Bart De Win?

• 20 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 *Trusted Software*
  • (Web) Application tester (pentesting, arch. review, code review, ...)
  • Proficiency in Secure Software Development Lifecycle (SDLC) and Software Quality
• OWASP SAMM co-leader
• Contact me at bart.de.win@be.pwc.com
Agenda

1. Motivation
2. SAMM At A Glance
3. SAMM Practices
4. Conclusion
**Typical questions**

What should we be doing in our SDLC?

What are others doing in terms of software assurance?

What are good practices for software assurance?

Should we focus on threat modelling or code reviews?

How much time/effort/cost will this take?


**Maturity models to the rescue**

According to Wikipedia:

“*Maturity* is a measurement of the ability of an organisation for continuous improvement in a particular discipline.”

A *maturity model* is a structure that represents different levels of maturity for one or more domains.
Why Maturity Models for SDLC?

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 tailored 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**
Agenda

1. Motivation
2. SAMM At A Glance
3. SAMM Practices
4. Conclusion
OWASP SAMM

Scope: Entire software lifecycle, rather than just development.

https://www.owasp.org/index.php/OWASP_SAMM_Project

Version 1.1, 2016
SAMM Business Functions

- Start with the core activities tied to any organization performing software development

- Named generically, but should resonate with any developer or manager
# 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>

Changing in version 1.5!
# An example

![Code Review Table]

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

*More on page 62*
OpenSAMM also defines

Objective
Activities
Results
Success Metrics
Costs
Personnel
Related Levels
Conducting assessments

<table>
<thead>
<tr>
<th>Secure Architecture</th>
<th>Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are project teams provided with a list of recommended third-party components?</td>
<td></td>
</tr>
<tr>
<td>Are most project teams aware of secure design principles and applying them?</td>
<td></td>
</tr>
<tr>
<td>Do you advertise shared security services with guidance for project teams?</td>
<td></td>
</tr>
<tr>
<td>Are project teams provided with prescriptive design patterns based on their application architecture?</td>
<td></td>
</tr>
<tr>
<td>Are project teams building software from centrally controlled platforms and frameworks?</td>
<td></td>
</tr>
<tr>
<td>Are project teams being audited for usage of secure architecture components?</td>
<td></td>
</tr>
</tbody>
</table>

SDLC Maturity Models
Assessment process

- Supports both lightweight and detailed assessments
Roadmap templates per company type (ISV)
Creating Scorecards

- Gap analysis
  - Capturing scores from detailed assessments versus expected performance levels
- Demonstrating improvement
  - Capturing scores from before and after an iteration of assurance program build-out
- Ongoing measurement
  - Capturing scores over consistent time frames for an assurance program that is already in place

SDLC Maturity Models

EXAMPLE

SecAppDev 2017
Roadmap templates

To make the “building blocks” usable, SAMM defines Roadmaps templates for typical kinds of organizations:

- Independent Software Vendors
- Online Service Providers
- Financial Services Organizations
- Government Organizations

Organization types chosen because:

- They represent common use-cases
- Each organization has variations in typical software-induced risk
- Optimal creation of an assurance program is different for each

SDLC Maturity Models
OpenSAMM Tools

Translations of the OpenSAMM model (Spanish, Japanese, German, Ukrainian, ...)
Assessment questionnaire(s)
Roadmap chart template
Project plan template
OpenSAMM-BSIMM mapping
Benchmark Project
Mappings to security standards
  • ISO/IEC 27034, PCI, ...
**BSIMM7 statistics: summary**

![Earth Spider Chart](image)

This chart represents the maturity levels of various security practices and processes, as measured by BSIMM7 for SDLC (Software Development Life Cycle) Maturity Models.
**BSIMM7 statistics per activity**

<table>
<thead>
<tr>
<th>BSIMM7 SCORECARD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>GOVERNANCE</td>
</tr>
<tr>
<td>ACTIVITY</td>
</tr>
<tr>
<td>SM1.1</td>
</tr>
<tr>
<td>SM1.2</td>
</tr>
<tr>
<td>SM1.3</td>
</tr>
<tr>
<td>SM1.4</td>
</tr>
<tr>
<td>SM2.1</td>
</tr>
<tr>
<td>SM2.2</td>
</tr>
<tr>
<td>SM2.3</td>
</tr>
<tr>
<td>SM2.5</td>
</tr>
<tr>
<td>SM2.6</td>
</tr>
<tr>
<td>SM3.1</td>
</tr>
<tr>
<td>SM3.2</td>
</tr>
<tr>
<td>CP1.1</td>
</tr>
<tr>
<td>CP1.2</td>
</tr>
<tr>
<td>CP1.3</td>
</tr>
<tr>
<td>CP2.1</td>
</tr>
<tr>
<td>CP2.2</td>
</tr>
<tr>
<td>CP2.3</td>
</tr>
<tr>
<td>CP2.4</td>
</tr>
<tr>
<td>CP2.5</td>
</tr>
<tr>
<td>CP3.1</td>
</tr>
<tr>
<td>CP3.2</td>
</tr>
<tr>
<td>CP3.3</td>
</tr>
<tr>
<td>T1.1</td>
</tr>
<tr>
<td>T1.5</td>
</tr>
<tr>
<td>T1.6</td>
</tr>
<tr>
<td>T1.7</td>
</tr>
<tr>
<td>T2.5</td>
</tr>
<tr>
<td>T2.6</td>
</tr>
<tr>
<td>T2.7</td>
</tr>
<tr>
<td>T3.1</td>
</tr>
<tr>
<td>T3.2</td>
</tr>
<tr>
<td>T3.3</td>
</tr>
<tr>
<td>T3.4</td>
</tr>
<tr>
<td>T3.5</td>
</tr>
</tbody>
</table>
Agenda

1. Motivation
2. SAMM At A Glance
3. SAMM Practices
4. Conclusion
SAMM Security Practices - Governance

**Security Practices**
- Strategy & Metrics
- Education & Guidance
- Policy & Compliance

**Business Functions**
- Governance
- Construction
- Verification
- Operations

- Software Development
- Security Requirements
- Design Review
- Security Testing
- Environment Hardening
- Implementation Review
- Issue Management
- Operational Enablement
Strategy & Metrics

Goal is to establish a software assurance framework within an organisation

- Foundation for all other OpenSAMM practices

Characteristics:

- Measurable
- Aligned with business risk

Driver for continuous improvement and financial guidance
## Strategy & Metrics

<table>
<thead>
<tr>
<th>Objective</th>
<th>SM 1</th>
<th>SM 2</th>
<th>SM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establish unified strategic roadmap for software security within the organization</td>
<td>Measure relative value of data and software assets and choose risk tolerance</td>
<td>Align security expenditure with relevant business indicators and asset value</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Activities</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Estimate overall business risk profile</td>
<td>A. Classify data and applications based on business risk</td>
<td>A. Conduct periodic industry-wide cost comparisons</td>
<td></td>
</tr>
<tr>
<td>B. Build and maintain assurance program roadmap</td>
<td>B. Establish and measure per-classification security goals</td>
<td>B. Collect metrics for historic security spend</td>
<td></td>
</tr>
</tbody>
</table>

---

• SDLC Maturity Models
Policy & Compliance

Goal is to understand and adhere to legal and regulatory requirements
• Typically external in nature
• This is often a very informal practice in organisations!

Characteristics
• Organisation-wide vs. project-specific
• Scope

Important driver for software security requirements
## Policy & Compliance

<table>
<thead>
<tr>
<th>Policy &amp; Compliance</th>
<th>...more on page 38</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon PC1] PC 1</td>
<td></td>
</tr>
<tr>
<td><strong>OBJECTIVE</strong></td>
<td></td>
</tr>
<tr>
<td>Understand relevant governance and compliance drivers to the organization</td>
<td></td>
</tr>
</tbody>
</table>

| ![Icon PC2] PC 2    |                    |
| **ACTIVITIES**      |                    |
| A. Identify and monitor external compliance drivers |
| B. Build and maintain compliance guidelines |
| Establish security and compliance baseline and understand per-project risks |

| ![Icon PC3] PC 3    |                    |
| **ACTIVITIES**      |                    |
| A. Build policies and standards for security and compliance |
| B. Establish project audit practice |
| Require compliance and measure projects against organization-wide policies and standards |
| A. Create compliance gates for projects |
| B. Adopt solution for audit data collection |
**Education & Guidance**

Goal is to disseminate security-oriented information to *all* stakeholders involved in the software development lifecycle

- By means of standards, trainings, ...

To be integrated with organisation training curriculum

A once-of effort is not sufficient

Teach a fisherman to fish

3. Technical guidelines form the basis for several other practices
## Education & Guidance

<table>
<thead>
<tr>
<th>Education &amp; Guidance</th>
<th>...more on page 42</th>
</tr>
</thead>
<tbody>
<tr>
<td>EG 1</td>
<td>EG 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>OBJECTIVE</strong></th>
<th>Offer development staff access to resources around the topics of secure programming and deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Educate all personnel in the software life-cycle with role-specific guidance on secure development</td>
</tr>
<tr>
<td></td>
<td>Mandate comprehensive security training and certify personnel for baseline knowledge</td>
</tr>
</tbody>
</table>

| **ACTIVITIES** | A. Conduct technical security awareness training |
|               | B. Build and maintain technical guidelines |
|               | A. Conduct role-specific application security training |
|               | B. Utilize security coaches to enhance project teams |
|               | A. Create formal application security support portal |
|               | B. Establish role-based examination/certification |
SAMM Security Practices - Construction
Threat Assessment

The goal of this practice is to focus on the attacker perspective of things

- To make sure that security is not only functionality-driven
- Remember that software security = white + black

Very common practice in safety-critical systems

- Less so in others

This is where “the magic” kicks in

- Your imagination is the limit
## Threat Assessment

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activities</th>
</tr>
</thead>
</table>
| **TA 1** Identify and understand high-level threats to the organization and individual projects | A. Build and maintain application-specific threat models  
B. Develop attacker profile from software architecture |
| **TA 2** Increase accuracy of threat assessment and improve granularity of per-project understanding | A. Build and maintain abuse-case models per project  
B. Adopt a weighting system for measurement of threats |
| **TA 3** Concretely tie compensating controls to each threat against internal and third-party software | A. Explicitly evaluate risk from third-party components  
B. Elaborate threat models with compensating controls |

*SDLC Maturity Models*
Security Requirements

Goal is to make security specification more explicit
• Turn security into a positively-spaced problem

Source of security requirements
• Compliance
• Standard
• Functionality
• Quality

Requirements should be specified in a S.M.A.R.T. way
# Security Requirements

<table>
<thead>
<tr>
<th>Objective</th>
<th>Security Requirements</th>
<th>Activities</th>
</tr>
</thead>
</table>
| SR 1      | Consider security explicitly during the software requirements process | A. Derive security requirements from business functionality  
B. Evaluate security and compliance guidance for requirements |
| SR 2      | Increase granularity of security requirements derived from business logic and known risks | A. Build an access control matrix for resources and capabilities  
B. Specify security requirements based on known risks |
| SR 3      | Mandate security requirements process for all software projects and third-party dependencies | A. Build security requirements into supplier agreements  
B. Expand audit program for security requirements |

...more on page 50

- SDLC Maturity Models
**Secure Architecture**

Key practice for security

Poor decisions at this step can have major impact, and are often difficult (or costly) to fix.

2. Characteristics
   - Take into account security principles
   - Risk is a factor of all components (incl. 3rd party)

3. Use proven solutions
   - Don’t roll you own crypto
   - Use company standards and best practices
Secure Architecture

**Secure Architecture**

<table>
<thead>
<tr>
<th>Objective</th>
<th>SA 1</th>
<th>SA 2</th>
<th>SA 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td>Insert consideration of proactive security guidance into the software design process</td>
<td>Direct the software design process toward known-secure services and secure-by-default designs</td>
<td>Formally control the software design process and validate utilization of secure components</td>
</tr>
<tr>
<td><strong>Activities</strong></td>
<td>A. Maintain list of recommended software frameworks&lt;br&gt;B. Explicitly apply security principles to design</td>
<td>A. Identify and promote security services and infrastructure&lt;br&gt;B. Identify security design patterns from architecture</td>
<td>A. Establish formal reference architectures and platforms&lt;br&gt;B. Validate usage of frameworks, patterns, and platforms</td>
</tr>
</tbody>
</table>
SAMM Security Practices - Verification
**Design Review**

- security assessment of attack surface, software design and architecture
- lightweight activities => formal inspection of data flows & security mechanisms
- enforcement of baseline expectations for conducting design assessments and reviewing findings before releases are accepted.

⇒ Assess and validate artifacts to understand protection mechanisms
## Design Review

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activities</th>
</tr>
</thead>
</table>
| DR 1      | A. Identify software attack surface  
B. Analyze design against known security requirements |
| DR 2      | A. Inspect for complete provision of security mechanisms  
B. Deploy design review service for project teams |
| DR 3      | A. Develop data-flow diagrams for sensitive resources  
B. Establish release gates for design review |

Support ad hoc reviews of software design to ensure baseline mitigations for known risks. Offer assessment services to review software design against comprehensive best practices for security. Require assessments and validate artifacts to develop detailed understanding of protection mechanisms.
**Implementation Review**

Assessment of source code:
- vulnerability discovery
- related mitigation activities
- establish secure coding baseline

Will require tool investment:
- Language specific
- Basic open source tooling
- Commercial tools maturing

Process & education important!

- Lightweight checklists
- Inspect critical software
- Automation
- Increase coverage / efficacy
- Integrate in development
- Produce audit evidence
- Test & production release gates

*SDLC Maturity Models*
# Implementation Review

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activities</th>
</tr>
</thead>
</table>
| Opportunistically find basic code-level vulnerabilities and other high-risk security issues | A. Create review checklists from known security requirements  
B. Perform point-review of high-risk code |
| Make implementation review during development more accurate and efficient through automation | A. Utilize automated code analysis tools  
B. Integrate code analysis into development process |
| Mandate comprehensive implementation review process to discover language-level and application-specific risks | A. Customize code analysis for application-specific concerns  
B. Establish release gates for code review |

...more on page 52
**Security Testing**

- Based on security & compliance requirements / checklist of common vulnerabilities
- Manual testing can be done, scaled with tooling: intercepting proxy and/or scanner
- Detected defects will require validation, risk analysis & recommendations to fix
- Automate to repeat tests for each release
- Introduce security test-driven development
- Test results to be reported to & accepted by owner for each deployment
## Security Testing

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activities</th>
</tr>
</thead>
</table>
| Establish process to perform basic security tests based on implementation and software requirements | A. Derive test cases from known security requirements  
B. Conduct penetration testing on software releases |
| Make security testing during development more complete and efficient through automation | A. Utilize automated security testing tools  
B. Integrate security testing into development process |
| Require application-specific security testing to ensure baseline security before deployment | A. Employ application-specific security testing automation  
B. Establish release gates for security testing |
Security Practices - Operations

SAMM Overview

Software Development

Business Functions
- Governance
- Construction
- Verification
- Operations

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

•SDLC Maturity Models
Issue Management

Prepare for WHEN, not IF!

Symptoms of malfunctioning SDLC

• handling vulnerability reports and operational incidents
• lightweight assignment of roles => formal incident response & communication process
• Use vulnerability metrics and root-cause analysis to improve SDLC

• spoc per team & security response team
• communication & information flow is key!
• patch release process & responsible/legal disclosure
## Issue Management

### Objectives

<table>
<thead>
<tr>
<th>IM 1</th>
<th>IM 2</th>
<th>IM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understand high-level plan for responding to issue reports or incidents</td>
<td>Elaborate expectations for response process to improve consistency and communications</td>
<td>Improve analysis and data gathering within response process for feedback into proactive planning</td>
</tr>
</tbody>
</table>

### Activities

<table>
<thead>
<tr>
<th>IM 1</th>
<th>IM 2</th>
<th>IM 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Identify point of contact for security issues</td>
<td>A. Establish consistent issue response process</td>
<td>A. Conduct root cause analysis for issues</td>
</tr>
<tr>
<td>B. Create informal security response team(s)</td>
<td>B. Adopt a security issue disclosure process</td>
<td>B. Collect per-issue metrics</td>
</tr>
</tbody>
</table>

---

*SDLC Maturity Models*
Environment Hardening

- Underlying infrastructure hardening & patching
- Track (3rd party) libraries & components
  TOP-10 - A9 – Using Known Vulnerable
- Add WAF layer (virtual patching)
  ModSecurity

SDLC Maturity Models
# Environment Hardening

<table>
<thead>
<tr>
<th>Environment Hardening</th>
<th>...more on page 74</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EH 1</strong></td>
<td></td>
</tr>
<tr>
<td><strong>EH 2</strong></td>
<td></td>
</tr>
<tr>
<td><strong>EH 3</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>OBJECTIVE</strong></th>
<th><strong>Activities</strong></th>
<th><strong>Activities</strong></th>
<th><strong>Activities</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understand baseline operational environment for applications and software components</td>
<td>A. Maintain operational environment specification&lt;br&gt; B. Identify and install critical security upgrades and patches</td>
<td>A. Establish routine patch management process&lt;br&gt; B. Monitor baseline environment configuration status</td>
<td>A. Identify and deploy relevant operations protection tools&lt;br&gt; B. Expand audit program for environment configuration</td>
</tr>
<tr>
<td>Improve confidence in application operations by hardening the operating environment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validate application health and status of operational environment against known best practices</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Operational Enablement**

- Support users & operators
- Security documentation!
- Feed/document application security logs into SIEM
- Lightweight documentation => operational security guides
- Change management & end to end deployment integrity
- Even more important for outsourced development!
## Operational Enablement

<table>
<thead>
<tr>
<th>Objective</th>
<th>Activities A</th>
<th>Activities B</th>
</tr>
</thead>
<tbody>
<tr>
<td>OE 1: Enable communications between development teams and operators for critical security-relevant data</td>
<td>A. Capture critical security information for deployment</td>
<td>B. Document procedures for typical application alerts</td>
</tr>
<tr>
<td>OE 2: Improve expectations for continuous secure operations through provision of detailed procedures</td>
<td>A. Create per-release change management procedures</td>
<td>B. Maintain formal operational security guides</td>
</tr>
<tr>
<td>OE 3: Mandate communication of security information and validate artifacts for completeness</td>
<td>A. Expand audit program for operational information</td>
<td>B. Perform code signing for application components</td>
</tr>
</tbody>
</table>
Agenda

1. Motivation
2. SAMM At A Glance
3. SAMM Practices
4. Conclusion
Conclusions

Maturity models (such as SAMM) provide an excellent framework for reasoning on software assurance, on a strategic level:

- Evaluate your as-is
- Define and improve towards your to-be
- Compare against peers

Popular approach for companies today that work on software assurance
Different flavours exist, choose one that fits your company’s context.

The models are easy to start with, but challenging to fully grasp. Don’t let this scare you, and get started!