ENPM 809W
Introduction to Secure Software Engineering

Gananand Kini

Lecture 11

Secrets of a Secure Code Review

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

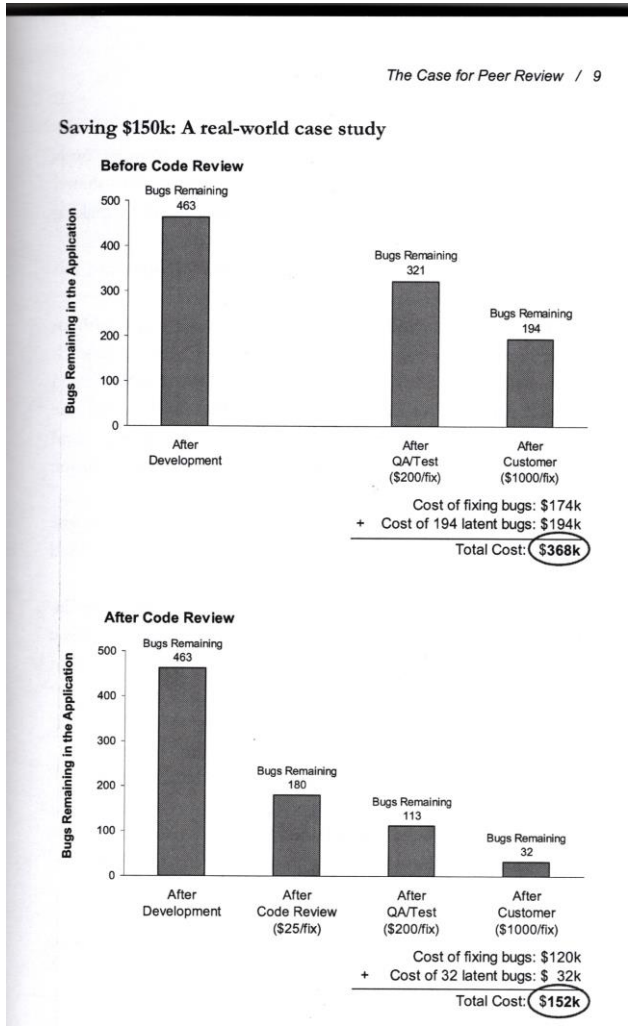
Outline

- What is Code Review?
- Why have a Secure Code Review?

What is a Code Review?

The case for code reviews in general

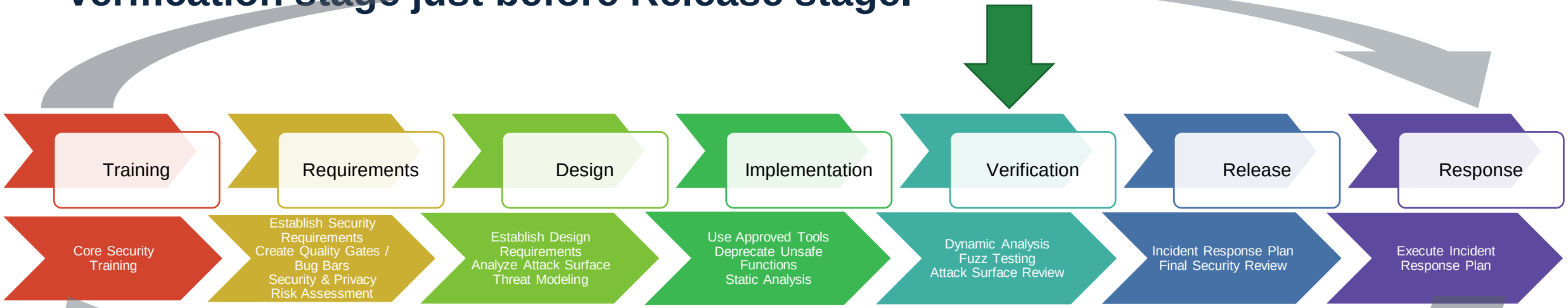
- **Reduces bugs/defects.**
- **Saves money.**
- **Saves time (less iterations of the development cycle).**
- **Increases trust in product.**



Sources:
1. Jason A. Cohen. Best Kept Secrets of Peer Code Review: Modern Approach. Practical Advice. 2006. pp9-129.

Code Review (and Secure Code Review)

- **Code Review: The practice of deliberately reviewing code to find bugs (related to design, functionality, software requirements etc.)**
- **Secure Code Review: Similar practice to find security weaknesses and security related bugs!**
- **Typically done as part of the Secure Development Lifecycle during the Verification stage just before Release stage.**

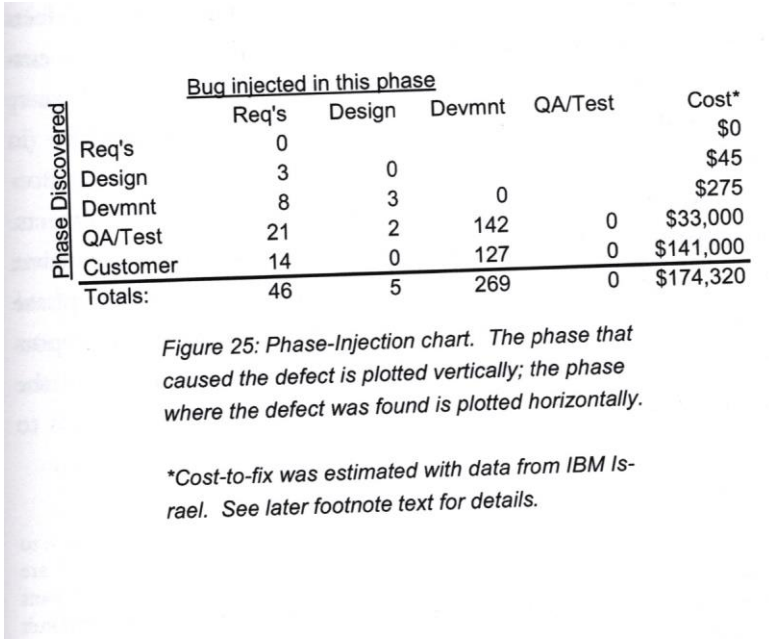


Sources:

1. https://en.wikipedia.org/wiki/Npm_(software)#Notable_breakages

Phases Bugs injected versus Bugs detected

- **Typically, you don’t want bugs to be found when the product reaches the customer.**
- **That is when it becomes the most expensive to fix.**
- **Usually lose trust in the software and the brand.**
- **Want to detect issues early!**
- **Secure Code Reviews must be done before the Release Phase.**



Phase Discovered	Bug injected in this phase: Req's	Design	Devmnt	QA/Test	Cost*
Req's	0				$0
Design	3	0			$45
Devmnt	8	3	0		$275
QA/Test	21	2	142	0	$33,000
Customer	14	0	127	0	$141,000
Totals:	46	5	269	0	$174,320

Figure 25: Phase-Injection chart. The phase that caused the defect is plotted vertically; the phase where the defect was found is plotted horizontally.

**Cost-to-fix was estimated with data from IBM Israel. See later footnote text for details.*

Sources:
1. Jason A. Cohen. Best Kept Secrets of Peer Code Review: Modern Approach. Practical Advice. 2006. pp9-129.

Why do a secure code review?

- **Just like normal code reviews for bugs, will save time and resources spent during the Secure Software Development Lifecycle.**
- **However, hard to measure and predict return on investment in dollar amounts.**
- **Gives a big return-on-investment for spending a few hours in terms of:**
  - Reduction of attack surface, and stepping stones for allowing an attacker to break the software system.
  - Helping all interested parties (users, developers, managers etc.) understand and manage expectations of security in the software system.
  - Helps increase trust in the software system and the organization.

Phases of a Peer Code Review

- **Planning**
  - Plan to hold a meeting between participants of the code review.
  - Keep source code files ready.
- **Inspection**
  - Use the time to find defects or bugs
  - Comment and discuss findings
- **Rework**
  - Developers can rework the code and attempt to fix the defects/bugs found
  - Reiterate with Inspection and Rework phases.
- **Complete**
  - Commit their changes to source control repository.

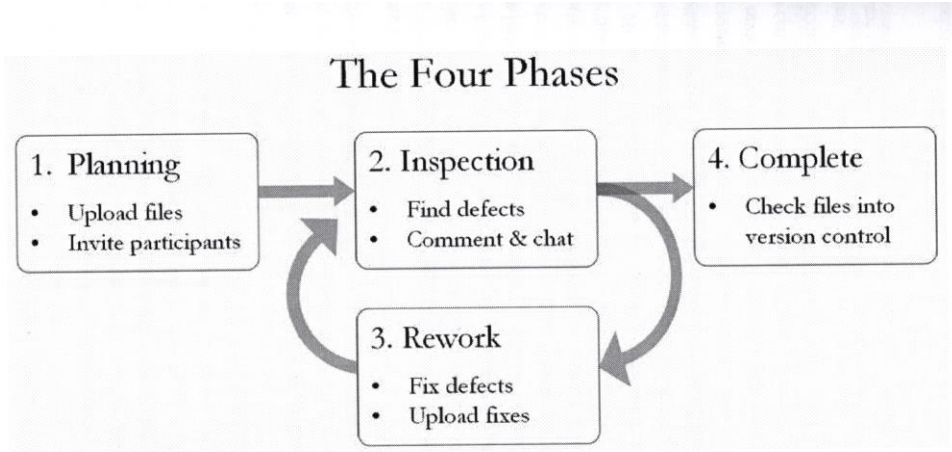


Figure 34: Lifecycle of a review.

Sources:
1. Jason A. Cohen. Best Kept Secrets of Peer Code Review: Modern Approach. Practical Advice. 2006. pp9-129.

Secure Code Review

- **Similar to Peer Code Review, however, good to have it performed by a third party (outside the developer group), preferably by a group with a security focus or another developer group.**
- **This allows for a non-vested third-party to determine the security flaws and bugs.**
- **Goal is to keep it:**
  - Judgement free – Don't want to point out developers that committed the bugs or security violations.
  - Control free – Don't want to control and force developers to implement fixes, nor force them to implement fixes in a certain way. Treat developers as the subject matter experts and equal partners in getting the security issues resolved!
  - Positive – Help developers implement secure software systems and help users of the software system to be satisfied with its usability and security!
- **If these goals are not observed, it can be catastrophic to developer morale and counter-productive to the development and security of the software system!**

Secure Code Review Phases

- **Planning**
  - Collect requests for a secure code review.
  - Plan a meeting with the developer team and the code reviewers.
- **Introductory Meeting**
  - Set expectations –
    - Discuss the secure code review process and what the developers can expect - typically a professional report with recommendations directed back to the developer team.
    - Don’t expect the review to find all security bugs and get them all fixed! It is going to be a balance between time, resources and what *can* be fixed.
  - Developers –
    - Go over the software system at a high level and point out some relevant details for security.
  - Code Reviewers –
    - Ask questions of the software system as it relates to each of the topics (covered in this course) like Authentication & Authorization, Session Management, Input Handling, Cryptography, Error Handling & Logging etc.

Secure Code Review Phases contd…

- **Code Analysis and Inspection**
  - You are not necessarily going to find all the security weaknesses possible!
  - Perform the code analysis using a combination of automated tools **and** manual inspection of the code.
  - Document and record the security defects found.
  - Deliver a professional report citing the security weaknesses found to the developer team.
- **Rework**
  - Allow developers to fix and resubmit for final Secure Code Review.
- **Post release follow up**
  - Follow up on how the software is being released and maintained.
  - Garner feedback on the review process from developers, managers etc.

How do I report weaknesses found in code?

- **Use the CWE™ identifiers discussed during this course!**
- **Report:**
  - The weakness found,
  - Specific locations of the weaknesses found (code filename, line number),
  - A brief description of the finding,
  - Potential steps to mitigate it
- **Discuss how it is applicable to the issue found and what its impact is.**
- **Important: What is the impact of the findings on the software system in terms of security?**
- **Can also look at whether issues found are in the CWE Top 25 Most Dangerous Software Weaknesses!**

Source:
1. Wilme Tanabi. DeviantArt: Jedi Order Scratched. https://www.deviantart.com/wilmetanabi/art/Jedi-Order-Scratched-246623989. CC BY-NC-ND-3.0.

How much time should the manual code inspection take?

- Goal is to find high level security weaknesses in code that can easily be fixed thereby reducing the attack surface and weakness chain that an attacker can take advantage of.
- You will not be able to find all weaknesses and bugs.
- Studies have found that the number of defects found by a reviewer tend to start tapering off at around 60 minutes to 90 minutes.
- The review can be broken up into chunks of 60-90 minute blocks for as long as it takes to cover all the source code for the software system.

	Checklist	Systematic	Use-Case
Defects (of 14)	7.3	6.2	5.7
False-Positives	3.4	3.2	2.9
Inspection Time	72.1	77.0	81.9
Defect Rate	6.07	4.83	4.18

Figure 6: Comparing results from three types of reviews. Inspection time is in minutes. Defect rate is in defects per hour.

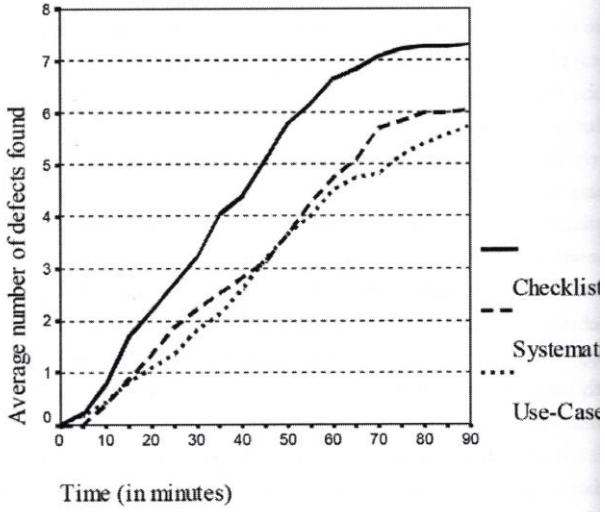


Figure 7: Elapsed time versus cumulative number of defects found for each of the three types of review.

The defect rate is constant until about 60 minutes into the inspection at which point it levels off with no defects found at all after 90 minutes.

Do Manual Review or use automated tools?

- Use both.
- First review manually based on interview with developers focusing on areas critical to the topics covered in the course like inputs, cryptography, authentication & authorization, session management, error handling and logging etc.
- Then apply tools to look deeper at findings. Typically, tools will be applied already as part of Implementation & Verification in the lifecycle. However, it can guide you on where to manually review as well.

Security Code Scan (.NET)

- https://security-code-scan.github.io/
- Static code analysis tool that is also available as a Visual Studio plugin or a standalone runner (integration with Git hooks etc).
- Can configure with custom data sources and sinks, as well as custom sanitizers and validators.

Microsoft.CodeAnalysis.NetAnalyzers

- `PM> Install-Package Microsoft.CodeAnalysis.NetAnalyzers -Version 5.0.3`
- https://github.com/dotnet/roslyn-analyzers#microsoftcodeanalysisnetanalyzers
- **This supercedes FxCop .NET analyzer since all the rules have been ported to NetAnalyzers.**
- **Enabled automatically by default in .NET5.**
- **For older .NET versions:**

```
<PropertyGroup>
    <EnableNETAnalyzers>true</EnableNETAnalyzers>
    <AnalysisLevel>latest</AnalysisLevel>
</PropertyGroup>
```

- **The documentation on rules can be found: https://docs.microsoft.com/en-us/dotnet/fundamentals/code-analysis/quality-rules/?view=vs-2019**
- **Includes design, style and security rules.**