Design Review of a Web Application

April 15, 2009

1 Overview

In this exercise, students will analyze a web-based scheduling system identify security risks and vulnerabilities, with a focus on design level issues. The customer's main concern is the security of personally identifiable information. Students will work in teams to complete the analysis. The Fortify Source Code Analyzer (SCA) static analysis tool will be used to assist students in detecting 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 RMH Homebase web application that we are auditing for this assignment. The application can be found from `http://hfoss.bowdoin.edu/`. The test site from that page can be used to explore the application's functionality, but it **cannot** be used for security testing.

3 Design 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 security controls does the application use? Describe each control, noting the file and function where it is located.
4. What are the classes of users supported by the application? Which application functions are each class of users permitted to access?

5. What is the design of the application? Document the design with data flow diagrams, including at least two levels of diagrams, starting with a level 0 diagram showing the entire system and its interactions with external entities, such as user types and databases, and drilling down to show interactions between system components (modules, files, functions.)

6. What are the security strengths in the application's design? Compare the application's security design strengths with that of other applications studied during the class. What security design patterns are applied in the application?

7. What security weaknesses exist in the application's design? Compare the application's security design weaknesses with that of other applications studied during the class. What security design patterns could be applied in the application to improve its security?

4 Auditing Vulnerabilities

For each type of vulnerability that SCA detects, write a description in the format of the following example:

1. The type of vulnerability discovered (SQL injection, XSS, etc.)

2. Filenames and line numbers where the vulnerability was detected.

3. A design pattern that would mitigate the vulnerability type and an explanation of how the mitigation works and how much it would reduce the risk. Some patterns may completely remove the risk; others will reduce the risk in various ways.

4. A list of files and functions that would have to be removed, added, or modified by the introduction of the design pattern, along with a high level description of the changes that would need to be made to apply the pattern.

Use the following format for the vulnerability descriptions:

```
Type: SQL Injection
Occurrences: dbConnector.php, lines: 36, 48, 122
             dbShifts.php, lines: 17, 182
Mitigations: X design pattern from source Y.  It will reduce the risk
of SQL injection by Z.
Affected Files: The mitigation will require that the following files
be modified: dbShifts.php, dbConnect.php.  The following
functions/classes will need to be removed: a, b, c.  Additional
functions will need to be added: d, e.  The function d will perform
the following tasks: f, g.  The following functions will need to be
modified as follows: h, i, j.
```

5 Implementation

Implement one of your design patterns in the application, applying the changes described above. Document all of the modifications made by applying your design pattern, and comment all source code modifications with your names and a brief description of the change. Turn in a copy of the application's source code with your changes.

6 Verifying the Fixes

After the pattern has been applied, analyze the fixed version of the program with SCA. The updated reports should show the effect of the improved security design. If the number of vulnerabilities of the type mitigated by the design pattern you implemented have not been reduced, document why this is the case.

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.

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