Software Assurance Maturity Model (SAMM) 1.0

Colin Watson
Watson Hall Ltd
colin.watson(at)owasp.org

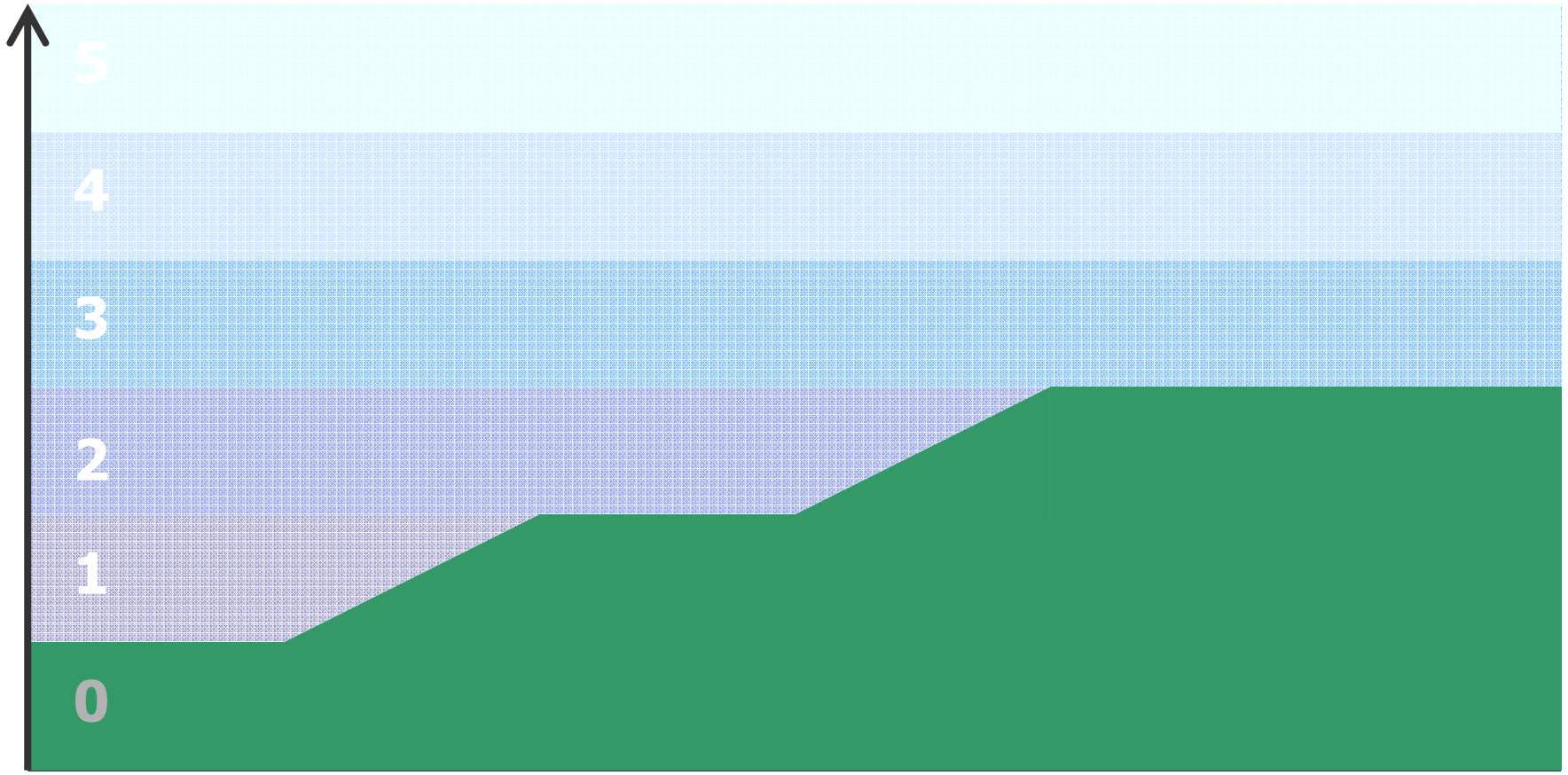
OWASP
Athens, Greece

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

Outline

- Terminology
- Structure
- Using SAMM
- Supporting materials
- Future developments
- Other models and standards

Generic Individual Practice Over Time



SAMM

- Software (Security) Assurance Maturity Model (S[S]AMM)
- Framework to formulate and implement a strategy for software security
- Tailored to an organisation's specific risks
- Vendor neutral
- Sequential, measurable goals
- Open and freely available
- OWASP project since January 2009

History/plan

- Author and project lead
  - Pravir Chandra, United States
- Comprehensive, Lightweight Application Security Process (CLASP)
  - Ongoing development
  - Current version 1.2, 2006
- Open SAMM 0.8 beta
  - August 2008
- Open SAMM 1.0
  - March 2009
- Open SAMM 2.0
  - ? 2011

Aims

- 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

Four Critical Business Functions

Governance

Software development management activities and organisation-wide business processes

Construction

Goal definition and software creation processes

Verification

Checking, evaluation and testing of software development artifacts

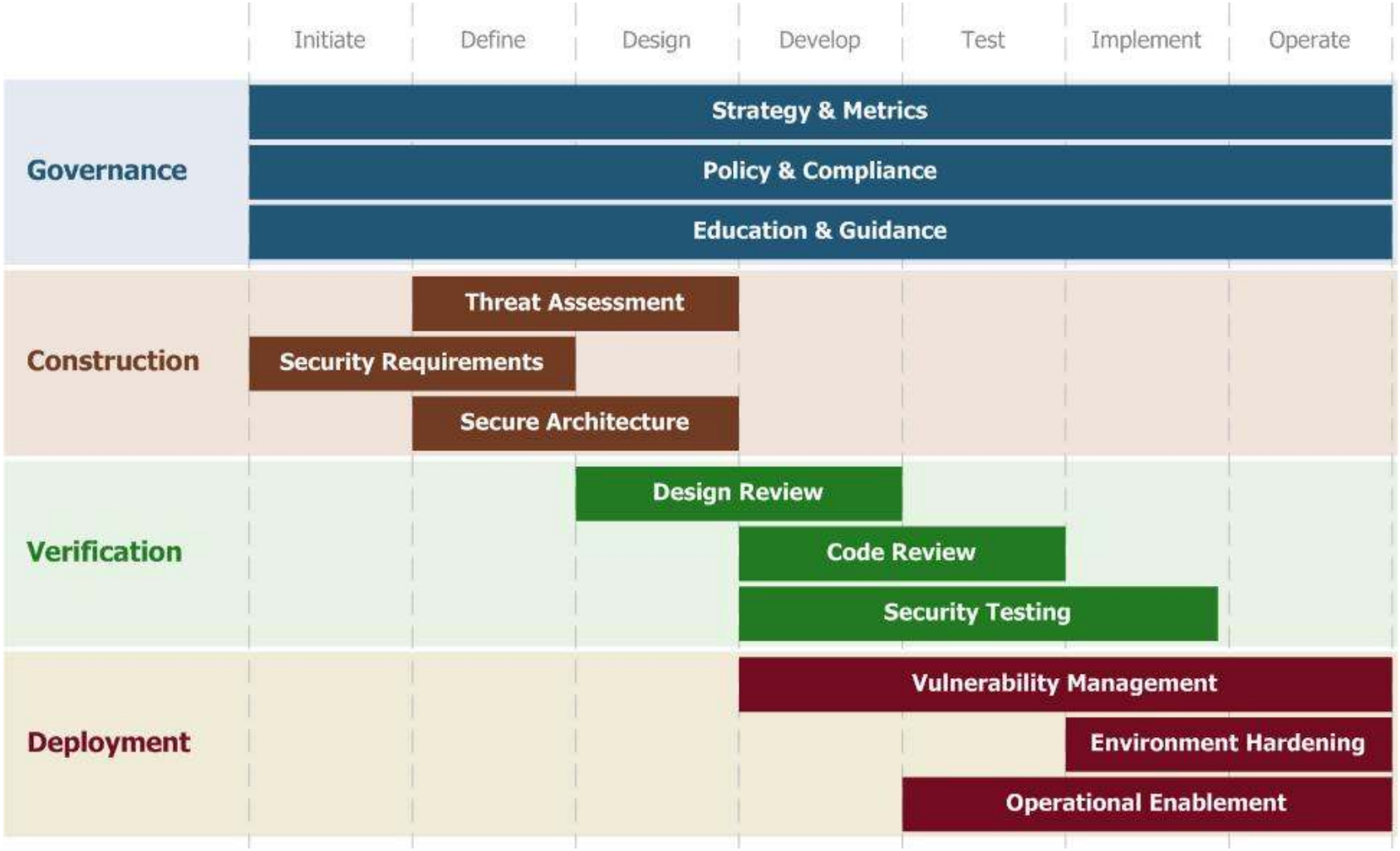
Deployment

Software release management and normal operational management

Structure

- Four business practices, each with:
  - Three security practices, each with:
    - One objective
    - Two activities
    - Assessment method
    - Expected results
- Software security is assessed against every security practice, giving each a maturity level (score) of between 0 and 3:
  1, 0, 0+, 1, 2, 3, 0+ 2, 0+, 1, 1+, 0

SAMM and an SDLC



Each with Three Security Practices

Governance	Construction	Verification	Deployment
			
Strategy & Metrics	Threat Assessment	Design Review	Vulnerability Management
Policy & Compliance	Security Requirements	Code Review	Environment Hardening
Education & Guidance	Secure Architecture	Security Testing	Operational Enablement

Each Practice has 3 levels (objectives) 1/2

Governance

Security & Metrics

Policy & Compliance

Education & Guidance

	PC1	PC2	PC3
Objectives	Understand relevant governance and compliance drivers to the organization	Establish security and compliance baseline and understand per-project risks	Require compliance and measure projects against organization-wide policies and standards
Activitves	Identify and monitor external compliance drivers Build and maintain compliance guidelines	Build policies and standards for security compliance Establish project audit practice	Create compliance gates for projects Adopt solution for audit data collection

Each Practice has 3 levels (objectives) 2/2

Verification

- Design Review
- Code Review
- Security Testing

	DR1	DR2	DR3
Objectives	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
Activitves	Identify software attack surface Analyze design against known security requirements	Inspect for complete provision of security mechanisms Deploy design review service for project teams	Develop data-flow diagrams for sensitive resources Establish release gates for design review

DR3 Detail 1/4

Design Review

Require assessments and validate artifacts to develop detailed understanding of protection mechanisms

Activities

A. Develop data-flow diagrams for sensitive resources

Based on the business function of the software project, conduct analysis to identify details on system behavior around high-risk functionality. Typically, high-risk functionality will correlate to features implementing creation, access, update, and deletion of sensitive data. Beyond data, high-risk functionality also includes project-specific business logic that is critical in nature, either from a denial-of-service or compromise perspective.

For each identified data source or business function, select and use a standardized notation to capture relevant software modules, data sources, actors, and messages that flow amongst them. It is often helpful to start with a high-level design diagram and iteratively flesh out relevant detail while removing elements that do not correspond to the sensitive resource.

With data-flow diagrams created for a project, conduct analysis over them to determine internal choke-points in the design. Generally, these will be individual software modules that handle data with differing sensitivity levels or those that gate access to several business functions of various levels of business criticality.

B. Establish release gates for design review

Having established a consistent design review program, the next step of enforcement is to set a particular point in the software development life-cycle where a project cannot pass until an design review is conducted and findings are reviewed and accepted. In order to accomplish this, a baseline level of expectations should be set, e.g. no projects with any high-severity findings will be allowed to pass and all other findings must be accepted by the business owner.

Generally, design reviews should occur toward the end of the design phase to aide early detection of security issues, but it must occur before releases can be made from the project team.

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 each to be reviewed to illuminate any hidden vulnerabilities in the existing systems. Exceptions for should be limited to no more than 20% of all projects.

Results

- Granular view of weak points in a system design to encourage better compartmentalization
- Organization-level awareness of project standing against baseline security expectations for architecture
- Comparisons between projects for efficiency and progress toward mitigating known flaws

Add'l Success Metrics

- >80% of projects with updated data-flow diagrams in past 6 months
- >75% of projects passing design review audit in past 6 months

Add'l Costs

- Ongoing project overhead from maintenance of data-flow diagrams
- Organization overhead from project delays caused by failed design review audits

Add'l Personnel

- Developers (2 days/yr)
- Architects (1 day/yr)
- Managers (1-2 days/yr)
- Business Owners (1-2 days/yr)
- Security Auditors (2-3 days/yr)

Related Levels

- Secure Architecture - 3
- Code Review - 3

DR3 Detail 2/4

Activities

A. Develop data-flow diagrams for sensitive resources

Based on the business function of the software project, conduct analysis to identify details on system behavior around high-risk functionality. Typically, high-risk functionality will correlate to features implementing creation, access, update, and deletion of sensitive data. Beyond data, high-risk functionality also includes project-specific business logic that is critical in nature, either from a denial-of-service or compromise perspective.

For each identified data source or business function, select and use a standardized notation to capture relevant software modules, data sources, actors, and messages that flow amongst them. It is often helpful to start with a high-level design diagram and iteratively flesh out relevant detail while removing elements that do not correspond to the sensitive resource.

With data-flow diagrams created for a project, conduct analysis over them to determine internal choke-points in the design. Generally, these will be individual software modules that handle data with differing sensitivity levels or those that gate access to several business functions of various levels of business criticality.

DR3 Detail 3/4

B. Establish release gates for design review

Having established a consistent design review program, the next step of enforcement is to set a particular point in the software development life-cycle where a project cannot pass until an design review is conducted and findings are reviewed and accepted. In order to accomplish this, a baseline level of expectations should be set, e.g. no projects with any high-severity findings will be allowed to pass and all other findings must be accepted by the business owner.

Generally, design reviews should occur toward the end of the design phase to aide early detection of security issues, but it must occur before releases can be made from the project team.

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 each to be reviewed to illuminate any hidden vulnerabilities in the existing systems. Exceptions for should be limited to no more than 20% of all projects.

DR3 Detail 4/4

Results

- Granular view of weak points in a system design to encourage better compartmentalization
- Organization-level awareness of project standing against baseline security expectations for architecture
- Comparisons between projects for efficiency and progress toward mitigating known flaws

Add'l Success Metrics

- >80% of projects with updated data-flow diagrams in past 6 months
- >75% of projects passing design review audit in past 6 months

Add'l Costs

- Ongoing project overhead from maintenance of data-flow diagrams
- Organization overhead from project delays caused by failed design review audits

Add'l Personnel

- Developers (2 days/yr)
- Architects (1 day/yr)
- Managers (1-2 days/yr)
- Business Owners (1-2 days/yr)
- Security Auditors (2-3 days/yr)

Related Levels

- Secure Architecture - 3
- Code Review - 3

SAMM procedure

- Conduct an assessment
  - Lightweight
  - Detailed
- Create a score card
- Build an assurance programme
  - Metrics
  - Road map
- Implementation and re-assessment

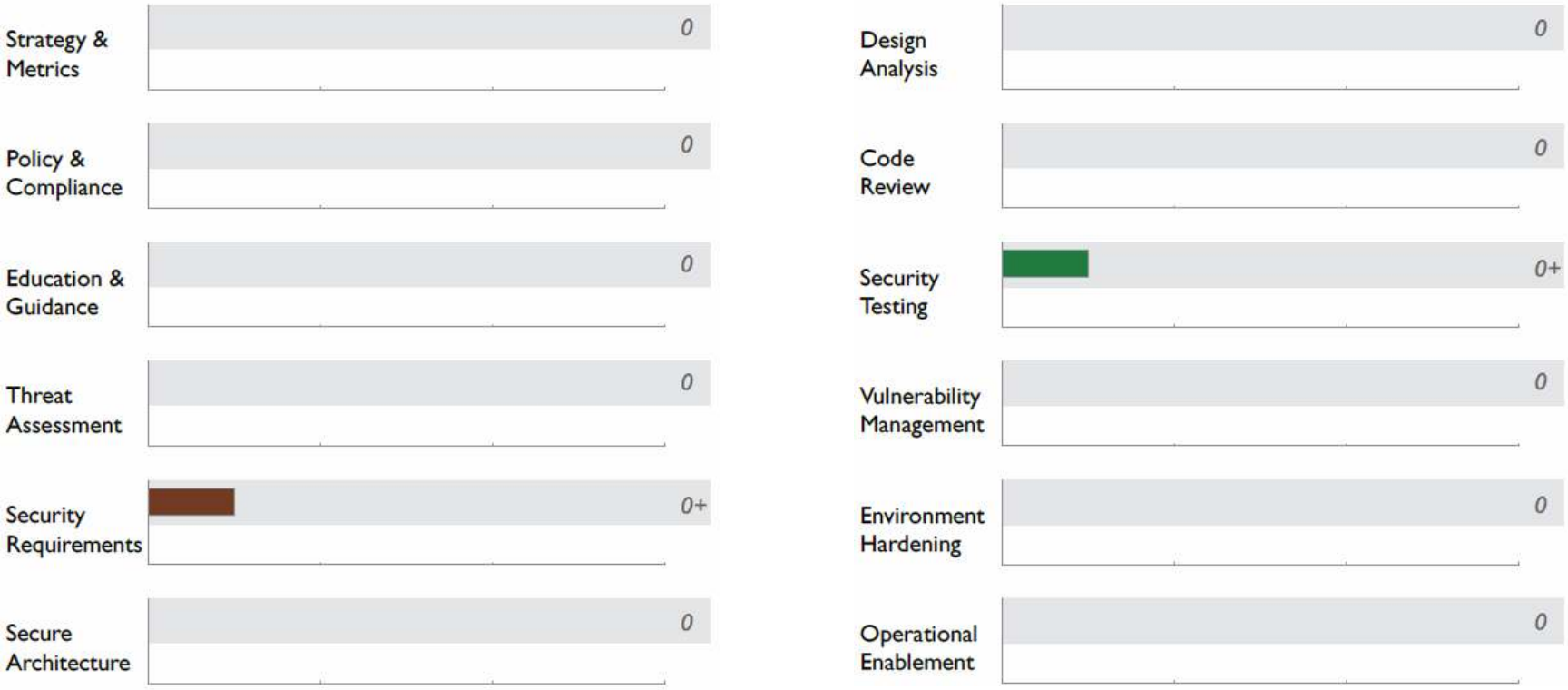
Assessment

Verification

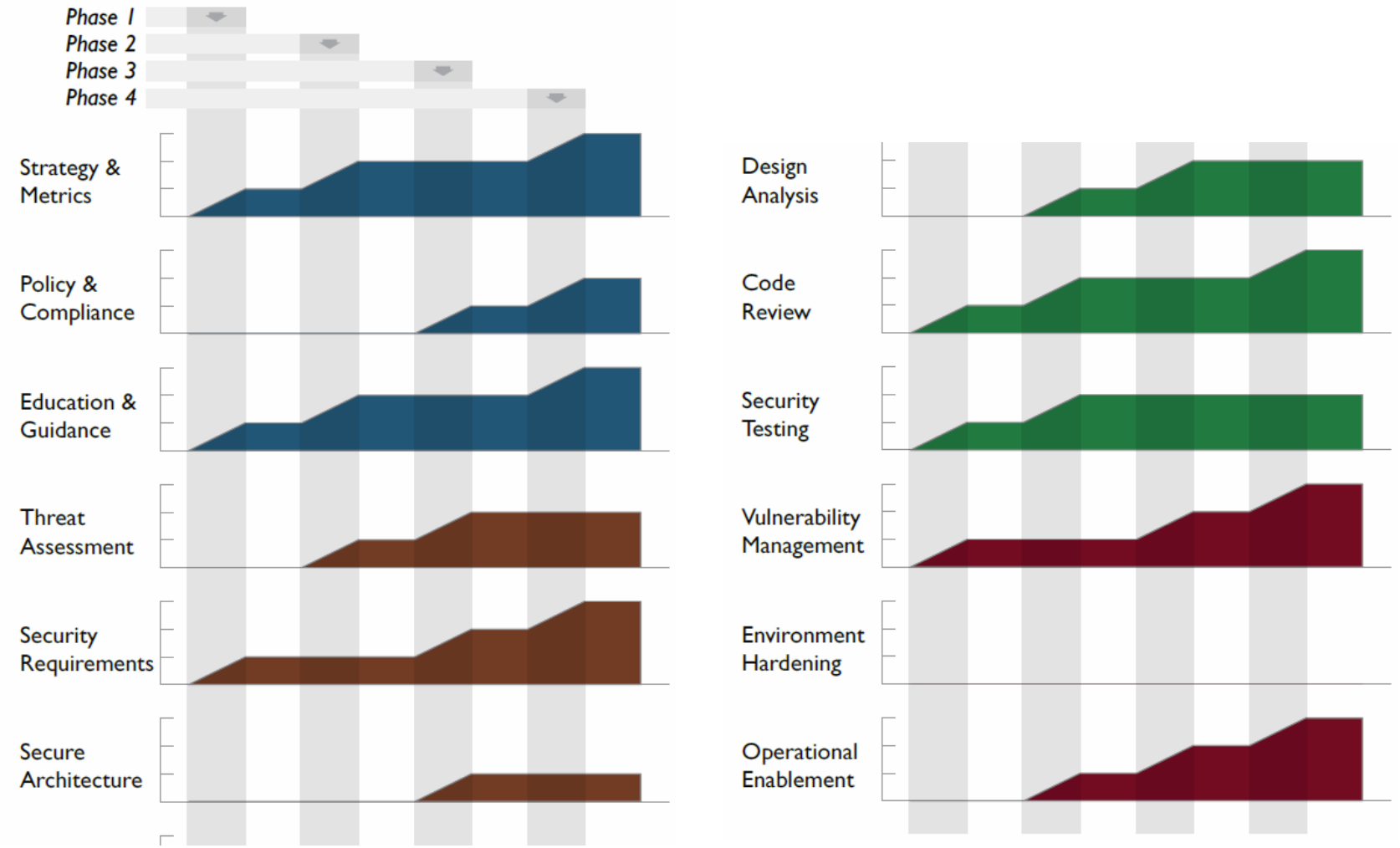
Assessment worksheet

Design Review	Yes/No
✦Do project teams document the attack perimeter of software designs?	
✦Do project teams check software designs against known security risks?	DR 1
✦Do most project teams specifically analyze design elements for security mechanisms?	
✦Are most project stakeholders aware of how to obtain a formal design review?	DR 2
✦Does the design review process incorporate detailed data-level analysis?	
✦Does routine project audit require a baseline for design review results?	DR 3

Scorecards


Strategy & Metrics
0
Policy & Compliance
0
Education & Guidance
0
Threat Assessment
0
Security Requirements
0+
Secure Architecture
0
Design Analysis
0
Code Review
0
Security Testing
0+
Vulnerability Management
0
Environment Hardening
0
Operational Enablement
0

Roadmaps


Phase 1
Phase 2
Phase 3
Phase 4
Strategy & Metrics
Policy & Compliance
Education & Guidance
Threat Assessment
Security Requirements
Secure Architecture
Design Analysis
Code Review
Security Testing
Vulnerability Management
Environment Hardening
Operational Enablement

SAMM in Use

- OpenSAMM Shows a Way, Building Real Software Blog, Jim Bird, 17 April 2009
- Feedback from client engagements using SAMM by Eoin Keary, Rahim Jina and Aidan Lynch (Ernst & Young), 10 July 2009
- Applicability
  - business maturity
  - organisation scale

Supporting Resources

- Wiki, PDF download, eBook and Lulu book in monochrome and colour
- Pravir Chandra's presentation at AppSec EU09
- Zate Berg's presentation at OWASP Tampa
- Matt Bartoldus' presentation at OWASP London
- Templates for assessments and managing software security strategies
- Roadmap charts

Success Measures

- Improve software security
- Promoted beyond the security community
- Metrics to measure improvements actually achieved (real projects)
- Reduce complexity
- Implemented in a wide range of organisations
- Supporting materials, tools, templates, papers and integration with other business process models and standards

Future Path

- Refinement based on experience and feedback
- Interview template assertions
- Additional case studies
- Use SAMM to assess OWASP project(s)
- Mappings to other resources (CLASP, BSIMM, NIST SP800-53, CobiT) and OWASP projects
- Translations (Spanish, French, Chinese, …)
- Success metrics as business results

Success Metrics as Business Results

- In SAMM 1.0, most metrics are activity based
  - ⅞ activity success metrics
    e.g. “>80% of staff briefed on assurance program roadmap in past 3 months” and “>50% of projects with updated change management procedures in past 6 months”
  - ⅛ business success metrics
    e.g. “>50% of all security incidents identified a priori by threat models in past 12 months” and perhaps “>75% of projects passing infrastructure audits in past 6 months”
- Greater emphasis on business success?

Further reading 1/2

- Software Assurance Maturity Model (SAMM)
  http://www.opensamm.org/
- OWASP SAMM Project
  http://www.owasp.org/index.php/Category:OWASP_Software_Assurance_Maturity_Model_Project
- OWASP CLASP Project
  http://www.owasp.org/index.php/Category:OWASP_CLASP_Project
- SAMM presentation at AppSec EU09 by Pravir Chandra
  http://www.owasp.org/images/4/49/AppSecEU09_OpenSAMM-1.0.ppt
- SAMM presentation at OWASP Tampa by Zate Berg
  http://lists.owasp.org/pipermail/samm/attachments/20090602/6d0d864c/attachment-0001.ppt
- SAMM presentation at OWASP London by Matt Bartoldus
  http://www.owasp.org/images/d/df/OpenSAMM.pdf
- Software Security Assurance, State-of-the-Art Report, 31 July 2007, Information Assurance Technology Analysis Center (IATAC) and Data and Analysis Center for Software (DACS)
  http://iac.dtic.mil/iatac/download/security.pdf

Further reading 2/2

- OpenSAMM Shows a Way, Jim Bird, 17 April 2009
  http://swreflections.blogspot.com/2009/04/opensamm-shows-way.html
- Team Software Process (TSP)
  http://www.sei.cmu.edu/tsp/
- Common Criteria (CC)
  http://www.commoncriteriaportal.org/thecc.html
- CESG
  http://www.cesg.gov.uk
- Build Security In (BSI)
  https://buildsecurityin.us-cert.gov
- Software Assurance Metrics And Tool Evaluation (SAMATE)
  http://samate.nist.gov/
- Software Assurance Forum for Excellence in Code (SAFECode)
  http://www.safecode.org/
- Trustworthy Computing Security Development Lifecycle, Microsoft
  http://msdn.microsoft.com/en-us/library/ms995349.aspx
- Correctness by Construction (CbyC)
  https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/sdlc/613-BSI.html
- Building Security In Maturity Model, Cigital
  http://www.bsi-mm.com/

Additional SAMM resources

- OWASP SAMM Project Mailing List
  https://lists.owasp.org/pipermail/samm/
- Open SAMM Blog
  http://www.opensamm.org/news/
- Tools
- http://www.opensamm.org/download/
  - OpenSAMM-BSIMM Mapping
    OWASP Summit 2011
  - Assessment Interview Template
    Nick Coblentz
  - Roadmap Chart Template
    Colin Watson
  - Assessment Worksheet
    Christian Frichot
  - Project Plan Template
    Jim Weiler
  - Vulnerability Manager
    Denim Group

End

- Colin Watson
  - Member, OWASP Global Industry Committee
    https://www.owasp.org/index.php/Global_Industry_Committee
  - Technical Director, Watson Hall Ltd
    https://www.watsonhall.com
- colin.watson(at)owasp.org