Secure Development LifeCycles (SDLC)
Bart De Win

SecAppDev 2014
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)
• OWASP OpenSAMM co-leader
• Contact me at bart.de.win@be.pwc.com
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Agile Development
5. Conclusion
Application Security Problem

- Software complexity
- Technology stacks
- Adaptability
- Mobile
- Growing connectivity
- Cloud
- Training
- Better
- Faster

75% of vulnerabilities are application related
Application Security Symbiosis
Application Security during Software Development

Secure Development LifeCycles (SDLC)
SecAppDev 2014
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

- **Risk**

- **Training**
Strategic?

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

91% DECREASE

Vulnerabilities disclosed one year after release

45% DECREASE

Source: Analysis by Jeff's

Source: Browser Vulnerability Database

(Some) SDLC-related initiatives

- Microsoft SDL
- BSIMM
- TouchPoints
- SSE-CMM
- SAMM
- SP800-64
- CLASP
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Agile Development
5. Conclusion
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
Design

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
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**

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

**Public pre-release review**

1. Privacy
2. Security

**Planning**

Preparation for incident response
Release

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. Agile Development
5. Conclusion
**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
# 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

**Code Review**

**Objective**

- CR 1: Opportunistically find basic code-level vulnerabilities and other high-risk security issues
- CR 2: Make code review during development more accurate and efficient through automation
- CR 3: Mandate comprehensive code review process to discover language-level and application-specific risks

**Activities**

<table>
<thead>
<tr>
<th>CR 1</th>
<th>CR 2</th>
<th>CR 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Create review checklists from known security requirements</td>
<td>A. Utilize automated code analysis tools</td>
<td>A. Customize code analysis for application-specific concerns</td>
</tr>
<tr>
<td>B. Perform point-review of high-risk code</td>
<td>B. Integrate code analysis into development process</td>
<td>B. Establish release gates for code review</td>
</tr>
</tbody>
</table>

...more on page 62
OpenSAMM also defines

Objective
Activities
Results
Success Metrics
Costs
Personnel
Related Levels

Security Testing

Require application-specific security testing to ensure baseline security before deployment

ACTIVITIES
A. Employ application-specific security testing automation
   Through either customization of security testing tools, enhancement to generic test case execution tools, or buildup of custom test harnesses, project teams should formally berate through security requirements and build a set of automated checklists 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 pre-project management stability 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 gates 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 inactive 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
Ado'L Success Metrics
- >50% of projects using security testing customizations
- >75% of projects passing all security tests in past 6 months

Ado'L 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

Ado'L Personnel
- Architects (1 daily)
- Developers (4 daily)
- Security Auditors (1-2 daily)
- QA Testers (1-2 daily)
- Business Owners (1 daily)
- Managers (1 daily)

RELATED LEVELS
- Policy & Compliance - 2
- Secure Architecture - 3
Assessments
Roadmap templates per company type (ISV)
BSIMM5 statistics: summary

Earth (67)

Top Ten (of 67)
### BSIMM5 statistics: per activity

<table>
<thead>
<tr>
<th>Governance</th>
<th>Intellige...</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>[SM1.1]</td>
<td>44</td>
<td>[AM1.1]</td>
<td>21</td>
</tr>
<tr>
<td>[SM1.2]</td>
<td>34</td>
<td>[AM1.2]</td>
<td>43</td>
</tr>
<tr>
<td>[SM1.3]</td>
<td>34</td>
<td>[AM1.3]</td>
<td>30</td>
</tr>
<tr>
<td>[SM1.5]</td>
<td>36</td>
<td>[AM1.5]</td>
<td>42</td>
</tr>
<tr>
<td>[SM2.1]</td>
<td>26</td>
<td>[AM2.1]</td>
<td>16</td>
</tr>
<tr>
<td>[SM2.2]</td>
<td>31</td>
<td>[AM2.2]</td>
<td>7</td>
</tr>
<tr>
<td>[SM2.3]</td>
<td>27</td>
<td>[AM2.3]</td>
<td>11</td>
</tr>
<tr>
<td>[SM2.5]</td>
<td>20</td>
<td>[AM3.1]</td>
<td>4</td>
</tr>
<tr>
<td>[SM3.1]</td>
<td>16</td>
<td>[AM3.2]</td>
<td>6</td>
</tr>
<tr>
<td>[SM3.2]</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[CP1.1]</td>
<td>[SFD1.1]</td>
<td>[CR1.1]</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>54</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>[CP1.2]</td>
<td>[SFD1.2]</td>
<td>[CR1.2]</td>
</tr>
<tr>
<td></td>
<td>52</td>
<td>53</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>[CP1.3]</td>
<td>[SFD2.1]</td>
<td>[CR1.4]</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>26</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>[CP2.1]</td>
<td>[SFD2.2]</td>
<td>[CR1.5]</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>[CP2.2]</td>
<td>[SFD3.1]</td>
<td>[CR1.6]</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>9</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>[CP2.3]</td>
<td>[SFD3.2]</td>
<td>[CR2.1]</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[CP2.4]</td>
<td>[SFD3.3]</td>
<td>[CR2.5]</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>[CP2.5]</td>
<td>[SR1.1]</td>
<td>[CR3.1]</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>48</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>[CP3.1]</td>
<td>[SR1.2]</td>
<td>[CR3.2]</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>[CP3.2]</td>
<td>[SR1.3]</td>
<td>[CR3.3]</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>[CP3.3]</td>
<td>[SR3.1]</td>
<td>[CR3.4]</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[T1.1]</td>
<td>[SR1.1]</td>
<td>[ST1.1]</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>48</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>[T1.2]</td>
<td>[SR1.2]</td>
<td>[ST1.2]</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>43</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>[T1.3]</td>
<td>[SR1.3]</td>
<td>[ST1.3]</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>45</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>[T1.4]</td>
<td>[SR1.4]</td>
<td>[ST2.1]</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>27</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>[T2.1]</td>
<td>[SR2.2]</td>
<td>[ST2.2]</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>23</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>[T2.2]</td>
<td>[SR2.3]</td>
<td>[ST3.1]</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>[T2.3]</td>
<td>[SR2.4]</td>
<td>[ST3.2]</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>[T3.1]</td>
<td>[SR3.1]</td>
<td>[ST3.3]</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>22</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>[T3.2]</td>
<td>[SR3.2]</td>
<td>[ST3.4]</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>[T3.3]</td>
<td>[SR3.3]</td>
<td>[ST3.5]</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[T3.4]</td>
<td>[SR3.4]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[T3.5]</td>
<td>[SR3.5]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</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. Agile Development
5. Conclusion
Agile Models: Rationale and Fundamentals

• Many traditional, large-scale software development projects are going wrong
  • Combination of business and technical causes
• Software is delivered late in the lifecycle
• Little flexibility during the process

Agile models focus on:
• Frequent interaction with stakeholders
• Short cycles
=> to increase flexibility and reduce risk
Agile Models: Scrum

- **Product Owner**
- **Team + P.O.**
- **Scrum Master**

- **Sprint Planning Meeting**
- **Sprint Backlog**

- **Sprint 2-3 weeks**

- **Daily Scrum W/Team**

- **Review W/Team + P.O.**

- **Team, P.O. Opt.**

- **Retrospective**

- **Product Component**
- **Usable Deliveries**

- **Don’t Touch Scope & Time During Sprint**

- **Changes!**

- **Product Backlog**
Agile & Secure development: a mismatch?

<table>
<thead>
<tr>
<th>Agile Dev.</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed &amp; Flexibility</td>
<td>Stable &amp; Rigorous</td>
</tr>
<tr>
<td>Short cycles</td>
<td>Extra activities</td>
</tr>
<tr>
<td>Limited documentation</td>
<td>Extensive analysis</td>
</tr>
<tr>
<td>Functionality-driven</td>
<td>Non-functional</td>
</tr>
</tbody>
</table>
Basic approach: Fit SLD tasks to the backlog as non-functional stories

Non-Technical vs. Technical
Requirement vs. Recommendation

Each SDL task goes in one of three types of requirements:

- Every Sprint
- Bucket
- One-Time
Every-Sprint Requirements (excerpt)

- All team members must have had security training in the past year
- All database access via parameterized queries
- Fix security issues identified by static analysis
- Mitigate against Cross-Site Request Forgery
- Update Threat models for new features
- Use Secure cookies over HTTPS
- Link all code with the /nxcompat linker option
- Encrypt all secrets such as credentials, keys and passwords
- Conduct internal security design review
Bucket Requirements (excerpt)

Bucket A: Security Verification
• Perform fuzzing (network/ActiveX/File/RPC/…)
• Manual and automated code review for high-risk code
• Penetration testing

Bucket B: Design Review
• Conduct a privacy review
• Complete threat model training

Bucket C: Planning
• Define or update the security/privacy bug bar
• Define a BC/DR plan
One-Time Requirements (excerpt)

- Create a baseline threat model
- Establish a security response plan
- Identify your team’s security expert
- Use latest compiler versions
Abuser Stories

Treat application security into software development by writing up application security risks as stories

• **Security stories:** “As a developer, I want to prevent SQL injection into my application”
  
  • Not a real user story (not relevant for product owner, but to help the development team)
  
  • Never really finished
  
  • Cfr MS examples

• **Thinking like the bad guy:** “User X should not have access to this type of data”
  
  • Think about what users don’t want to and can’t do, how to trust users, what data is involved, ...
Thou shall use Iteration Zero

Many agile projects start with an “Iteration Zero” to

- Get the team together
- Choose tools and frameworks
- Get to know the domain

This is an opportunity for security too, to

- Assign security responsible
- Select security tools
- Determine risk levels
Security Involvement in the Process

Ensure that security-savvy people are involved at important phases:

- Planning game (to enhance/verify requirements)
- Development (daily follow-up)
- Review (to support acceptance)
- Retrospective (to improve dev. Practices for security)

Different profiles can be distinguished:

- Security architect
- Security engineer
- Risk Manager/Governance
Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Agile Development
5. Conclusion
Conclusions

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

No model is perfect, but they provide good guidance

Agile development can be improved as well

Take into account all cornerstones

Risk Management is key for rationalizing effort