

A Code Inspection Process for Security Reviews

Date: Mar 6, 2009
Author: Gabriele Garzoglio
Document version: v1.0

1 Abstract

This document presents a process to conduct a code inspection of a software artifact. The inspection focuses on reviewing the security properties of this artifact.

2 Scope and Goals

This document defines a process to assess security issues of a software artifact. Goal of the process is to identify technical risks associated with the application and the impact of these technical risks. Examples of technical risks are vulnerabilities, flaws in application code and architecture, and other technical problems or complications. Examples of the impacts of these risks are unexpected system crashes and avoidance of security control, including unauthorized data modification or disclosure. Optionally, this process can generate application quality metrics, such as the number or “density” of defects in the code, number of critical risks, etc. In addition, a side-goal for the process is improving the process itself, by reevaluating it with each inspection.

The process focuses on studying security issues with the code itself, rather than with the operations of the software. For example, the process would focus on uncovering avoidance of security controls, rather than on how system passwords are protected and distributed or on how patches are maintained. In any case, since the line between software and operational issues is often blurred, in the process, reviewers are asked to familiarize themselves with high-level operational practices. It should be noted that this process does not discuss how to select the software artifact that should be reviewed. The process; however, allows for a quick rejection of an artifact, in case its security sensitivity is evaluated to be too low-risk to justify the cost of a full review.

To achieve the goals of the process, the reviewers should study the software artifact with the following in mind:

- What are the business context and risk? What does the software do and protect?
- What does an exploiter gain? What is the threat and exploit community?
- What defects can be exploited? What are the potential vulnerabilities?

- Loosely defining risks as vulnerabilities × threats, what are the risks?

This inspection process is loosely inspired by the Fagan inspection [1]. This defines certain personnel roles and a workflow of activities, including meetings, and is discussed in sec. 3. The core of the inspection is the security review itself. The security review process consists of six steps and is discussed in sec. 4.

3 The Inspection Process

This inspection process is inspired by the Fagan inspection and uses a similar workflow and organization of personnel roles. The following sections describe the roles, deliverables, and workflow of the process.

3.1 Personnel Roles

This process defines the following roles for the personnel participating in the inspection

- **Management:** the line and project management of the personnel participating in the inspection. Management must support the inspection review activity in order for it to be successful, allowing for the appropriate time to be spent on the review and on addressing the issues listed in the Inspection Report. The inspection is typically initiated by a manager with a stake in a software artifact: the selection process for the artifact is outside of the scope of this document.
- **Moderator:** the person responsible for the inspection. The Moderator collaborates with Management to appoint the personnel participating in the review. He or she organizes the inspection according to the inspection workflow (sec. 3.3). The Moderator is also responsible to “moderate” the interactions between members of the inspection with different roles (e.g. Reviewers and Authors).
- **Author:** the main developer(s) of the software artifact. Author(s) are responsible for working with the Reviewers, providing material and expertise on the artifact. Author(s) are also responsible for following up with the recommendations for improving the artifact, as documented in the Inspection Report.
- **Reviewer:** a person reviewing the software artifact. Reviewers typically have particular expertise or interest in the domain of the software artifact. Reviewers are highly encouraged to follow the Security Review Process described in sec. 4 and provide feedback on it. Reviewers are responsible for documenting the outcomes of their review in a Security Review Report.
- **Scribe:** a person participating in the meetings and taking notes on behalf of the group. The scribe should always feel free to interrupt the conversation to make sure that the record is accurate.
- **Security Expert:** this is typically a representative of the Security Team at Fermilab. His/her input is useful to weight the sensitivity of the issues exposed by the review.

3.2 Deliverables of the Inspection

The deliverables of the process is an Inspection Report. This report consists of the Security Review Report from the Reviewers (sec. 4) and documentation of relevant discussions between Authors and Reviewers, as compiled by the Moderator using the notes from the Scribe.

The outcome of the process should give Authors enough information to improve the quality of the software artifact. It is the responsibility of the Authors, possibly in consultation with their Management, to decide what recommendations of the Inspection Report to implement.

3.3 Inspection Workflow

The following is a description of the inspection workflow. Fig. 1 shows a diagram of the workflow.

1. **Moderator's appointment:** Management appoints a Moderator to coordinate the inspection process for a software artifact. The process for management to select what software artifacts should be reviewed is outside of the scope of this document.
2. **Author's appointment:** The Moderator contacts the leadership of the project responsible for the development of the software artifact. The Moderator and the project manager appoint one or more Authors to be the point of contact for the inspection. The Authors' line management must support the activity, allowing that the Authors spend the appropriate amount of time to work with the reviewers and to follow up with the Reviewers' recommendations. At this point, Authors are asked to gather the documentation for the Reviewers. Such documentation should give information on the issues listed in sec. 4.1.
3. **Weighting the sensitivity of the review:** Moderator, Authors, and Management agree on what components of the software artifact should be reviewed. At this point, it is assessed whether the artifact presents potential security risks sensitive enough to be worth the cost of a full review or not. If not, the moderator should write the Inspection Report, detailing the reasons for the decision.
4. **Reviewer's appointment:** The Moderator appoints two or three Reviewers. It is advantageous to include at least one reviewer from outside of the software project group. Reviewers and their line management should understand that working on a review is a commitment of several hours of effort (4-16 hours), distributed throughout several calendar days (typically, 2 weeks). One Reviewer should be appointed as Editor Reviewer. Her responsibility consists in gathering all input from the Reviewers and in writing the Security Review Report.
5. **Scriber's appointment:** The Moderator appoints a Scribe. The Scribe should have some general understanding of the software domain, even if detailed knowledge should not be required.
6. **Hold Initial Briefing:** This meeting has three purposes: (1) the Moderator gives an overview of the inspection and security review process; (2) Authors give Reviewers the documentation on the issues listed in sec. 4.1; (3) Authors and Reviewers agree on their future interactions to complete the security review process. The Moderator can appoint the Editor Reviewer at this meeting. The meeting is attended by the Moderator, Authors, Reviewers, and the Scribe. At

this time, the author can optionally discuss the software artifact, if Reviewers are not familiar with it.

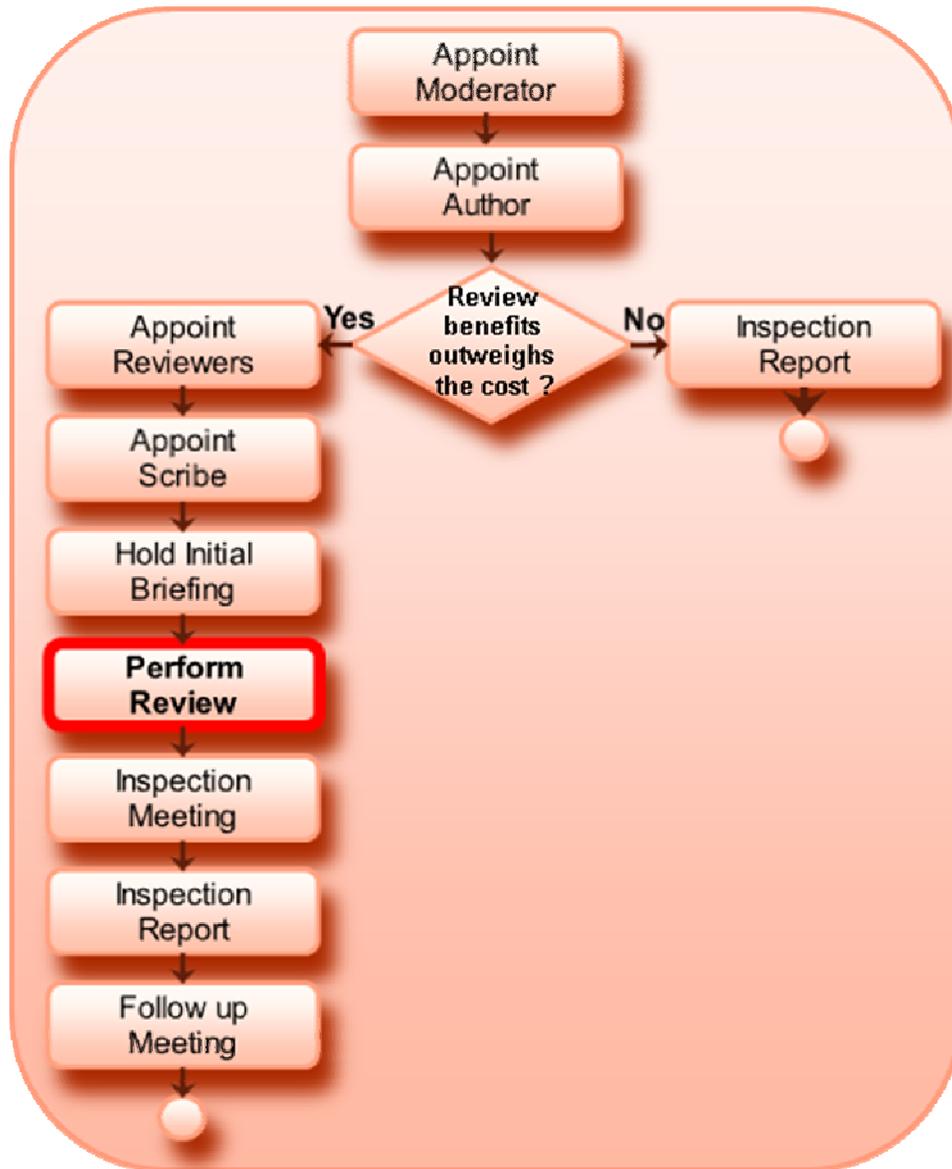


Fig. 1: A diagram of the inspection workflow.

7. **Perform the Security Review:** Reviewers perform the Security Review, following the process in sec. 4. The review should take about 2 calendar weeks (of non-continuous effort).
8. **Hold the inspection meeting:** Reviewers discuss their report with the Authors. The meeting is attended by the Moderator, Authors, Reviewers, the Scribe, and possibly a representative of the Security Team. The Moderator should keep the meeting running smoothly and on track; it is useful to remind Reviewers that they inspect a software artifact, not its Authors. It is important to keep this meeting to about two hours in length. Always bring donuts to put the attendees in the proper mood. The meeting should consist of the following 2 phases:

- 8.1. Reviewers present their report to the Authors. Authors and Reviewers discuss the sensitivity of each finding. The Scribe takes notes on the exchanges.
- 8.2. At the end of the inspection meeting, the Moderator solicits comments from those present on improvements to the inspection process itself.
9. **Generate Inspection Report:** The Moderator writes the final report. This consists of the Security Review from the Reviewers and relevant notes from the meetings (sec. 3.2). The Inspection Report should include a list of issues for the Authors. In all cases, it is up to the Authors to decide whether an issue that has been raised will require some action. If the Authors decide that a particular issue can be ignored, a very good reason must be stated as to why this is the best course of action. This reason must be documented in the Report. The Security Team at Fermilab can be involved to provide input on the relevance of each issue.
10. **Follow up meeting:** This meeting is optional. Its goal is to follow up with the Authors on the issues listed in the Inspection Report. It should happen several weeks after the Inspection Report has been finalized.

4 The Security Review

This security review process has the goal of identifying technical risks and their impact. Scope and goals of the review are discussed in sec. 2. The Security Review is the core part of the Inspection process (point 7 of sec. 3.3). The deliverable of this review is a Security Review Report.

The Security Review process consists of 6 steps, each described in a separate subsection below.

4.1 Application Review

This step is useful to familiarize the Reviewers with the software artifact. The principal investigation mechanism consists in reviewing documentation and interviewing the Authors. The documentation can be gathered during the initial briefing (point 6 sec. 3.3). Some issues of interest for the review are listed hereby:

1. General Functionalities
2. Environment (Users, Security Policies, etc.)
3. Use Cases
4. Specific Features
5. Architecture
6. Relevant code portions
7. Project management practices
8. Operational practices
9. Risk Analysis / Security Requirements / Security Operations (if available)

4.2 Abuse Cases Analysis

Abuse case analysis is discussed in [2] and [3]. “Abuse cases” are malicious ways of misusing of the software. Studying abuse cases helps prepare for abnormal or exceptional application behavior. Abuse case analysis can help document how the software reacts in the face of illegitimate usage.

To determine and study abuse cases, a Reviewer can loosely follow the following process:

1. Identify the most likely applicable attack patterns, by studying use cases and requirements. A list of attack patterns is available in the appendix (sec. 6).
2. Using the applicable attack patterns, build an attack model.
3. Determine misuses of the software and abuse cases

Remember to talk to the Authors, as they might be already aware of potential abuses of the software.

4.3 Architectural Risk Analysis

Architectural risk analysis is discussed in [3] and [4]. Reviewers can loosely follow the following process.

1. Build a one page architectural overview of the software system, if not available. This can be built with the help of the Authors.
2. Analyze the architecture, focusing on the following properties:
 - a. Attack resistance: how resilient is the application in the face of an attack? What attack patterns (sec. 6) do not apply to the software system because of its architectural properties?
 - b. Presence of ambiguity: are there ambiguities in the architecture, in terms of functionalities, responsibilities, etc.? Modules that are involved in any architectural ambiguity are more likely to present security issues.
 - c. Presence of weaknesses: are there weaknesses built into the architecture, such as single points of failures, communication throttles, etc.? Modules that are involved in architectural weaknesses are more likely to present security issues.
3. Identify and rank architectural risks. Focus on what the application protects. Understand what would an exploiter gain with a successful attack (threat) and what defects can be exploited (vulnerability). Risks are often loosely defined as the product of threats \times vulnerabilities.
4. Help the Authors by defining possible mitigation strategies for each risk.

4.4 Code Review

Various recommendations on how to do a Code review are discussed in [3] and [4].

In general, it is best using automated tools to guide the reviewers toward portions of the code more likely to have problems. Some of these tools, such as Fortify, have advanced analysis features but are expensive to use. There are various resources on the web that discuss free tools [5]. These include ITS4 [6] and Flawfinder [7], which are static analyzers for C / C++.

If the code review is done by reading the code, the Reviewer should especially look for the following properties:

1. Input validation and representation
2. API abuse
3. Security-related features
4. Time and state
5. Error handling
6. Code quality
7. Encapsulation
8. Environment

Potential concerns are often one of the following:

1. An area of the artifact that the Author is uncertain about
2. An area that in the past has shown to be problematic
3. A special quality that the deliverable must contain (i.e. be memory efficient, incorporate all previous requirements etc.)

Reviewers can learn about these issues by interviewing the Authors.

4.5 Application tests

The following is detailed in [3].

There are various types of security testing techniques. For web-based application, OWASP [8] offers step by step guides to uncover security vulnerabilities, using similar attack patterns as in sec. 6. Tools for testing the response of software to a large set of input parameters are called “fuzzing” tools [9]. These tools include Peach [10] and OWASP WSFuzzer.

Tests for the security review should be selected according to the outcomes of previous analyses.

We expect that this section of the document will be enhanced with recommendations for effective tests as we build experience using this process.

4.6 Write Security Review Report

The Security Review report is the central piece of the Inspection Report. The report should include

1. The findings of the review for each of the steps detailed above. It should be clear from the review what issues need to be discussed with and potentially fixed by Authors.
2. A discussion on what the application does and protects, what the threats and exploit community are, what potential vulnerabilities have been uncovered, and what the risks are.
3. The impact of each risk, by linking risks with the business needs of the application (Availability, confidentiality, integrity, authenticity/non-repudiation, etc.)
4. An analysis of what mitigations can be adopted for the highest impact risks.

The Moderator appoints an Editor Reviewer at the initial Inspection Briefing (point 6 sec 3.3). The Editor Reviewer is responsible for the delivery of the report. Reviewers discuss the report at the Inspection Meeting (point 8 sec. 3.3) and hand over the report to the Moderator.

5 Conclusions

This document describes an inspection process for security reviews. It defines personnel roles, recommended meetings, and six steps to review the security properties of the investigated software.

This security review process was extended from software inspection guidelines [1] that have been successfully used for years at Fermilab. The process has been successfully used to review the Site AuthoriZation service (SAZ) software at Fermilab and is meant to be continuously refined.

6 Appendix A: 48 Attack Patterns

The following attack patterns are described in detail in [2].

1. Make the Client invisible
2. Target Programs That Write to Privileged OS Resources
3. Use a User-Supplied Configuration File to Run Commands That Elevate Privilege
4. Make Use of Configuration File Search Paths
5. Direct Access to Executable Files
6. Embedding Scripts within Scripts
7. Leverage Executable Code in Non-executable Files
8. Argument Injection
9. Command Delimiters
10. Multiple Parsers and Double Escapes
11. User-Supplied Variable Passed to File System Calls
12. Postfix NULL Terminator and Backslash
13. Relative Path Traversal

14. Client-Controlled Environment Variables
15. User-Supplied Global Variables (DEBUG=1, PHP Globals, etc.)
16. Session ID, Resource 10, and Blind Trust
17. Analog In-Band Switching Signals (aka "Blue Boxing")
18. Attack Pattern Fragment: Manipulating Terminal Devices
19. Simple Script Injection
20. Embedding Script in Nonscript Elements
21. XSS in HTTP Headers
22. HTTP Query Strings
23. User-Controlled Filename
24. Passing Local Filenames to Functions That Expect a URL
25. Meta-characters in E-mail Header
26. File System Function Injection, Content Based
27. Client-side Injection, Buffer Overflow
28. Cause Web Server Misclassification
29. Alternate Encoding the Leading Ghost Characters
30. Using Slashes in Alternate Encoding
31. Using Escaped Slashes in Alternate Encoding
32. Unicode Encoding
33. UTF-8 Encoding
34. URL Encoding
35. Alternative IP Addresses
36. Slashes and URL.Encoding Combined
37. Web Logs
38. Overflow Binary Resource File
39. Overflow Variables and Tags
40. Overflow Symbolic Links
41. MIME Conversion
42. HTTP Cookies
43. Filter Failure through Buffer Overflow
44. Buffer Overflow with Environment Variables
45. Buffer Overflow in API Calls
46. Buffer Overflow in Local Command-Line Utilities
47. Parameter Expansion
48. String Format Overflow in syslog()

7 References

- [1] The Fagan inspection process at Fermilab:
<http://ods.fnal.gov/ods/www/process/faganInsp.html>
- [2] G. Hoglund and G. McGraw, “Exploiting Software: How to break the code”, Ed: Addison-Wesley
- [3] G. McGraw, “Software Security: Building Security in”, Ed: Addison-Wesley
- [4] J. Viega & G. McGraw, “Building Secure Software”, Ed: Addison-Wesley
- [5] Code review tools: http://www.softpanorama.org/SE/code_reviews_and_inspections.shtml
- [6] ITS4 code scanning tool: <http://www.citigal.com/its4/>
- [7] Flawfinder code scanning tool: <http://www.dwheeler.com/flawfinder/>
- [8] OWASP testing portal:
http://www.owasp.org/index.php/OWASP_Testing_Guide_v2_Table_of_Contents
- [9] Resources on Fuzzing tools:
<http://www.dragoslungu.com/2007/05/12/my-favorite-10-web-application-security-fuzzing-tools/>
- [10] Peach Fuzzing tool: <http://peachfuzzer.com/>