A Secure Code Review Retrospective

Presenter: Rushi Purohit
Authors: Andrew Buttner, Richard Piazza, Rushi Purohit, and Alec Summers
A Secure Code Review is a specialized task involving a combination of analysis methods intended to reduce the likelihood of security-related flaws in code

- An important part of the Software Development Lifecycle
- Usually performed as part of testing phase
- Does not replace typical peer reviews
- Not a silver bullet … rather, it is a tool in the toolbox
Use Cases

- MITRE-developed code for internal use
- MITRE-developed code for external use
- MITRE-developed code for sponsors
- MITRE-developed code for open source
- Sponsor and/or contractor-developed code for an independent review
Code Review Guidelines

- **Don’t create a battleground**
  The goal is better software, not who’s right

- **Layout the ground rules**
  Establish clear expectations about how the review will be performed, including how long it will take, how much it will cost, and what role everyone is playing

- **Maintain professionalism**
  Don’t take the criticism personally and offer only technical advice that will improve the code. Respect others’ opinions, comments, and suggestions

- **Scope the review properly**
  Determine the size and scope of the code being reviewed. Don’t bite off more than can be chewed

- **Document what happens**
  Write everything down, especially decisions and action items

## MITRE Secure Code Review Types

<table>
<thead>
<tr>
<th>Developer Interview</th>
<th>Verbal Review</th>
<th>Automated Review</th>
<th>Manual Review (&lt;100K LOC)</th>
<th>Verify Mitigations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level of Effort:</strong> 4 hours/2-staff</td>
<td><strong>Level of Effort:</strong> 2 hours/2-staff</td>
<td><strong>Level of Effort:</strong> 8 hours/tool/staff (add staff per tool)</td>
<td><strong>Level of Effort:</strong> 48 hours/2-staff (plus 2 hours for management)</td>
<td><strong>Level of Effort:</strong> Min. 2 hours – Max. 8 hours/staff</td>
</tr>
<tr>
<td><strong>Scope:</strong> Meeting with the development team pre-development</td>
<td><strong>Scope:</strong> Meeting with the development team post-development</td>
<td><strong>Scope:</strong> Verbal Review + an automated static analysis scan(s) with the appropriate licensed tool(s)</td>
<td><strong>Scope:</strong> Verbal Review + Automated Review + Manual inspection of the code</td>
<td><strong>Scope:</strong> Assess diffs along with justifications of the fixes as well as validate the Plan of Actions &amp; Milestones (POA&amp;M).</td>
</tr>
<tr>
<td><strong>Outcome:</strong> High-level identification of different attack-vectors and ways to guard against them</td>
<td><strong>Outcome:</strong> Learn about the application and if/how the coding and certain mitigations were implemented</td>
<td><strong>Outcome:</strong> Raw tool(s) findings to be shared with the development team with a high-level analysis of those findings</td>
<td><strong>Outcome:</strong> A tailored findings report written by review team consisting of confirmed issues, their relevant context, and considerations for remediation</td>
<td><strong>Outcome:</strong> Confirmation on which findings have been fixed and how as well as which findings are still outstanding or incorrectly fixed, and what should be done to fix them.</td>
</tr>
</tbody>
</table>
Findings Report

• Overview of secure code review process
• Documentation of high-level findings consumable by project and technical leadership

Detailed findings report:

• File name & line number
• Rating and their definitions
• Finding description & potential remediation strategy
• Common Weakness Enumeration (CWE)
Lessons Learned [1 & 2]

Tailor reviews to needs of the team
Offer different types of support, as previously highlighted

Understand the developers’ approach
Work with the development team to learn about the application, and how security features have been implemented
Lessons Learned [3 & 4]

Use multiple tools

Tools have different strengths and weaknesses, so having multiple tools provides for more comprehensive review

Focus on the big picture

Avoid reading every single line of code, rather understand what the whole code is doing and use security mechanisms to focus on what code is relevant
Lessons Learned [5 & 6]

Check for small finite set of CWEs

Having a set number of weaknesses to focus on provides for better quality. Prioritize based on the languages & technologies used, context of the application, etc.

Stick to the intent of the review

Use static analysis and avoid executing the code. Pen-testing and other dynamic analysis activities are important and should be performed by others.
Lessons Learned [7 & 8]

Do not assess level of risk

A secure code review shouldn’t make judgements about the associated risk. Provide ratings so that the development team can prioritize mitigations

Follow up on review points

Be available to answer/clarify development teams’ questions and comments. Assist in mitigation strategies. Verify mitigations and POA&M
Thank You

Rushi Purohit
rpurohit@mitre.org