Secure Development LifeCycles (SDLC)
Bart De Win

SecAppDev 2013
Bart De Win

- 15+ years of Information Security Experience
  - Ph.D. in Computer Science - Application Security
- Author of >60 scientific publications
- ISC² CSSLP certified
- Senior Manager @ PwC Belgium:
  - Expertise Center Leader Secure Software
  - (Web) Application tester (pentesting, arch. review, code review, ...)
  - Trainer for several courses related to secure software
  - Specialized in Secure Software Development Lifecycle (SDLC)

Contact me at bart.de.win@pwc.be
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion
Application Security Problem

Software complexity

Mobile

Cloud

Technology stacks

Growing connectivity

Adaptability

Training

Better

Faster

75% of vulnerabilities are application related
The Nature of Application Security

Secure Development LifeCycles (SDLC)
SecAppDev 2013
The State-of-Practice in Secure Software Development

Problematic, since:

• Focus on bugs, not flaws
• Penetration can cause major harm
• Not cost efficient
• No security assurance
  - All bugs found?
  - Bug fix fixes all occurrences? (also future?)
  - Bug fix might introduce new security vulnerabilities
Enterprise-wide software security improvement program

- Strategic approach to assure software quality
- Goal is to increase systematicity
- Focus on security functionality and security hygiene
**SDLC Objectives**

To develop (and maintain) software in a **consistent** and **efficient** way with a **demonstrable & standards-compliant security quality**, inline with the organizational **risks**.
SDLC Cornerstones

- People
  - Roles & Responsibilities

- Process
  - Activities
  - Deliverables
  - Control Gates

- Knowledge
  - Standards & Guidelines
  - Compliance
  - Transfer methods

- Tools & Components
  - Development support
  - Assessment tools
  - Management tools
Organizations with a proper SDLC will experience an 80 percent decrease in critical vulnerabilities.

Organizations that acquire products and services with just a 50 percent reduction in vulnerabilities will reduce configuration management and incident response costs by 75 percent each.
Does it really work?

Vulnerabilities disclosed three years after release

- 34
- 3
- 91% DECREASE

Vulnerabilities disclosed one year after release

- Windows XP Before SDL: 119
- Windows Vista After SDL: 66
- OSI: 400
- OS II: 242
- OS III: 157

45% DECREASE

**SDLC-related initiatives**

<table>
<thead>
<tr>
<th>Training</th>
<th>Requirements</th>
<th>Design</th>
<th>Implementation</th>
<th>Verification</th>
<th>Release</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core training</td>
<td>Analyze security and privacy risks</td>
<td>Define quality gates</td>
<td>Threat modeling</td>
<td>Attack surface analysis</td>
<td>Specify tools</td>
<td>Enforce banned functions</td>
</tr>
</tbody>
</table>

- Microsoft SDL
- Microsoft SDL
- BSIMM
- GGASSP
- CLASP

- TouchPoints
- SP800-64
- SSE-CMM
- SAMM
- TSP-Secure
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
- GASSP
- SAMM
- CMU
- TSP-Secure Secure Development LifeCycles (SDLC)
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion
Selected Example: Microsoft SDL (SD3+C)
Training

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

Content

• Secure design
• Threat modeling
• Secure coding
• Security testing
• Privacy

Why?
Requirements

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

Project inception

When you consider security and privacy at a foundational level

Cost analysis

Determine if development and support costs for improving security and privacy are consistent with business needs
Establish and follow best practices for Design

≠ secure-coding best practices

Risk analysis

Threat modeling

STRIDE
Implementation

Creating documentation and tools for users that address security and privacy

Establish and follow best practices for development

1. Review available information resources
2. Review recommended development tools
3. Define, communicate and document all best practices and policies

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response
Verification

Security and privacy testing

1. Confidentiality, integrity and availability of the software and data processed by the software
2. Freedom from issues that could result in security vulnerabilities

Security push
Release

Public pre-release review

1. Privacy
2. Security

Planning

Preparation for incident response
Release

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

Final security and privacy review

Outcomes:
- Passed FSR
- Passed FSR with exceptions
- FSR escalation

Release to manufacturing/release to web

Sign-off process to ensure security, privacy and other policy compliance
Execute Incident Response Plan

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

=> able to respond appropriately to reports of vulnerabilities in their software products, and to attempted exploitation of those vulnerabilities.
Process Models: wrapup

Microsoft SDL:
- Mature, long-term practical experience
- Heavyweight, ISV flavour
- Several supporting tools and methods

Other process models exist, with their pro’s and con’s

In general, no process will fit your organization perfectly
Mix-and-Match + adaptation are necessary
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion
**Why Maturity Models?**

An organization’s behavior changes slowly over time.
- Changes must be iterative while working toward long-term goals

There is no single recipe that works for all organizations
- A solution must enable risk-based choices tailor to the organization

Guidance related to security activities must be prescriptive
- A solution must provide enough details for non-security-people

Overall, must be simple, well-defined, and measurable
Selected example: OpenSAMM

http://www.opensamm.org

Version 1.0, 2009
Core Structure

SAMM Overview

Business Functions
- Governance
- Construction
- Verification
- Deployment

Security Practices
- Strategy & Metrics
- Education & Guidance
- Security Requirements
- Design Review
- Security Testing
- Environment Hardening
- Policy & Compliance
- Threat Assessment
- Secure Architecture
- Code Review
- Vulnerability Management
- Operational Enablement

Software Development
## Notion of Maturity

<table>
<thead>
<tr>
<th>Level</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Implicit starting point representing the activities in the practice being unfulfilled</td>
</tr>
<tr>
<td>1</td>
<td>Initial understanding and ad-hoc provision of the security practice</td>
</tr>
<tr>
<td>2</td>
<td>Increase efficiency and/of effectiveness of the security practice</td>
</tr>
<tr>
<td>3</td>
<td>Comprehensive mastery of the security practice at scale</td>
</tr>
</tbody>
</table>
An example

<table>
<thead>
<tr>
<th>Code Review</th>
<th>...more on page 62</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CR 1</strong></td>
<td><strong>CR 2</strong></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Objective</strong></td>
<td><strong>Activities</strong></td>
</tr>
<tr>
<td>Opportunistically find basic code-level vulnerabilities and other high-risk security issues</td>
<td>Make code review during development more accurate and efficient through automation</td>
</tr>
<tr>
<td>Mandate comprehensive code review process to discover language-level and application-specific risks</td>
<td>A. Create review checklists from known security requirements</td>
</tr>
<tr>
<td></td>
<td>B. Perform point-review of high-risk code</td>
</tr>
<tr>
<td></td>
<td>A. Utilize automated code analysis tools</td>
</tr>
<tr>
<td></td>
<td>B. Integrate code analysis into development process</td>
</tr>
<tr>
<td></td>
<td>A. Customize code analysis for application-specific concerns</td>
</tr>
<tr>
<td></td>
<td>B. Establish release gates for code review</td>
</tr>
</tbody>
</table>
OpenSAMM also defines

**Objective**

**Activities**

**Results**

**Success Metrics**

**Costs**

**Personnel**

**Related Levels**

ACTIVITIES

A. Employ application-specific security testing automation

Through either customization of security testing tools, enhancements to generic test case execution tools, or buildout of custom test harnesses, project teams should formally iterate through security requirements and build a set of automated checkers to test the security of the implemented business logic.

Additionally, many automated security testing tools can be greatly improved in accuracy and depth of coverage if they are customized to understand more detail about the specific software interfaces in the project under test. Further, organization-specific concerns from compliance or technical standards can be codified as a reusable, central test battery to make audit data collection and per-project management visibility simpler.

Project teams should focus on buildout of granular security test cases based on the business functionality of their software, and an organization-level team led by a security auditor should focus on specification of automated tests for compliance and internal standards.

B. Establish release gate for security testing

To prevent software from being released with easily found security bugs, a particular point in the software development life-cycle should be identified as a checkpoint where an established set of security test cases must pass in order to make a release from the project. This establishes a baseline for the kinds of security tests all projects are expected to pass.

Since adding too many test cases initially can result in an overhead cost bubble, begin by choosing one or two security issues and include a wide variety of test cases for each with the expectation that no project may pass if any test fails. Over time, this baseline should be improved by selecting additional security issues and adding a variety of corresponding test cases.

Generally, this security testing checkpoint should occur toward the end of the implementation or testing, but must occur before release.

For legacy systems or in-scope projects, an exception process should be created to allow those projects to continue operations, but with an explicitly assigned timeframe for mitigation of findings. Exceptions should be limited to no more than 20% of all projects.

RESULTS

- Organization-wide baseline for expected application performance against attacks
- Customized security test suites to improve accuracy of automated analysis
- Project teams aware of objective goals for attack resistance

ADDITIONAL SUCCESS METRICS

- >50% of projects using security testing customizations
- >75% of projects passing all security tests in past 6 months

ADDITIONAL COSTS

- Buildout and maintenance of customizations to security testing automation
- Ongoing project overhead from security testing audit process
- Organization overhead from project delays caused by failed security testing audits

ADDITIONAL PERSONNEL

- Architecture (1 day/yr)
- Developers (1 day/yr)
- Security Auditors (1-2 day/yr)
- QA Testers (1-2 day/yr)
- Business Owners (1 day/yr)
- Managers (1 day/yr)

RELATED LEVELS

- Policy & Compliance - 2
- Secure Architecture - 3
Assessments

Secure Architecture

- Are project teams provided with a list of recommended third-party components?
- Are most project teams aware of secure design principles and applying them?
- Do you advertise shared security services with guidance for project teams?
- Are project teams provided with prescriptive design patterns based on their application architecture?
- Are project teams building software from centrally controlled platforms and frameworks?
- Are project teams being audited for usage of secure architecture components?

Yes/No

<table>
<thead>
<tr>
<th>Secure Architecture</th>
<th>Yes/No</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SA 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SA 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy &amp; Metrics</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy &amp; Compliance</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Education &amp; Guidance</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Threat Assessment</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Security Requirements</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Secure Architecture</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Design Review</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Code Review</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Security Testing</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Vulnerability Management</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Environment Hardening</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Operational Enablement</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

March 2013
Roadmap templates per company type (ISV)
BSIMM4 statistics: summary
BSIMM4 statistics: per activity (TODO)

<table>
<thead>
<tr>
<th>Governance</th>
<th>Intelligence</th>
<th>SSDL Touchpoints</th>
<th>Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Activity</td>
<td>Activity</td>
<td>Activity</td>
</tr>
<tr>
<td>Observed</td>
<td>Observed</td>
<td>Observed</td>
<td>Observed</td>
</tr>
<tr>
<td>SM1.1</td>
<td>35</td>
<td>[AM1.1]</td>
<td>15</td>
</tr>
<tr>
<td>SM1.2</td>
<td>30</td>
<td>[AM1.2]</td>
<td>31</td>
</tr>
<tr>
<td>SM1.3</td>
<td>33</td>
<td>[AM1.3]</td>
<td>25</td>
</tr>
<tr>
<td>SM1.4</td>
<td>44</td>
<td>[AM1.4]</td>
<td>13</td>
</tr>
<tr>
<td>SM1.6</td>
<td>35</td>
<td>[AM1.5]</td>
<td>32</td>
</tr>
<tr>
<td>SM2.1</td>
<td>21</td>
<td>[AM1.6]</td>
<td>17</td>
</tr>
<tr>
<td>SM2.2</td>
<td>26</td>
<td>[AM2.1]</td>
<td>12</td>
</tr>
<tr>
<td>SM2.3</td>
<td>26</td>
<td>[AM2.2]</td>
<td>13</td>
</tr>
<tr>
<td>SM2.5</td>
<td>22</td>
<td>[AM3.1]</td>
<td>3</td>
</tr>
<tr>
<td>SM3.1</td>
<td>15</td>
<td>[AM3.2]</td>
<td>5</td>
</tr>
<tr>
<td>SM3.2</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP1.1</td>
<td>40</td>
<td>[SFD1.1]</td>
<td>44</td>
</tr>
<tr>
<td>CP1.2</td>
<td>45</td>
<td>[SFD1.2]</td>
<td>37</td>
</tr>
<tr>
<td>CP1.3</td>
<td>36</td>
<td>[SFD2.1]</td>
<td>25</td>
</tr>
<tr>
<td>CP2.1</td>
<td>21</td>
<td>[SFD2.2]</td>
<td>19</td>
</tr>
<tr>
<td>CP2.2</td>
<td>20</td>
<td>[SFD2.3]</td>
<td>15</td>
</tr>
<tr>
<td>CP2.3</td>
<td>25</td>
<td>[SFD3.1]</td>
<td>8</td>
</tr>
<tr>
<td>CP2.4</td>
<td>22</td>
<td>[SFD3.2]</td>
<td>9</td>
</tr>
<tr>
<td>CP2.5</td>
<td>31</td>
<td>[CR1.1]</td>
<td>23</td>
</tr>
<tr>
<td>CP3.1</td>
<td>7</td>
<td>[CR1.2]</td>
<td>20</td>
</tr>
<tr>
<td>CP3.2</td>
<td>12</td>
<td>[CR1.4]</td>
<td>33</td>
</tr>
<tr>
<td>CP3.3</td>
<td>8</td>
<td>[CR1.5]</td>
<td>11</td>
</tr>
<tr>
<td>[T1.1]</td>
<td>38</td>
<td>[SR1.1]</td>
<td>38</td>
</tr>
<tr>
<td>T1.5</td>
<td>19</td>
<td>[SR1.2]</td>
<td>27</td>
</tr>
<tr>
<td>T1.6</td>
<td>21</td>
<td>[SR1.3]</td>
<td>34</td>
</tr>
<tr>
<td>T1.7</td>
<td>23</td>
<td>[SR1.4]</td>
<td>21</td>
</tr>
<tr>
<td>T2.5</td>
<td>10</td>
<td>[SR2.1]</td>
<td>12</td>
</tr>
<tr>
<td>T2.6</td>
<td>12</td>
<td>[SR2.2]</td>
<td>20</td>
</tr>
<tr>
<td>T2.7</td>
<td>11</td>
<td>[SR2.3]</td>
<td>18</td>
</tr>
<tr>
<td>T3.1</td>
<td>5</td>
<td>[SR2.4]</td>
<td>5</td>
</tr>
<tr>
<td>T3.2</td>
<td>5</td>
<td>[SR2.5]</td>
<td>21</td>
</tr>
<tr>
<td>T3.3</td>
<td>8</td>
<td>[SR3.1]</td>
<td>8</td>
</tr>
<tr>
<td>T3.4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3.5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Maturity Models wrapup

OpenSAMM

- Comprehensive and rich model, more than just activities
- Supporting tools are available
- Real-world case studies, but few are openly shared

Other models exist with their pro’s and con’s

Maturity models provide an excellent framework for reasoning on software assurance, on a strategic level.
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion
Before you begin

Organizational Context

Realistic Goals?

Scope?

Constraints (budget, timing, resources)

Affinity with a particular model?
What’s your Company Maturity?

- In terms of IT strategy and application landscape
- In terms of software Development practices
  - Analysis, Design, Implementation, Testing, Release, Maintenance
- In terms of ITSM practices

Company Maturity $\approx$ Feasibility SDLC Program
Complicating factors, anyone?

- Different development teams
- Different technology stacks
- Business-IT alignment issues
- Outsourced development
- ...
Common SDLC strategies

- **Enterprise-wide**
  - Focus on overall methods and practices
  - Fundamental approach

- **Project-specific**
  - Focus on 1 particular project
  - Targeted approach

- **Problem-specific**
  - Focus on 1 specific problem
  - Ad-hoc approach
Typical Approach

As-Is  To-Be  Improvements
As-Is

Maturity Evaluation (in your favorite model)

Depending on (your knowledge of) the organisation, you might be able to do this on your own

If not, interviews with different stakeholders will be necessary
  Analyst, Architect, Tech Lead, QA, Ops, Governance

Discuss outcome with the stakeholders and present findings to the project advisory board
**Scoping**

For large companies, teams will perform differently

=> difficult to come up with a single result

Consider

• Reducing the scope to a single, uniform unit
• splitting the assessment into different organizational subunits

Splitting might be awkward at first, but can be helpful later on for motivational purposes
**To-Be**

Identify the targets for your company

Define staged roadmap and overall planning

Define application migration strategy

Gradual improvements work better than big bang

Have this validated by the project advisory board
## Staged Roadmap

<table>
<thead>
<tr>
<th>Security Practices/Phase</th>
<th>Start</th>
<th>One</th>
<th>Two</th>
<th>Three</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strategy &amp; metrics</strong></td>
<td>0,5</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>Policy &amp; Compliance</strong></td>
<td>0</td>
<td>0,5</td>
<td>1</td>
<td>1,5</td>
</tr>
<tr>
<td><strong>Education &amp; Guidance</strong></td>
<td>0,5</td>
<td>1</td>
<td>2</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Threat Assessment</strong></td>
<td>0</td>
<td>0,5</td>
<td>2</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Security Requirements</strong></td>
<td>0,5</td>
<td>1,5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>Secure Architecture</strong></td>
<td>0,5</td>
<td>1,5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>Design Review</strong></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Code Review</strong></td>
<td>0</td>
<td>0,5</td>
<td>1,5</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Security Testing</strong></td>
<td>0,5</td>
<td>1</td>
<td>1,5</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Vulnerability</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Management</strong></td>
<td>2,5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Environment Hardening</strong></td>
<td>2,5</td>
<td>2,5</td>
<td>2,5</td>
<td>2,5</td>
</tr>
<tr>
<td><strong>Operational Enablement</strong></td>
<td>0,5</td>
<td>0,5</td>
<td>1,5</td>
<td>3</td>
</tr>
<tr>
<td><strong>Total Effort per Phase</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* March 2013

**Secure Development LifeCycles (SDLC)**

**SecAppDev 2013**
Implementation

Implementation of dedicated activities according to the plan

Iterative, Continuous Process

Leverage good existing practices
**Entry Points**

- Pick the weak spots that can demonstrate short-term ROI

- Typical examples
  - Awareness training
  - Coding Guidelines
  - External Pentesting

- Success will help you in continuing your effort
Application categorization

Use this to rationalize security effort (according to the application risk)

Granularity!

Inter-Connectivity!
Communication & Support

Critical success factor!

Spreading the message – broad audience

Setup a secure applications portal!

Regular status updates towards management
Monitoring & Metrics

The only valid measurement of code quality: WTFs/minute

(c) 2008 Focus Shift
Responsibilities

Core Security team

Security Satellite

Analysts

Architects

Developers

Operations

Management

Formalized RACI will be a challenge
The Power of Default Security

Construct development frameworks that are secure by default

Minimizes work for developers

Will lower number of vulns.
What about Agile Development?

Security improvements must be aligned to the company practices.

Security typically better aligns to waterfall-like processes, however can be used in agile methods as well

- Organisation of activities is different
- Setup of activities needs to be adapted to the techniques used in the concrete process (e.g., abuser stories for threat modelling)
SDLC Cornerstones (revisited)

- People:
  - Roles & Responsibilities

- Process:
  - Activities
  - Deliverables
  - Control Gates

- Knowledge:
  - Standards & Guidelines
  - Compliance
  - Transfer methods

- Tools & Components:
  - Development support
  - Assessment tools
  - Management tools

Risk

Training
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion
Discussion Topics

Practical experiences

Agile

Mobile

...
Conclusions

SDLC is the framework for most of this week’s sessions.

No model is perfect, but they provide good guidance.

Find balance for all cornerstones.

Risk Management is key for rationalizing effort.

Beware the big bang.