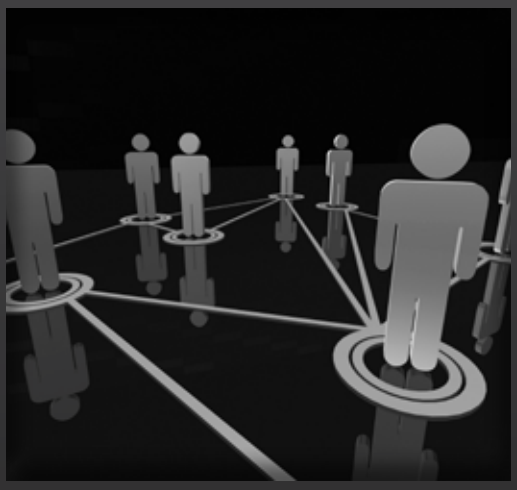


Software Assurance Maturity Model

A guide to building security into software development

Version 1.5

For the latest version and additional info, please see the project web site at

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

Acknowledgements

This document was originally created through the OpenSAMM Project led by Pravir Chandra (chandra@owasp.org), an independent software security consultant. Creation of the first draft was made possible through funding from Fortify Software, Inc. Since the initial release of SAMM, this project has become part of the Open Web Application Security Project (OWASP). This document is currently maintained and updated through the OWASP SAMM Project led by Sebastien Deleersnyder, Bart De Win & Brian Glas. Thanks also go to many supporting organizations that are listed on back cover.

Contributors & Reviewers

This work would not be possible without the support of many individual reviewers and experts that offered contributions and critical feedback.

- Fabio Arciniegas
- Matt Bartoldus
- Jonathan Carter
- Darren Challey
- Brian Chess
- Justin Clarke
- Dan Cornell
- Michael Craigue
- Dinis Cruz
- Sebastien Deleersnyder
- Justin Derry
- Bart De Win
- John Dickson
- Alexios Fakos
- David Fern
- Brian Glas
- Kuai Hinojosa
- Jerry Hoff
- Carsten Huth
- Bruce Jenkins
- Daniel Kefer
- Yan Kravchenko
- James McGovern
- Matteo Meucci
- Jeff Payne
- Gunnar Peterson
- Jeff Piper
- Andy Steingruebl
- John Steven
- Chad Thunberg
- Colin Watson
- Jeff Williams
- Steven Wierckx

This is an OWASP Project

OWASP

The Open Web Application Security Project

OWASP is an international organization and the OWASP Foundation supports OWASP efforts around the world. OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security. We advocate approaching application security as a people, process, and technology problem because the most effective approaches to application security include improvements in all of these areas. We can be found at https://www.owasp.org.

Executive Summary

The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for software security that is tailored to the specific risks facing the organization. The resources provided by SAMM will aid in:

- *Evaluating an organization's existing software security practices.*
- *Building a balanced software security assurance program in well-defined iterations.*
- *Demonstrating concrete improvements to a security assurance program.*
- *Defining and measuring security-related activities throughout an organization.*

Version 1.1 of SAMM expanded and restructured its predecessor into four complementary resources: this document that describes the core SAMM model, the How-To Guide that explains how to apply the model, the Quick Start Guide to help accelerate learning and adoption, and the toolbox that provides simple automation for data collection, metrics, and graphs. Furthermore, a number of elements have been renamed to better represent their purpose.

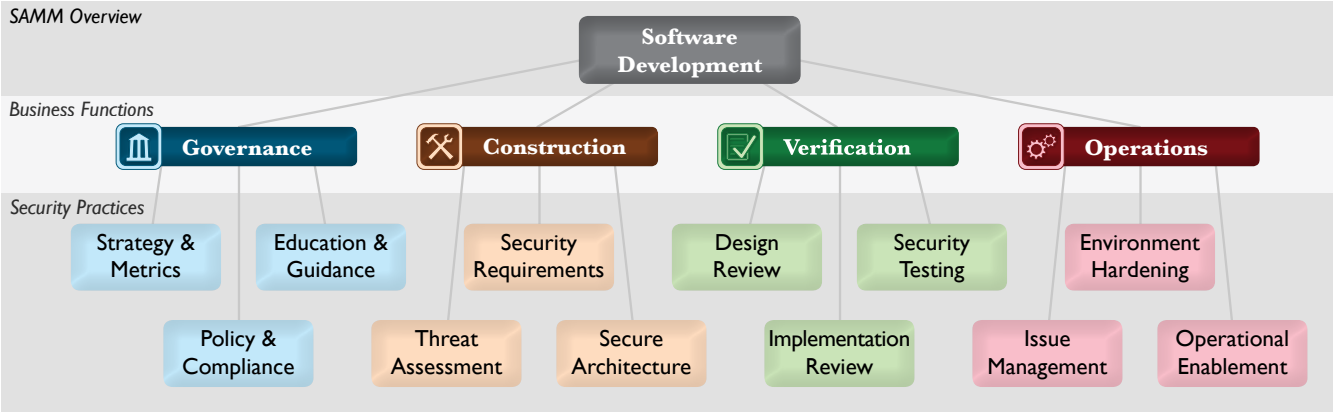
Version 1.5 of SAMM incorporates a refinement of the scoring model to provide more granularity to the scoring in an assessment. Now an organization will get credit for all the related work done in a practice rather than having the base number held at the highest completed maturity level. The updated scoring model has been designed to help SAMM assessors and organizations avoid the awkward discussion on whether to mark an answer yes or no when it is honestly something in between, and to show incremental improvements.

SAMM was defined with flexibility in mind such that it can be utilized by small, medium, and large organizations using any style of development. Additionally, this model can be applied organization-wide, for a single line-of-business, or even for an individual project. Beyond these traits, SAMM was built on the following principles:

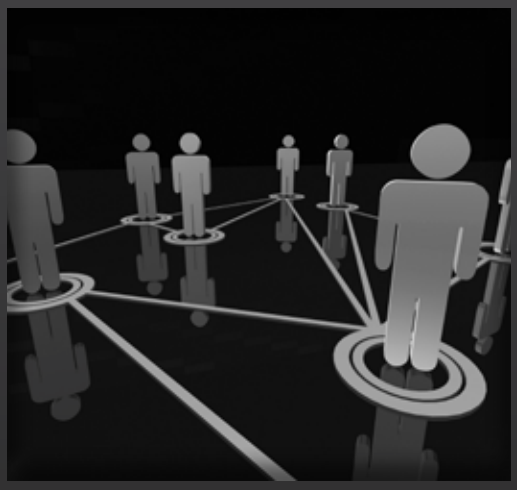
- *An organization's behavior changes slowly over time* - A successful software security program should be specified in small iterations that deliver tangible assurance gains while incrementally working toward long-term goals.
- *There is no single recipe that works for all organizations* - A software security framework must be flexible and allow organizations to tailor their choices based on their risk tolerance and the way in which they build and use software.
- *Guidance related to security activities must be prescriptive* - All the steps in building and assessing an assurance program should be simple, well-defined, and measurable. This model also provides roadmap templates for common types of organizations.

The foundation of the model is built upon the core business functions of software development with security practices tied to each (see diagram below). The building blocks of the model are the three maturity levels defined for each of the twelve security practices. These define a wide variety of activities in which an organization could engage to reduce security risks and increase software assurance. Additional details are included to measure successful activity performance, understand the associated assurance benefits, estimate personnel and other costs.

As an open project, SAMM content shall always remain vendor-neutral and freely available for all to use.



Contents



Understanding the Model

A view of the big picture

SAMM is built upon a collection of security practices that are tied back into the core business functions involved in software development. This section introduces those business functions and the corresponding security practices for each. After covering the high-level framework, the maturity levels for each security practice are also discussed briefly in order to paint a picture of how each can be iteratively improved over time.

Business Functions

***At the highest level, SAMM defines four critical* business functions.** Each business function is a category of activities related to the nuts-and-bolts of software development, or stated another way, any organization involved with software development must fulfill each of these business functions to some degree.

***For each business function, SAMM defines three* security practices.** Each security practice is an area of security-related activities that build assurance for the related business function. There are twelve security practices that are the independent silos for improvement that map to the four business functions of software development.

***For each security practice, SAMM defines three* maturity levels *as* objectives**. Each level within a security practice is characterized by a successively more sophisticated objective defined by specific activities, and more stringent success metrics than the previous level. Additionally, each security practice can be improved independently, though related activities can lead to optimizations.

Governance

Governance is centered on the processes and activities related to how an organization manages overall software development activities. More specifically, this includes concerns that impact cross-functional groups involved in development, as well as business processes that are established at the organization level.

Strategy & Metrics involves the overall strategic direction of the software assurance program and instrumentation of processes and activities to collect metrics about an organization's security posture.

Policy & Compliance involves setting up a security, compliance, and audit control framework throughout an organization to achieve increased assurance in software under contruction and in operation.

Education & Guidance involves increasing security knowledge amongst personnel in software development through training and guidance on security topics relevant to individual job functions.

...more on page 10

Construction

Construction concerns the processes and activities related to how an organization defines goals and creates software within development projects. In general, this will include product management, requirements gathering, high-level architecture specification, detailed design, and implementation.

Threat Assessment involves accurately identifying and characterizing potential attacks upon an organization's software in order to better understand the risks and facilitate risk management.

Security Requirements involves promoting the inclusion of security-related requirements during the software development process in order to specify correct functional-ity from inception.

Secure Architecture involves bolstering the design process with activities to promote secure-by-default designs and control over technologies and frameworks upon which software is built.

...more on page 12

Verification

Verification is focused on the processes and activities related to how an organization checks, and tests artifacts produced throughout software development. This typically includes quality assurance work such as testing, but it can also include other review and evaluation activities.

Design Review involves inspection of the artifacts created from the design process to ensure provision of adequate security mechanisms, and adherence to an organization's expectations for security.

Implementation Review involves assessment of an organization's source code to aid vulnerability discovery and related mitigation activities as well as establish a baseline for secure coding expectations.

Security Testing involves testing the organization's software in its runtime environment, in order to both discover vulnerabilities, and establish a minimum standard for software releases.

...more on page 14

Operations

Operations entails the processes and activities related to how an organization manages software releases that has been created. This can involve shipping products to end users, deploying products to internal or external hosts, and normal operations of software in the runtime environment.

Issue Management involves establishing consistent processes for managing internal and external vulnerability reports to limit exposure and gather data to enhance the security assurance program.

Environment Hardening involves implementing controls for the operating environment surrounding an organization's software to bolster the security posture of applications that have been deployed.

Operational Enablement involves identifying and capturing security-relevant information needed by an operator to properly configure, deploy, and run an organization's software.

...more on page 16

Maturity Levels

Each of the twelve security practices has three defined maturity levels and an implicit starting point at zero. The details for each level differs between the practices, but they generally represent:

0 Implicit starting point representing the activities in the practice being unfulfilled

1 Initial understanding and adhoc provision of security practice

2 Increase efficiency and/or effectiveness of the security practice

3 Comprehensive mastery of the security practice at scale

Notation

Throughout this document, the following terms will be reserved words that refer to the SAMM components defined in this section:

- ✦Business Function
- ✦Security Practice
- ✦Maturity Level or Objective

Assurance programs might not always consist of activities that neatly fall on a boundary between maturity levels, e.g. an organization that assesses to a Level 1 for a given practice might also have additional activities in place but not such that Level 2 is completed. Prior to v1.5, the organization's score should be annotated with a "**+**" symbol to indicate there's additional assurances in place beyond those indicated by the Level obtained. For example, an organization that is performing all Level 1 activities for operational enablement as well as one Level 2 or 3 activity would be assigned a "**1+**" score. Likewise, an organization performing all activities for a security practice, including some beyond the scope of SAMM, would be given a "**3+**" score.

The scoring model has changed in v1.5 to provide more granularity to the scoring in an assessment. Now an organization will get credit for different levels of work they have done within a practice rather than having the base number held at the highest completed maturity level. The scoring is now fractional to two decimal places for each practice and a single decimal for an answer. Questions have also been changed from Yes/No to four options that represent different levels of coverage or maturity. This change will assist practitioners completing SAMM assessments with the inevitable debate whether to mark an answer yes or no when it is honestly something in between.

The primary reason for the scoring change was to ensure organizations would receive full credit for their work in software security and to make it easier to show improvements in scoring when activities and programs grow and mature. The hope is this change will bring us closer to understanding what works in different scenarios for different organizations to benefit all.

The toolbox spreadsheet has been updated to reflect more context aware answers for each of the questions in the assessment. The formulas in the toolbox will also average the answers to calculate the score for each practice, a roll up average for each business function, and an overall score. The toolbox also has updated scorecard graphics that help represent the current score and can help show improvements to the program as the answers to the questions change. The worksheets later in this document are also updated to align with the new scoring model.

No = 0 **Few/Some = .2** **At Least Half = .5** **Many/Most = 1**