No reason to be SAD?
The Integration of a Secure Software Development Life Cycle in OSS Enterprise Web Applications

Thomas Biege
Project Manager IT-Security
SUSE Linux Products GmbH
thomas@suse.de

OWASP
2011/11/17

The OWASP Foundation
http://www.owasp.org

Agenda

- Why do we want secure software?
- SAD: SUSE's secure life cycle approach
- Lessons learned
- What's up next?
- Tools, docs, processes used

Why do we want secure software?

- To safe money, work, time, avoid headache... choose a pain based on your role.
- And the customer?
- Information Security's biggest challenge is (industrial) espionage and criminals.
- Secure software is the key element in Information Security.
- But you need a holistic approach that is transparent and measurable... and comes CHEAP!

SAD: SUSE's secure Life Cycle Approach

- … but first some numbers:
  - Even if you have your maintenance infrastructure in place and experienced teams are working for you, each update still costs you several hundreds to thousand USD. *
  - Each update costs the customer about 600 USD and therefore makes your product **more** expensive and **less** attractive. **
  - Fixing a bug late in the life cycle makes it **much** more (up to 100x) expensive. ***
  - Toyota returned about 8 million cars because of broken breaks, costs 1.93 billion USD :-)

* developer, QA, coordination, build system, storage, etc. more than you would expect probably

** Forbath, Theo; Kalaher, Patrick; O'Grady, Thomas: The Total Cost of Security Patch Management – A Comparison of Microsoft Windows and Open Source Software.

*** Boehm, B.; The high bosts of software. In: Practical Strategies for Developing Large Software Systems

SAD: Cost of Fixing Bugs – 1 *

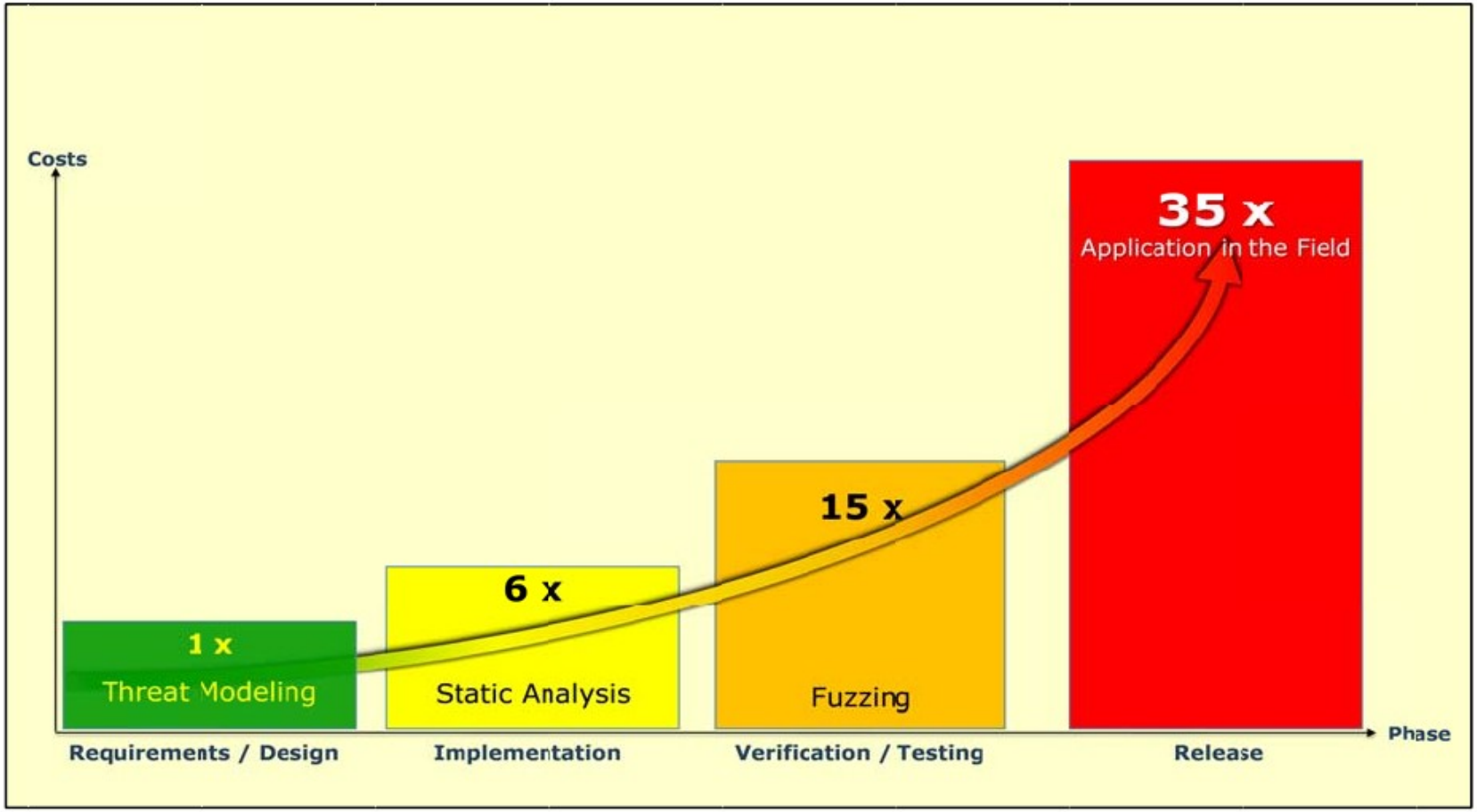


Abb. 1: Kosten der Fehlerbehebung in den Software-Entwicklungsphasen

* from: Prof. Dr. Pohl, Helmut; Identifizierung nicht-erkannter Sicherheitslücken, page 3

SAD: Cost of Fixing Bugs – 2 *

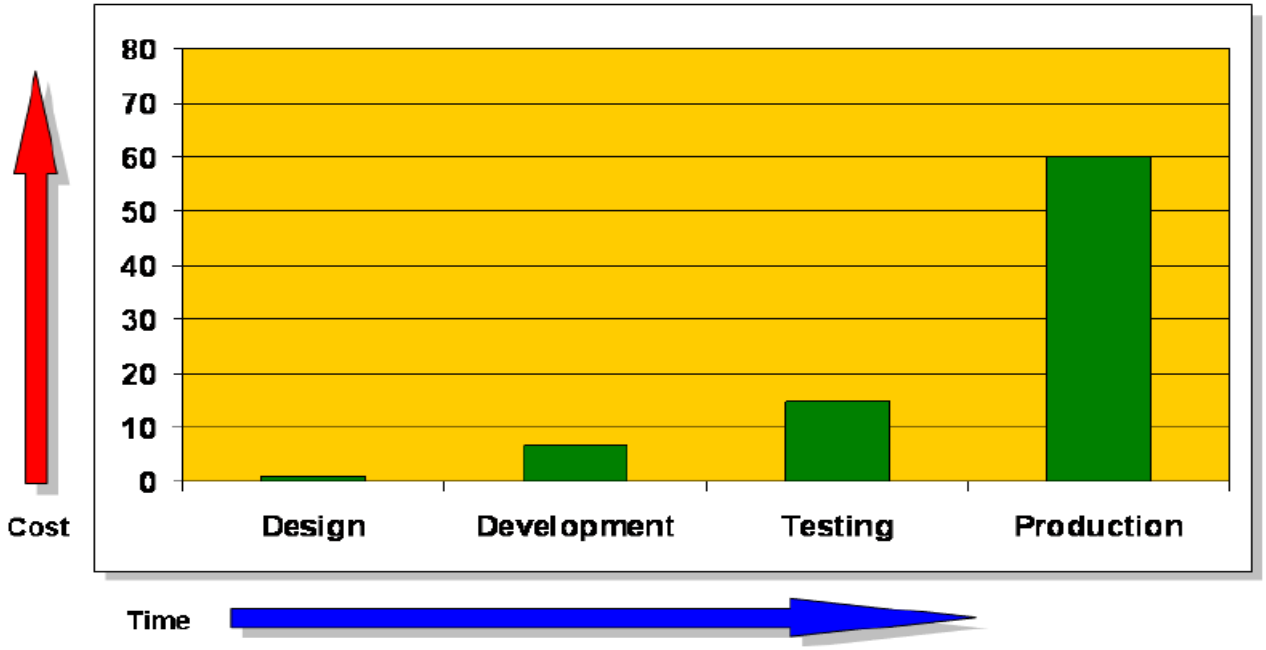
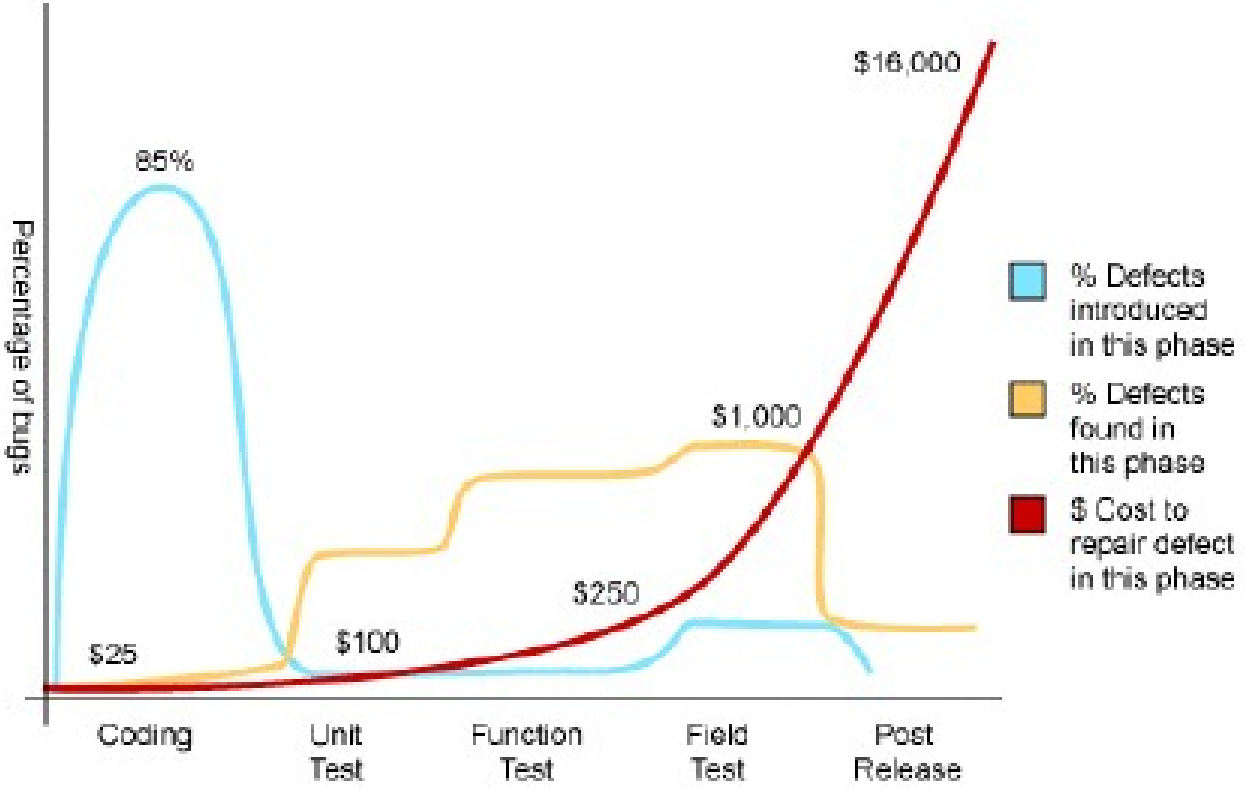


Figure 3: The cost of security

A 2002 study illustrates this point perfectly: the U.S. Department of Commerce National Institute of Standards and Technology concluded that software errors (or *bugs*) cost the U.S. economy $59 billion annually, or about 0.6% of the gross domestic product. Microsoft estimates that their average cost to fix a bug after product release is $250,000. Of course, your organization's costs likely will be much lower than Microsoft's, but it is worth considering how much effort is required to regress and compatibility test every change.

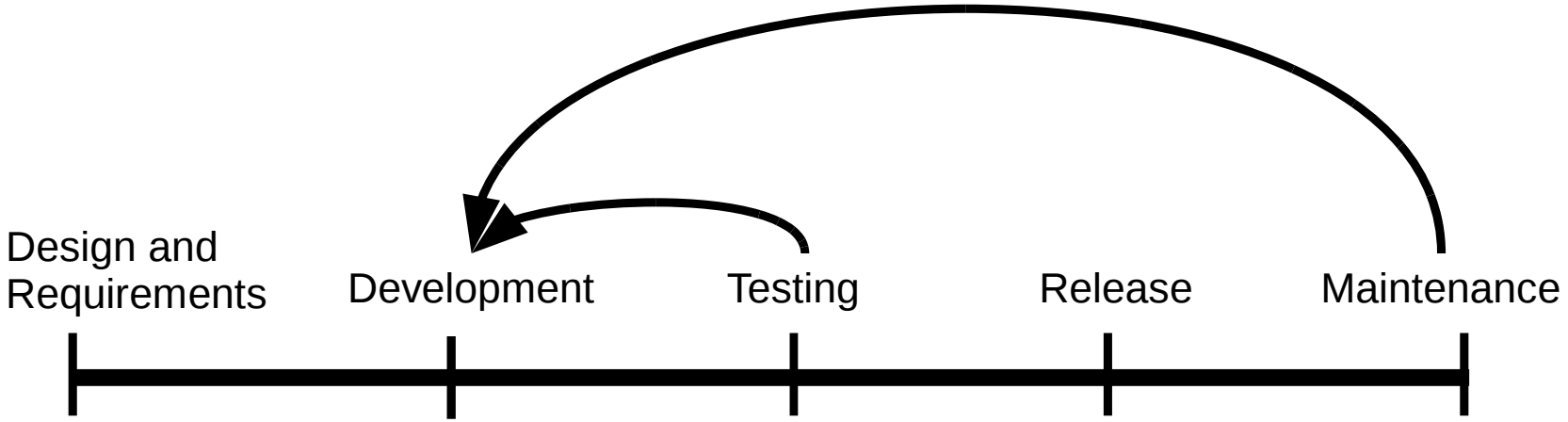
* from: IOActive; Improving RoI by Using an SDL:
http://www.ioactive.com/pdfs/ImprovingRoIByUsingAnSDL.pdf

SAD: Cost of Fixing Bugs – 3

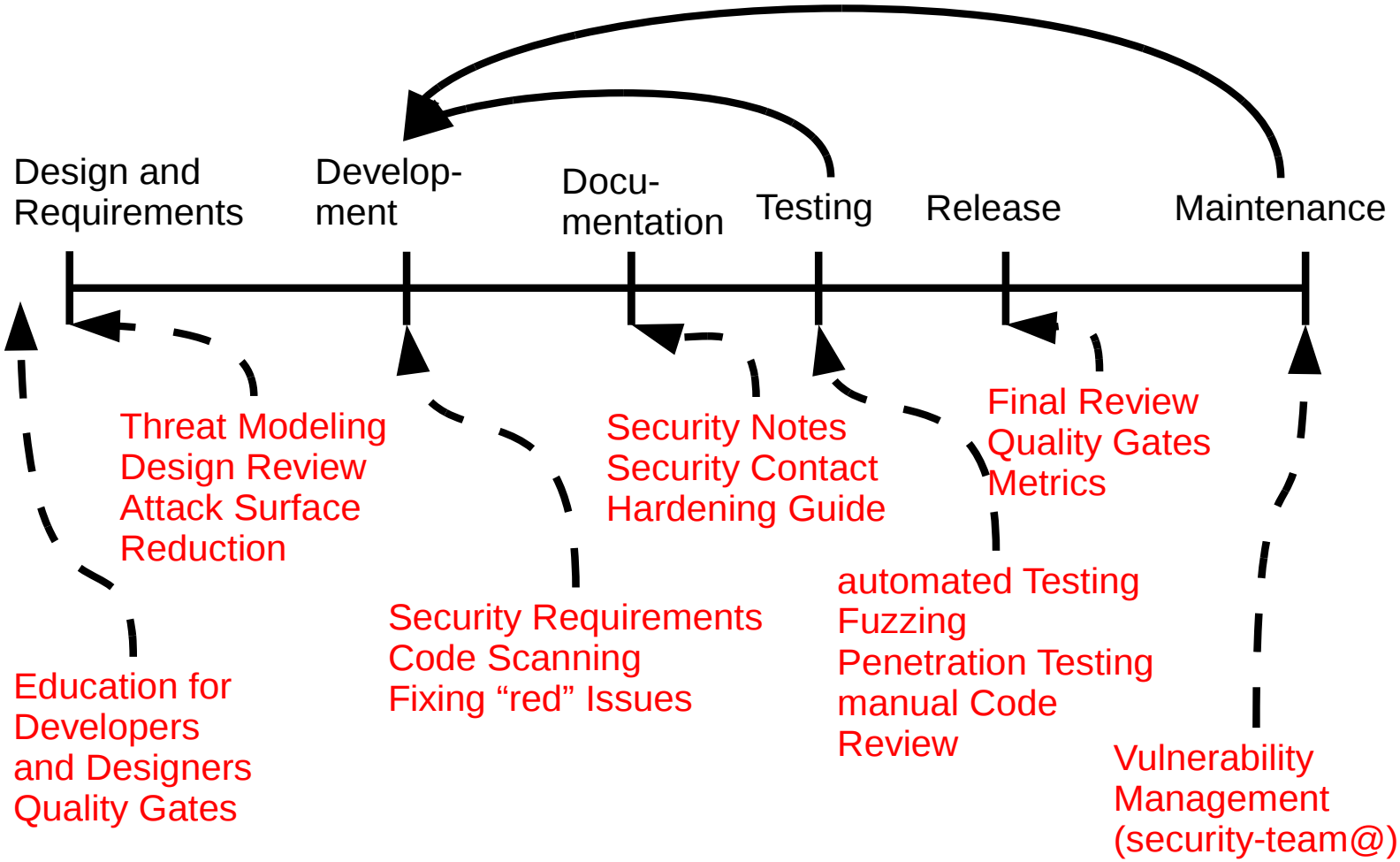


Source: Applied Software Measurement, Capers Jones, 1996

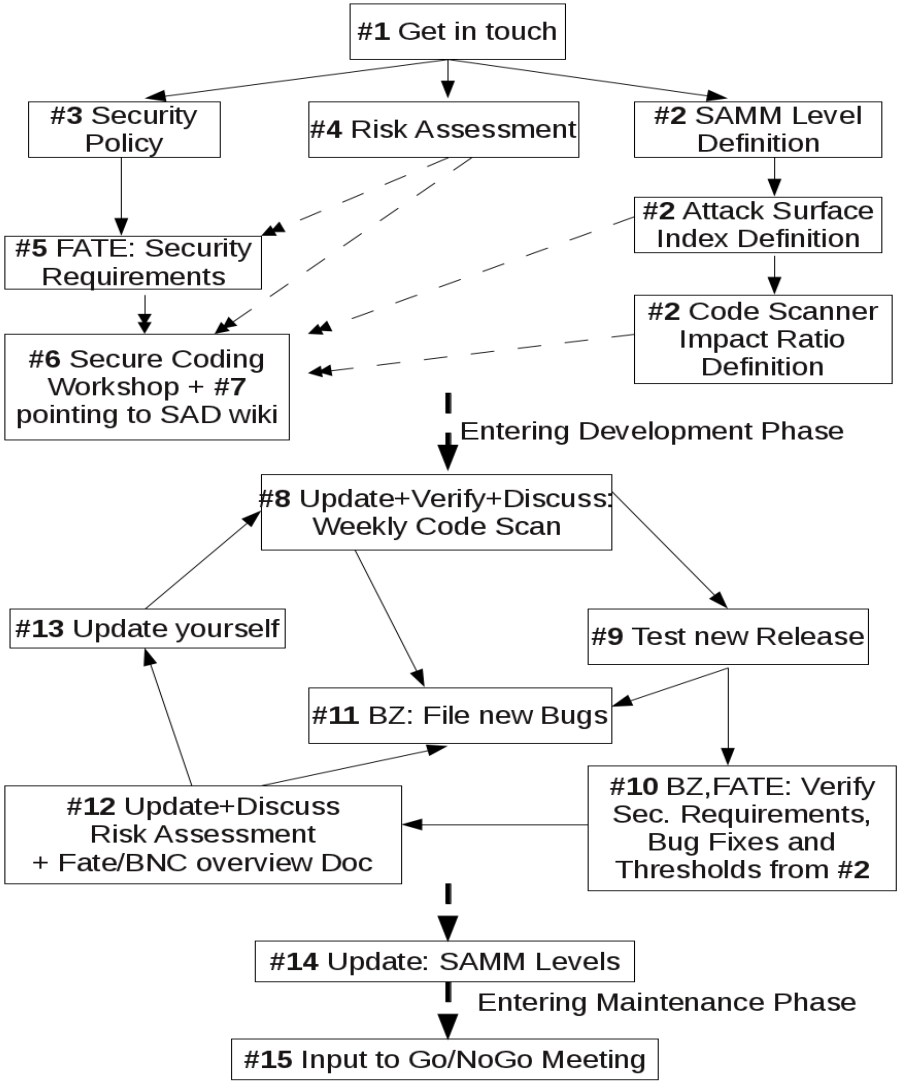
SAD: abstract Development Life Cycle



SAD: secure Development Life Cycle



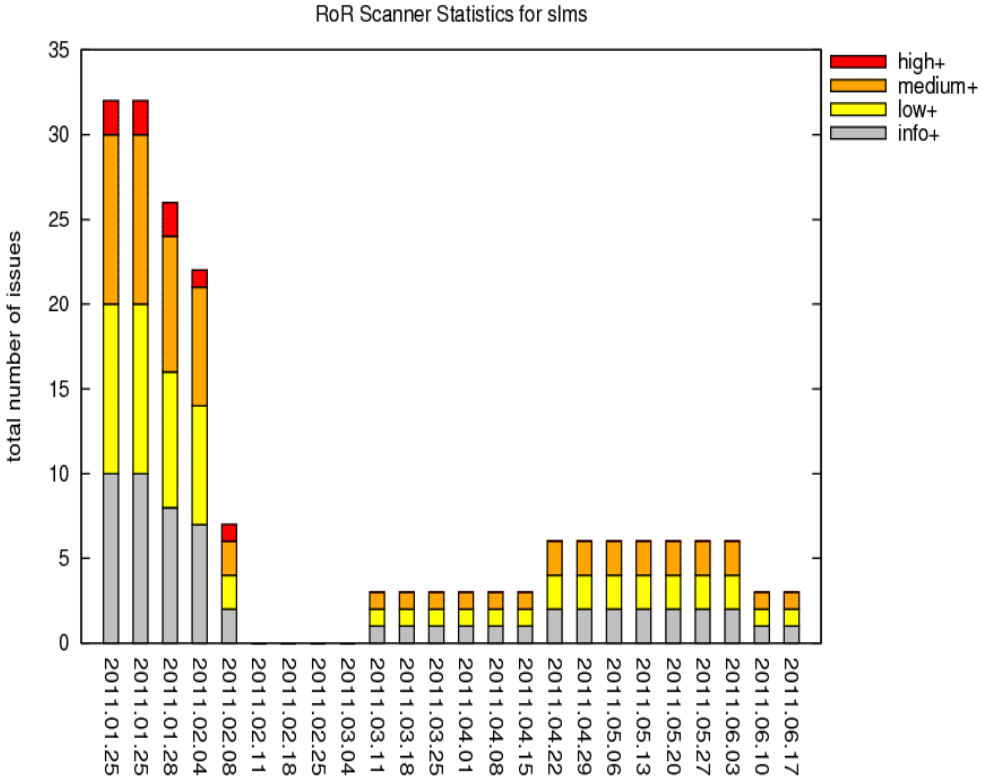
SAD: Our Process

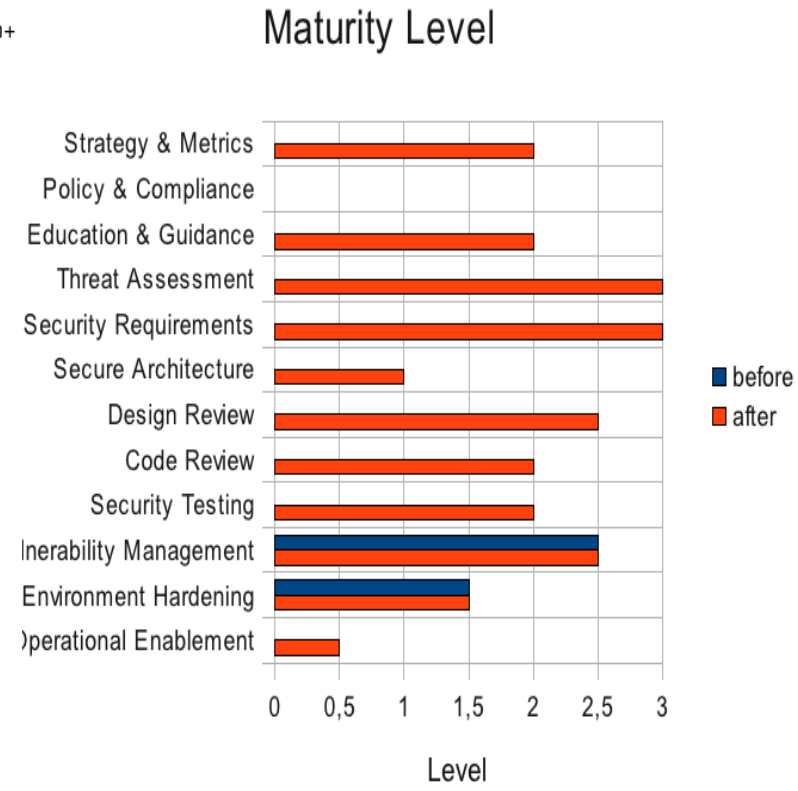


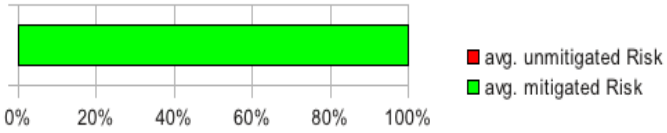
Lessons learned

- *Step 1*: Biggest hurdle was to create awareness but after that and...
- ... due to the Agile/Scrum/Kanban development processes which are based on cooperation, pragmatism and prioritizing it becomes a fruitful work
- *Step 2*: It is always good to implement the most effective methods (like, threat modeling, static code analysis, sec. testing, and fuzzing) first
- *Step 3*: Measurement (SAMM, Attack Surface Index, Code Defect Ratio per Impact Level)

Lessons learned: Case Study Product X







What's up next?

- Get all teams on board.
- Hand over to development/testing teams.
  - Static code scanner results and fixing
  - Security testing
- Agree on measurable results.
- More complete and easy to use tools for testing and verification.

Tools, Docs, Processes used

- OWASP testing guide
- OWASP testing guide suite:
  - https://gitorious.org/sectestsuite/websec
- SAMM:
  - http://www.opensamm.org
- OWASP CLASP:
  - https://www.owasp.org/index.php/Category:OWASP_CLASP_Project
- WebScarab:
  - https://www.owasp.org/index.php/Category:OWASP_WebScarab_Project
- Attack Surface Analyzer:
  - https://gitorious.org/sectestsuite/asi
- XMLRPC/REST fuzzer:
  - https://gitorious.org/fuzzer/fuzz-xmlrpc
- Ruby on Rails code scanner:
  - https://gitorious.org/code-scanner/ror-sec-scanner
  - https://github.com/openSUSE/scanny
- and more: https://gitorious.org/~thomasbiege

Questions?