

SDLC Maturity Models

SecAppDev 2017

Bart De Win

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

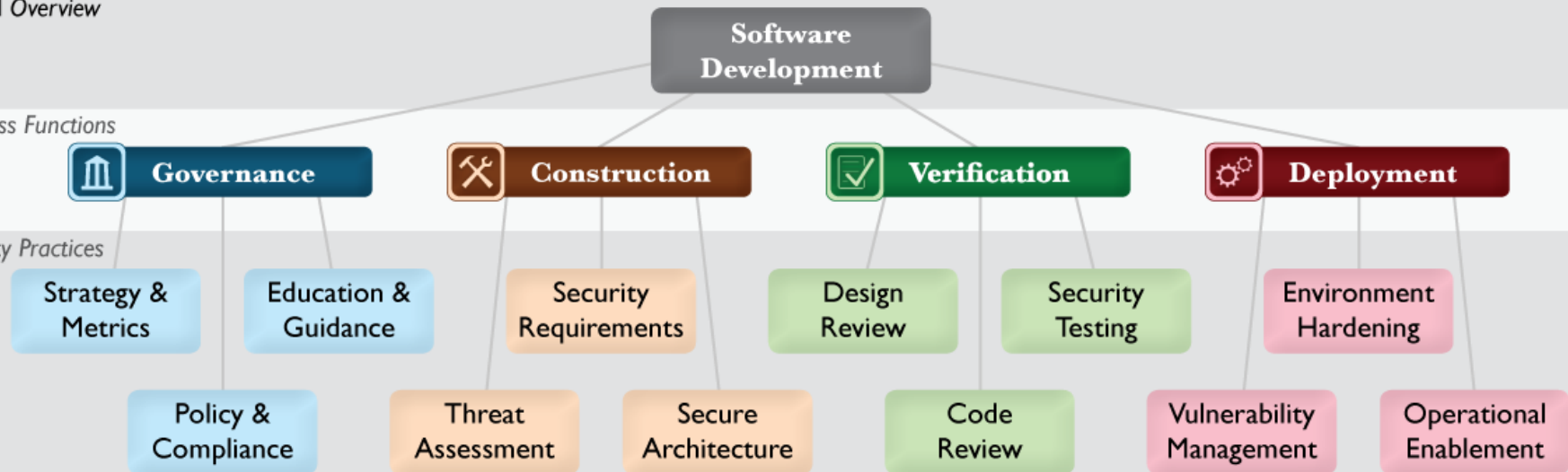


Core Structure

SAMM Overview

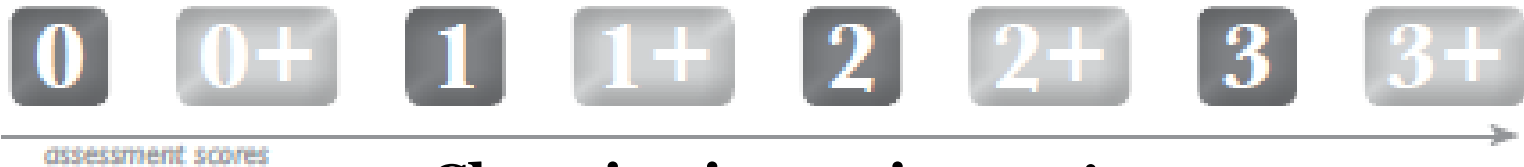
Business Functions

Security Practices






Notion of Maturity

Level	Interpretation
0	Implicit starting point representing the activities in the practice being unfulfilled
1	Initial understanding and ad-hoc provision of the security practice
2	Increase efficiency and/of effectiveness of the security practice
3	Comprehensive mastery of the security practice at scale



Changing in version 1.5 !

An example

Code Review ...more on page 62			
	 CR 1	 CR 2	 CR 3
OBJECTIVE	Opportunistically find basic code-level vulnerabilities and other high-risk security issues	Make code review during development more accurate and efficient through automation	Mandate comprehensive code review process to discover language-level and application-specific risks
ACTIVITIES	<ul style="list-style-type: none">A. Create review checklists from known security requirementsB. Perform point-review of high-risk code	<ul style="list-style-type: none">A. Utilize automated code analysis toolsB. Integrate code analysis into development process	<ul style="list-style-type: none">A. Customize code analysis for application-specific concernsB. Establish release gates for code review

OpenSAMM also defines

Security Testing



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

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 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

- ✦ 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

ADD'L SUCCESS METRICS

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

ADD'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




ADD'L PERSONNEL

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

RELATED LEVELS

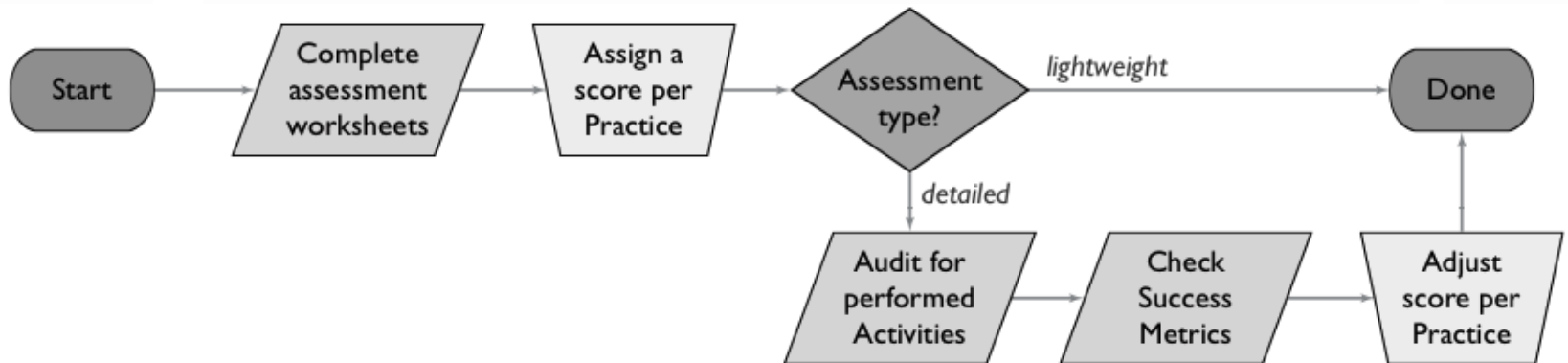
- ✦ Policy & Compliance - 2
- ✦ Secure Architecture - 3

Conducting assessments

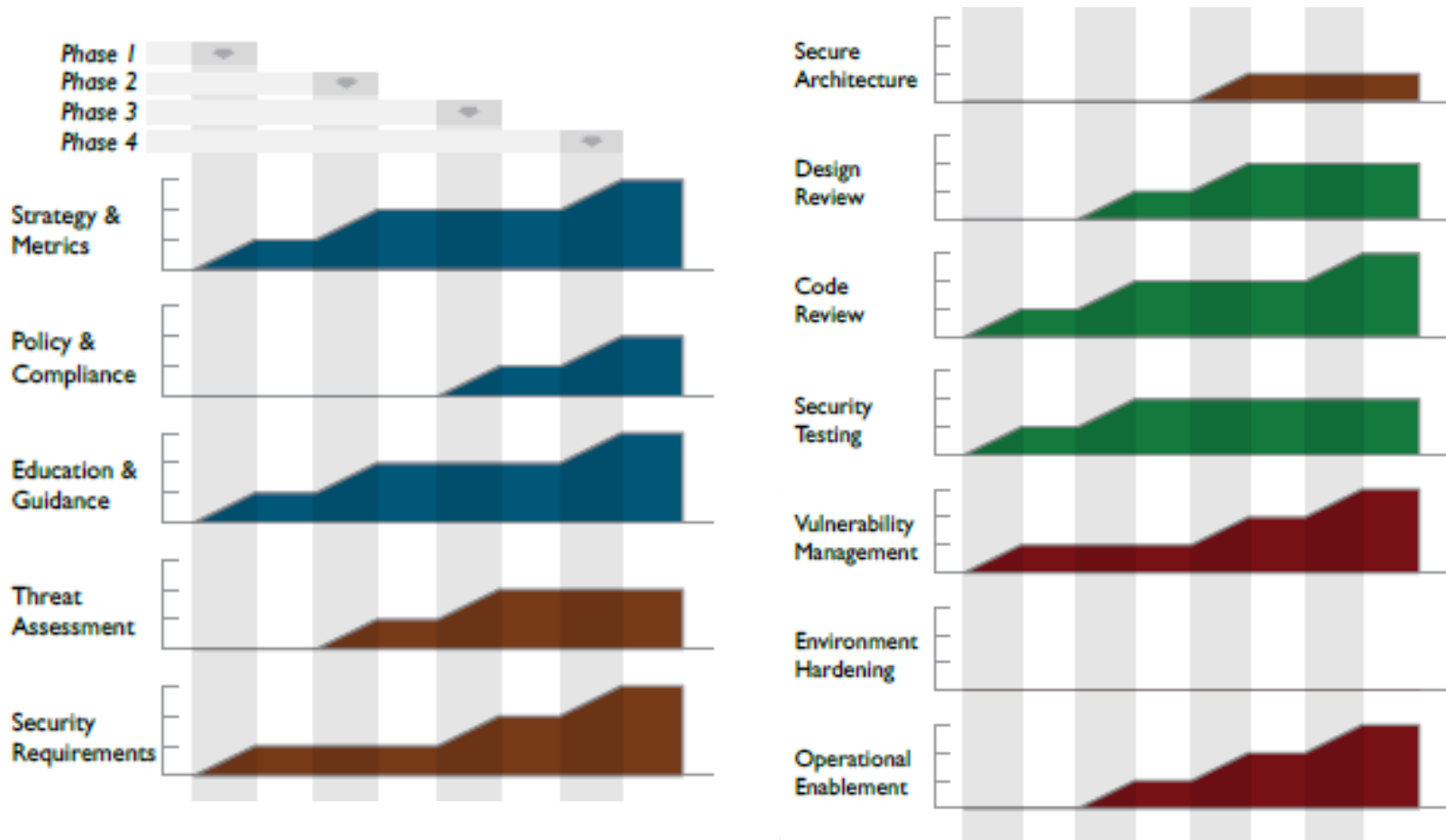
Secure Architecture	Yes/No	
✦ Are project teams provided with a list of recommended third-party components?		
✦ Are most project teams aware of secure design principles and applying them?		 SA 1
✦ Do you advertise shared security services with guidance for project teams?		
✦ Are project teams provided with prescriptive design patterns based on their application architecture?		 SA 2
✦ Are project teams building software from centrally controlled platforms and frameworks?		
✦ Are project teams being audited for usage of secure architecture components?		 SA 3

Assessment process

- Supports both lightweight and detailed assessments

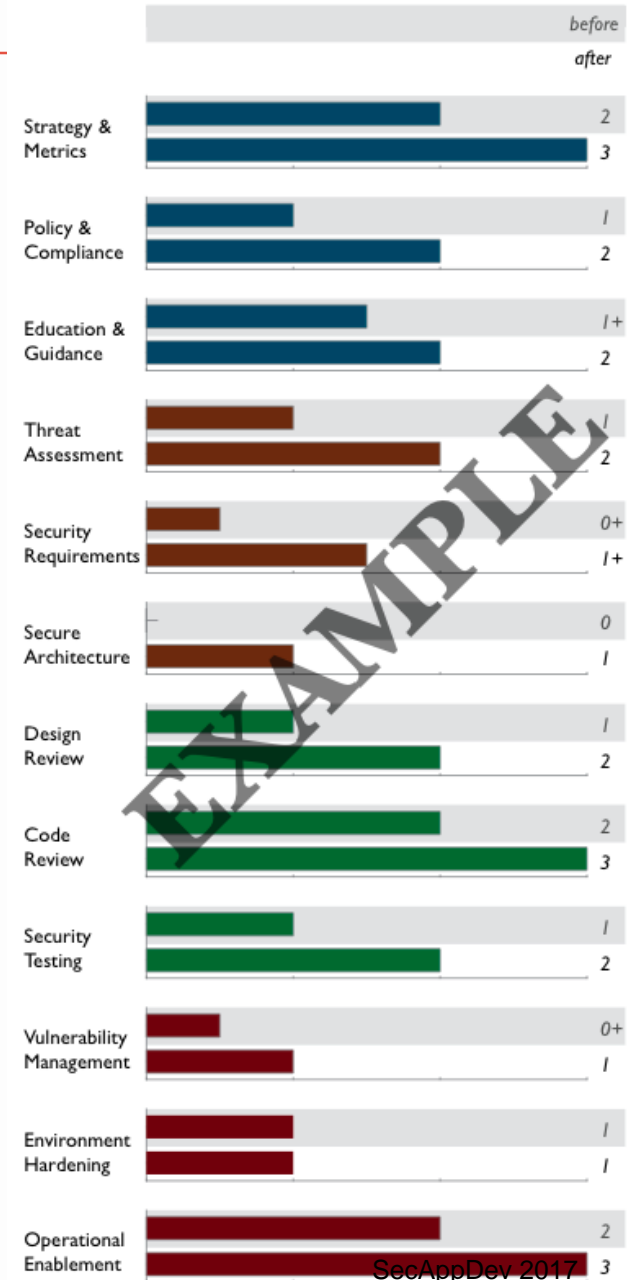


Roadmap templates per company type (ISV)



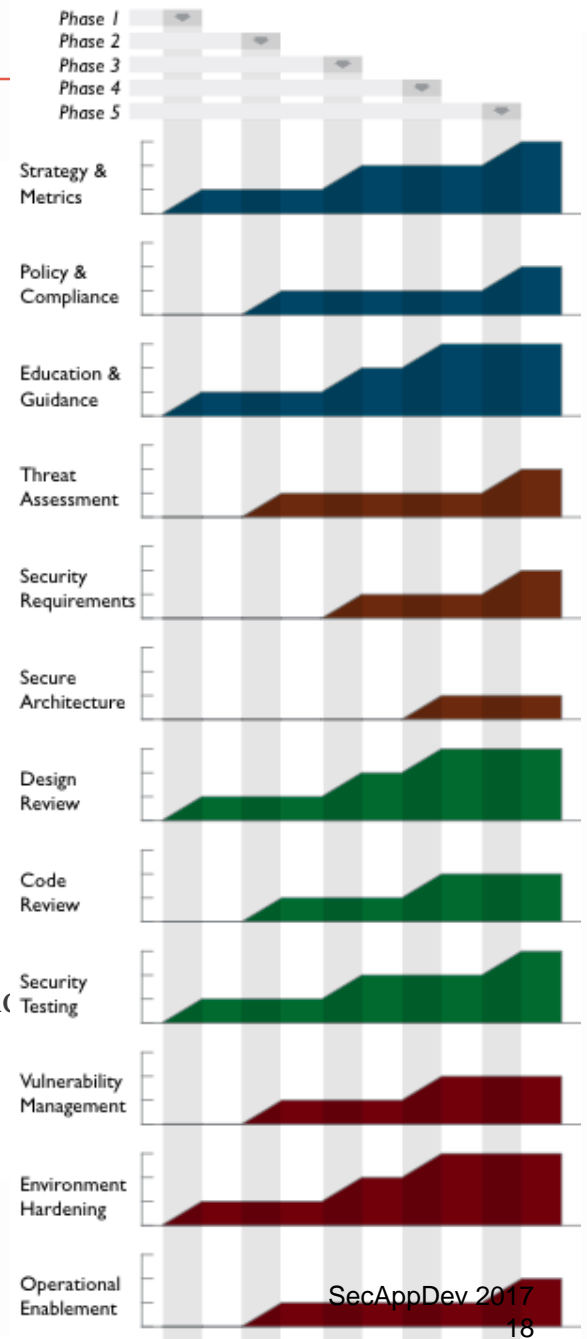
Creating Scorecards

- 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



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-inc
 - Optimal creation of an assurance program is different for each



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

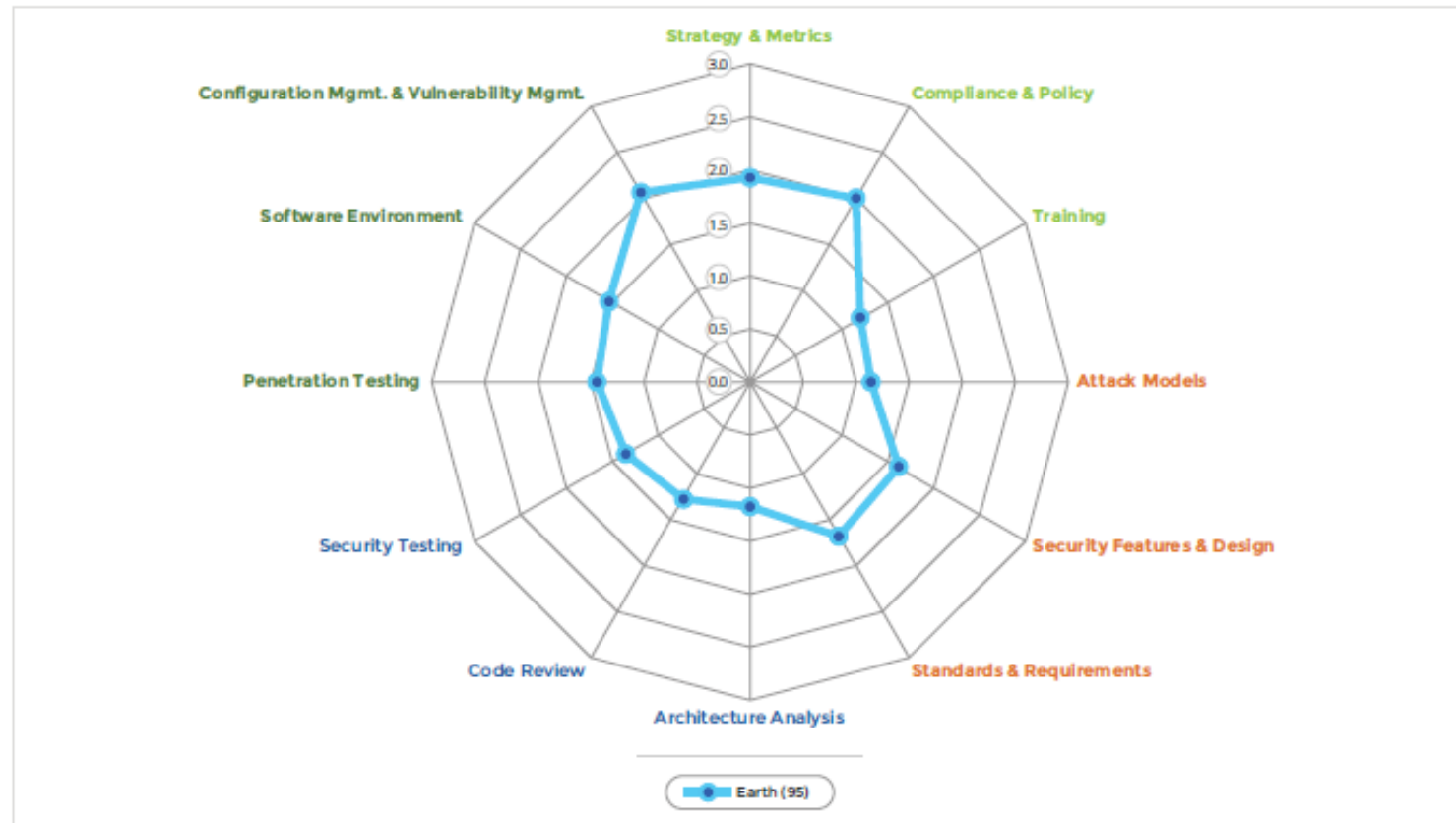
Mappings to security standards

- ISO/IEC 27034, PCI, ...

**OpenSAMM Assessment
Toolbox**

BSIMM7 statistics: summary

EARTH SPIDER CHART



BSIMM7 statistics per activity

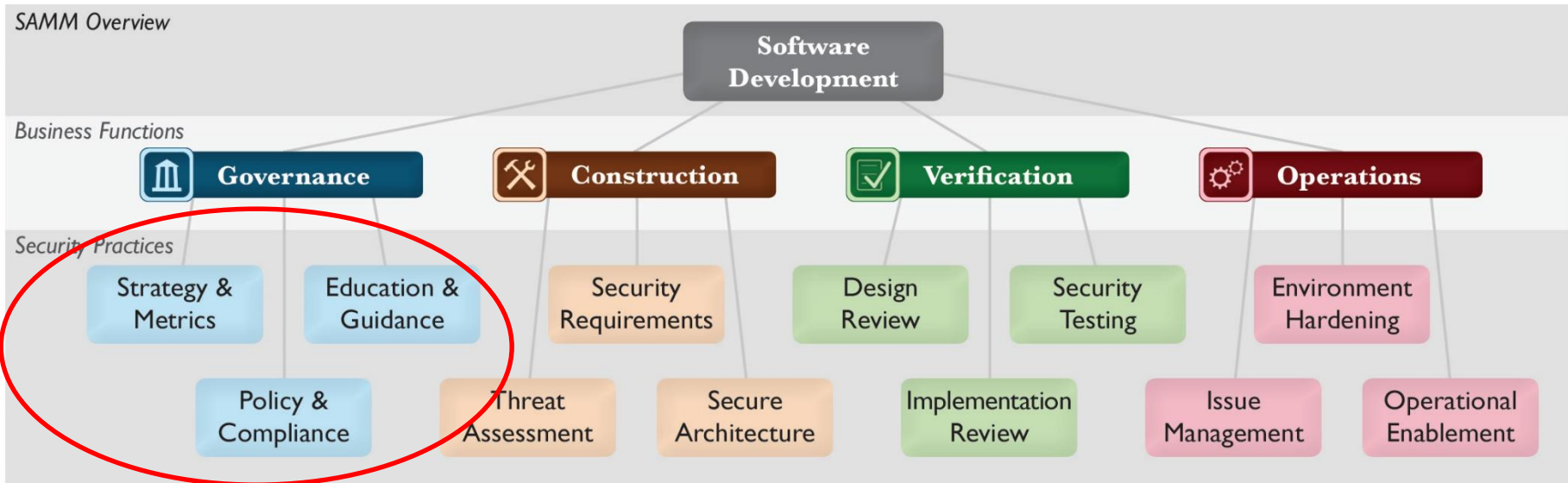
BSIMM7 SCORECARD

GOVERNANCE		INTELLIGENCE		SSDL TOUCHPOINTS		DEPLOYMENT	
ACTIVITY	OBSERVED	ACTIVITY	OBSERVED	ACTIVITY	OBSERVED	ACTIVITY	OBSERVED
[SM1.1]	47	[AM1.2]	63	[AA1.1]	81	[PT1.1]	82
[SM1.2]	48	[AM1.3]	34	[AA1.2]	29	[PT1.2]	58
[SM1.3]	46	[AM1.5]	48	[AA1.3]	23	[PT1.3]	54
[SM1.4]	81	[AM2.1]	8	[AA1.4]	47	[PT2.2]	21
[SM2.1]	41	[AM2.2]	8	[AA2.1]	15	[PT2.3]	16
[SM2.2]	35	[AM2.5]	13	[AA2.2]	12	[PT3.1]	10
[SM2.3]	33	[AM2.6]	9	[AA2.3]	5	[PT3.2]	6
[SM2.5]	19	[AM2.7]	9	[AA3.1]	4		
[SM2.6]	33	[AM3.1]	4	[AA3.2]	0		
[SM3.1]	14	[AM3.2]	2				
[SM3.2]	9						
[CP1.1]	56	[SFD1.1]	74	[CR1.2]	58	[SE1.1]	46
[CP1.2]	84	[SFD1.2]	65	[CR1.4]	63	[SE1.2]	78
[CP1.3]	50	[SFD2.1]	27	[CR1.5]	28	[SE2.2]	27
[CP2.1]	24	[SFD2.2]	40	[CR1.6]	34	[SE2.4]	24
[CP2.2]	31	[SFD3.1]	6	[CR2.5]	22	[SE3.2]	12
[CP2.3]	34	[SFD3.2]	10	[CR2.6]	15	[SE3.3]	3
[CP2.4]	36	[SFD3.3]	1	[CR2.7]	19	[SE3.4]	0
[CP2.5]	38			[CR3.2]	3		
[CP3.1]	19			[CR3.3]	2		
[CP3.2]	13			[CR3.4]	3		
[CP3.3]	5			[CR3.5]	5		
[T1.1]	69	[SR1.1]	60	[ST1.1]	78	[CMVM1.1]	82
[T1.5]	27	[SR1.2]	66	[ST1.3]	72	[CMVM1.2]	84
[T1.6]	17	[SR1.3]	64	[ST2.1]	22	[CMVM2.1]	69
[T1.7]	37	[SR2.2]	28	[ST2.4]	10	[CMVM2.2]	74
[T2.5]	13	[SR2.3]	22	[ST2.5]	7	[CMVM2.3]	41
[T2.6]	14	[SR2.4]	21	[ST2.6]	9	[CMVM3.1]	3
[T2.7]	5	[SR2.5]	22	[ST3.3]	4	[CMVM3.2]	5
[T3.1]	3	[SR2.6]	17	[ST3.4]	2	[CMVM3.3]	8
[T3.2]	5	[SR3.1]	8	[ST3.5]	4	[CMVM3.4]	6
[T3.3]	2	[SR3.2]	11				
[T3.4]	7						
[T3.5]	2						

Agenda

1. Motivation
2. SAMMM At A Glance
- 3. SAMMM Practices**
4. Conclusion

SAMM Security Practices - Governance



Strategy & Metrics

Goal is to establish a software assurance framework within an organisation

- Foundation for all other OpenSAMM practices

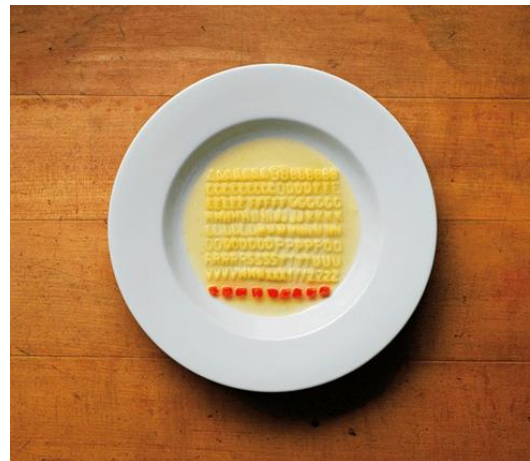
Characteristics:

- Measurable
- Aligned with business risk




Driver for continuous improvement and financial guidance



VS.



Strategy & Metrics

	Strategy & Metrics ...more on page 34		
	 SM 1	 SM 2	 SM 3
OBJECTIVE	Establish unified strategic roadmap for software security within the organization	Measure relative value of data and software assets and choose risk tolerance	Align security expenditure with relevant business indicators and asset value
ACTIVITIES	<ul style="list-style-type: none">A. Estimate overall business risk profileB. Build and maintain assurance program roadmap	<ul style="list-style-type: none">A. Classify data and applications based on business riskB. Establish and measure per-classification security goals	<ul style="list-style-type: none">A. Conduct periodic industry-wide cost comparisonsB. Collect metrics for historic security spend

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

Privacy Policy



Policy & Compliance

	Policy & Compliance ...more on page 38		
	 PC 1	 PC 2	 PC 3
OBJECTIVE	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
ACTIVITIES	A. Identify and monitor external compliance drivers B. Build and maintain compliance guidelines	A. Build policies and standards for security and compliance B. Establish project audit practice	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

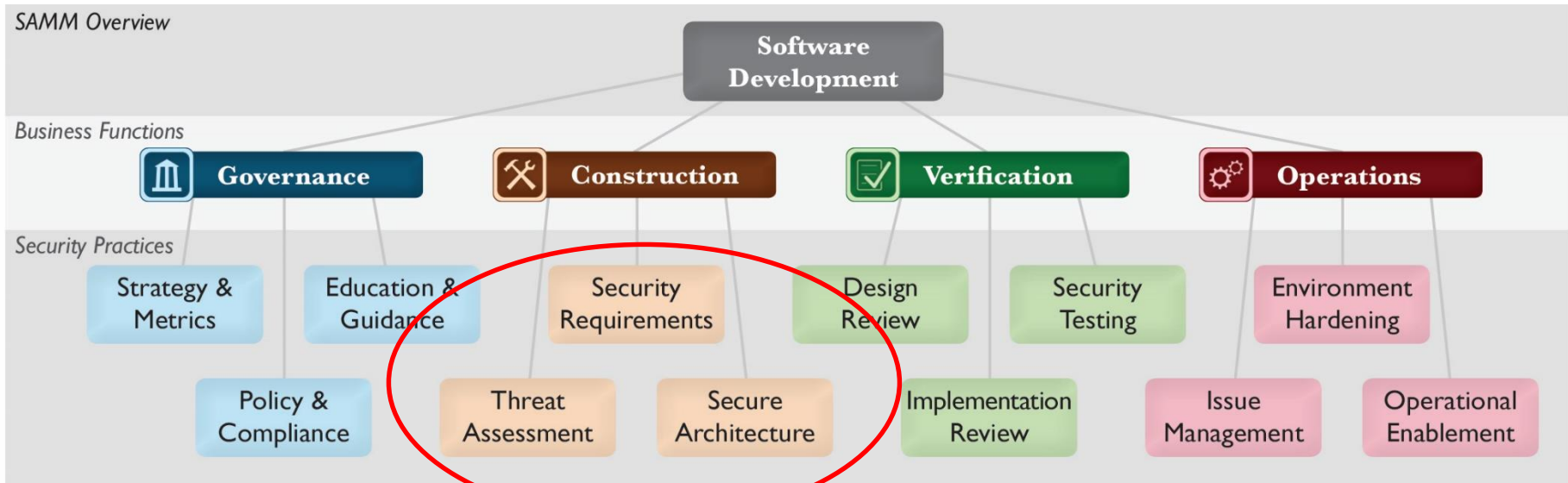
Teach a fisherman to fish

3. Technical guidelines form the basis for several other practices

Education & Guidance

	Education & Guidance ...more on page 42		
	 EG 1	 EG 2	 EG 3
OBJECTIVE	Offer development staff access to resources around the topics of secure programming and deployment	Educate all personnel in the software life-cycle with role-specific guidance on secure development	Mandate comprehensive security training and certify personnel for baseline knowledge
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

Threat Assessment			
...more on page 46			
	 TA 1	 TA 2	 TA 3
OBJECTIVE	Identify and understand high-level threats to the organization and individual projects	Increase accuracy of threat assessment and improve granularity of per-project understanding	Concretely tie compensating controls to each threat against internal and third-party software
ACTIVITIES	A. Build and maintain application-specific threat models B. Develop attacker profile from software architecture	A. Build and maintain abuse-case models per project B. Adopt a weighting system for measurement of threats	A. Explicitly evaluate risk from third-party components B. Elaborate threat models with compensating controls

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

	Security Requirements ...more on page 50		
	 SR 1	 SR 2	 SR 3
OBJECTIVE	Consider security explicitly during the software requirements process	Increase granularity of security requirements derived from business logic and known risks	Mandate security requirements process for all software projects and third-party dependencies
ACTIVITIES	<p>A. Derive security requirements from business functionality</p> <p>B. Evaluate security and compliance guidance for requirements</p>	<p>A. Build an access control matrix for resources and capabilities</p> <p>B. Specify security requirements based on known risks</p>	<p>A. Build security requirements into supplier agreements</p> <p>B. Expand audit program for security requirements</p>

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

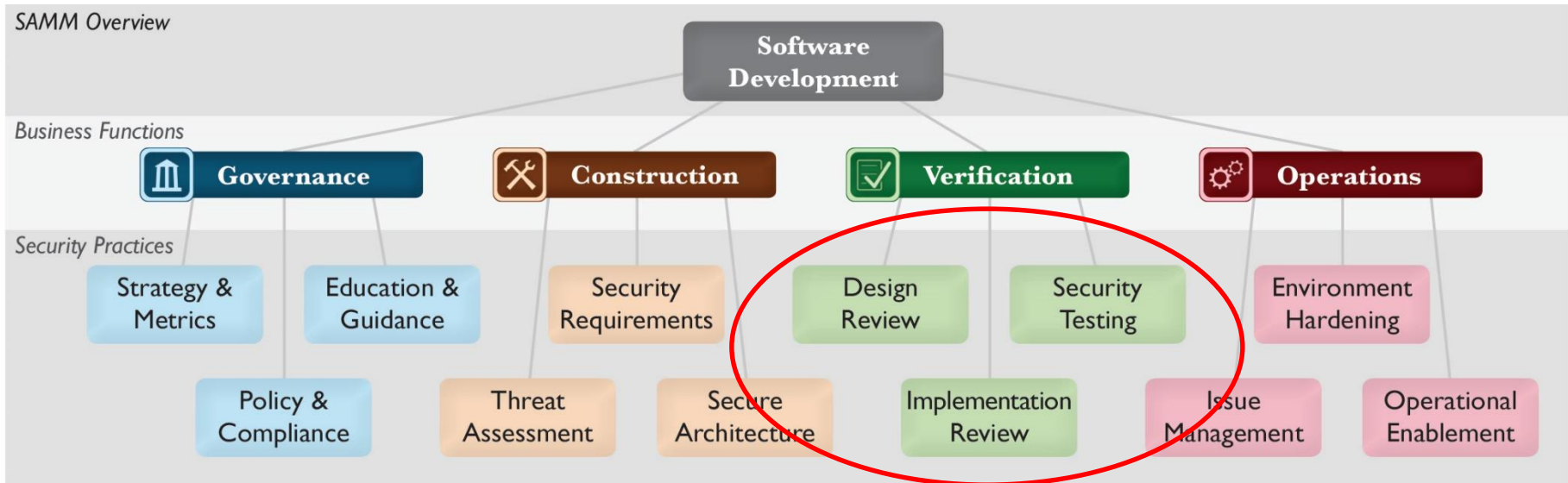
- Don't roll your own crypto
- Use company standards and best practices



Secure Architecture

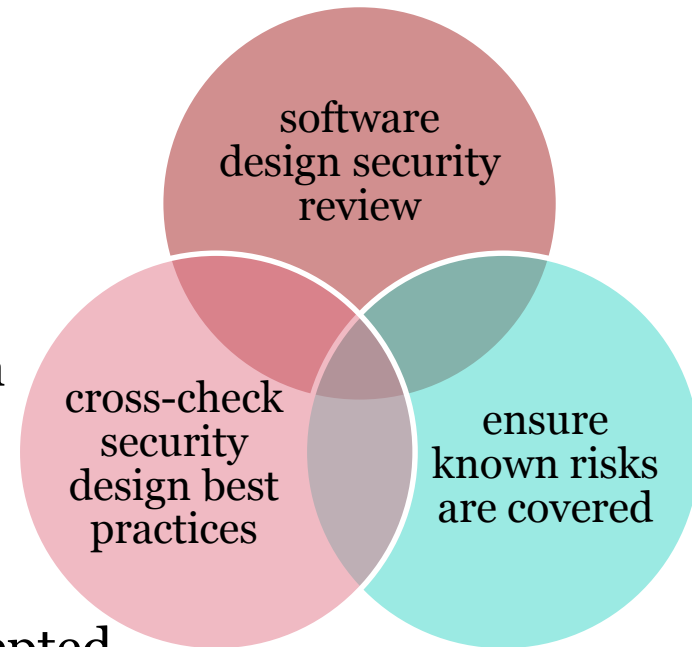
Secure Architecture ...more on page 54			
	 SA 1	 SA 2	 SA 3
OBJECTIVE	Insert consideration of proactive security guidance into the software design process	Direct the software design process toward known-secure services and secure-by-default designs	Formally control the software design process and validate utilization of secure components
ACTIVITIES	<ul style="list-style-type: none">A. Maintain list of recommended software frameworksB. Explicitly apply security principles to design	<ul style="list-style-type: none">A. Identify and promote security services and infrastructureB. Identify security design patterns from architecture	<ul style="list-style-type: none">A. Establish formal reference architectures and platformsB. Validate usage of frameworks, patterns, and platforms

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

Design Review ...more on page 58			
	 DR 1	 DR 2	 DR 3
OBJECTIVE	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
ACTIVITIES	<ul style="list-style-type: none">A. Identify software attack surfaceB. Analyze design against known security requirements	<ul style="list-style-type: none">A. Inspect for complete provision of security mechanismsB. Deploy design review service for project teams	<ul style="list-style-type: none">A. Develop data-flow diagrams for sensitive resourcesB. Establish release gates for design review

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!

Start

- lightweight checklists
- inspect critical software

Improve




- Automation
- Increase coverage / efficacy

Mature

- Integrate in development
- Produce audit evidence
- Test & production release gates

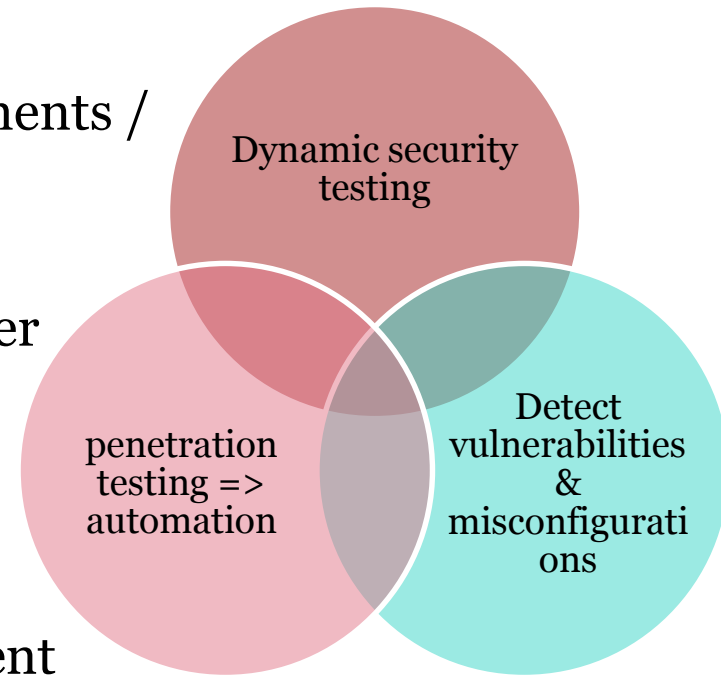


Implementation Review




Implementation Review ...more on page 52			
	 IR 1	 IR 2	 IR 3
OBJECTIVE	Opportunisticly find basic code-level vulnerabilities and other high-risk security issues	Make implementation review during development more accurate and efficient through automation	Mandate comprehensive implementation review process to discover language-level and application-specific risks
ACTIVITIES	A. Create review checklists from known security requirements B. Perform point-review of high-risk code	A. Utilize automated code analysis tools B. Integrate code analysis into development process	A. Customize code analysis for application-specific concerns B. Establish release gates for code review

Security Testing

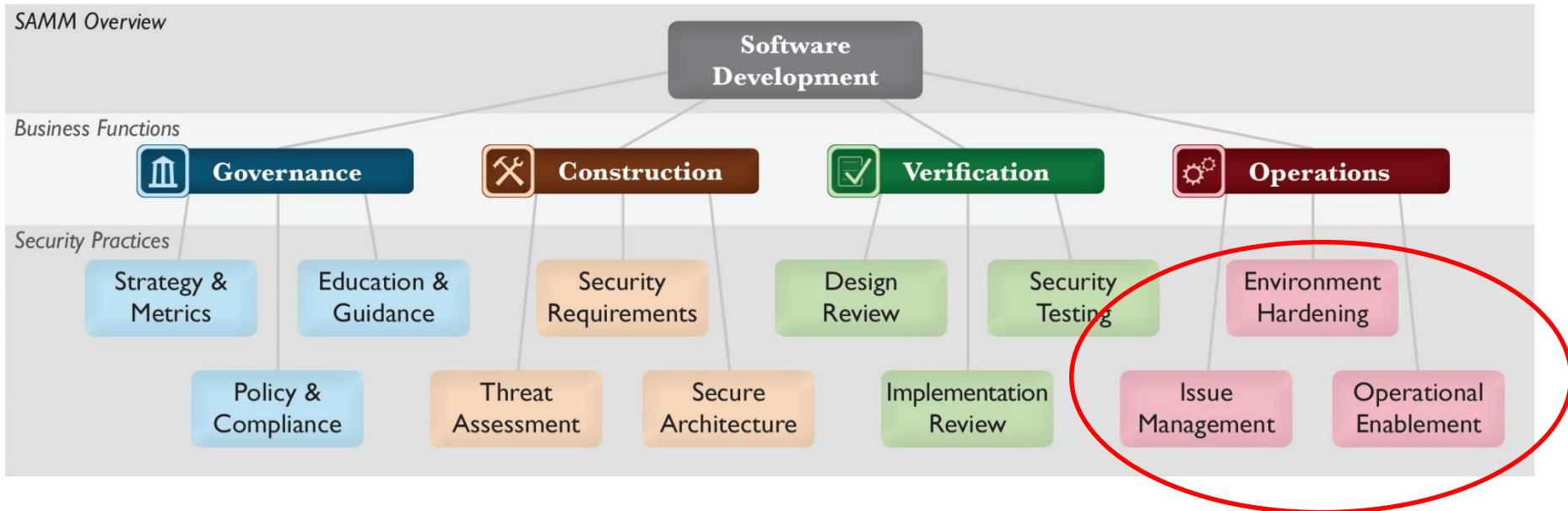
- Based on security & compliance requirements / checklist of common vulnerabilities
- Manual testing can be done, scaled with tooling: intercepting proxy and/or scanner
- Detected defects will require validation, risk analysis & recommendations to fix
- Automate to repeat tests for each release
- Introduce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



Security Testing

	Security Testing ...more on page 66		
	 ST 1	 ST 2	 ST 3
OBJECTIVE	Establish process to perform basic security tests based on implementation and software requirements	Make security testing during development more complete and efficient through automation	Require application-specific security testing to ensure baseline security before deployment
ACTIVITIES	<p>A. Derive test cases from known security requirements</p> <p>B. Conduct penetration testing on software releases</p>	<p>A. Utilize automated security testing tools</p> <p>B. Integrate security testing into development process</p>	<p>A. Employ application-specific security testing automation</p> <p>B. Establish release gates for security testing</p>

Security Practices - Operations



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

Issue Management

...more on page 60



OBJECTIVE

Understand high-level plan for responding to issue reports or incidents

Elaborate expectations for response process to improve consistency and communications

Improve analysis and data gathering within response process for feedback into proactive planning

ACTIVITIES

A. Identify point of contact for security issues
B. Create informal security response team(s)

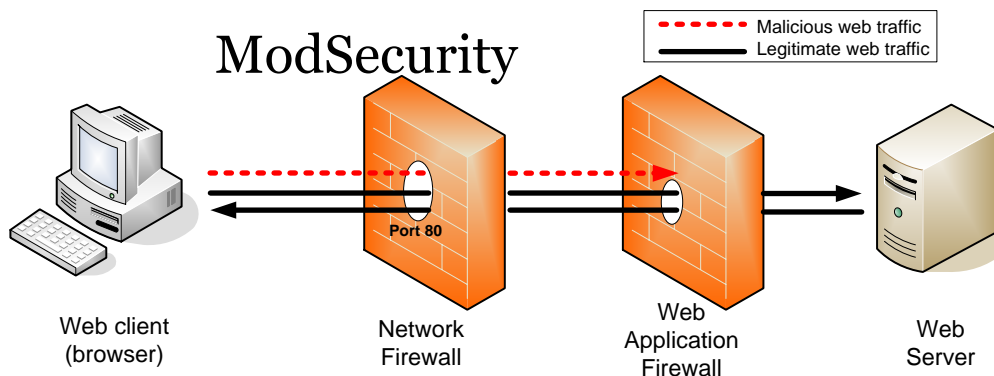
A. Establish consistent issue response process
B. Adopt a security issue disclosure process

A. Conduct root cause analysis for issues
B. Collect per-issue metrics




Environment Hardening

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

- Add WAF layer (virtual patching)



Environment Hardening

Environment Hardening ...more on page 74			
	 EH 1	 EH 2	 EH 3
OBJECTIVE	Understand baseline operational environment for applications and software components	Improve confidence in application operations by hardening the operating environment	Validate application health and status of operational environment against known best practices
ACTIVITIES	<p>A. Maintain operational environment specification</p> <p>B. Identify and install critical security upgrades and patches</p>	<p>A. Establish routine patch management process</p> <p>B. Monitor baseline environment configuration status</p>	<p>A. Identify and deploy relevant operations protection tools</p> <p>B. Expand audit program for environment configuration</p>

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

	Operational Enablement ...more on page 78		
			
OBJECTIVE	Enable communications between development teams and operators for critical security-relevant data	Improve expectations for continuous secure operations through provision of detailed procedures	Mandate communication of security information and validate artifacts for completeness
ACTIVITIES	<p>A. Capture critical security information for deployment</p> <p>B. Document procedures for typical application alerts</p>	<p>A. Create per-release change management procedures</p> <p>B. Maintain formal operational security guides</p>	<p>A. Expand audit program for operational information</p> <p>B. Perform code signing for application components</p>

Agenda

1. Motivation
2. SAMMM At A Glance
3. SAMMM 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!