other additions to the program. It may be best to modify the design of the application to improve its security architecture than to attempt to fix each vulnerability with the smallest code changes possible. Comment all source code modifications with your names and a brief description of the change. Turn in the fixed source code, along with a one paragraph description of each vulnerability and how the code modifications mitigate it.

6 Verifying the Fixes

After the vulnerabilities have been fixed, analyze the fixed version of each program with SCA. The updated reports should only contain false positive results. In addition to the false positives carried over from the original source code, there may be new false positives arising from the code written to fix the vulnerabilities. Write a paragraph analysis of each new false positive, explaining why a vulnerability is detected despite the changes they made to the code.

7 Submission

The final assignment submission will include the following documents:

1. High-level security analysis, answering the questions in section 3.
2. FPR file from the analysis of the original source code with Source Code Analyzer.
3. Vulnerability reports, as described in section 4.
4. Fixed source code, with comments documenting each fix.
5. FPR file from the analysis of the fixed source code with Source Code Analyzer.
6. Vulnerability reports for the fixed version, if any, as described in section 6.

Encapsulate all of the documents in a single ZIP file for submission.