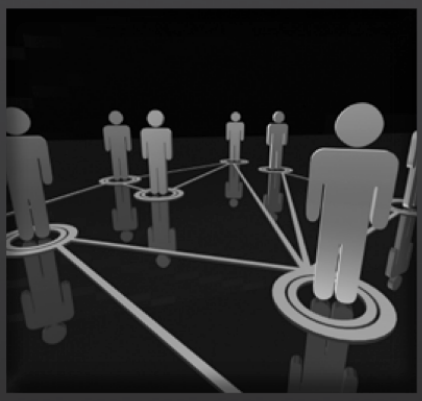


Software Assurance Maturity Model

A guide to building security into software development

VERSION 1.6

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 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.

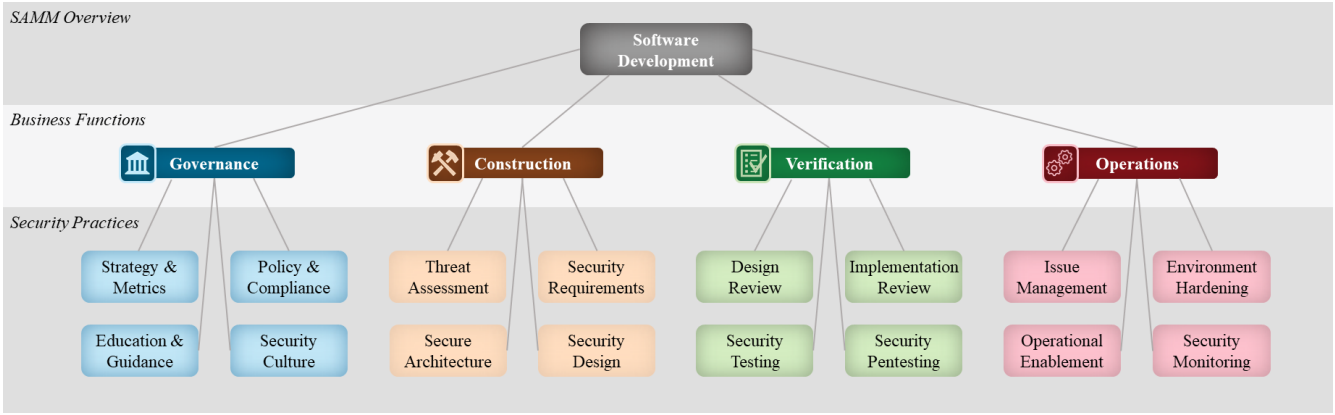
Version 1.5 of SAMM incorporates a refinement of the scoring model to provide more granularity to the scoring in an assessment. Now an organization will get credit for all the related work done in a practice rather than having the base number held at the highest completed maturity level. The updated 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:

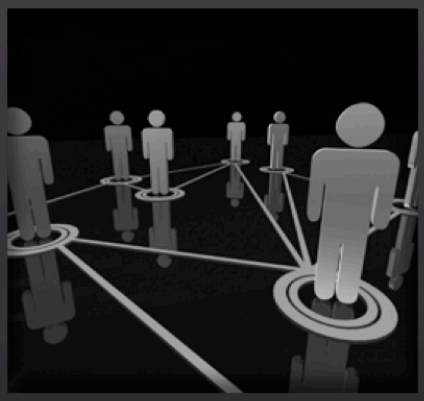
- ✦ An organization's behavior changes slowly over time
  A successful software security program should be specified in small iterations that deliver tangible assurance gains while incrementally working toward long-term goals.
- ✦ There is no single recipe that works for all organizations
  A software 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.


SAMM Overview
Software Development
Business Functions
Governance
Construction
Verification
Operations
Security Practices
Strategy & Metrics
Policy & Compliance
Education & Guidance
Security Culture
Threat Assessment
Security Requirements
Secure Architecture
Security Design
Design Review
Implementation Review
Security Testing
Security Pentesting
Issue Management
Environment Hardening
Operational Enablement
Security Monitoring

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 sixteen 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 or- ganization’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 construction 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.

Security Culture Security Culture reduces vulnerabilities by human faults, and it grows up by the human players of the software organization.

...more on page 11

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.

Security Design involves planning and setting up of penetration tests and test environment.

...more on page 15

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.

Penetration Testing involves testing the latest software releases in the system's environments against previously designed penetration testing routines as well as the integration of these testing routines into continuous integration/delivery mechanisms with feedback loops to software design and implementation.

...more on page 19

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.

Security Monitoring involves process and organization wide structural abilities and skills for finding and dealing with cyber security threats, reacting to them and rolling out updates/degradations to the end user.

...more on page 23

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 differ 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

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

Governance

Description of Security Practices

Strategy & Metrics

The Strategy & Metrics (SM) practice is focused on establishing the framework within an organization for a software security assurance program. This is the most fundamental step in defining security goals in a way that's both measurable and aligned with the organization's real business risk.

By starting with lightweight risk profiles, an organization grows into more advanced risk classification schemes for application and data assets over time. With additional insight on relative risk measures, an organization can tune its project-level security goals and develop granular roadmaps to make the security program more efficient. At the more advanced levels within this practice, an organization draws upon many data sources, both internal and external, to collect metrics and qualitative feedback on the security program. This allows fine tuning of cost outlay versus the realized benefit at the program level.

Policy & Compliance

The Policy & Compliance (PC) practice is focused on understanding and meeting external legal and regulatory requirements while also driving internal security standards to ensure compliance in a way that's aligned with the business purpose of the organization.

A driving theme for improvement within this practice is focus on project-level audits that gather information about the organization's behavior in order to check that expectations are being met. By introducing routine audits that start out lightweight and grow in depth over time, organizational change is achieved iteratively.

In a sophisticated form, provision of this practice entails organization-wide understanding of both internal standards and external compliance drivers while also maintaining low-latency checkpoints with project teams to ensure no project is operating outside expectations without visibility.

Education & Guidance

The Education & Guidance (EG) practice is focused on arming personnel involved in the software life-cycle with knowledge and resources to design, develop, and deploy secure software. With improved access to information, project teams will be better able to proactively identify and mitigate specific security risks that apply to their organization.

One major theme for improvement across the objectives is providing training for employees, either through instructor-led sessions or computer-based modules. As an organization progresses, a broad base of training is built by starting with developers and moving to other roles throughout the organization, culminating with the addition of role-based certification to ensure comprehension of the material.

In addition to training, this practice also requires pulling security-relevant information into guidelines that serve as reference information to staff. This builds a foundation for establishing a baseline expectation for security practices in your organization, and later allows for incremental improvement once usage of the guidelines has been adopted.

Security Culture

The Security Culture (SC) focuses on establishing a framework of communication and value work within a software organization. An open discussion about faults regarding the development process and product is initiated. Therefore, a list of organizational deficiencies, human faults and intentional activities is created.

The staff members will have a value set and they will be aware of vulnerabilities. Improvement is possible with accurate tracking of lifelong learning activities and with an organization-wide participation in a reward system.

In a sophisticated form, a regular assessment of the security culture and continuous enhancement of the security culture is established. Teams learn with retrospective. Teams regularly measure their collaboration maturity level.

Governance

Activities Overview

Strategy & Metrics

...more on page 38

	SM 1	SM 2	SM 3
Objective:	Establish a 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:	A. Estimate overall business risk profile B. Build and maintain assurance program roadmap	A. Classify data and applications based on business risk B. Establish and measure per–classification security goals	A. Conduct periodic industry-wide cost comparisons B. Collect metrics for historic security spend

Policy & Compliance

...more on page 45

	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

...more on page 52

	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 lifecycle 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

Security Culture

…more on page 59

	SC 1	SC 2	SC 3
Objective:	Establish first elements of a security culture.	Refinement of the security culture.	Regular assessment and continuous enhancement of the security culture.
Activities:	A. Establish communication B. Creating values diversity	A. Giving culture commitment B. Acting as a team and apply lifelong learning	A. Culture benchmarking B. Culture retrospective and reflection

Construction

Description of Security Practices

Threat Assessment

The Threat Assessment (TA) practice is centered on identification and understanding the project-level risks based on the functionality of the software being developed and characteristics of the runtime environment. From details about threats and likely attacks against each project, the organization as a whole operates more effectively through better decisions about prioritization of initiatives for security. Additionally, decisions for risk acceptance are more informed, therefore better aligned to the business.

By starting with simple threat models and building to more detailed methods of threat analysis and weighting, an organization improves over time. Ultimately, a sophisticated organization would maintain this information in a way that is tightly coupled to the compensating factors and pass-through risks from external entities. This provides greater breadth of understanding for potential downstream impacts from security issues while keeping a close watch on the organization's current performance against known threats.

Security Requirements

The Security Requirements (SR) practice is focused on proactively specifying the expected behavior of software with respect to security. Through addition of analysis activities at the project level, security requirements are initially gathered based on the high-level business purpose of the software. As an orga-nization advances, more advanced techniques are used such as access control specifications to discover new security requirements that may not have been initially obvious to development.

In a sophisticated form, provision of this practice also entails pushing the security requirements of the organization into its relationships with suppliers and then auditing projects to ensure all are adhering to expectations with regard to specification of security requirements.

Secure Architecture

The Secure Architecture (SA) practice is focused on proactive steps for an organization to design and build secure software by default. By enhancing the software design process with reusable services and components, the overall security risk from software development can be dramatically reduced.

Beginning from simple recommendations about software frameworks and explicit consideration of secure design principles, an organization evolves toward consistently using design patterns for security functionality. Also, activities encourage project teams to increased utilization of centralized security services and infrastructure.

As an organization evolves over time, sophisticated provision of this practice entails organizations building reference platforms to cover the generic types of software they build. These serve as frameworks upon which developers can build custom software with lower risk of vulnerabilities.

Security Design

The Security Design (SD) method focuses on proactive planning of penetration tests and its test environment on project level. The analysis of security risks and potential attacks during development ensures purposefully testing. A verification of implications on robustness against attacks and reliability is especially important.

Furthermore, planning of legal principles for security/safety will be integrated and complemented by evaluations of executed penetration tests, which will lead on to iterative development cycles. While most of tests are completed during roll out, the SD requires further planning for ongoing tests to test products against new occurring threats.

Ongoing auditing and testing steadily expands the company’s knowledge. Tools for hardware independent testing, which enables partial automation to simplify the process, can be developed due to diligent planning and knowledge acquisition.

Construction

Activities Overview

Threat Assessment

...more on page 66

	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 align 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

...more on page 73

	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:	A. Derive security requirements from business functionality B. Evaluate security and compliance guidance for requirements	A. Build an access control matrix for resources and capabilities B. Specify security requirements based on known risks	A. Build security requirements into supplier agreements B. Expand audit program for security requirements

Secure Architecture

...more on page 80

	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:	A. Maintain list of recommended software frameworks B. Explicitly apply security principles to design	A. . Identify and promote security services and infrastructure B. Identify security design patterns from architecture	A. Establish formal reference architectures and platforms B. Validate usage of frameworks, patterns, and platforms

Security Design

…more on page 87

	SD 1	SD 2	SD 3
Objective:	Identify security risks and attack potentialities for individual projects.	Plan penetration tests in advance and take future attack potentialities into account.	Facilitate hardware independent and ongoing penetration tests.
Activities:	A. Compile a risk analysis in due consideration of security goals. B. Set up an attack pattern catalog	A. Define test environment and categorize penetration tests. B. Downscale security risks of external software and hardware components.	A. Facilitate hardware independent penetration tests and implement development cycles by the use of structured data exchange. B. Categorize, acquire and store all relevant hardware and software versions.

Verification

Description of Security Practices

Design Review

The Design Review (DR) practice is focused on assessment of software design and architecture for security-related problems. This allows an organization to detect architecture-level issues early in software development and thereby avoid potentially large costs from refactoring later due to security concerns.

Beginning with lightweight activities to build understanding of the security-relevant details about an architecture, an organization evolves toward more formal inspection methods that verify completeness in provision of security mechanisms. At the organization level, design review services are built and offered to stakeholders.

In a sophisticated form, provision of this practice involves detailed, data-level inspection of designs, and enforcement of baseline expectations for conducting design assessments and reviewing findings before releases are accepted.

Implementation Review

The Implementation Review (IR) practice is focused on inspection of software at the source code and configuration level in order to find security vulnerabilities. Code-level vulnerabilities are generally simple to understand conceptually, but even informed developers can easily make mistakes that leave software open to potential compromise.

To begin, an organization uses lightweight checklists and for efficiency, only inspects the most critical software modules. However, as an organization evolves it uses automation technology to dramatically improve coverage and efficacy of implementation review activities.

Sophisticated provision of this practice involves deeper integration of implementation review into the development process to enable project teams to find problems earlier. This also enables organizations to better audit and set expectations for implementation review findings before releases can be made.

Security Testing

The Security Testing (ST) practice is focused on inspection of software in the runtime environment in order to find security problems. These testing activities bolster the assurance case for software by checking it in the same context in which it is expected to run, thus making visible operational misconfigurations or errors in business logic that are difficult to otherwise find.

Starting with penetration testing and high-level test cases based on the functionality of software, an organization evolves toward usage of security testing automation to cover the wide variety of test cases that might demonstrate a vulnerability in the system.

In an advanced form, provision of this practice involves customization of testing automation to build a battery of security tests covering application-specific concerns in detail. With additional visibility at the organization level, security testing enables organizations to set minimum expectations for security testing results before a project release is accepted.

Security Pentesting

The Security Pentesting (SP) practice is focused on the execution of penetration tests in the test and delivery environment on a project level in order to find security and safety related problems.

During the completion of the SP, methods for a complete coverage of attack vectors are used, mostly attack pattern concatenation in the form of attack pattern paths. Version management ensures the consistent and thorough processing of attack vectors.

The introduction of automated penetration testing builds and continuous integration mechanisms lead to more frequent automated penetration testing and constant availability of a current robust build for all purposes.

Furthermore, a detailed documentation for vulnerability mitigation and exploitation guidance, feedback loops for design and architecture as well as release gates for penetration testing guarantee a safe execution of the complete software development lifecycle.

Verification

Activities Overview

Design Review

…more on page 94

	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:	A. Identify software attack surface B. Analyze design against known security requirements	A. Inspect for complete provision of security mechanisms B. Deploy design review service for project teams	A. Develop data-flow diagrams for sensitive resources B. Establish release gates for design review

Implementation Review

…more on page 101

	IR 1	IR 2	IR 3
Objective:	Opportunistically 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

…more on page 108

	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:	A. Derive test cases from known security requirements B. Conduct penetration testing on software releases	A. Utilize automated security testing tools B. Integrate security testing into development process	A. Employ application-specific security testing automation B. Establish release gates for security testing

Security Pentesting

…more on page 115

	SP 1	SP 2	SP 3
Objective:	Establish and execute a complete workflow for active penetration testing.	Combine the project-wide penetration testing workflow with continuous integration (CI) techniques.	Establish feedback loops and release gates within the continuous integration for the software design and development cycle.
Activities:	A. Lay out an appropriate testing structure in order to test the system against all possible vectors of attacks B. Add version management for maximum covering rates and criteria documentation for expressiveness of test results	A. Expand the automated build with related penetration testing procedures B. Establish a continuous integration mechanism of the automated build into the project’s version management	A. Report (un-) successfully tested builds on a feedback platform B. Establish release gates for penetration testing

Operations

Description of Security Practices

Issue Management

The Issue Management (IM) practice is focused on the processes within an organization with respect to handling issue reports and operational incidents. By having these processes in place, an organization's projects will have consistent expectations and increased efficiency for handling these events, rather than chaotic and uninformed responses.

Starting from lightweight assignment of roles in the event of an incident, an organization grows into a more formal incident response process that ensures visibility and tracking on issues that occur. Communications are also improved to improve overall understanding of the processes.

In an advanced form, issue management involves thorough dissecting of incidents and issue reports to collect detailed metrics and other root-cause information to feedback into the organization's downstream behavior.

Environment Hardening

The Environment Hardening (EH) practice is focused on building assurance for the runtime environment that hosts the organization's software. Since secure operation of an application can be deteriorated by problems in external components, hardening this underlying infrastructure directly improves the overall security posture of the software.

By starting with simple tracking and distributing of information about the operating environment to keep development teams better informed, an organization evolves to scalable methods for managing deployment of security patches and instrumenting the operating environment with early-warning detectors for potential security issues before damage is done.

As an organization advances, the operating environment is further reviewed and hardened by deployment of protection tools to add layers of defenses and safety nets to limit damage in case any vulnerabilities are exploited.

Operational Enablement

The Operational Enablement (OE) practice is focused on gathering security critical information from the project teams building software and communicating it to the users and operators of the software. Without this information, even the most securely designed software carries undue risks since important security characteristics and choices will not be known at a deployment site.

Starting from lightweight documentation to capture the most important details for users and operators, an organization evolves toward building complete operational security guides that are delivered with each release.

In an advanced form, operational enablement also entails organization-level checks against individual project teams to ensure that information is being captured and shared according to expectations.

Security Monitoring

The Security Monitoring (MO) practice is focused on the overall process and operations which have to be installed in the organization to guarantee cyber security for products during the whole lifecycle. This includes several actions as well as the need of structural facilities in any organization which is involved in embedded software development.

More advanced organizations should structure their knowledge gathered from penetratrion tests and other information sources. This should lead to a complete test of every combination of softtware and hardware out in the field for each new threat situation. The configuration management of potential products therfore should be available in real time.

Advanced organizations need a complete security strategy and team. Also they should have a clear deployment strategy and update channel for every product at the end user. Also, the need for continuously watching the state of the art, and thus derive the evaluation of current threat risks has to be maintained by a centrally organized cyber security team.

Operations

Activities Overview

Issue Management

...more on page 122

	IM 1	IM 2	IM 3
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

...more on page 129

	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:	A. Maintain operational environment specification B. Identify and install critical security upgrades and patches	A. Establish routine patch management process B. Monitor baseline environment configuration status	A. Identify and deploy relevant operations protection tools B. Expand audit program for environment configuration

Operational Enablement

...more on page 136

	OE 1	OE 2	OE 3
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:	A. Capture critical security information for deployment B. Document procedures for typical application alerts	A. Create per-release change management procedures B. Maintain formal operational security guides	A. Expand audit program for operational information B. Perform code signing for application components

Security Monitoring

…more on page 143

	MO 1	MO 2	MO 3
Objective:	Awareness of potential attack patterns.	Knowledge about impact of attack patterns to released hardware and software versions and variants.	Minimize negative impacts for customers.
Activities:	A. Maintenance of the attack pattern catalog	A. Implement all relevant penetration tests on all relevant hardware and software versions and variants B. Execute all relevant penetration tests on all relevant hardware and software	A. Operation of deployment infrastructure B. Deploy degrada-tions/updates to customer

Governance

Assessment worksheet

Strategy & Metrics

SCORE	0.0	0.2	0.5	1.0
SM1				
✦Is there a software security assurance program in place?	NO	<1 YR	>1 YR	MATURE
✦Are development staff aware of future plans for the assurance program?	NO	SOME	HALF	MOST
✦Do the business stakeholders understand your organization’s risk profile?	NO	SOME	HALF	MOST
SM2				
✦Are many of your applications and resources categorized by risk?	NO	SOME	HALF	MOST
✦Are risk ratings used to tailor the required assurance activities?	NO	SOME	HALF	MOST
✦Does the organization know about what’s required based on risk ratings?	NO	SOME	HALF	MOST
SM3				
✦Is per-project data for the cost of assurance activities collected?	NO	SOME	HALF	MOST
✦Does your organization regularly compare your security spend with that of other organizations?	NO	ONCE	EVERY 2-3 YRS	ANNUALLY

Policy & Compliance

SCORE	0.0	0.2	0.5	1.0
PC1				
✦Do project stakeholders know their project’s compliance status?	NO	SOME	HALF	MOST
✦Are compliance requirements specifically considered by project teams?	NO	NOT APPLY	AD-HOC	YES
PC2				
✦Does the organization utilize a set of policies and standards to control software development?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Are project teams able to request an audit for compliance with policies and standards?	NO	SOME	HALF	MOST
PC3				
✦Are projects periodically audited to ensure a baseline of compliance with policies and standards?	NO	SOME	HALF	MOST
✦Does the organization systematically use audits to collect and control compliance evidence?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED

Education & Guidance

SCORE	0.0	0.2	0.5	1.0
EG1				
✦Have developers been given high-level security awareness training?	NO	ONCE	EVERY 2-3 YEARS	ANNUALLY
✦Does each project team understand where to find secure development best-practices and guidance?	NO	SOME	HALF	MOST
EG2				
✦Are those involved in the development process given role-specific security training and guidance?	NO	SOME	HALF	MOST
✦Are stakeholders able to pull in security coaches for use on projects?	NO	SOME	HALF	MOST
EG3				
✦Is security-related guidance centrally controlled and consistently distributed throughout the organization?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Are developers tested to ensure a baseline skill- set for secure development practices?	NO	ONCE	EVERY 2-3 YEARS	ANNUALLY

Security Culture

SCORE	0.0	0.2	0.5	1.0
SC1				
✦Does your organization have an open discussion about organizational deficiencies, human faults and intentional activities?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Are diversity principles used in the organization?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Do employees consider security a high-priority issue?	NO	SOME	HALF	MOST
SC2				
✦Is a value development established?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Is lifelong learning established in the organization?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Is a fundamental collaboration visible in the organization?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
SC3				
✦Is a customized security culture benchmarking (assessment) established?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Do teams perform value retrospective and reflection of the security culture?	NO	SOME	HALF	MOST
✦Is a mature collaboration visible in the organization?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED

Construction

Assessment worksheet

Threat Assessment

SCORE	0.0	0.2	0.5	1.0
TA1				
✦Do projects in your organization consider and document likely threats?	NO	SOME	HALF	MOST
✦Does your organization understand and document the types of attackers it faces?	NO	SOME	HALF	MOST
TA2				
✦Do project teams regularly analyze functional requirements for likely abuses?	NO	SOME	HALF	MOST
✦Do project teams use a method of rating threats for relative comparison?	NO	SOME	HALF	MOST
✦Are stakeholders aware of relevant threats and ratings?	NO	SOME	HALF	MOST
TA3				
✦Do project teams specifically consider risk from external software?	NO	SOME	HALF	MOST
✦Are the majority of the protection mechanisms and controls captured and mapped back to threats?	NO	SOME	HALF	MOST

Security Requirements

SCORE	0.0	0.2	0.5	1.0
SR1				
✦Do project teams specify security requirements during development?	NO	SOME	HALF	MOST
✦Do project teams pull requirements from best practices and compliance guidance?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
SR2				
✦Do stakeholders review access control matrices for relevant projects?				
NO	SOME	HALF	MOST	
✦Do project teams specify requirements based on feedback from other security activities?	NO	SOME	HALF	MOST
SR3				
✦Do stakeholders review vendor agreements for security requirements?	NO	SOME	HALF	MOST
✦Are audits performed against the security requirements specified by project teams?	NO	ONCE	EVERY 2-3 YEARS	ANNUALLY

Secure Architecture

SCORE	0.0	0.2	0.5	1.0
SA1				
✦Are project teams provided with a list of recommended third-party components?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Are project teams aware of secure design principles and do they apply them consistently?	NO	SOME	HALF	MOST
SA2				
✦Do you advertise shared security services with guidance for project teams?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Are project teams provided with prescriptive design patterns based on their application architecture?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
SA3				
✦Do project teams build software from centrally- controlled platforms and frameworks?	NO	SOME	HALF	MOST
✦Are project teams audited for the use of secure architecture components?	NO	ONCE	EVERY 2-3 YEARS	ANNUALLY

Security Design

SCORE	0.0	0.2	0.5	1.0
SD1				
✦Do projects execute risk analysis (e.g., in regard to (A)SIL)?	NO	SOME	HALF	MOST
✦Are identified attack patterns documented in a reusable catalog?	NO	SOME	HALF	MOST
SD2				
✦Are penetration tests planned and categorized for all products?	NO	SOME	HALF	MOST
✦How often are external security reports reviewed?	NO	ONCE	EVERY 2-3 YEARS	ANNUALLY
✦Is special staff trained or are external certified testers employed?	NO	SOME	HALF	MOST
✦Is the security of external software and hardware components guaranteed?	NO	SOME	HALF	MOST
SD3				
✦Can products be tested hardware independently?	NO	SOME	HALF	MOST
✦Can products be updated subsequently after rollout?	NO	SOME	HALF	MOST
✦Can results from penetration tests be reused?	NO	SOME	HALF	MOST
✦Is security already implemented into the lifecycle?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED

Verification

Assessment worksheet

Design Review

SCORE	0.0	0.2	0.5	1.0
DR1				
✦Do project teams document the attack perimeter of software designs?	NO	SOME	HALF	MOST
✦Do project teams check software designs against known security risks?	NO	SOME	HALF	MOST
DR2				
✦Do project teams specifically analyze design elements for security mechanisms?	NO	SOME	HALF	MOST
✦Are project stakeholders aware of how to obtain a formal secure design review?	NO	SOME	HALF	MOST
DR3				
✦Does the secure design review process incorporate detailed data-level analysis?	NO	SOME	HALF	MOST
✦Does a minimum security baseline exist for secure design review results?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS

Implementation Review

SCORE	0.0	0.2	0.5	1.0
IR1				
✦Do project teams have review checklists based on common security related problems?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Do project teams review selected high-risk code?	NO	SOME	HALF	MOST
IR2				
✦Can project teams access automated code analysis tools to find security problems?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Do stakeholders consistently review results from code reviews?	NO	SOME	HALF	MOST
IR3				
✦Do project teams utilize automation to check code against application-specific coding standards?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Does a minimum security baseline exist for code review results?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS

Security Testing

SCORE	0.0	0.2	0.5	1.0
ST1				
✦Do projects specify security testing based on defined security requirements?	NO	SOME	HALF	MOST
✦Is penetration testing performed on high-risk projects prior to release?	NO	SOME	HALF	MOST
✦Are stakeholders aware of the security test status prior to release?	NO	SOME	HALF	MOST
ST2				
✦Do projects use automation to evaluate security test cases?	NO	SOME	HALF	MOST
✦Do projects follow a consistent process to evaluate and report on security tests to stakeholders?	NO	SOME	HALF	MOST
ST3				
✦Are security test cases comprehensively generated for application-specific logic?	NO	SOME	HALF	MOST
✦Does a minimum security baseline exist for security testing?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS

Security Pentesting

SCORE	0.0	0.2	0.5	1.0
SP1				
✦Are identified attack pattern structured and tested orderly?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Do projects use version control management for penetration testing?	NO	SOME	HALF	MOST
✦Do projects integrate practical exploitation and mitigation comments for every attack pattern?	NO	SOME	HALF	MOST
✦Are all relevant and found attacks tested on the software?	NO	SOME	HALF	MOST
SP2				
✦Do penetration testers have access to automatic software builds that include their tests?	NO	SOME	HALF	MOST
✦Is the penetration testing routine embedded in a continuous integration mechanism?	NO	NOT APPLY	AD-HOC	YES
✦Do developers and penetration testers discuss and handle the results of penetration test cases?	NO	SOME	HALF	MOST
SP3				
✦Do projects integrate preventive security feedback loops including penetration testing?	NO	SOME	HALF	MOST
✦Do projects establish release gates in the software development lifecycle for penetration testing?	NO	SOME	HALF	MOST

Operations

Assessment worksheet

Issue Management

SCORE	0.0	0.2	0.5	1.0
IM1				
✦Do projects have a point of contact for security issues or incidents?	NO	SOME	HALF	MOST
✦Does your organization have an assigned security response team?	NO	<1 YEAR	>1 YEAR	MATURE
✦Are project teams aware of their security point(s) of contact and response team(s)?	NO	SOME	HALF	MOST
IM2				
✦Does the organization utilize a consistent process for incident reporting and handling?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Are project stakeholders aware of relevant security disclosures related to their software projects?	NO	SOME	HALF	MOST
IM3				
✦Are incidents inspected for root causes to generate further recommendations?	NO	SOME	HALF	MOST
✦Do projects consistently collect and report data and metrics related to incidents?	NO	SOME	HALF	MOST

Environment Hardening

SCORE	0.0	0.2	0.5	1.0
EH1				
✦Do projects document operational environment security requirements?	NO	SOME	HALF	MOST
✦Do projects check for security updates to third-party software components?	NO	SOME	HALF	MOST
EH2				
✦Is a consistent process used to apply upgrades and patches to critical dependencies?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Do projects leverage automation to check application and environment health?	NO	SOME	HALF	MOST
EH3				
✦ Are stakeholders aware of options for additional tools to protect software while running in operations?	NO	PER TEAM	ORG WIDE	INTEGRATED PROCESS
✦Does a minimum security baseline exist for environment health (versioning, patching, etc)?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED

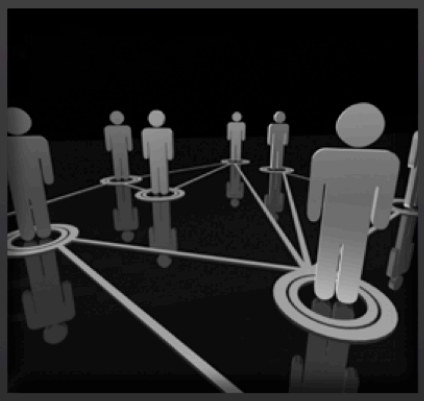
Operational Enablement

SCORE	0.0	0.2	0.5	1.0
OE1				
✦Are security notes delivered with each software release?	NO	SOME	HALF	MOST
✦Are security-related alerts and error conditions documented on a per-project basis?	NO	SOME	HALF	MOST
OE2				
✦Do projects utilize a change management process that’s well understood?	NO	SOME	HALF	MOST
✦Do project teams deliver an operational security guide with each product release?	NO	SOME	HALF	MOST
OE3				
✦Are project releases audited for appropriate operational security information?	NO	ONCE	EVERY 2-3 YRS	ANNUALLY
✦Is code signing routinely performed on software components using a consistent process?	NO	NOT APPLY	AD-HOC	YES

Security Monitoring

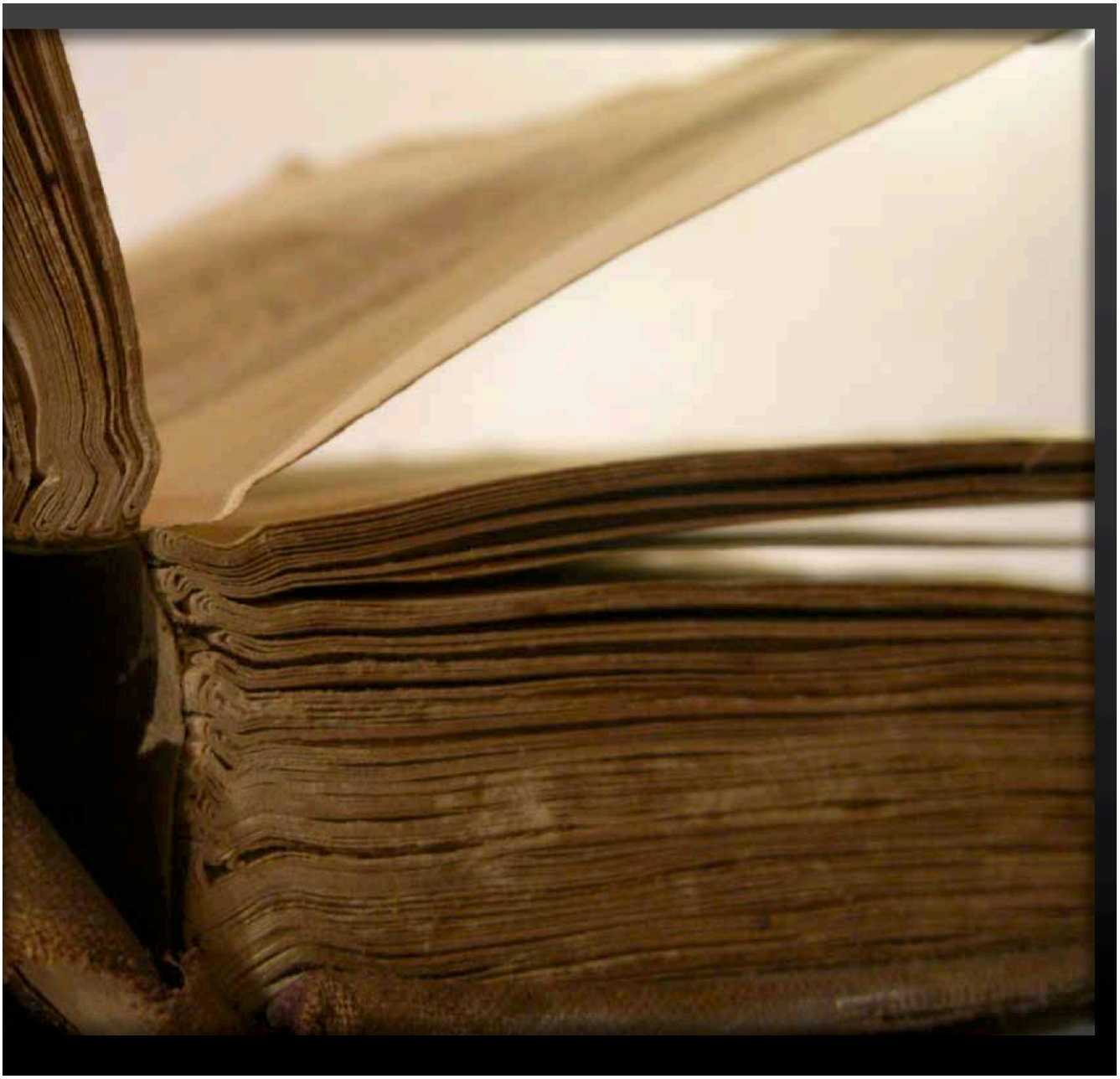
SCORE	0.0	0.2	0.5	1.0
MO1				
✦Does your organization gather information about new vulnerability issues?	NO	ONCE	EVERY 2-3 YRS	ANNUALLY
✦Are vulnerability issues filtered to dedicated project, used hardware and software stacks?	NO	NOT APPLY	AD-HOC	YES
✦Are there project teams after final (serial) release responsible for these issues?	NO	SOME	HALF	MOST
MO2				
✦Is your penetration test continuously developed?	NO	NOT APPLY	AD-HOC	YES
✦Do you assure that penetration tests are executed for every released hardware/software combination?	NO	NOT APPLY	AD-HOC	YES
✦Does your company collect and archive all results of these penetration tests?	NO	SOME	HALF	MOST
MO3				
✦Does your company know how to contact the end users of your devices?	NO	SOME	HALF	MOST
✦Does your company have a process which defines the propagation of security vulnerabilities to your customers?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED
✦Do your project teams provide a security degradation guidance model?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED

✦Does your company have a process which defines the propagation of security updates to end users?	NO	BUS AREA	ORG WIDE	ORG WIDE & REQUIRED



The Security Practices

An explanation of the details



This section defines the building blocks of SAMM, the maturity levels under each security practice. For each practice, the three levels are covered in a summary table. Following that, the description for each level includes detailed explanations of the required activities, results an organization can expect from attaining the level, success metrics to gauge performance, required ongoing personnel investment, and additional associated costs.

Strategy & Metrics

	SM1	SM2	SM3
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	A. Estimate overall business risk profile B. Build and maintain assurance program roadmap	A. Classify data and applications based on business risk B. Establish and measure per- classification security goals	A. Conduct periodic industry-wide cost comparisons B. Collect metrics for historic security spend
Assessment	✦Is there a software security assurance program in place? ✦Are development staff aware of future plans for the assurance program? ✦Do the business stakeholders understand your organization’s risk profile?	✦Are many of your applications and resources categorized by risk? ✦Are risk ratings used to tailor the required assurance activities? ✦Does the organization know about what’s required based on risk ratings?	✦Is per-project data for the cost of assurance activities collected? ✦Does your organization regularly compare your security spend with that of other organizations?
Results	✦Concrete list of the most critical business-level risks caused by software ✦Tailored roadmap that addresses the security needs for your organization with minimal overhead ✦Organization-wide understanding of how the assurance program will grow over time	✦Customized assurance plans per project based on core value to the business ✦Organization-wide understanding of security-relevance of data and application assets ✦Better informed stakeholders with respect to understanding and accepting risks	✦Information to make informed case-by-case decisions on security expenditures ✦Estimates of past loss due to security issues ✦Per-project consideration of security expense versus loss potential ✦Industry-wide due diligence with regard to security

Strategy & Metrics: SM1

Establish unified strategic roadmap for software security within the organization

Activities

A. Estimate overall business risk profile Interview business owners and stakeholders and create a list of worst-case scenarios across the organization's various application and data assets. Based on the way in which your organization builds, uses, or sells software, the list of worst-case scenarios can vary widely, but common issues include data theft or corruption, service outages, monetary loss, reverse engineering, account compromise, etc.

After broadly capturing worst-case scenario ideas, collate and select the most important based on collected information and knowledge about the core business. Any number can be selected but aim for at least three and no more than seven to make efficient use of time and keep the exercise focused.

Elaborate a description of each of the selected items and document details of contributing worst-case scenarios, potential contributing factors, and potential mitigating factors for the organization.

The final business risk profile should be reviewed with business owners and other stakeholders for understanding.

B. Build and maintain assurance program roadmap Understanding the main business risks to the organization, evaluate the current performance of the organization against each the twelve practices. Calculate a score for each practice based on the answers to the multiple-choice questions using the toolbox spreadsheet or SAMM survey application.

Once a good understanding of current status is obtained, the next goal is to identify the practices that will be improved in the next iteration. Select them based on business risk profile, other business drivers, compliance requirements, budget tolerance, etc. Once practices are selected, the goals of the iteration are to achieve the next objective under each. Iterations of improvement on the assurance program should be approximately 3-6 months, but an assurance strategy session should take place at least every three months to review progress on activities, performance against success metrics and other business drivers that may require program changes.

ASSESSMENT

- Is there a software security assurance program in place?
- Are development staff aware of future plans for the assurance program?
- Do the business stakeholders understand your organization's risk profile?

RESULTS

- Concrete list of the most critical business-level risks caused by software
- Tailored roadmap that addresses the security needs for your organization with minimal overhead
- Organization-wide understanding of how the assurance program will grow over time

SUCCESS METRICS

- >80% of stakeholders briefed on business risk profile in the past six months
- >80% of staff briefed on assurance program roadmap in the past three months
- >1 assurance program strategy session in the past three months

COSTS

- Buildout and maintenance of business risk profile
- Quarterly evaluation of assurance program

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- QA Testers
- Security Auditor

RELATED LEVELS

- Policy & Compliance - 1
- Threat Assessment - 1
- Security Requirements - 2

Strategy & Metrics: SM2

Measure relative value of data and software assets and choose risk tolerance

Activities

A. Classify data and applications based on business risk Establish a simple classification system to represent risk-tiers for applications. In its simplest form, this can be a High/Medium/Low categorization. More sophisticated classifications can be used, but there should be no more than seven categories and they should roughly represent a gradient from high to low impact against business risks.

Working from the organization's business risk profile, create project evaluation criteria that maps each project to one of the risk categories. A similar but separate classification scheme should be created for data assets and each item should be weighted and categorized based on potential impact to business risks.

Evaluate collected information about each application and assign each a risk category based upon overall evaluation criteria and the risk categories of data assets in use. This can be done centrally by a security group or by individual project teams through a customized questionnaire to gather the requisite information.

An ongoing process for application and data asset risk categorization should be established to assign categories to new assets and keep the existing information updated at least biannually.

B. Establish and measure per-classification security goals With a classification scheme for the organization's application portfolio in place, direct security goals and assurance program roadmap choices can be made more granular.

The assurance program's roadmap should be modified to account for each application risk category by specifying emphasis on particular practices for each category. For each iteration of the assurance program, this would typically take the form of prioritizing more higher-level objectives on the highest risk application tier and progressively less stringent objectives for lower/other categories.

This process establishes the organization's risk tolerance since active decisions must be made as to what specific objectives are expected of applications in each risk category. By choosing to keep lower risk applications at lower levels of performance with respect to the security practices, resources are saved in exchange for acceptance of a weighted risk. However, it is not necessary to arbitrarily build a separate roadmap for each risk category since that can leads to inefficiency in management of the assurance program itself.

ASSESSMENT

- Are many of your applications and resources categorized by risk?
- Are risk ratings used to tailor the required assurance activities?
- Does the organization know about what's required based on risk ratings?

Results

- Customized assurance plans per project based on core value to the business
- Organization-wide understanding of security-relevance of data
- and application assets
- Better informed stakeholders with respect to understanding and accepting risks

SUCCESS METRICS

- >90% applications and data assets evaluated for risk classification in the past 12 months
- >80% of staff briefed on relevant application and data risk ratings in the past six months
- >80% of staff briefed on relevant assurance program roadmap in the past three months

COSTS

- Buildout or license of application and data risk categorization scheme
- Program overhead from more granular roadmap planning

PERSONNEL

- Architects
- Managers
- Business Owners
- Security Auditor

RELATED LEVELS

- Policy & Compliance - 2
- Threat Assessment - 2
- Design Review - 2

Strategy & Metrics: SM3

Align security expenditure with relevant business indicators and asset value

Activities

B. Conduct periodic industry-wide cost comparisons Research and gather information about security costs from intra-industry communication forums, business analyst and consulting firms, or other external sources. In particular, there are a few key factors that need to be identified.

First, use collected information to identify the average amount of security effort being applied by similar types of organizations in your industry. This can be done either top- down from estimates of total percentage of budget, revenue, etc. or it can be done bottom-up by identifying security-related activities that are considered normal for your type of organization. Overall, this can be hard to gauge for certain industries, so collect information from as many relevant sources as are accessible.

The next goal of researching security costs is to determine if there are potential cost savings on third-party security products and services that your organization currently uses. When weighing the decision of switching vendors, account for hidden costs such as retraining staff or other program overhead.

Overall, these cost-comparison exercises should be conducted at least annually prior to the subsequent assurance program strategy session. Comparison information should be pre- sented to stakeholders in order to better align the assurance program with the business.

B. Collect metrics for historic security spend Collect project-specific information on the cost of past security incidents. For instance, time and money spent in cleaning up a breach, monetary loss from system outages, fines and fees to regulatory agencies, project-specific one-off security expenditures for tools or services, etc.

Using the application risk categories and the respective prescribed assurance program roadmaps for each, a baseline security cost for each application can be initially estimated from the costs associated with the corresponding risk category.

Combine the application-specific cost information with the general cost model based on risk category, and then evaluate projects for outliers, i.e. sums disproportionate to the risk rating. These indicate either an error in risk evaluation/classification or the necessity to tune the organization’s assurance program to address root causes for security cost more effectively.

The tracking of security spend per project should be done quarterly at the assurance program strategy session, and the information should be reviewed and evaluated by stakeholders at least annually. Outliers and other unforeseen costs should be discussed for potential effect on assurance program roadmap.

ASSESSMENT

- Is per-project data for the cost of assurance activities collected?
- Does your organization regularly compare your security spend with that of other organizations?

RESULTS

- Information to make informed case-by- case decisions on security expenditures
- Estimates of past loss due to security issues
- Per-project consideration of security expense versus loss potential
- Industry-wide due diligence with regard to security

SUCCESS METRICS

- >80% of projects reporting security costs in the past three months
- >1 industry-wide cost comparison in the past year
- >1 historic security spend evaluation in the past year

COSTS

- Buildout or license industry intelligence on security programs
- Program overhead from cost estimation, tracking, and evaluation

PERSONNEL

- Architects
- Managers
- Business Owners
- Security Auditor

RELATED LEVELS

- Issue Management - 1

Policy & Compliance

	PC1	PC2	PC3
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
Assessment	✦Do project stakeholders know their project's compliance status? ✦Are compliance requirements specifically considered by project teams?	✦Does the organization utilize a set of policies and standards to control software development? ✦Are project teams able to request an audit for compliance with policies and standards?	✦Are projects periodically audited to ensure a baseline of compliance with policies and standards? ✦Does the organization systematically use audits to collect and control compliance evidence?
Results	✦Increased assurance for handling third-party audit with positive outcome ✦Alignment of internal resources based on priority of compliance requirements ✦Timely discovery of evolving regulatory requirements that affect your organization	✦Awareness for project teams regarding expectations for both security and compliance ✦Business owners that better understand specific compliance risks in their product lines ✦Optimized approach for efficiently meeting compliance with opportunistic security improvement	✦Organization-level visibility of accepted risks due to non-compliance ✦Concrete assurance for compliance at the project level ✦Accurate tracking of past project compliance history ✦Efficient audit process leveraging tools to cut manual effort

Policy & Compliance: PC1

Understand relevant governance and compliance drivers to the organization

Activities

A. Identify and monitor external compliance drivers While an organization might have a wide variety of compliance requirements, this activity is specifically oriented around those that either directly or indirectly affect the way in which the organization builds or uses software and/or data. Leverage internal staff focused on compliance if available.

Based on the organization's core business, conduct research and identify third-party regulatory standards with which compliance is required or considered an industry norm. Possibilities include the Sarbanes-Oxley Act (SOX), the Payment Card Industry Data Security Standards (PCI-DSS), the Health Insurance Portability and Accountability Act (HIPAA), etc. After reading and understanding each third-party standard, collect specific requirements related to software and data and build a consolidated list that maps each driver (third-party standard) to each of its specific requirements for security. At this stage, try to limit the amount of requirements by dropping anything considered optional or only recommended.

At a minimum, conduct research at least biannually to ensure the organization is keeping up-dated on changes to third-party standards. Depending upon the industry and the importance of compliance, this activity can vary in effort and personnel involvement, but should always be done explicitly.

B. Build and maintain compliance guidelines Based upon the consolidated list of software and data-related requirements from compliance drivers, elaborate the list by creating a corresponding response statement to each requirement. Sometimes called control statements, each response should capture the concept of what the organization does to ensure the requirement is met (or to note why it does not apply).

Since typical audit practice often involves checking a control statement for sufficiency and then measuring the organization against the control statement itself, it is critical that they accurately represent actual organizational practices. Also, many requirements can be met by instituting simple, lightweight process elements to cover base-line compliance prior to evolving the organization for better assurance down the road.

Working from the consolidated list, identify major gaps to feed the future planning efforts with regard to building the assurance program. Communicate information about compliance gaps with stakeholders to ensure awareness of the risk from non-compliance.

At a minimum, update and review control statements with stakeholders at least biannually. Depending on the number of compliance drivers, it may make sense to perform updates more often.

ASSESSMENT

- Do project stakeholders know their project's compliance status?
- Are compliance requirements specifically considered by project teams?

RESULTS

- Increased assurance for handling third- party audit with positive outcome
- Alignment of internal resources based on priority of compliance requirements
- Timely discovery of evolving regulatory requirements that affect your organization

SUCCESS METRICS

- >1 compliance discovery meeting in the past six months
- Compliance checklist completed and updated within the past six months
- >1 compliance review meeting with stakeholders in the past six months

COSTS

- Initial creation and ongoing maintenance of compliance checklist

PERSONNEL

- Architects
- Managers
- Business Owners

RELATED LEVELS

- Strategy & Metrics - 1

Policy & Compliance: PC2

Establish security and compliance baseline and understand per-project risks

Activities

A. Build policies and standards for security and compliance Beginning with a current compliance guideline, review regulatory standards and note any optional or recommended security requirements. Also, the organization should conduct a small amount of research to discover any potential future changes in compliance requirements that are relevant.

Augment the list with any additional requirements based on known business drivers for security. Often it is simplest to consult existing guidance being provided to development staff and gather a set of best practices.

Group common/similar requirements and rewrite each group as more generalized/simplified statements that meet all the compliance drivers as well as provide some additional security value. Work through this process for each grouping with the goal of building a set of internal policies and standards that can be directly mapped back to compliance drivers and best practices.

It is important for the set of policies and standards to not contain requirements that are too difficult or excessively costly for project teams to comply. A useful heuristic is that approximately 80% of projects should be able to comply with minimal disruption. This requires a good communications program being set up to advertise the new policies/ standards and assist teams with compliance if needed.

B. Establish project audit practice Create a simple audit process for project teams to request and receive an audit against internal standards. Audits are typically performed by security auditors but can also be conducted by security-savvy staff as long as they are knowledgeable about the internal standards.

Based upon any known business risk indicators, projects can be prioritized concurrently with audit queue triage such that high-risk software is assessed sooner or more frequently. Additionally, low-risk projects can have internal audit requirements loosened to make the audit practice more cost-effective.

Overall, each active project should undergo an audit at least biannually. Generally, subse-quent audits after the initial will be simpler to perform if sufficient audit information about the application is retained.

Advertise this service to business owners and other stakeholders so that they may request an audit for their projects. Detailed pass/fail results per requirement from the internal standards should be delivered to project stakeholders for evaluation. Where practical, audit results should also contain explanations of impact and remediation recommendations.

ASSESSMENT

- Does the organization utilize a set of policies and standards to control software development?
- Are project teams able to request an audit for compliance with policies and standards?

RESULTS

- Awareness for project teams regarding expectations for both security and compliance
- Business owners that better understand specific compliance risks in their product lines
- Optimized approach for efficiently meeting compliance with opportunistic security improvement

SUCCESS METRICS

- >75% of staff briefed on policies and standards in the past six months
- >80% stakeholders aware of compliance status against policies and standards

COSTS

- Internal standards buildout or license
- Per-project overhead from compliance with internal standards and audit

PERSONNEL

- Architects
- Managers
- Security Auditors

RELATED LEVELS

- Strategy & Metrics - 2
- Education & Guidance - 1 & 3

- Security Requirements - 1 & 3
- Secure Architecture - 3
- Design Review - 3
- Implementation Review - 3
- Environment Hardening - 3

Policy & Compliance: PC3

Require compliance and measure projects against organization-wide policies and standards

Activities

A. Create compliance gates for projects Once an organization has established internal standards for security, the next level of en-forcement is to set particular points in the project lifecycle where a project cannot pass until it is audited against the internal standards and found to be in compliance.

Usually, the compliance gate is placed at the point of software release such that they are not allowed to publish a release until the compliance check is passed. It is important to provide enough time for the audit to take place and remediation to occur, so generally the audit should begin earlier, for instance when a release is given to quality assurance.

Despite being a firm compliance gate, legacy or other specialized projects may not be able to comply, so an exception approval process must also be created. No more than about 20% of all projects should have exception approval.

B. Adopt solution for audit data collection Organizations conducting regular audits of project teams generate a large amount of audit data over time. Automation should be utilized to assist in automated collection, manage collation for storage and retrieval, and to limit individual access to sensitive audit data.

For many concrete requirements from the internal standards, existing tools such as code analyzers, application penetration testing tools, monitoring software, etc. can be customized and leveraged to automate compliance checks against internal standards. The purpose of automating compliance checks is to both improve efficiency of audit as well as enable more staff to self-check for compliance before a formal audit takes place. Additionally, automated checks are less error-prone and allow for lower latency on discovery of problems.

Information storage features should allow centralized access to current and historic audit data per project. Automation solutions must also provide detailed access control features to limit access to approved individuals with valid business purpose for accessing the audit data.

All instructions and procedures related to accessing compliance data as well as requesting access privileges should be advertised to project teams. Additional time may be initially required from security auditors to bootstrap project teams.

ASSESSMENT

- Are projects periodically audited to ensure a baseline of compliance with policies and standards?
- Does the organization systematically use audits to collect and control compliance evidence?

RESULTS

- Organization-level visibility of accepted risks due to non-compliance
- Concrete assurance for compliance at the project level
- Accurate tracking of past project compliance history
- Efficient audit process leveraging tools to cut manual effort

SUCCESS METRICS

- >80% projects in compliance with policies and standards as seen by audit
- <50% time per audit as compared to manual

COSTS

- Buildout or license tools to automate audit against internal standards
- Ongoing maintenance of audit gates and exception process

PERSONNEL

- Developers
- Architects
- Managers

RELATED LEVELS

- Education & Guidance - 3
- Implementation Review - 2
- Security Testing - 2

Education & Guidance

	EG1	EG2	EG3
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
Assessment	✦Have developers been given high-level security awareness training? ✦Does each project team understand where to find secure development best-practices and guidance?	✦Are those involved in the development process given role-specific security training and guidance? ✦Are stakeholders able to pull in security coaches for use on projects?	✦Is security-related guidance centrally controlled and consistently distributed throughout the organization? ✦Are developers tested to ensure a baseline skill-set for secure development practices?
Results	✦Increased developer awareness on the most common problems at the code level ✦Maintain software with rudimentary security best-practices in place ✦Set baseline for security know-how among technical staff ✦Enable qualitative security checks for baseline security knowledge	✦End-to-end awareness of the issues that leads to security vulnerabilities at the product, design, and code levels ✦Build plans to remediate vulnerabilities and design flaws in ongoing projects ✦Enable qualitative security checkpoints at requirements, design, and development stages ✦Deeper understanding of security issues encourages more proactive security planning	✦Efficient remediation of vulnerabilities in both ongoing and legacy code bases ✦Quickly understand and mitigate against new attacks and threats ✦Measure the amount of security knowledge of the staff and measure against a common standard ✦Establish fair incentives toward security awareness

Education & Guidance: EG1

Offer development staff access to resources around the topics of secure programming and deployment

Activities

A. Conduct technical security awareness training Either internally or externally sourced, conduct security training for technical staff that covers the basic tenets of application security. Generally, this can be accomplished via instructor-led training in 1-2 days or via computer-based training with modules taking about the same amount of time per developer.

Course content should cover both conceptual and technical information. Appropriate topics include high-level best practices surrounding input validation, output encoding, error handling, logging, authentication, authorization, and data protection. Additional coverage of commonplace software vulnerabilities is also desirable such as a Top 10 list appropriate to the software being devel-oped (web applications, embedded devices, client- server applications, back-end transaction systems, etc.). Wherever possible, use code samples and lab exercises in the specific pro-gramming language(s) that applies.

To rollout such training, it is recommended to mandate annual security training and then hold courses (either instructor-led or computer-based) as often as required based on development head-count.

B. Build and maintain technical guidelines For development staff, assemble a list of approved documents, web pages, and technical notes that provide technology-specific security advice. These references can be assembled from many publicly available resources on the Internet. In cases where very specialized or pro- prietary technologies permeate the development environment, utilize senior, security-savvy staff to build security notes over time to create such a knowledge base in an ad hoc fashion.

Ensure management is aware of the resources and briefs oncoming staff about their expected usage. Try to keep the guidelines lightweight and up-to-date to avoid clutter and irrelevance. Once a comfort-level has been established, they can be used as a qualitative checklist to ensure that the guidelines have been read, understood, and followed in the development process.

ASSESSMENT

- Have developers been given high-level security awareness training?
- Does each project team understand where to find secure development best-practices and guidance?

RESULTS

- Increased developer awareness on the most common problems at the code level
- Maintain software with rudimentary security best-practices in place
- Set baseline for security know-how among technical staff
- Enable qualitative security checks for baseline security knowledge

SUCCESS METRICS

- >50% development staff briefed on security issues within the past year
- >75% senior development/ architect staff briefed on security issues within the past year
- Launch technical guidance within three months of first training

COSTS

- Training course buildout or license
- Ongoing maintenance of technical guidance

PERSONNEL

- Developers
- Architects

RELATED LEVELS

- Policy & Compliance - 2
- Security Requirements - 1
- Secure Architecture - 1

Education & Guidance: EG2

Educate all personnel in the software lifecycle with role-specific guidance on secure development

Activities

A. Conduct role-specific application security training Conduct security training for staff that highlights application security in the context of each role's job function. Generally, this can be accomplished via instructor-led training in 1-2 days or via computer-based training with modules taking about the same amount of time per person.

For managers and requirements specifiers, course content should feature security requirements planning, vulnerability and incident management, threat modeling, and misuse/abuse case design.

Tester and auditor training should focus on training staff to understand and more effectively analyze software for security-relevant issues. As such, it should feature techniques for code review, architecture and design analysis, runtime analysis, and effective security test planning.

Expand technical training targeting developers and architects to include other relevant topics such as security design patterns, tool-specific training, threat modeling and software assessment techniques.

To rollout such training, it is recommended to mandate annual security awareness training and periodic specialized topics training. Course should be available (either instructor-led or computer-based) as often as required based on head-count per role.

B. Utilize security coaches to enhance project teams Using either internal or external experts, make security-savvy staff available to project teams for consultation. Further, this coaching resource should be advertised internally to ensure that staff are aware of its availability. The coaching staff can be created by recruiting experienced individuals within the organization to spend some percentage of their time, around 10% maximum, performing coaching activities. The coaches should communicate between one another to ensure they are aware of each other's area of expertise and route questions accordingly for efficiency.

While coaches can be used at any point in the software lifecycle, appropriate times to use the coaches include during initial product conception, before completion of functional or detailed design specification(s), when issues arise during development, test planning, and when operational security incidents occur.

Over time, the internal network of coaching resources can be used as points-of-contact for communicating security-relevant information throughout the organization as well as being local resources that have greater familiarity with the ongoing project teams than a purely centralized security team might.

ASSESSMENT

- Are those involved in the development process given role-specific security training and guidance?
- Are stakeholders able to pull in security coaches for use on projects?

RESULTS

- End-to-end awareness of the issues that leads to security vulnerabilities at the product, design, and code levels
- Build plans to remediate vulnerabilities and design flaws in ongoing projects
- Enable qualitative security checkpoints at requirements, design, and development stages
- Deeper understanding of security issues encourages more proactive security planning

SUCCESS METRICS

- >60% development staff trained within the past year
- >50% management/analyst staff trained within the past year
- >80% senior development/architect staff trained within the past year
- >3.0 Likert Scale on usefulness of training courses

COSTS

- Training library build-out or license
- Security-savvy staff for hands-on coaching

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- QA Testers
- Security Auditors

RELATED LEVELS

- Secure Architecture - 2
- Design Review - 2
- Issue Management - 1

Education & Guidance: EG3

Mandate comprehensive security training and certify personnel for baseline knowledge

Activities

A. Create formal application security support portal Building upon written resources on topics relevant to application security, create and advertise a centralized repository (usually an internal web site). The guidelines themselves can be created in any way that makes sense for the organization, but an approval board and straightforward change control processes must be established.

Beyond static content in the form of best-practices lists, tool-specific guides, FAQs, and other articles, the support portal should feature interactive components such as mailing lists, web- based forums, or wikis to allow internal resources to cross-communicate security relevant topics and have the information cataloged for future reference.

The content should be cataloged and easily searchable based upon several common factors such as platform, programming language, pertinence to specific third-party libraries or frameworks, lifecycle stage, etc. Project teams creating software should align themselves early in product development to the specific guidelines that they will follow. In product as-sessments, the list of applicable guidelines and product-related discussions should be used as audit criteria.

B. Establish role-based examination/certification Either per role or per training class/module, create and administer aptitude exams that test people for comprehension and utilization of security knowledge. Typically, exams should be created based on the role-based curricula and target a minimum passing score around 75% correct. While staff should be required to take applicable training or refresher courses an- nually, certification exams should be required biannually at a minimum.

Based upon pass/fail criteria or exceptional performance, staff should be ranked into tiers such that other security-related activities could require individuals of a particular certification level to sign-off before the activity is complete, e.g. an uncertified developer cannot pass a design into implementation without explicit approval from a certified architect. This provides granular visibility on a per-project basis for tracking security decisions with individual accountability. Overall, this provides a foundation for rewarding or penalizing staff for making good business decisions regarding application security.

ASSESSMENT

- Is security-related guidance centrally controlled and consistently distributed throughout the organization?
- Are developers tested to ensure a baseline skill-set for secure development practices?

RESULTS

- Efficient remediation of vulnerabilities in both ongoing and legacy code bases
- Quickly understand and mitigate against new attacks and threats
- Judge security-savvy of staff and measure against a common standard
- Establish fair incentives toward security awareness

SUCCESS METRICS

- >80% staff certified within the past year

COSTS

- Certification examination build-out or license
- Ongoing maintenance and change control for application security support portal
- Human-resources and overhead cost for implementing employee certification

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- QA Testers
- Security Auditors

RELATED LEVELS

- Policy & Compliance - 2 & 3

Security Culture

	SC1	SC2	SC3
Objective	Establish first elements of a security culture.	Refinement of the security culture.	Regular assessment and continuous enhancement of the security culture.
Activities	A. Establish communication B. Creating values diversity	A. Giving culture commitment B. Acting as a team and apply lifelong learning	A. Culture benchmarking B. Culture retrospective and reflection
Assessment	✦Does your organization have an open discussion about organizational deficiencies, human faults and Intentional activities? ✦Are diversity principles used in the organization? ✦Do employees consider security a high-priority issue?	✦Is a value development established? ✦Is lifelong learning established in the organization? ✦Is a fundamental collaboration visible in the organization?	✦Is a customized security culture benchmarking (assessment) established? ✦Do teams perform value retrospective and reflection of the security culture? ✦Is a mature collaboration visible in the organization?
Results	✦Established fault reporting regarding development process and product ✦Concrete list of organizational deficiencies, human faults and intentional activities ✦Diversity in profile of team members and diversity in systems architecture (hardware, software)	✦Awareness of staff members regarding values ✦Customized plans for lifelong learning ✦Approach for development of collaboration within the organization ✦Accurate tracking of lifelong learning activities ✦Organization-wide participation in a reward system	✦Regular Assessment report of the security culture ✦Established retrospective process ✦Regular measurement of team collaboration

Security Culture: SC1

Establish first elements of a security culture

Activities

A. Establish communication An open discussion about organizational personal factors in security involves organizational deficiencies, human faults and intentional activities. The team characteristic can be described as working group or pseudo team.

There is an individual artifacts integration. The collaboration maturity model (Col_MM) Collaboration Maturity Model (Col_MM) assesses organization's team collaboration maturity: The collaboration characteristic based on Col_MM is, e.g., low interdependence, the collaboration management based on Col_MM is decisions made randomly, the collaboration process based on Col_MM is missing conflict handling, and the information and knowledge integration based on Col_MM is randomly knowledge generation.

The lessons learned form the penetration tests have to be evaluated regarding the effectiveness in order to review the vulnerabilities with the team and to make sure your incident response plan covers the uncovered vulnerabilities.

B. Creating values diversity The motivation of the team members helps to grow up a mindset of security awareness. Security is acknowledged as a high-level value. The team uses diversity in the development process regarding, e.g. in different opinions in review meetings. There is a proactive attitude regarding handling vulnerabilities and system hardening.

A positive concept of human being is settled. Social awareness is sporadic individual, but not organization wide.

The team creates security values—like good design, face-to-face meetings, motivated individuals, velocity—in Schneider's model using the topics of culture, collaboration, control and competency.

The privacy perception is described by empowerment of privacy awareness.

ASSESSMENT

- Does your organization have an open discussion about organizational deficiencies, human faults and intentional activities?
- Are diversity principles used in the organization?
- Do employees consider security a high-priority issue?

RESULTS

- Established fault reporting regarding development process and product
- Concrete list of organizational deficiencies, human faults and intentional activities
- Diversity in profile of team members and diversity in systems architecture (hardware, software)

SUCCESS METRICS

- Compliance discovery in past 6 months
- Collaboration characteristic based on Col_MM questionnaire
- Security values in Schneider's model
- Compliance checklist completed within the past 6 months
- Compliance review meeting with stakeholders in past 6 months

COSTS

- Initial creation and adoption of Col_MM questionnaire
- Initial creation of security values in Schneider’s model

PERSONNEL

- Architects
- Business Owners
- Developers
- Managers
- Security Auditors
- QA-Testers

RELATED LEVELS

- Policy & Compliance – 1

Security Culture: SC2

Refinement of the security culture

Activities

A. Giving culture commitment An advanced value development starts with a commitment to security. It comprises involvement (employees take extra effort and voice their concerns regarding security even when it is not their direct responsibility), personal responsibility (employees understand that not only IT/ Information security department is responsible for security assurance; thus, they take responsibility for acting security), and impact – my actions matter (employees understand how their actions and decisions affect the “bigger security picture”: they see the connection between their everyday work and security).

The social awareness is organization wide. The white hat attitude as penetration tester can be described as finding vulnerabilities by performing attacks on systems based on a contract.

Trust is established, because the team assesses the perceptions of users regarding the safekeeping of private information and their trust in the communications of the organization.

B. Acting as a team and apply lifelong learning A lifelong learning process is applied, which establishes self-controlled learning with constructivism-based methods, e.g., self-controlled learning. The team establishes a reward system for security. Team characteristic can be categorized as potential team or real team.

The collaboration characteristic based on Col_MM is non regular strong interactions, the collaboration management based on Col_MM is decisions made reactive, the collaboration process based on Col_MM is hierarchical conflict interventions, and the information and knowledge integration based on Col_MM is individual knowledge generation.

ASSESSMENT

- Is a value development established?
- Is lifelong learning established in the organization?
- Is a fundamental collaboration visible in the organization?

RESULTS

- Awareness of staff members regarding values
- Customized plans for lifelong learning
- Approach for development of collaboration within the organization
- Accurate tracking of lifelong learning activities
- Organization-wide participation in a reward system

SUCCESS METRICS

- >75% of the staff is involved in the security culture

COSTS

- Performing the Col_MM questionnaire evaluation and its analysis
- Overhead of accurate tracking of lifelong learning activities
- Performing an organization-wide participation in a security reward system

PERSONNEL

- Architects
- Business Owners
- Developers
- Managers
- Security Auditors
- QA-Testers

RELATED LEVELS

- Policy & Compliance – 2

Security Culture: SC3

Regular assessment and continuous enhancement of the security culture

Activities

A. Culture benchmarking The organization performs a security culture benchmarking by using a security culture questionnaire. The team characteristic is categorized as a “high efficiency team”. There is a tactic knowledge sharing with staged communication distribution (depends on domain driven strategies e.g. automotive business).

The collaboration characteristic based on Col_MM is regular strong interactions, the collaboration management based on Col_MM is decisions made proactive and collective with consensus, the collaboration process based on Col_MM is conflicts solved collectively, and the information and knowledge integration based on Col_MM is knowledge sharing in a co-creation manner.

B. Culture retrospective and reflection There is a value retrospective and reflection by adopting and modifying the organization security values with Schneider‘s model. The organization has independent penetration testers. The process uses external strategies and auditors for penetration testing. The penetration testers have a consciousness about their own bias view.

ASSESSMENT

- Is a customized security culture benchmarking (assessment) established?
- Do teams perform value retrospective and reflection of the security culture?
- Is a mature collaboration visible in the organization?

RESULTS

- Regular Assessment report of the security culture
- Established retrospective process
- Regular measurement of team collaboration

SUCCESS METRICS

- All projects have a high maturity level in the Col_MM
- All projects use security values in Schneider’s model
- The organization has a well performed value retrospective and reflection process.

COSTS

- Performing the Col_MM questionnaire evaluation and its analysis
- Overhead of continuous improvement of security values in Schneider’s model
- Performing value retrospective and reflection

PERSONNEL

- Architects
- Business Owners
- Developers
- Managers
- Security Auditors
- QA-Testers

RELATED LEVELS

- Policy & Compliance – 3

Threat Assessment

	TA1	TA2	TA3
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 align 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
Assessment	✦Do projects in your organization consider and document likely threats? ✦Does your organization understand and document the types of attackers it faces?	✦Do project teams regularly analyze functional requirements for likely abuses? ✦Do project teams use a method of rating threats for relative comparison? ✦Are stakeholders aware of relevant threats and ratings?	✦Do project teams specifically consider risk from external software? ✦Are the majority of the protection mechanisms and controls captured and mapped back to threats?
Results	✦High-level understanding of factors that may lead to negative outcomes ✦Increased awareness of threats amongst project teams ✦Inventory of threats for your organization	✦Granular understanding of likely threats to individual projects ✦Framework for better tradeoff decisions within project teams ✦Ability to prioritize development efforts within a project team based on risk weighting	✦Deeper consideration of full threat profile for each software project ✦Detailed mapping of assurance features to established threats against each software project ✦Artifacts to document due diligence based on business function of each software project

Threat Assessment: TA1

Identify and understand high-level threats to the organization and individual projects

Activities

A. Build and maintain application-specific threat models Based purely on the business purpose of each software project and the business risk profile (if available) identify likely worst-case scenarios for the software under development in each project team. This can be conducted using simple attack trees or through a more formal threat modeling process such as Microsoft's STRIDE, Trike, etc.

To build attack trees, identify each worst-case scenario in one sentence and label these as the high-level goals of an attacker. From each attacker goal identified, identify preconditions that must hold in order for each goal to be realized. This information should be captured in branches underneath each goal where each branch is either a logical AND or a logical OR of the statements contained underneath. An AND branch indicates that each directly attached child nodes must be true in order to realize the parent node. An OR branch indicates that any one of the directly attached child nodes must be true in order to achieve the parent node.

Regardless of the threat modeling approach, review each current and historic functional requirement to augment the attack tree to indicate security failures relevant to each. Brain- storm by iteratively dissecting each failure scenario into all the possible ways in which an attacker might be able to reach one of the goals. After initial creation, the threat model for an application should be updated when significant changes to the software are made. This assessment should be conducted with senior developers and architects as well as one or more security auditors.

B. Develop attacker profile rom software architecture Initially, conduct an assessment to identify all likely threats to the organization based on software projects. For this assessment, consider threats to be limited to agents of malicious intent and omit other risks such as known vulnerabilities, potential weaknesses, etc.

Begin by generally considering external agents and their corresponding motivations for attack. To this list, add internal roles that could cause damage and their motivations for insider attack. Based on the architecture of the software project(s) under consideration, it can be more efficient to conduct this analysis once per architecture type instead of for each project individually since applications of architecture and business purpose will generally be suscep-tible to similar threats.

This assessment should be conducted with business owners and other stakeholders but also include one or more security auditors for additional perspective on threats. In the end, the goal is to have a concise list of threat agents and their corresponding motivations for attack.

ASSESSMENT

- Do projects in your organization consider and document likely threats?
- Does your organization understand and document the types of attackers it faces?

RESULTS

- High-level understanding of factors that may lead to negative outcomes
- Increased awareness of threats amongst project teams
- Inventory of threats for your organization

SUCCESS METRICS

- >50% of project stakeholders briefed on the threat models of relevant projects within the past 12 months
- >75% of project stakeholders briefed on attacker profiles for relevant architectures

COSTS

- Buildout and maintenance of project artifacts for threat models

PERSONNEL

- Business Owners
- Developers
- Architects
- Security Auditors
- Managers

RELATED LEVELS

- Strategy & Metrics - 1
- Security Requirements - 2

Threat Assessment: TA2

Increase accuracy of threat assessment and improve granularity of per-project understanding

Activities

A. Build and maintain abuse-case models per project Further considering the threats to the organization, conduct a more formal analysis to determine potential misuse or abuse of functionality. Typically, this process begins with identification of normal usage scenarios, e.g. use-case diagrams if available.

If a formal abuse-case technique isn't used, generate a set of abuse-cases for each scenario by starting with a statement of normal usage and brainstorming ways in which the statement might be negated, in whole or in part. The simplest way to get started is to insert the word "no" or "not" into the usage statement in as many ways as possible, typically around nouns and verbs. Each usage scenario should generate several possible abuse-case statements.

Further elaborate the abuse-case statements to include any application-specific concerns based on the business function of the software. The ultimate goal is for the completed set of abuse statements to form a model for usage patterns that should be disallowed by the software. If desired, these abuse cases can be combined with existing threat models.

After initial creation, abuse-case models should be updated for active projects during the design phase. For existing projects, new requirements should be analyzed for potential abuse, and existing projects should opportunistically build abuse-cases for established functionality where practical.

B. Adopt a weighting system for measurement of threats Based on the established attacker profiles, identify a rating system to allow relative compari- son between the threats. Initially, this can be a simple high-medium-low rating based upon business risk, but any scale can be used provided that there are no more than five categories.

After identification of a rating system, build evaluation criteria that allow each threat to be assigned a rating. In order to do this properly, additional factors about each threat must be considered beyond motivation. Important factors include capital and human resources, inherent access privilege, technical ability, relevant goals on the threat model(s), likelihood of successful attack, etc.

After assigning each threat to a rating, use this information to prioritize risk mitigation activities within the development lifecycle. Once built for a project team, it should be updated during design of new features or refactoring efforts.

ASSESSMENT

- Do project teams regularly analyze functional requirements for likely abuses?
- Do project teams use a method of rating threats for relative comparison?
- Are stakeholders aware of relevant threats and ratings?

RESULTS

- Granular understanding of likely threats to individual projects
- Framework for better tradeoff decisions within project teams
- Ability to prioritize development efforts within a project team based on risk weighting

SUCCESS METRICS

- >75% of project teams with identified and rated threats
- >75% of project stakeholders briefed on threat and abuse models of relevant projects within the past six months

COSTS

- Project overhead from maintenance of threat models and attacker profiles

PERSONNEL

- Security Auditor
- Business Owner
- Managers

RELATED LEVELS

- Strategy & Metrics - 2
- Secure Architecture - 2

Threat Assessment: TA3

Concretely tie compensating controls to each threat against internal and third-party software

Activities

A. Explicitly evaluate risk from third-party components Conduct an assessment of your software code-base and identify any components that are of external origin. Typically, these will include open-source projects, purchased consumer of the shelf (COTS) software, and online services which your software uses.

For each identified component, elaborate attacker profiles for the software project based upon potential compromise of third-party components. Based upon the newly identified attacker profiles, update software threat models to incorporate any likely risks based upon new attacker goals or capabilities.

In addition to threat scenarios, also consider ways in which vulnerabilities or design flaws in the third-party software might affect your code and design. Elaborate your threat models accordingly with the potential risks from vulnerabilities and knowledge of the updated attacker profile.

After initially conducted for a project, this must be updated and reviewed during the design phase or every development cycle. This activity should be conducted by a security auditor with relevant technical and business stakeholders.

B. Elaborate threat models with compensating controls Conduct an assessment to formally identify factors that directly prevent preconditions for compromise represented by the threat models. These mitigating factors are the compensat- ing controls that formally address the direct risks from software. Factors can be technical features in the software itself, but can also be process elements in the development lifecycle, infrastructure features, etc.

If using attack trees, the logical relationship represented by each branch will be either an AND or an OR. Therefore, by mitigating against just one precondition on an AND branch, the parent and all connected leaf nodes can be marked as mitigated. However, all child nodes on an OR node must be prevented before the parent can be marked as mitigated.

Regardless of threat modeling technique, identify compensating controls and annotate the threat models directly. The goal is to maximize coverage in terms of controls that mark parts of the threat model as mitigated. For any viable paths remaining, identify potential compensating controls for feedback into organizational strategy.

After initially conducted for a project, this must be updated and reviewed during the design phase or every development cycle. This activity should be conducted by a security auditor with relevant technical and business stakeholders.

ASSESSMENT

- Do project teams specifically consider risk from external software?
- Are the majority of the protection mechanisms and controls captured and mapped back to threats?

RESULTS

- Deeper consideration of full threat profile for each software project
- Detailed mapping of assurance features to established threats against each software project
- Artifacts to document due diligence based on business function of each software project

SUCCESS METRICS

- >80% of project teams with updated threat models prior to every implementation cycle
- >80% of project teams with updated inventory of third-party components prior to every release
- >50% of all security incidents identified a priori by threat models in the past 12 months

COSTS

- Project overhead from maintenance of detailed threat models and expanded attacker profiles
- Discovery of all third-party dependencies

PERSONNEL

- Business Owners
- Developers
- Architects
- Security Auditors
- Managers

RELATED LEVELS

- Security Requirements - 2 & 3

Security Requirements

	SR1	SR2	SR3
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	A. Derive security requirements from business functionality B. Evaluate security and compliance guidance for requirements	A. Build an access control matrix for resources and capabilities B. Specify security requirements based on known risks	A. Build security requirements into supplier agreements B. Expand audit program for security requirements
Assessment	✦Do project teams specify security requirements during development? ✦Do project teams pull requirements from best practices and compliance guidance?	✦Do stakeholders review access control matrices for relevant projects? ✦Do project teams specify requirements based on feedback from other security activities?	✦Do stakeholders review vendor agreements for security requirements? ✦Are audits performed against the security requirements specified by project teams?
Results	✦High-level alignment of development effort with business risks ✦Ad hoc capturing of industry best-practices for security as explicit requirements ✦Awareness amongst stakeholders of measures being taken to mitigate risk from software	✦Detailed understanding of attack scenarios against business logic ✦Prioritized development effort for security features based on likely attacks ✦More educated decision-making for tradeoffs between features and security efforts ✦Stakeholders that can better avoid functional requirements that inherently have security flaws	✦Formally set baseline for security expectations from external code ✦Centralized information on security effort undertaken by each project team ✦Ability to align resources to projects based on application risk and desired security requirements

Security Requirements: SR1

Consider security explicitly during the software requirements process

Activities

A. Derive security requirements from business functionality Conduct a review of functional requirements that specify the business logic and overall behavior for each software project. After gathering requirements for a project, conduct an assessment to derive relevant security requirements. Even if software is being built by a third-party, these requirements, once identified, should be included with functional requirements delivered to vendors.

For each functional requirement, a security auditor should lead stakeholders through the process of explicitly noting any expectations with regard to security. Typically, questions to clarify for each requirement include expectations for data security, access control, transaction integrity, criticality of business function, separation of duties, uptime, etc.

It is important to ensure that all security requirements follow the same principles for writing good requirements in general. Specifically, they should be specific, measurable, and reasonable. Conduct this process for all new requirements on active projects. For existing features, it is recommended to conduct the same process as a gap analysis to fuel future refactoring for security.

B. Evaluate security and compliance guidance for requirements Determine industry best-practices that project teams should treat as requirements. These can be chosen from publicly available guidelines, internal or external guidelines/standards/ policies, or established compliance requirements.

It is important to not attempt to bring in too many best-practice requirements into each development iteration since there is a time trade-off with design and implementation. The recommended approach is to slowly add best-practices over successive development cycles to bolster the software's overall assurance profile over time.

For existing systems, refactoring for security best practices can be a complex undertaking. Where possible, add security requirements opportunistically when adding new features. At a minimum, conducting the analysis to identify applicable best practices should be done to help fuel future planning efforts.

This review should be performed by a security auditor with input from business stakeholders. Senior developers, architects, and other technical stakeholders should also be involved to bring design and implementation-specific knowledge into the decision process.

ASSESSMENT

- Do project teams specify security requirements during development?
- Do project teams pull requirements from best practices and compliance guidance?

RESULTS

- High-level alignment of development effort with business risks
- Ad hoc capturing of industry best-practices for security as explicit requirements
- Awareness amongst stakeholders of measures being taken to mitigate risk from software

SUCCESS METRICS

- >50% of project teams with explicitly defined security requirements

COSTS

- Project overhead from addition of security requirements to each development cycle

PERSONNEL

- Security Auditor
- Business Owners
- Managers
- Architects

RELATED LEVELS

- Policy & Compliance - 2
- Education & Guidance - 1
- Design Review - 1
- Implementation Review - 1
- Security Testing - 1

Security Requirements: SR2

Increase granularity of security requirements derived from business logic and known risks

Activities

A. Build an access control matrix for resources and capabilities Based upon the business purpose of the application, identify user and operator roles. Additionally, build a list of resources and capabilities by gathering all relevant data assets and application-specific features that are guarded by any form of access control.

In a simple matrix with roles on one axis and resources on the other, consider the relationships between each role and each resource and note in each intersection the correct behavior of the system in terms of access control according to stakeholders.

For data resources, it is important to note access rights in terms of creation, read access, update, and deletion. For resources that are features, gradation of access rights will likely be application-specific, but at a minimum, note if the role should be permitted access to the feature.

This permission matrix will serve as an artifact to document the correct access control rights for the business logic of the overall system. As such, it should be created by the project teams with input from business stakeholders. After initial creation, it should be updated by business stakeholders before every release, but usually toward the beginning of the design phase.

B. Specify security requirements based on known risks Explicitly review existing artifacts that indicate organization or project-specific security risk in order to better understand the overall risk profile for the software. When available, draw on resources such as the high-level business risk profile, individual application threat models, findings from design review, code review, security testing, etc.

In addition to review of existing artifacts, use abuse-case models for an application to serve as fuel for identification of concrete security requirements that directly or indirectly mitigate the abuse scenarios. This process should be conducted by business owners and security auditors as needed.

Ultimately, the notion of risks leading to new security requirements should become a built-in step in the planning phase whereby newly discovered risks are specifically assessed by project teams.

ASSESSMENT

- Do stakeholders review access control matrices for relevant projects?
- Do project teams specify requirements based on feedback from other security activities?

RESULTS

- Detailed understanding of attack scenarios against business logic
- Prioritized development effort for security features based on likely attacks
- More educated decision-making for tradeoffs between features and security efforts
- Stakeholders that can better avoid functional requirements that inherently have security flaws

SUCCESS METRICS

- >75% of all projects with updated abuse-case models within past six months

COSTS

- Project overhead from buildout and maintenance of abuse-case models

PERSONNEL

- Security Auditor
- Managers
- Architects
- Business Owners

RELATED LEVELS

- Strategy & Metrics - 1
- Threat Assessment - 1 & 3

Security Requirements: SR3

Mandate security requirements process for all software projects and third-party dependencies

Activities

A. Build security requirements into supplier agreements Beyond the kinds of security requirements already identified by previous analysis, additional security benefits can be derived from third-party agreements. Typically, requirements and perhaps high-level design will be developed internally while detailed design and implementation is often left up to suppliers.

Based on the specific division of labor for each externally developed component, identify specific security activities and technical assessment criteria to add to the vendor contracts. Commonly, this is a set of activities from the design review, code review, and security testing practices.

Modifications of agreement language should be handled on a case-by-case basis with each supplier since adding additional requirements will generally mean an increase in cost. The cost of each potential security activity should be balanced against the benefit of the activity as per the usage of the component or system being considered.

B. Expand audit program for security requirements Incorporate checks for completeness of security requirements into routine project audits. Since this can be difficult to gauge without project-specific knowledge, the audit should focus on checking project artifacts such as requirements or design documentation for evidence that the proper types of analysis were conducted.

Particularly, each functional requirement should be annotated with security requirements based on business drivers as well as expected abuse scenarios. The overall project requirements should contain a list of requirements generated from best-practices in guidelines and standards. Additionally, there should be a clear list of unfulfilled security requirements and an estimated timeline for their provision in future releases.

This audit should be performed during every development iteration, ideally toward the end of the requirements process, but it must be performed before a release can be made.

ASSESSMENT

- Do stakeholders review vendor agreements for security requirements?
- Are audits performed against the security requirements specified by project teams?

RESULTS

- Formally set baseline for security expectations from external code
- Centralized information on security effort undertaken by each project team
- Ability to align resources to projects based on application risk and desired security requirements

SUCCESS METRICS

- >80% of projects passing security requirements audit in past six months
- >80% of vendor agreements analyzed for contractual security requirements in the past 12 months

COSTS

- Increased cost from outsourced development from additional security requirements
- Ongoing project overhead from release gates for security requirements

PERSONNEL

- Security Auditor
- Managers
- Business Owners

RELATED LEVELS

- Policy & Compliance - 2
- Threat Assessment - 3

Secure Architecture

	SA1	SA2	SA3
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	A. Maintain list of recommended software frameworks B. Explicitly apply security principles to design	A. Identify and promote security services and infrastructure B. Identify security design patterns from architecture	A. Establish formal reference architectures and platforms B. Validate usage of frameworks, patterns, and platforms
Assessment	✦Are project teams provided with a list of recommended third-party components? ✦Are project teams aware of secure design principles and do they apply them consistently?	✦Do you advertise shared security services with guidance for project teams? ✦Are project teams provided with prescriptive design patterns based on their application architecture?	✦Do project teams build software from centrally-controlled platforms and frameworks? ✦Are project teams audited for the use of secure architecture components?
Results	✦Ad hoc prevention of unexpected dependencies and one-off implementation choices ✦Stakeholders aware of increased project risk due to libraries and frameworks chosen ✦Established protocol within development for proactively applying security mechanisms to a design	✦Detailed mapping of assets to user roles to encourage better compartmentalization in design ✦Reusable design building blocks for provision of security protections and functionality ✦Increased confidence for software projects from use of established design techniques for security	✦Customized application development platforms that provide built-in security protections ✦Organization-wide expectations for proactive security effort in development ✦Stakeholders better able to make tradeoff decisions based on business need for secure design

Secure Architecture: SA1

Insert consideration of proactive security guidance into the software design process

Activities

A. Maintain list of recommended software frameworks Across software projects within the organization identify commonly used third-party software libraries and frameworks in use. Generally, this need not be an exhaustive search for dependencies, but rather focus on capturing the high-level components that are most often used.

From the list of components, group them into functional categories based on the core features provided by the third-party component. Also, note the usage prevalence of each component across project teams to weight the reliance upon the third-party code. Using this weighted list as a guide, create a list of components to be advertised across the development organization as recommended components.

Several factors should contribute to decisions for inclusion on the recommended list. Although a list can be created without conducting research specifically, it is advisable to inspect each for incident history, track record for responding to vulnerabilities, appropriateness of functionality for the organization, excessive complexity in usage of the third-party component, etc.

This list should be created by senior developers and architects, but also include input from managers and security auditors. After creation, this list of recommended components matched against functional categories should be advertised to the development organization. Ultimately, the goal is to provide well-known defaults for project teams.

B. Explicitly apply security principles to design During design, technical staff on the project team should use a short list of guiding security principles as a checklist against detailed system designs. Typically, security principles include defense in depth, securing the weakest link, use of secure defaults, simplicity in design of security functionality, secure failure, balance of security and usability, running with least privilege, avoidance of security by obscurity, etc.

In particular for perimeter interfaces, the design team should consider each principle in the context of the overall system and identify features that can be added to bolster security at each such interface. Generally, these should be limited such that they only take a small amount of extra effort beyond the normal implementation cost of functional requirements and anything larger should be noted and scheduled for future releases.

While this process should be conducted by each project team after being trained with security awareness, it is helpful to incorporate more security-savvy staff to aid in making design decisions.

ASSESSMENT

- Are project teams provided with a list of recommended third-party components?
- Are project teams aware of secure design principles and do they apply them consistently?

RESULTS

- Ad hoc prevention of unexpected dependencies and one-off implementation choices
- Stakeholders aware of increased project risk due to libraries and frameworks chosen
- Established protocol within development for proactively applying security mechanisms to a design

SUCCESS METRICS

- >80% of development staff briefed on software framework recommendations in the past year
- >50% of projects self-reporting application of security principles to design

COSTS

- Buildout, maintenance, and awareness of software framework recommendations
- Ongoing project overhead from analysis and application of security principles

PERSONNEL

- Architects
- Developers
- Security Auditors
- Managers

RELATED LEVELS

- Education & Guidance - 1

Secure Architecture: SA2

Direct the software design process toward known-secure services and secure-by-default designs

Activities

A. Maintain list of recommended software frameworks Across software projects within the organization identify commonly used third-party software libraries and frameworks in use. Generally, this need not be an exhaustive search for dependencies, but rather focus on capturing the high-level components that are most often used.

From the list of components, group them into functional categories based on the core features provided by the third-party component. Also, note the usage prevalence of each component across project teams to weight the reliance upon the third-party code. Using this weighted list as a guide, create a list of components to be advertised across the development organization as recommended components.

Several factors should contribute to decisions for inclusion on the recommended list. Although a list can be created without conducting research specifically, it is advisable to inspect each for incident history, track record for responding to vulnerabilities, appropriateness of functionality for the organization, excessive complexity in usage of the third-party component, etc.

This list should be created by senior developers and architects, but also include input from managers and security auditors. After creation, this list of recommended components matched against functional categories should be advertised to the development organization. Ultimately, the goal is to provide well-known defaults for project teams.

B. Identify security design patterns from architecture Across software projects at an organization, each should be categorized in terms of the generic architecture type. Common categories include client-server applications, embedded systems, desktop applications, web-facing applications, web services platforms, transactional middleware systems, mainframe applications, etc. Depending on your organization's specialty, more detailed categories may need to be developed based upon language, or processor architecture, or even era of deployment.

For the generic software architecture type, a set of general design patterns representing sound methods of implementing security functionality can be derived and applied to the individual designs of an organization's software projects. These security design patterns represent general definitions of generic design elements they can be researched or purchased, and it is often even more effective if these patterns are customized to be made more specific to your organization. Example patterns include a single-sign-on subsystem, a cross-tier delegation model, a hardened interface design, separation-of-duties authorization model, a centralized logging pattern, etc.

The process of identification of applicable and appropriate patterns should be carried out by architects, senior developers, and other technical stakeholders during the design phase.

ASSESSMENT

- Do you advertise shared security services with guidance for project teams?
- Are project teams provided with prescriptive design patterns based on their application architecture?

RESULTS

- Detailed mapping of assets to user roles to encourage better compartmentalization in design
- Reusable design building blocks for provision of security protections and functionality
- Increased confidence for software projects from use of established design techniques for security

SUCCESS METRICS

- >80% of projects with updated permission matrix in the past six months
- >80% of project teams briefed on applicable security patterns in the past six months

COSTS

- Buildout or license of applicable security patterns
- Ongoing project overhead from maintenance of permission matrix

PERSONNEL

- Architects
- Developers
- Managers
- Business Owners
- Security Auditors

RELATED LEVELS

- Education & Guidance - 1

Secure Architecture: SA3

Formally control the software design process and validate utilization of secure components

Activities

A. Establish formal reference architectures and platforms After promoting integration with shared security services and working with security patterns specific to each type of architecture, a collection of code implementing these pieces of functionality should be selected from project teams and used as the basis for a shared code-base. This shared code-base can initially start as a collection of commonly recommended libraries that each project needs to use, and it can grow over time into one or more software frameworks representing reference platforms upon which project teams build their software. Examples of reference platforms include frameworks for model view-controller web applications, libraries supporting transactional back-end systems, frameworks for web services platforms, scaffolding for client-server applications, frameworks for middle-ware with pluggable business logic, etc.

Another method of building initial reference platforms is to select a particular project early in the life-cycle and have security-savvy staff work with them to build the security functionality in a generic way so that it could be extracted from the project and utilized elsewhere in the organization.

Regardless of approach to creation, reference platforms have advantages in terms of shortening audit and security-related reviews, increasing efficiency in development, and lowering maintenance overhead.

Architects, senior developers and other technical stakeholders should participate in design and creation of reference platforms. After creation, a team must maintain ongoing support and updates.

B. Validate usage of frameworks, patterns, and platforms During routine audits of projects conduct additional analysis of project artifacts to measure usage of recommended frameworks, design patterns, shared security services, and reference platforms. Though conducted during routine audits, the goal of this activity is to collect feedback from project teams as much as to measure their individual proactive security effort.

Overall, it is important to verify several factors with project teams. Identify use of non-recommended frameworks to determine if there may be a gap in recommendations versus the organization's functionality needs. Examine unused or incorrectly used design patterns and reference platform modules to determine if updates are needed. Additionally, there may be more or different functionality that project teams would like to see implemented in the reference platforms as the organization evolves. This analysis can be conducted by any security-savvy technical staff.

Metrics collected from each project should be collated for analysis by managers and stakeholders.

ASSESSMENT

- Do project teams build software from centrally-controlled platforms and frameworks?
- Are project teams audited for the use of secure architecture components?

RESULTS

- Customized application development platforms that provide built-in security protections
- Organization-wide expectations for proactive security effort in development
- Stakeholders better able to make tradeoff decisions based on business need for secure design

SUCCESS METRICS

- >50% of active projects using reference platforms
- >80% of projects reporting framework, pattern, and platform usage feedback in the past six months
- >3.0 Likert Scale on usefulness of guidance/platforms reported by project teams

COSTS

- Buildout or license of reference platform(s)
- Ongoing maintenance and support of reference platforms
- Ongoing project overhead from usage validation during audit

PERSONNEL

- Managers
- Business Owners
- Architects
- Developers
- Security Auditors

RELATED LEVELS

- Policy & Compliance - 2
- Design Review - 3
- Implementation Review - 3
- Security Testing - 3

Security Design

	SD1	SD2	SD3
Objective	Identify security risks and attack potentialities for individual projects.	Plan penetration tests in advance and take future attack potentialities into account.	Facilitate hardware independent and ongoing penetration tests.
Activities	A. Compile a risk analysis in due consideration of security and safety goals. B. Set up an attack pattern catalog.	A. Define test environment and categorize penetration tests. B. Downscale security risks of external software and hardware components.	A. Facilitate hardware independent penetration tests and implement development cycles by the use of structured data exchange. B. Categorize, acquire and store all relevant hardware and software versions.
Assessment	✦Do projects execute risk analysis (e.g., in regard to (A)SIL)? ✦Are identified attack patterns documented in a reusable catalog?	✦Are penetration tests planned and categorized for all products? ✦How often are external security reports reviewed? ✦Is special staff trained or are external certified testers employed? ✦Is the security of external software and hardware components guaranteed?	✦Can products be tested hardware independently? ✦Can products be updated subsequently after rollout? ✦Can results from penetration tests be reused? ✦Is security already implemented into the lifecycle?
Results	✦Understanding of possible security risks. ✦Better understanding of risk functions that need to be tested. ✦Reusable catalog of possible attacks. ✦Mapping of security vulnerabilities to safety-relevant functions.	✦Better preparation against emerging security vulnerabilities. ✦Simpler testing through tightly defined test environments. ✦Improved test coverage by documenting all penetration testing procedures. ✦All tests are up-to-date with latest security vulnerabilities.	✦Possibility to test products independently of their hardware. ✦Possibility to conduct follow-up tests for new security vulnerabilities for every product. ✦Always up to date security standards. ✦Outline of the individual steps of penetration testing within the project’s lifecycle. ✦Guideline on appropriate methods for processing the feedback of penetration tests.

Security Design: SD1

Identify security risks and attack potentialities for individual projects

Activities

A. Compile a risk analysis in due consideration of security and safety goals Start with a summary of security relevant features and compile a security and risk analysis in regard to potential attacks for every feature. A risk function or risk matrix according to EN61508 or ISO26262 can be used to support the analysis. EN61508 defines Safety Integrity Level (SIL) that are also used by ISO26262 as Automotive SIL (ASIL). (A)SIL provides a way to classify risks and validate whether they meet safety requirements, however, security requirements can be validated in a similar manner. In fact, ISO21434, which will be published at the end of 2020, introduces Cybersecurity Assurance Levels (CAL) that are similar to ASIL serve as qualitative assessment of how security is established in the software developing company. Security relevant features can be identified by listing all features of a product and come up with different scenarios and situations in which security could be affected by the feature. Then identify all internal and external components that influence a product's relevant security features. Security and safety goals should be drafted in such a way that the elaborated risks will not occur.

In the second step, define all known attacks on the identified functions—also think about functions that are similar to the identified ones—and carve out attack surfaces. Assign a risk level to every found attack by, e.g., considering the frequency of occurrence of such an attack and the severity of resulting harms. The Common Vulnerability Scoring System (CVSS) calculator can be used to assess the risk level.

During these steps, penetration testers should support the developers. They should work together in several meetings. Offer a security training for staff members. Different views of external advisors can be helpful, too.

B. Set up an attack pattern catalog To prepare for penetration tests you should create and maintain an attack pattern catalog across all products. For this, known attacks that are relevant to one or more products should be gathered. In addition, attacks that are only known by your company are to be included, e.g., from previous penetration tests. The gathered information can then be used to create attack patterns. Experiences from previous penetration tests can also be of great help here.

An attack pattern is an abstract mechanism that describes how an attack works. It should begin with a name and include its probability, the exploitability for security, and the severity of the attack. Then, it should list the affected domain, the used mechanisms, and link possible attacks that follow the same schema. In addition, a short text with a description should be included. The goal, the motivation and the necessary abilities of the attacker should be documented as well. Then follows a detailed explanation of the attack. This should include the exploited vulnerability, necessary tools, and the exact steps of the process, dependencies, possible follow-up attacks, indicators and obfuscation. In addition, known examples should be provided in the form of code or known attacks and possible defense mechanisms should be proposed (mitigation). In the end, existing relationships with other known patterns or vulnerabilities can also be listed. For the creation and maintaining of the catalog several meetings are needed.

ASSESSMENT

- Do projects execute risk analysis (e.g., in regard to (A)SIL)?
- Are identified attack patterns documented in a reusable catalog?

RESULTS

- Understanding of possible security risks
- Better understanding of risk functions that need to be tested
- Reusable catalog of possible attacks
- Mapping of security vulnerabilities to safety-relevant functions

SUCCESS METRICS

- >75% of all projects have compiled a risk analysis
- >50% of all projects have compiled attack patterns

COSTS

- Setup and maintenance of project artifacts for risk analysis of attack patterns

PERSONNEL

- Developers
- Architects
- Penetration Testers
- Managers
- Security Auditors

RELATED LEVELS

- Education & Guidance -1
- Threat Assessment - 1 & 2
- Security Requirements - 1
- Security Design - 2 & 3

Security Design: SD2

Plan penetration tests in advance and take future attack potentialities into account

Activities

A. Define test environment and categorize penetration tests Penetration tests have to be planned in advance with regard to the security goals. For that reason, standards and test environments should be established within the company according to which the test environment can be defined. Furthermore, a policy has to be compiled, which includes a list of used tools, whether it's black box or white box testing and which interfaces are to be implemented in software or hardware. Please note that these suggestions should be a guideline and no mandatory regulation. Before the tests start, the test environment has to be set up, e.g., required software and hardware has to be provided and the approach has to be planned and documented.

After defining the test environment, the penetration tests dependent on security goals and found risk functions can be planned, e.g., in regard to (A)SIL. Furthermore, the penetration tests have to be selected and a prioritization and categorization of these tests should be made.

Use the BSI guideline for IT-Security penetration tests and the BSI Study of Penetration Tests if applicable.

Alternatively, external penetration testers can be tasked for certification. Additional measures for planning the penetration tests can be offered in form of training and certifications for internal penetration testers.

Experience and previously made results should be taken into account during planning of penetration tests.

B. Downscale security risks of external software and hardware components Every project team that uses external software or hardware has to ensure the security of said hardware.

Extensive research of the vendor is essential. Especially consider the history of security risks and the vendor's reaction to these issues.

Moreover, the chosen vendor should have a notification system that informs about occurring problems as soon as possible. If such a system is not available, a reporting obligation of all security exploits at the time they become known should be included in the contract. Additionally, public services, which publish security exploits, should be searched and either subscribed to or periodically examined.

ASSESSMENT

- Are penetration tests planned and categorized for all products?
- How often are external security reports reviewed?
- Is special staff trained or are external certified testers employed?
- Is the security of external software and hardware components guaranteed?

RESULTS

- Better preparation against emerging security vulnerabilities
- Simpler testing through tightly defined test environments
- Improved test coverage by documenting all penetration testing procedures
- All tests are up-to-date with latest security vulnerabilities

SUCCESS METRICS

- >90% of all penetration testers are specifically trained
- >80% of external software and hardware components are from verified vendors

COSTS

- Common costs of projects increase due to additional verification
- Additional training

PERSONNEL

- Developers
- Architects
- Penetration Testers
- Managers
- Security Auditors

RELATED LEVELS

- Education & Guidance - 2
- Threat Assessment - 3
- Security Testing - 2

Security Design: SD3

Facilitate hardware independent and ongoing penetration tests

Activities

A. Facilitate hardware independent penetration tests and implement development cycles by the use of structured data exchange In order to ensure the safety of a product over a longer period of time, it must be repeatedly tested against new threats.

To be able to test old and new products quickly and efficiently against newly detected security vulnerabilities at any time, a hardware-independent test environment is needed. For this, great effort has to be invested in implementing a hardware-independent test interface. The interface should be compatible with as many products as possible. As a result, only the test software needs to be adapted to the respective product.

A further measure includes the expansion of development cycles. This means that new development processes have to be created within the company. Instead of a strict schedule with one phase of planning, development, testing and deployment, these steps should be repeated continuously. In this case, the company generally or each project has to find a suitable procedure for itself. However, it is important that results from tests are evaluated early on and that the planning and development make changes accordingly. Therefore, the planning process has to be repeated several times.

B. Categorize, acquire and store all relevant hardware and software versions and variants It should be possible to test newly identified vulnerabilities immediately for all affected products.

This can be difficult and lead to an immense overhead since a product that has been rolled out for some time often has several software and hardware versions. However, not being able to test every “relevant” version or variant can lead to major problems afterwards. Therefore, a secure update channel and key management must be designed for every product and a plan for stopping the support after the project is dismissed. Additionally, “relevant” software and hardware versions and variants should always be present for testing. To make this possible, the affected departments of a company must agree on a standard for “relevant” versions and variants. Then, if a new “relevant” hardware or software version or variant is rolled out, it should be stored at least once in the company. In order to protect against possible losses, you should consider storing every version and variant several times (redundancies). To do this, a company must create the necessary capacity and infrastructure. Therefore, managers, developers, testers, and security experts need to design a plan for the implementation. As a point of reference for relevant versions and variants, the complexity of version and/or variant updates should be taken into consideration, i.e., how many adaptations a user needs to make during an update.

To integrate the results of the tests into the software development process, appropriate languages and formats must be defined for documenting them. Clear responsibilities have to be defined, such that the test results can be adequately addressed and security gaps closed for all versions.

ASSESSMENT

- Can products be tested hardware independently?
- Can products be updated subsequently after rollout?
- Can results from penetration tests be reused?
- Is security already implemented into the lifecycle?

RESULTS

- Possibility to test products independently of their hardware
- Possibility to conduct follow up tests for new security vulnerabilities for every product

- Always up to date security standards

- Outline of the individual steps of penetration testing within the project’s lifecycle
- Guideline on appropriate methods for processing the feedback of penetration tests

SUCCESS METRICS

- >80% of relevant hardware and software versions are stored
- >70% of all tests for a project are tested with a hardware-independent test interface

COSTS

- Increasing effort through the intermediate storage of hardware and software versions
- Development time for a test interface

PERSONNEL

- Developers
- Architects
- Penetration Testers
- Managers
- Security Auditors

RELATED LEVELS

- Strategy & Metrics - 1
- Policy & Compliance - 3
- Education & Guidance - 2
- Secure Architecture - 1
- Security Testing - 2
- Issue Management - 2 & 3
- Operational Enablement - 1
- Security Monitoring - 1

Design Review

	DR1	DR2	DR3
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	A. Identify software attack surface B. Analyze design against known security requirements	A. Inspect for complete provision of security mechanisms B. Deploy design review service for project teams	A. Develop data-flow diagrams for sensitive resources B. Establish release gates for design review
Assessment	✦Do project teams document the attack perimeter of software designs? ✦Do project teams check software designs against known security risks?	✦Do project teams specifically analyze design elements for security mechanisms? ✦Are project stakeholders aware of how to obtain a formal secure design review?	✦Does the secure design review process incorporate detailed data-level analysis? ✦Does a minimum security baseline exist for secure design review results?
Results	✦High-level understanding of security implications from perimeter architecture ✦Enable development teams to self-check designs for security best-practices ✦Lightweight process for conducting project-level design reviews	✦Formally offered assessment service to consistently review architecture for security ✦Pinpoint security flaws in maintenance-mode and legacy systems ✦Deeper understanding amongst project stakeholders on how the software provides assurance protections	✦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

Design Review: DR1

Support ad hoc reviews of software design to ensure baseline mitigations for known risks

Activities

A. Identify software attack surface For each software project, create a simplified view of the overall architecture. Typically, this should be created based on project artifacts such as high-level requirements and design documents, interviews with technical staff, or module-level review of the code base. It is important to capture the high-level modules in the system, but a good rule of thumb for granularity is to ensure that the diagram of the whole system under review fits onto one page.

From the single page architecture view, analyze each component in terms of accessibility of the interfaces from authorized users, anonymous users, operators, application-specific roles, etc. The components providing the interfaces should also be considered in the context of the one-page view to find points of functional delegation or data pass-through to other components on the diagram. Group interfaces and components with similar accessibility profiles and capture this as the software attack surface.

For each interface, further elaborate the one-page diagram to note any security-related functionality. Based on the identified interface groups comprising the attack surface, check the model for design-level consistency for how interfaces with similar access are secured. Any breaks in consistency can be noted as assessment findings.

This analysis should be conducted by security-savvy technical staff, either within the project team or external. Typically, after initial creation, the diagram and attack surface analysis only need to be updated during the design phase when additions or changes are made to the edge system interfaces.

B. Analyze design against known security requirements Security requirements, either formally identified or informally known, should be identified and collected. Additionally, identify and include any security assumptions upon which safe operation of the system relies.

Review each item on the list of known security requirements against the one-page diagram of the system architecture. Elaborate the diagram to show the design-level features that address each security requirement. Separate, granular diagrams can be created to simplify capturing this information if the system is large and/or complex. The overall goal is to verify that each known security requirement has been addressed by the system design. Any security requirements that are not clearly provided at the design level should be noted as assessment findings.

This analysis should be conducted by security-savvy technical staff with input from architects, developers, managers, and business owners as needed. It should be updated during the design phase when there are changes in security requirements or high-level system design.

ASSESSMENT

- Do project teams document the attack perimeter of software designs?
- Do project teams check software designs against known security risks?

RESULTS

- High-level understanding of security implications from perimeter architecture
- Enable development teams to self-check designs for security best-practices
- Lightweight process for conducting project-level design reviews

SUCCESS METRICS

- >50% of projects with updated attack surface analysis in past 12 months
- >50% of projects with updated security requirements design-level analysis in past 12 months

COSTS

- Buildout and maintenance of architecture diagrams for each project
- Ongoing project overhead from attack surface and security requirement design inspection

PERSONNEL

- Architects
- Developers
- Managers
- Security Auditor

RELATED LEVELS

- Security Requirements - 1

Design Review: DR2

Offer assessment services to review software design against comprehensive best practices for security

Activities

A. Inspect for complete provision of security mechanisms For each interface on a module in the high-level architecture diagram, formally iterate through the list of security mechanisms and analyze the system for their provision. This type of analysis should be performed on both internal interfaces, e.g. between tiers, as well as external ones, e.g. those comprising the attack surface.

The six main security mechanisms to consider are authentication, authorization, input validation, output encoding, error handling, and logging. Where relevant, also consider the mechanisms of cryptography and session management. For each interface, determine where in the system design each mechanism is provided and note any missing or unclear features as findings.

This analysis should be conducted by security-savvy staff with assistance from the project team for application-specific knowledge. This analysis should be performed once per release, usually toward the end of the design phase. After initial analysis, subsequent releases are required to update the findings based on changes being made during the development cycle.

B. Deploy design review service for project teams Institute a process whereby project stakeholders can request a design review. This service may be provided centrally within the organization or distributed across existing staff, but all reviewers must be trained on performing the reviews completely and consistently.

The review service should be centrally managed in that the review request queue should be triaged by senior managers, architects, and stakeholders that are familiar with the overall business risk profile for the organization. This allows prioritization of project reviews in alignment with overall business risk.

During a design review, the review team should work with project teams to collect information sufficient to formulate an understanding of the attack surface, match project-specific security requirements to design elements, and verify security mechanisms at module interfaces.

ASSESSMENT

- Do project teams specifically analyze design elements for security mechanisms?
- Are project stakeholders aware of how to obtain a formal secure design review?

RESULTS

- Formally offered assessment service to consistently review architecture for security
- Pinpoint security flaws in maintenance mode and legacy systems
- Deeper understanding amongst project stakeholders on how the software provides assurance protections

SUCCESS METRICS

- >80% of stakeholders briefed on status of review requests in the past six months
- >75% of projects undergoing design review in the past 12 months

COSTS

- Buildout, training, and maintenance of design review team
- Ongoing project overhead from review activities

PERSONNEL

- Architects
- Developers
- Managers
- Security Auditors

RELATED LEVELS

- Strategy & Metrics - 2
- Education & Guidance - 2

Design Review: DR3

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 lifecycle where a project cannot pass until a 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 time-frame for each to be reviewed to illuminate any hidden vulnerabilities in the existing systems. Exceptions should be limited to no more than 20% of all projects.

ASSESSMENT

- Does the secure design review process incorporate detailed data-level analysis?
- Does a minimum security baseline exist for secure design review results?

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

SUCCESS METRICS

- >80% of projects with updated dataflow diagrams in the past six months
- >75% of projects passing design review audit in the past six months

COSTS

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

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- Security Auditors

RELATED LEVELS

- Secure Architecture - 3
- Implemenation Review - 3

Implementation Review

	IR1	IR2	IR3
Objective	Opportunistically find basic code-level vulnerabilities and other high-risk security issues.	Make implementation review during development more accurate and efficient through automation.	Mandate comprehensive code 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 implementation review
Assessment	✦Do project teams have review checklists based on common security-related problems? ✦Do project teams review selected high-risk code?	✦Can project teams access automated code analysis tools to find security problems? ✦Do stakeholders consistently review results from code reviews?	✦Do project teams utilize automation to check code against application-specific coding standards? ✦Does a minimum security baseline exist for code review results?
Results	✦Inspection for common configuration or code vulnerabilities that lead to likely discovery or attack ✦Lightweight review for coding errors that lead to severe security impact ✦Basic code-level due diligence for security assurance	✦Development enabled to consistently self-check for code-level security vulnerabilities ✦Routine analysis results to compile historic data on per-team secure coding habits ✦Stakeholders aware of unmitigated vulnerabilities to support better tradeoff analysis	✦Increased confidence in accuracy and applicability of code analysis results ✦Organization-wide baseline for secure coding expectations ✦Project teams with an objective goal for judging code-level security

Implementation Review: IR1

Opportunistically find basic code-level vulnerabilities and other high-risk security issues

Activities

A. Create review checklists from known security requirements From the known security requirements for a project, derive a lightweight implementation review checklist for security. These can be checks specific to the security concerns surrounding the functional requirements or checks for secure coding best practices based on the implementation language, platform, typical technology stack, etc. Due to these variations, often a set of checklists are needed to cover the different types of software development within an organization.

Regardless of whether created from publicly available resources or purchased, technical stakeholders such as development managers, architects, developers, and security auditors should review the checklists for efficacy and feasibility. It is important to keep the lists short and simple, aiming to catch high-priority issues that are straightforward to find in code either manually or with simple search tools. Code analysis automation tools may also be used to achieve this same end but should also be customized to reduce the overall set of security checks to a small, valuable set in order to make the scan and review process efficient.

Developers should be briefed on the goals of checklists appropriate to their job function.

B. Perform point-review of high-risk code Since code-level vulnerabilities can have dramatically increased impacts if they occur in security-critical parts of software, project teams should review high-risk modules for common vulnerabilities. Common examples of high-risk functionality include authentication modules, access control enforcement points, session management schemes, external interfaces, input validators and data parsers, etc.

Utilizing the implementation review checklists, the analysis can be performed as a normal part of the development process where members of the project team are assigned modules to review when changes are made. Security auditors and automated review tools can also be utilized for the review.

During development cycles where high-risk code is being changed and reviewed, development managers should triage the findings and prioritize remediation appropriately with input from other project stakeholders.

ASSESSMENT

- Do project teams have review checklists based on common security related problems?
- Do project teams review selected high-risk code

RESULTS

- Inspection for common configuration or code vulnerabilities that lead to likely discovery or attack
- Lightweight review for coding errors that lead to severe security impact
- Basic code-level due diligence for security assurance

SUCCESS METRICS

- >80% of project teams briefed on relevant code review checklists in the past six months
- >50% of project teams performing code review on high-risk code in the past six months
- >3.0 Likert Scale on usefulness of code review checklists reported by developers

COSTS

- Buildout or license of code review checklists
- Ongoing project overhead from code review activities of high-risk code

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners

RELATED LEVELS

- Security Requirements - 1

Implementation Review: IR2

Make implementation review during development more accurate and efficient through automation

Activities

A. Utilize automated code analysis tools Although any such tool can produce false positives, it can save a lot of time and energy, by helping focus attention on the most suspicious sections of code.

Many security vulnerabilities at the code level are complex to understand and require careful inspection for discovery. However, there are many useful automation solutions available to automatically analyze code for bugs and vulnerabilities.

There are both commercial and open-source products available to cover popular programming languages and frameworks. Selection of an appropriate code analysis solution is based on several factors including depth and accuracy of inspection, product usability and usage model, expandability and customization features, applicability to the organization's architec-ture and technology stack(s), etc.

Utilize input from security-savvy technical staff as well as developers and development managers in the selection process and review overall results with stakeholders.

B. Integrate code analysis into development process Once a code analysis solution is selected, it must be integrated into the development process to encourage project teams to utilize its capabilities. An effective way to accomplish this is to setup the infrastructure for the scans to run automatically at build time or from code in the project's code repository. In this fashion, results are available earlier thus enabling development teams to self-check along the way before release.

A potential problem with legacy systems or large ongoing projects is that code scanners will typically report findings in modules that were not being updated in the release. If automatic scanning is setup to run periodically, an effective strategy to avoid review overhead is to limit consideration of findings to those that have been added, removed, or changed since the previous scan. It is critical to not ignore the rest of the results however, so development managers should take input from security auditors, stakeholders, and the project team to formulate a concrete plan for addressing the rest of the findings.

If unaddressed findings from implementation review remain at time of release, these must be reviewed, assigned a risk rating and accepted by project stakeholders.

ASSESSMENT

- Can project teams access automated code analysis tools to find security problems?
- Do stakeholders consistently review results from code reviews?

RESULTS

- Development enabled to consistently self-check for code-level security vulnerabilities
- Routine analysis results to compile historic data on per-team secure coding habits
- Stakeholders aware of unmitigated vulnerabilities to support better tradeoff analysis

SUCCESS METRICS

- >50% of projects with code review and stakeholder sign-off in the past six months
- >80% of projects with access to automated code review results in the past month

COSTS

- Research and selection of code analysis solution
- Initial cost and maintenance of automation integration
- Ongoing project overhead from automated code review and mitigation

PERSONNEL

- Developers
- Architects
- Managers
- Security Auditors

RELATED LEVELS

- None

Implementation Review: IR3

Mandate comprehensive implementation review process to discover language-level and application-specific risks

Activities

A. Customize code analysis for application-specific concerns Code scanning tools are powered by built-in a knowledge-base of rules to check code based on language APIs and commonly used libraries but have limited ability to understand custom APIs and designs to apply analogous checks. However, through customization, a code scanner can be a powerful, generic analysis engine for finding organization and project-specific security concerns.

While details vary between tools in terms of ease and power of custom analysis, code scanner customization generally involves specifying checks to be performed at specific APIs and function call sites. Checks can include analysis for adherence to internal coding standards, unchecked tainted data being passed to custom interfaces, tracking and verification of sensitive data handling, correct usage of an internal API, etc.

Checkers for usage of shared code-bases are an effective place to begin scanner customizations since the created checkers can be utilized across multiple projects. To customize a tool for a code-base, a security auditor should inspect both code and high-level design to identify candidate checkers to discuss with development staff and stakeholders for implementation.

B. Establish release gates for implementation review To set a code-level security baseline for all software projects, a particular point in the software development life-cycle should be established as a checkpoint where a minimum standard for implementation review results must be met in order to make a release.

To begin, this standard should be straightforward to meet, for example by choosing one or two vulnerability types and a setting the standard that no project may pass with any corresponding findings. Over time, this baseline standard should be improved by adding additional criteria for passing the checkpoint.

Generally, the implementation review checkpoint should occur toward the end of the implementation phase 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 that 20% of all projects.

ASSESSMENT

- Do project teams utilize automation to check code against application-specific coding standards?
- Does a minimum security baseline exist for code review results?

RESULTS

- Increased confidence in accuracy and applicability of code analysis results
- Organization-wide baseline for secure coding expectations
- Project teams with an objective goal for judging code-level security

SUCCESS METRICS

- >50% of projects using code analysis customizations
- >75% of projects passing code review audit in the past six months

COSTS

- Buildout and maintenance of custom code review checks
- Ongoing project overhead from code review audit
- Organization overhead from project delays caused by failed code review audits

PERSONNEL

- Architects
- Developers
- Security Auditors
- Business Owners
- Managers

RELATED LEVELS

- Policy & Compliance - 2
- Secure Architecture - 3

Security Testing

	ST1	ST2	ST3
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	A. Derive test cases from known security requirements B. Conduct penetration testing on software releases	A. Utilize automated security testing tools B. Integrate security testing into development process	A. Employ application-specific security testing automation B. Establish release gates for security testing
Assessment	✦Do projects specify security testing based on defined security requirements? ✦Is penetration testing performed on high-risk projects prior to release? ✦Are stakeholders aware of the security test status prior to release?	✦Do projects use automation to evaluate security test cases? ✦Do projects follow a consistent process to evaluate and report on security tests to stakeholders?	✦Are security test cases comprehensively generated for application-specific logic? ✦Does a minimum security baseline exist for security testing?
Results	✦Independent verification of expected security mechanisms surrounding critical business functions ✦High-level due diligence toward security testing ✦Ad-hoc growth of a security test suite for each software project	✦✦Deeper and more consistent verification of software functionality for security ✦Development teams enabled to self-check and correct problems before release ✦Stakeholders better aware of open vulnerabilities when making risk acceptance decisions	✦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 SAMM

Security Testing: ST1

Establish process to perform basic security tests based on implementation and software requirements

Activities

A. Derive test cases from known security requirements From the known security requirements for a project, identify a set of test cases to check the software for correct functionality. Typically, these test cases are derived from security concerns surrounding the functional requirements and business logic of the system but should also include generic tests for common vulnerabilities based on the implementation language or technology stack.

Often, it is most effective to use the project team's time to build application-specific test cases and utilize publicly available resources or purchased knowledge bases to select applicable general test cases for security. Although not required, automated security testing tools can also be utilized to cover the general security test cases.

This test case planning should occur during the requirements and/or design phases but must occur before final testing prior to release. Candidate test cases should be reviewed for applicability, efficacy, and feasibility by relevant development, security, and quality assurance staff.

B. Conduct penetration testing on software releases Using the set of security test cases identified for each project, penetration testing should be conducted to evaluate the system's performance against each case. It is common for this to occur during the testing phase prior to release.

Penetration testing cases should include both application-specific tests to check soundness of business logic as well as common vulnerability tests to check the design and implementation. Once specified, security test cases can be executed by security-savvy quality assurance or development staff, but first-time execution of security test cases for a project team should be monitored by a security auditor to assist and coach team members.

Prior to release or deployment, stakeholders must review results of security tests and accept the risks indicated by failing security tests at release time. In the latter case, a concrete timeline should be established to address the gaps over time.

ASSESSMENT

- Do projects specify security testing based on defined security requirements?
- Is penetration testing performed on high-risk projects prior to release?
- Are stakeholders aware of the security test status prior to release?

RESULTS

- Independent verification of expected security mechanisms surrounding critical business functions
- High-level due diligence toward security testing
- Ad hoc growth of a security test suite for each software project

SUCCESS METRICS

- >50% of projects specifying security test cases in the past 12 months
- >50% of stakeholders briefed on project status against security tests in the past six months

COSTS

- Buildout or license of security test cases
- Ongoing project overhead from maintenance and evaluation of security test cases

PERSONNEL

- QA Testers
- Security Auditor
- Developers
- Architects
- Business Owners

RELATED LEVELS

- Security Requirements - 1

Security Testing: ST2

Make security testing during development more complete and efficient through automation

Activities

A. Utilize automated security testing tools In order to test for security issues, a potentially large number of input cases must be checked against each software interface, which can make effective security testing using manual test case implementation and execution unwieldy. Thus, automated security test tools should be used to automatically test software, resulting in more efficient security testing and higher quality results.

Both commercial and open-source products are available and should be reviewed for appropriateness for the organization. Selecting a suitable tool is based on several factors including robustness and accuracy of built-in security test cases, efficacy at testing architecture types important to organization, customization to change or add test cases, quality and usability of findings to the development organization, etc.

Utilize input from security-savvy technical staff as well as development and quality assurance staff in the selection process and review overall results with stakeholders.

B. Integrate security testing into development process With tools to run automated security tests, projects within the organization should routinely run security tests and review results during development. In order to make this scalable with low overhead, security testing tools should be configured to automatically run on a routine basis, e.g. nightly or weekly, and findings should be inspected as they occur.

Conducting security tests as early as the requirements or design phases can be beneficial. While traditionally, used for functional test cases, this type of test-driven development approach involves identifying and running relevant security test cases early in the development cycle, usually during design. With the automatic execution of security test cases, projects enter the implementation phase with a number of failing tests for the nonexistent functionality. Implementation is complete when all the tests pass. This provides a clear, upfront goal for developers early in the development cycle, thus lowering risk of release delays due to security concerns or forced acceptance of risk in order to meet project deadlines.

For each project release, results from automated and manual security tests should be presented to management and business stakeholders for review. If there are unaddressed findings that remain as accepted risks for the release, stakeholders and development managers should work together to establish a concrete timeframe for addressing them.

ASSESSMENT

- Do projects use automation to evaluate security test cases?
- Do projects follow a consistent process to evaluate and report on security tests to stakeholders?

RESULTS

- Deeper and more consistent verification of software functionality for security
- Development teams enabled to self-check and correct problems before release
- Stakeholders better aware of open vulnerabilities when making risk acceptance decisions

SUCCESS METRICS

- >50% of projects with security testing and stakeholder sign-off in the past six months
- >80% of projects with access to automated security testing results in the past month

COSTS

- Research and selection of automated security testing solution
- Initial cost and maintenance of automation integration
- Ongoing project overhead from automated security testing and mitigation

PERSONNEL

- Developers
- Architects
- Managers
- Security Auditors
- QA Testers

RELATED LEVELS

- None

Security Testing: ST3

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, 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 lifecycle 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 that 20% of all projects.

ASSESSMENT

- Are security test cases comprehensively generated for application-specific logic?
- Does a minimum security baseline exist for security testing?

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

SUCCESS METRICS

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

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

PERSONNEL

- Architects
- Developers
- Security Auditors
- QA Testers
- Business Owners
- Managers

RELATED LEVELS

- Policy & Compliance - 2
- Secure Architecture - 3

Security Pentesting

	SP1	SP2	SP3
Objective	Establish and execute a complete workflow for active penetration testing.	Combine the project-wide penetration testing workflow with continuous integration (CI) techniques.	Establish feedback loops and release gates within the continuous integration for the software design and development cycle.
Activities	A. Lay out an appropriate testing structure in order to test the system against all possible vectors of attacks B. Add version management for maximum covering rates and criteria documentation for expressiveness of test results	A. Expand the automated build with related penetration testing procedures B. Establish a continuous integration mechanism of the automated build into the project's version management	A. Report (un-) successfully tested builds on a feedback platform B. Establish release gates for penetration testing
Assessment	✦Are identified attack pattern structured and tested orderly? ✦Do projects use version control management for penetration testing? ✦Do projects integrate practical exploitation and mitigation comments for every attack pattern? ✦Are all relevant and found attacks tested on the software?	✦Do penetration testers have access to automatic software builds that include their tests? ✦Is the penetration testing routine embedded in a continuous integration mechanism? ✦Do developers and penetration testers discuss and handle the results of penetration test cases?	✦Do projects integrate preventive security feedback loops including penetration testing? ✦Do projects establish release gates in the software development lifecycle for penetration testing?
Results	✦Establishment of a structure for penetration testing in the attack pattern catalog ✦Introduction of additional layering of structure depending on one's own needs ✦Introduction of version management framework for penetration testing and detailed exploitation and mitigation library	✦Enforced discipline on frequent automated penetration testing ✦Immediate feedback on system-wide impact of local changes ✦Constant availability of a current robust build for all purposes ✦Early detection mechanisms avoiding last-minute chaos	✦Projects established a preventive security feedback loop ✦Projects can draw knowledge about secure software architecture and design from security threats catalog ✦Release gates hinder weakly conceptualized software from publishing

Security Penetesting: SP1

Establish and execute a complete workflow for active penetration testing

Activities

A. Lay out an appropriate testing structure in order to test the system against all possible vectors of attacks The project team should execute a smooth integration of testing processes including found attack patterns and should capture as many vulnerabilities as there may be in order to mitigate arising risks.

If the project team considers the twofold advice, the members will feel compelled to engage in the practice of attack pattern concatenation. Hence, it is recommended to build attack pattern paths that must be tested step-by-step (or in different orders/alternative ways) in order to capture as many potential vulnerabilities as possible. Attack pattern most of the time affect each other, thus finding one new attack pattern could enable other (even closed) attack patterns (again).

Another technical aid would be to classify and catalog attack patterns as universal or specific. Universal attack patterns tend to “awake” again when new patterns emerge. Attack pattern paths can be modeled on different levels and layers. Firstly, the penetration testing team could extract paths for module testing, component testing, system testing and conclusively for delivery testing. Secondly, the team should divide between application layered and network layered attack paths. Bring in mind that attack paths intertwine.

B. Add version management for maximum covering rates and criteria documentation for expressiveness of test results Even in the presence of an appropriate attack pattern structure, an impulsive and uncoordinated execution of penetration tests would lead to low covering rates and low expressiveness of the test results. Therefore, it is recommended to integrate a version management mechanism including a fine-grained documentation of test results into the penetration testing workflow.

Version management: The version management guarantees that all attack paths and therefore all attack patterns constellations are getting tested at least once, leading to an overall robustness of the system.
As a consequence, certain penetration testers or whole penetration testing teams are hold fully responsible for certain parts of a (branched) attack path, also called test cases or spectrums.
There are controlling structures that oversee and manage the work in progress and to-do lists, recognize occurring bottlenecks, odd delays and that solve conflicts. Additionally, burn-down charts can heighten the awareness of the current penetration testing process. Integrating a scorecard, labelled with “priority” and “severity” for existing attack pattern is also feasible.

Criteria documentation: One possible way to achieve great documentation after penetration testing an attack pattern against the current system is to expand an instance (object) of the attack pattern (class) with an adjusted exploitation and mitigation section. The exploitation section should describe in detail and reproducibly, how the exploit was carried out, including technical details. The criteria, as mentioned in the security design phase, describes the probability, the exploitability for security, and the severity of the attack. The penetration tester now has the task to add specific comments to each of these criteria according to the elaborated, practical exploit. The mitigation section should also follow the previously elaborated criteria schema. Here, the tester can attach mitigation strategies to the document. One way to add structure to this chapter is to derive mitigation procedures according to well-known security goals, i.e., system hardening, authentication, encryption, authorization, intrusion detection systems, auditing/logging as well as—-even if not recommended by the “Security of Design” principles—-obfuscation techniques.

ASSESSMENT

- Are identified attack pattern structured and tested orderly?
- Do projects use version control management for penetration testing?

- Do projects integrate practical exploitation and mitigation comments for every attack pattern?
- Are all relevant and found attacks tested on the software?

RESULTS

- Establishment of a structure for penetration testing in the attack pattern catalog
- Introduction of additional layering of structure depending on one's own needs
- Introduction of version management framework for penetration testing and detailed exploitation and mitigation library

SUCCESS METRICS

- >75% of projects assessed attack paths
- >50% of projects use version management for penetration testing
- >75% of projects document exploitation & mitigation with standard criteria

COSTS

- Ongoing project overhead from establishing and maintaining attack pattern concatenation
- Ongoing project overhead from new documentation format
- Additional training of personnel in version management systems

PERSONNEL

- Penetration Testers
- Managers
- Security Auditors
- Developers
- Architects

RELATED LEVELS

- Education & Guidance – 2 & 3
- Security Design – 1 & 2
- Security Testing – 1

Security Penetesting: SP2

Combine the project-wide penetration testing workflow with continuous integration (CI) techniques

Activities

A. Expand the automated build with related penetration testing procedures Modern software engineering suggests the usage of automated builds as part of a continuous integration mechanism.

Suppose that the current software development cycles are integrated in an automated build process. Most of the time, open source build automation tools that help the team to build and launch the system on a local machine using only a single command.

Add various test cases described to the automated build scripts. For a first shot, it will be infeasible to target highly project or hardware independent test cases. Rather, try to establish automated test cases that cover all attack patterns and related paths for a specific environment. The goal of this feature is to combine the ongoing software development of critical systems with highly integrated penetration security testing automation on the local machines (including all relevant hardware and software) of penetration testers. Therefore, the project team should seek the adaptation of test-driven development (TDD) for security related test cases.

B. Establish a continuous integration mechanism of the automated build into the project's version management Following up the last activity about the automated security test cases in the automated build routine, the team can expand this approach with continuous integration (CI) techniques.

Refer to common practices of continuous integration, e.g., Fowler's Continuous Integration Guideline

Thus, the new workflow looks as follows: Whenever a new software version is published by developers on the version management system, a penetration tester pulls this update into his local machine equipped with the necessary hardware and runs the previously established automated build. Afterwards, the penetration tester reviews the outcomes of the security test cases.

If a previously succeeded test case now fails (1) (vulnerability found) or does not compile anymore (2), it will be the duty of the penetration tester to elaborate reasoning and suggest either an issue to the latest iteration of software development (1) or to the automated test case (2).

If the penetration test case does not compile anymore (2), the penetration tester rewrites it so that it operates again. If the test case has failed (1), the following procedure is recommended: If the skill of the penetration tester is low, the tester should publish an issue about the failed test case on the version management console. If the skill set is high, the tester should update the code for the latest iteration of software that is no more vulnerable to the failed test case or. Additionally, succeeding lifecycle partner must be informed about the vulnerability, if parts of the software are already in production use.

ASSESSMENT

- Do penetration testers have access to automatic software builds that include their tests?
- Is the penetration testing routine embedded in a continuous integration mechanism?
- Do developers and penetration testers discuss and handle the results of penetration test cases?

RESULTS

- Enforced discipline on frequent automated penetration testing
- Immediate feedback on system-wide impact of local changes
- Constant availability of a current robust build for all purposes
- Early detection mechanisms avoiding last-minute chaos

SUCCESS METRICS

- >50% of builds per project that include penetration tests can be executed automatically on hardware-supported local machines
- >50% of projects work with continuous integration on a penetration testing level

COSTS

- Setting up or expanding the build system for automated penetration testing
- Spreading awareness and acceptance of continuous integration for developers AND penetration testers
- Heightened complexity of build routines due to demanded flexibility of penetration tests

PERSONNEL

- Penetration testers
- Testers
- Developers
- (Project) management

RELATED LEVELS

- Security Design – 3
- Implementation Review – 2 & 3
- Security Testing - 2 & 3

Security Penetesting: SP3

Establish feedback loops and release gates within the continuous integration for the software design and development cycle

Activities

A. Report (un-) successfully tested builds on a feedback platform A continuous integration platform is introduced and used not only by developers and project management but also by penetration testers. As an outcome, the penetration tester is part of a project-wide and software development cycle-spanning feedback loop that should be established in the following way:

1. Developers integrate new software
2. Penetration testers review the updates and change the test cases or publish an issue of the current software
3. Developers skilled with security training engage in solving the conflicts
4. Due to 3., failures in the software could force alteration of software design and also of software architecture on a higher level.
5. Due to 4., project management-wide loops have to be established that iteratively check whether current software architecture and design suit the security related attack patterns

It is recommended to document a catalog of occurred changesets in software design and architecture due to security threats. This documentation can help (new) projects to preventively take care of disclosing written security vulnerabilities by default. Link the attack pattern mitigation and exploitation sections to this catalog.

B. Establish release gates for penetration testing As mentioned in Security Testing 3, release gates prevent software from being released with easily found security bugs. Integrate penetration testing results into these release gates in order to establish precise points in the software development lifecycle where these penetration test cases must pass in order to enable release of the software.

Begin to draw an initial baseline (release gate 1) on a limited selection of attack pattern paths that must pass, so that improvement to this baseline is possible. Widen the scope of the baseline with additional attack pattern paths cases (release gate 2-4) for a more robust software release.

Another option to structure release gates could be to assign different release gates to (A) releases and (B) release candidates of software development. Consider this approach if you are working on a highly cyber-security relevant system. Release candidates in software development are not intended to be seen as a finished product, rather you test the release candidates in an additional branch of software development in order to fix (security-related) bugs. Add release gates 1-3 to the earlier stages and gate 4 to the latest stage of the release candidate branch.

ASSESSMENT

- Do projects integrate preventive security feedback loops including penetration testing?
- Do projects establish release gates in the software development lifecycle for penetration testing?

RESULTS

- Projects established a preventive security feedback loop
- Projects can draw knowledge about secure software architecture and design from security threats catalog
- Release gates hinder weakly conceptualized software from publishing

SUCCESS METRICS

- >50% of projects have project-wide security related feedback loops
- >50% of projects do have release gates

COSTS

- Project management overhead for establishing release gates
- Documentation overhead for security threats catalog

PERSONNEL

- Architects
- Developers
- Penetration Testers
- Security Auditors
- Business Owners
- Managers

RELATED LEVELS

- Secure Architecture – 1-3
- Design Review – 2
- Implementation Review – 2-3

Issue Management

	IM1	IM2	IM3
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 incident response process B. Adopt a security issue disclosure process	A. Conduct root cause analysis for incidents B. Collect per-incident metrics
Assessment	✦Do projects have a point of contact for security issues or incidents? ✦Does your organization have an assigned security response team? ✦Are project teams aware of their security point(s) of contact and response team(s)?	✦Does the organization utilize a consistent process for incident reporting and handling? ✦Are project stakeholders aware of relevant security disclosures related to their software projects?	✦Are incidents inspected for root causes to generate further recommendations? ✦Do projects consistently collect and report data and metrics related to incidents?
Results	✦Lightweight process in place to handle high-priority issues or incidents ✦Framework for stakeholder notification and reporting of events with security impact ✦High-level due diligence for handling security issues	✦Communications plan for dealing with issue reports from third-parties ✦Clear process for releasing security patches to software operators ✦Formal process for tracking, handling, and internally communicating about incidents	✦Detailed feedback for organizational improvement after each incident ✦Rough cost estimation from issue and compromises ✦Stakeholders better able to make tradeoff decisions based on historic incident trends

Issue Management: IM1

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

Activities

A. Identify point of contact for security issues For each division within the organization or for each project team, establish a point of contact to serve as a communications hub for security information. While generally this responsibility will not claim much time from the individuals, the purpose of having a predetermined point of contact is to add structure and governance for vulnerability management.

Examples of incidents that might cause the utilization include receipt of a vulnerability report from an external entity, compromise or other security failure of software in the field, internal discovery of high-risk vulnerabilities, etc. In case of an event, the closest contact would step in as an extra resource and advisor to the affected project team(s) to provide technical guidance and brief other stakeholders on progress of mitigation efforts.

The point of contact should be chosen from security-savvy technical or management staff with a breadth of knowledge over the software projects in the organization. A list of these assigned security points of contact should be centrally maintained and updated at least every six months. Additionally, publishing and advertising this list allows staff within the organization to request help and work directly with one another on security problems.

B. Create informal security response team(s) From the list of individuals assigned responsibility as a security point of contact or from dedicated security personnel, select a small group to serve as a centralized technical security response team. The responsibilities of the team will include directly taking ownership of security incidents or issue reports and being responsible for triage, mitigation, and reporting to stakeholders.

Given their responsibility when tapped, members of the security response team are also responsible for executive briefings and upward communication during an incident. It is likely that most of the time, the security response team would not be operating in this capacity, though they must be flexible enough to be able to respond quickly or a smooth process must exist for deferring and incident to another team member.

The response team should hold a meeting at least annually to brief security points of contact on the response process and high-level expectations for security-related reporting from project teams.

ASSESSMENT

- Do projects have a point of contact for security issues or incidents?
- Does your organization have an assigned security response team?
- Are project teams aware of their security point(s) of contact and response team(s)?

RESULTS

- Lightweight process in place to handle high-priority issues or incidents
- Framework for stakeholder notification and reporting of events with security impact
- High-level due diligence for handling security issues

SUCCESS METRICS

- >50% of the organization briefed on closest security point of contact in the past six months
- >1 meeting of security response team and points of contact in the past 12 months

COSTS

- Ongoing variable project overhead from staff filling the security point of contact roles
- Identification of appropriate security response team

PERSONNEL

- Security Auditors
- Architects
- Managers
- Business Owners

RELATED LEVELS

- Strategy & Metrics - 3
- Education & Guidance - 2

Issue Management: IM2

Elaborate expectations for response process to improve consistency and communications

Activities

A. Establish consistent incident response process Extending from the informal security response team, explicitly document the organization's incident response process as well as the procedures that team members are expected to follow. Additionally, each member of the security response team must be trained on this material at least annually.

There are several tenets to sound incident response process and they include initial triage to prevent additional damage, change management and patch application, managing project personnel and others involved in the incident, forensic evidence collection and preservation, limiting communication about the incident to stakeholders, well-defined reporting to stakeholders and/or communications trees, etc.

With development teams, the security responders should work together to conduct the technical analysis to verify facts and assumptions about each incident or issue report. Likewise, when project teams detect an incident or high-risk vulnerability, they should follow an internal process that puts them in contact with a member of the security response team.

B. Adopt a security issue disclosure process For most organizations, it is undesirable to let news of a security problem become public, but there are several important ways in which internal-to-external communications on security issues should be fulfilled.

The first and most common is through creation and deployment of security patches for the software produced by the organization. Generally, if all software projects are only use internally, then this becomes less critical, but for all contexts where the software is being operated by parties external to the organization, a patch release process must exist. It should provide for several factors including change management and regression testing prior to patch release, announcement to operators/users with assigned criticality category for the patch, sparse technical details so that an exploit cannot be directly derived, etc.

Another avenue for external communications is with third-parties that report security vulnerabilities in an organization's software. By adopting and externally posting the expected process with time-frames for response, vulnerability reporters are encouraged to follow responsible disclosure practices.

Lastly, many states and countries legally require external communications for incidents involving data theft of personally identifiable information and other sensitive data type. Should this type of incident occur, the security response team should work with managers and business stakeholders to determine appropriate next-steps.

ASSESSMENT

- Does the organization utilize a consistent process for incident reporting and handling?
- Are project stakeholders aware of relevant security disclosures related to their software projects?

RESULTS

- Communications plan for dealing with issue reports from third-parties
- Clear process for releasing security patches to software operators
- Formal process for tracking, handling, and internally communicating about incidents

SUCCESS METRICS

- >80% of project teams briefed on incident response process in the past six months
- >80% of stakeholders briefed on security issue disclosures in the past six months

COSTS

- Ongoing organization overhead from incident response process

PERSONNEL

- Security Auditors
- Managers
- Business Owners
- Support/Operators

RELATED LEVELS

- None

Issue Management: IM3

Elaborate expectations for response process to improve consistency and communications

Activities

A. Establish consistent incident response process Extending from the informal security response team, explicitly document the organization's incident response process as well as the procedures that team members are expected to follow. Additionally, each member of the security response team must be trained on this material at least annually.

There are several tenets to sound incident response process and they include initial triage to prevent additional damage, change management and patch application, managing project personnel and others involved in the incident, forensic evidence collection and preservation, limiting communication about the incident to stakeholders, well-defined reporting to stakeholders and/or communications trees, etc.

With development teams, the security responders should work together to conduct the technical analysis to verify facts and assumptions about each incident or issue report. Likewise, when project teams detect an incident or high-risk vulnerability, they should follow an internal process that puts them in contact with a member of the security response team.

B. Adopt a security issue disclosure process For most organizations, it is undesirable to let news of a security problem become public, but there are several important ways in which internal-to-external communications on security issues should be fulfilled.

The first and most common is through creation and deployment of security patches for the software produced by the organization. Generally, if all software projects are only used internally, then this becomes less critical, but for all contexts where the software is being operated by parties external to the organization, a patch release process must exist. It should provide for several factors including change management and regression testing prior to patch release, announcement to operators/users with assigned criticality category for the patch, sparse technical details so that an exploit cannot be directly derived, etc.

Another avenue for external communications is with third-parties that report security vulnerabilities in an organization's software. By adopting and externally posting the expected process with time-frames for response, vulnerability reporters are encouraged to follow responsible disclosure practices.

Lastly, many states and countries legally require external communications for incidents involving data theft of personally identifiable information and other sensitive data type. Should this type of incident occur, the security response team should work with managers and business stakeholders to determine appropriate next-steps.

ASSESSMENT

- Are incidents inspected for root causes to generate further recommendations?
- Do projects consistently collect and report data and metrics related to incidents?

RESULTS

- Detailed feedback for organizational improvement after each incident
- Rough cost estimation from issue and compromises
- Stakeholders better able to make tradeoff decisions based on historic incident trends

SUCCESS METRICS

- >80% of incidents documented with root causes and further recommendations in the past six months
- >80% of incidents collated for metrics in the past six months

COSTS

- Ongoing organization overhead from conducting deeper research and analysis of incidents
- Ongoing organization overhead from collection and review of incident metrics

PERSONNEL

- Security Auditors
- Managers
- Business Owners
- Support/Operators

RELATED LEVELS

- Strategy & Metrics - 3

Environment Hardening

	EH1	EH2	EH3
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	A. Maintain operational environment specification B. Identify and install critical security upgrades and patches	A. Establish routine patch management process B. Monitor baseline environment configuration status	A. Identify and deploy relevant operations protection tools B. Expand audit program for environment configuration
Assessment	✦Do projects document operational environment security requirements? ✦Do projects check for security updates to third-party software components?	✦Is a consistent process used to apply upgrades and patches to critical dependencies? ✦Do projects leverage automation to check application and environment health?	✦Are stakeholders aware of options for additional tools to protect software while running in operations? ✦Does a minimum security baseline exist for environment health (versioning, patching, etc.)?
Results	✦Clear understanding of operational expectations within the development team ✦High-priority risks from underlying infrastructure mitigated on a well-understood timeline ✦Software operators with a high-level plan for security-critical maintenance of infrastructure	✦Granular verification of security characteristics of systems in operations ✦Formal expectations on timelines for infrastructure risk mitigation ✦Stakeholders consistently aware of current operations status of software projects	✦Reinforced operational environment with layered checks for security ✦Established and measured goals for operational maintenance and performance ✦Reduced likelihood of successful attack via flaws in external dependencies

Environment Hardening: EH1

Understand baseline operational environment for applications and software components

Activities

A. Maintain operational environment specification For each project, a concrete definition of the expected operating platforms should be created and maintained. Depending on the organization, this specification should be jointly created with development staff, stakeholders, support and operations groups, etc.

Begin this specification by capturing all details that must be true about the operating environment based upon the business function of the software. These can include factors such as processor architecture, operating system versions, prerequisite software, conflicting software, etc. Further, note any known user or operator configurable options about the operating environment that affect the way in which the software will behave.

Additionally, identify any relevant assumptions about the operating environment that were made in design and implementation of the project and capture those assumptions in the specification.

This specification should be reviewed and updated at least every six months for active projects or more often if changes are being made to the software design or the expected operating environment.

B. Identify and install critical security upgrades and patches Most applications are software that runs on top of another large stack of software composed of built-in programming language libraries, third-party components and development frameworks, base operating systems, etc. Because security flaws contained in any module in that large software stack affect the overall security of the organization's software, critical security updates for elements of the technology stack must be installed.

As such, regular research or ongoing monitoring of high-risk dependencies should be performed to stay abreast of the latest fixes to security flaws. Upon identification of a critical upgrade or patch that would impact the security posture of the software project, plans should be made to get affected users and operators to update their installations. Depending on the type of software project, details on doing this can vary.

ASSESSMENT

- Do projects document operational environment security requirements?
- Do projects check for security updates to third-party software components?

RESULTS

- Clear understanding of operational expectations within the development team
- High-priority risks from underlying infrastructure mitigated on a well-understood timeline
- Software operators with a high-level plan for security-critical maintenance of infrastructure

SUCCESS METRICS

- >50% of projects with updated operational environment specification in the past six months
- >50% of projects with updated list of relevant critical security patches in the past six months

COSTS

- Ongoing project overhead from buildout and maintenance of operational environment specification
- Ongoing project overhead from monitoring and installing critical security updates

PERSONNEL

- Developers
- Architects
- Managers
- Support/Operators

RELATED LEVELS

- Operational Enablement - 2

Environment Hardening: EH2

Improve confidence in application operations by hardening the operating environment

Activities

A. Establish routine patch management process Moving to a more formal process than ad hoc application of critical upgrades and patches, an ongoing process should be created in the organization to consistently apply updates to software dependencies in the operating environment.

In the most basic form, the process should aim to make guarantees for time-lapse between release and application of security upgrades and patches. To make this process efficient, organizations typically accept high latency on lower priority updates, e.g. maximum of two days for critical patches spanning to a maximum of 30 days for low priority patches.

This activity should be primarily conducted by support and operations staff, but routine meetings with development should also be conducted to keep the whole project abreast of past changes and scheduled upgrades.

Additionally, development staff should share a list of third-party components upon which the software project internally depends so that support and operations staff can monitor those as well to cue development teams on when an upgrade is required.

B. Monitor baseline environment configuration status Given the complexity of monitoring and managing patches alone across the variety of components composing the infrastructure for a software project, automation tools should be utilized to automatically monitor systems for soundness of configuration.

There are both commercial and open-source tools available to provide this type of functionality, so project teams should select a solution based on appropriateness to the organization's needs. Typical selection criteria include ease of deployment and customization, applicability to the organization's platforms and technology stacks, built-in features for change management and alerting, metrics collection and trend tracking etc.

In addition to host and platform checks, monitoring automation should be customized to perform application-specific health checks and configuration verifications. Support and operations personnel should work with architects and developers to determine the optimal amount of monitoring for a given software project.

Ultimately, after a solution is deployed for monitoring the environment's configuration status, unexpected alerts or configuration changes should be collected and regularly reviewed by project stakeholders as often as weekly but at least once per quarter.

ASSESSMENT

- Is a consistent process used to apply upgrades and patches to critical dependencies?
- Do projects leverage automation to check application and environment health?

RESULTS

- Granular verification of security characteristics of systems in operations
- Formal expectations on timelines for infrastructure risk mitigation
- Stakeholders consistently aware of current operations status of software projects

SUCCESS METRICS

- >80% of project teams briefed on patch management process in the past 12 months
- >80% of stakeholders aware of current patch status in the past six months

COSTS

- Ongoing organization overhead from patch management and monitoring
- Buildout or license of infrastructure monitoring tools

PERSONNEL

- Architects
- Developers
- Business Owners
- Managers
- Support/Operators

RELATED LEVELS

- None

Environment Hardening: EH3

Validate application health and status of operational environment against known best practices

Activities

A. Identify and deploy relevant operations protection tools In order to build a better assurance case for software in its operating environment, additional tools can be used to enhance the security posture of the overall system. Operational environments can vary dramatically, thus the appropriateness of given protection technology should be considered in the project context.

Commonly used protections tools include web application firewalls, XML security gateways for web services, anti-tamper and obfuscation packages for client/embedded systems, network intrusion detection/prevention systems for legacy infrastructure, forensic log aggregation tools, host-based integrity verification tools, etc.

Based on the organization and project-specific knowledge, technical stakeholders should work with support and operations staff to identify and recommend selected operations protection tools to business stakeholders. If deemed a valuable investment in terms of risk reduction versus cost of implementation, stakeholders should agree on plans for a pilot, widespread rollout, and ongoing maintenance.

B. Expand audit program for environment configuration When conducting routine project-level audits, expand the review to include inspection of artifacts related to hardening the operating environment. Beyond an up-to-date specification for the operational environment, audits should inspect current patch status and historic data since the previous audit. By tapping into monitoring tools, audits can also verify key factors about application configuration management and historic changes. Audits should also inspect the usage of operations protections tools against those available for the software's architecture type.

Audits for infrastructure can occur at any point after a project's initial release and deployment but should occur at least every six months. For legacy systems or projects without active development, infrastructure audits should still be conducted and reviewed by business stakeholders. An exception process should be created to allow special-case projects to continue operations, but with an explicitly assigned timeframe for mitigation of findings. Exceptions should be limited to no more that 20% of all projects.

ASSESSMENT

- Are stakeholders aware of options for additional tools to protect software while running in operations?
- Does a minimum security baseline exist for environment health (versioning, patching, etc.)?

RESULTS

- Reinforced operational environment with layered checks for security
- Established and measured goals for operational maintenance and performance
- Reduced likelihood of successful attack via flaws in external dependencies

SUCCESS METRICS

- >80% of stakeholders briefed on relevant operations protection tools in the past six months
- >75% of projects passing infrastructure audits in the past six months

COSTS

- Research and selection of operations protection solutions
- Buildout or license of operations protections tools
- Ongoing operations overhead from maintenance of protection tools
- Ongoing project overhead from infrastructure-related audits

PERSONNEL

- Business Owners
- Managers
- Support/Operators

RELATED LEVELS

- Policy & Compliance - 2

Operational Enhancement

	OE1	OE2	OE3
Objective	Enable communications between development teams and operators for critical security-relevant data.	Enable communications between development teams and operators for critical security-relevant data.	Mandate communication of security information and validate artifacts for completeness.
Activities	A. Capture critical security information for deployment? B. Document procedures for typical application alerts	A. Create per-release change management procedures B. Maintain formal operational security guides	A. Expand audit program for operational information B. Perform code signing for application components
Assessment	✦Are security notes delivered with each software release? ✦Are security-related alerts and error conditions documented on a per-project basis?	✦Do projects utilize a change management process that's well understood? ✦Do project teams deliver an operational security guide with each product release?	✦Are project releases audited for appropriate operational security information? ✦Is code signing routinely performed on software components using a consistent process?
Results	✦Ad hoc improvements to software security posture through better understanding of correct operations ✦Operators and users aware of their role in ensuring secure deployment ✦Improved communications between software developers and users for security- critical information	✦Detailed guidance for security-relevant changes delivered with software releases ✦Updated information repository on secure operating procedures per application ✦Alignment of operations expectations among developers, operators, and users.	✦Organization-wide understanding of expectations for security-relevant documentation ✦Stakeholders better able to make tradeoff decisions based on feedback from deployment and operations ✦Operators and/or users able to independently verify integrity of software releases

Operational Enhancement: OE1

Enable communications between development teams and operators for critical security-relevant data

Activities

A. Capture critical security information for deployment With software-specific knowledge, project teams should identify any security-relevant configuration and operations information and communicate it to users and operators. This enables the actual security posture of software at deployment sites to function in the same way that designers in the project team intended.

This analysis should begin with architects and developers building a list of security features built-in to the software. From that list, information about configuration options and their security impact should be captured as well. For projects that offer several different deployment models, information about the security ramifications of each should be noted to better inform users and operators about the impact of their choices.

Overall, the list should be lightweight and aim to capture the most critical information. Once initially created, it should be reviewed by the project team and business stakeholders for agreement. Additionally, it is effective to review this list with select operators or users in order to ensure the information is understandable and actionable. Project teams should review and update this information with every release but must do so at least every six months.

B. Document procedures for typical application alerts With specific knowledge of ways in which software behaves, project teams should identify the most important error and alert messages which require user/operator attention. From each identified event, information related to appropriate user/operator actions in response to the event should be captured.

From the potentially large set of events that the software might generate, select the highest priority set based on relevance in terms of the business purpose of the software. This should include any security-related events, but also may include critical errors and alerts related to software health and configuration status.

For each event, actionable advice should be captured to inform users and operators of required next steps and potential root causes of the event. These procedures must be re-viewed by the project team and updated at every major product release, every 6 months, but can be done more frequently, e.g. with each release.

ASSESSMENT

- Are security notes delivered with each software release?
- Are security-related alerts and error conditions documented on a per-project basis?

RESULTS

- Ad-hoc improvements to software security posture through better understanding of correct operations
- Operators and users aware of their role in ensuring secure deployment
- Improved communications between software developers and users for security-critical information

SUCCESS METRICS

- >50% of projects with updated deployment security information in the past six months
- >50% of projects with operational procedures for events updated in the past six months

COSTS

- Ongoing project overhead from maintenance of deployment security information
- Ongoing project overhead from maintenance of critical operating procedures

PERSONNEL

- Developers
- Architects
- Managers
- Support/Operators

RELATED LEVELS

- None

Operational Enhancement: OE2

Improve expectations for continuous secure operations through provision of detailed procedures

Activities

A. Create per-release change management procedures To more formally update users and operators on relevant changes in the software, each release must include change management procedures relevant to upgrade and first-time installation. Overall, the goal is to capture the expected accompanying steps that ensure the deployment will be successful and not incur excessive downtime or degradation of security posture.

To build these procedures during development, the project teams should setup a lightweight internal process for capturing relevant items that would impact deployments. It is effective to have this process in place early in the development cycle so that this information can be retained as soon as it is identified while in the requirements, design, and implementation phases.

Before each release, the project team should review the list as a whole for completeness and feasibility. For some projects, extensive change procedures accompanying a given release may warrant special handling, such as building automated upgrade scripts to prevent errors during deployment.

B. Maintain formal operational security guides Starting from the information captured on critical software events and the procedures for handling each, project teams should build and maintain formal guides that capture all the security-relevant information that users and operators need to know.

Initially, this guide should be built from the known information about the system, such as security-related configuration options, event handling procedures, installation and upgrade guides, operational environment specifications, security-related assumptions about the deployment environment, etc. Extending this, the formal operational security guide should elaborate on each of these to cover more details such that the majority of the users and operators will be informed for all the questions they might have had. For large or complex systems, this can be challenging, so project teams should work with business stakeholders to determine the appropriate level of documentation. Additionally, project teams should document any recommendations for deployments that would enhance security.

The operational security guide, after initial creation, should be reviewed by project teams and updated with each release.

ASSESSMENT

- Do projects utilize a change management process that’s well understood?
- Do project teams deliver an operational security guide with each product release?

RESULTS

- Detailed guidance for security-relevant changes delivered with software releases
- Updated information repository on secure operating procedures per application
- Alignment of operations expectations among developers, operators, and users.

SUCCESS METRICS

- >50% of projects with updated change management procedures in the past six months
- >80% of stakeholders briefed on status of operational security guides in the past six months

COSTS

- Ongoing project overhead from maintenance of change management procedures
- Ongoing project overhead from maintenance of operational security guides

PERSONNEL

- Developers
- Architects
- Managers
- Support/Operators

RELATED LEVELS

- Environment Hardening - 1

Operational Enhancement: OE3

Mandate communication of security information and validate artifacts for completeness

Activities

A. Expand audit program for operational information When conducting routine project-level audits, expand the review to include inspection of artifacts related to operational enablement for security. Projects should be checked to ensure they have an updated and complete operational security guides as relevant to the specifics of the software.

These audits should begin toward the end of the development cycle close to release but must be completed and passed before a release can be made. For legacy systems or inactive projects, this type of audit should be conducted, and a one-time effort should be made to address findings and verify audit compliance, after which additional audits for operational enablement are no longer required.

Audit results must be reviewed with business stakeholders prior to release. An exception process should be created to allow projects failing an audit to continue with a release but these projects should have a concrete timeline for mitigation of findings. Exceptions should be limited to no more that 20% of all active projects.

B. Perform code signing for application components Though often used with special-purpose software, code signing allows users and operators to perform integrity checks on software such that they can cryptographically verify the authenticity of a module or release. By signing software modules, the project team enables deployments to operate with a greater degree of assurance against any corruption or modification of the deployed software in its operating environment.

Signing code incurs overhead for management of signing credentials for the organization. An organization must follow safe key management processes to ensure the ongoing confidentiality of the signing keys. When dealing with any cryptographic keys, project stakeholders must also consider plans for dealing with common operational problems related to cryptography such as key rotation, key compromise, or key loss.

Since code signing is not appropriate for everything, architects and developers should work with security auditors and business stakeholders to determine which parts of the software should be signed. As projects evolve, this list should be reviewed with each release, especially when adding new modules or making changes to previously signed components.

ASSESSMENT

- Are project releases audited for appropriate operational security information?
- Is code signing routinely performed on software components using a consistent process?

RESULTS

- Organization-wide understanding of expectations for security-relevant documentation
- Stakeholders better able to make tradeoff decisions based on feedback from deployment and operations
- Operators and/or users able to independently verify integrity of software releases

SUCCESS METRICS

- >80% of projects with updated operational security guide in the last six months
- >80% of stakeholders briefed on code signing options and status in the past six months

COSTS

- Ongoing project overhead from audit of operational guides
- Ongoing organization overhead from management of code signing credentials
- Ongoing project overhead from identification and signing of code modules

PERSONNEL

- Developers
- Architects
- Managers
- Security Auditors

RELATED LEVELS

- None

Security Monitoring

	MO1	MO2	MO3
Objective	Awareness of potential attack patterns.	Knowledge about impact of attack patterns to released hardware and software versions and variants.	Minimize negative impacts for customers.
Activities	A. Maintenance of the attack pattern catalog	A. Implement all relevant penetration tests on all relevant hardware and software versions and variants B. Execute all relevant penetration tests on all relevant hardware and software	A. Operation of deployment infrastructure B. Deploy degradations/updates to customer
Assessment	✦Does your organization gather information about new vulnerability issues? ✦Are vulnerability issues filtered to dedicated project, used hardware and software stacks? ✦Are there project teams after final (serial) release responsible for these issues?	✦Is your penetration test continuously developed? ✦Do you assure that penetration tests are executed for every released hardware/software Combination? ✦Does your company collect and archive all results of these penetration tests?	✦Does your company know how to contact the end users of your devices? ✦Does your company have a process which defines the propagation of security vulnerabilities to your customers? ✦Do your project teams provide a security degradation guidance model? ✦Does your company have a process which defines the propagation of security updates to end users?
Results	✦Company has improved security knowledge ✦Up to date overview of potential attack surfaces ✦Always up to date catalog of attack patterns linked with probably impacts	✦Detailed matrix which links attack patterns to devices ✦Database with references from software/hardware variants of devices and possible attack patterns ✦Detailed TODO-List/Bug-List for each product/device.	✦Company has knowledge about current security risk of their products ✦Company has process and infrastructure for deploying security information and updates to end user ✦Security is guaranteed because all patches and workarounds are deployed to customers/end users

Security Monitoring: MO1

Awareness of potential attack patterns

Activities

A. Maintenance of the attack pattern catalog A central division of the company has to know about every hardware derivate and every software library which is used in any product at the customer's side.

With this responsibility, this division has to monitor multiple appropriate feeds to keep the knowledge about security disclosures up to date.

Further potential combined attacks must be investigated, documented and the risks have to be estimated.

All the gathered information have to lead into attack pattern which must be stored in a structured database.

ASSESSMENT

- Does your organization gather information about new vulnerability issues?
- Are vulnerability issues filtered to dedicated project, used hardware and software stacks?
- Are there project teams after final (serial) release responsible for these issues?

RESULTS

- Company has improved security knowledge
- Up to date overview of potential attack surfaces
- Always up to date catalog of attack patterns linked with probably impacts

SUCCESS METRICS

- Database for attack patern installed
- At least one person fulltime monitoring security feeds
- >70% of products/derivatives/libraries in central configuration management database
- Respond team for after sale security management installed

COSTS

- Ongoing project overhead due to security engineering
- Ongoing organization overhead due to the central security team
- Ongoing organization overhead due to the maintenance of the attack pattern cataloge

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- Security Auditor

RELATED LEVELS

- Security Design - 3

Security Monitoring: MO2

Knowledge about impact of attack patterns to released hardware and software versions and variants

Activities

A. Implement all relevant penetration tests on all relevant hardware and software versions and variants At product level the knowledge about attack patterns should be classified. This classification should structure the attack patterns to the devices. Thereby only the devices which are in the field have to be considered. Devices during development for example can be lower prioritized.

Nevertheless, all affected versions and variants have to be evaluated. As result a mapping from attack pattern to device in combination with software variant must be developed. This mapping exceedes the current configuration management because the information should be available close to real time. With this knowledge in mind specific penetration tests have to be developed. These penetration tests should be linked to the attack pattern in the “attack pattern-device” mapping above.

Out of this mapping a coverage report of all potential attack pattern can be created.

B. Execute all relevant penetration tests on all relevant hardware and software After all, the penetration tests have to be executed for all relevant software/hardware combinations. After reviewing the results regarding risk, minimizing the impact at customer should be in focus. Therefore, patches or workarounds—maybe also degradation strategies—have to be developed.Having security issues in mind, one has to guarantee quality/functionality and availability of the device also. The security workaround should also be tested whith current functional tests to be sure that the customer still has his demanded functionality in his car.

ASSESSMENT

- Is your penetration test continuously developed?
- Do you assure that penetration tests are executed for every released hardware/software combination?
- Does your company collect and archive all results of these penetration tests?

RESULTS

- Detailed matrix which links attack patterns to devices
- Database with references from software/hardware variants of devices and possible attack patterns
- Detailed TODO list / bug list for each product/device

SUCCESS METRICS

- Independent penentration test development team with at least one person fulltime
- Company development process contains process steps for penetration tests at least for every release
- Archiving of penetration test results is guaranteed
- Database contains at least >70% of released/sold products and their configuration management

COSTS

- Ongoing project overhead due to implementation of specific security tests
- Ongoing project overhead due to execution of security tests
- Ongoing organization overhead from testing all relevant devices and variants
- Ongoing organization overhead from maintaining attack pattern catalog

PERSONNEL

- Developers
- Testers
- Managers
- Security Auditor

RELATED LEVELS

- None

Security Monitoring: MO3

Minimize negative impacts for customers

Activities

A. Operation of deployment infrastructure When starting a new embedded software project, one has to clarify a communication channel to the end user. This is essential to ensure later updates and/or degradations. Therefore, an infrastructure has to be set up. If your company is in a supplier role, this infrastructure has to be cleared with your customer who also has the contact to the end user. It has to be maintained and ensured that it is future save. Thus, the deployement of later updates has to be guaranteed. Secondly, the infrastructure is itself a high-risk channel to the device in the field. This leads to many efforts to be made to keep this channel secure and also still working.

B. Deploy degradations/updates to customer Security and reliability have to be highly prioritized. After a security update has been developed or a potential degradation has been decided, a customized action must be placed to the end user device. This means, the action has to be deployed to the devices out in the field. Thereby only affected devices should get an update. This can be dependent from hardware and/or software versions and variants. Therefore, usually at company's side—or if in supplier role at the customer's side—a database with software and hardware variants has to be maintained and used for tailoring and deployment of security updates.

ASSESSMENT

- Does your company know how to contact the end users of your devices?
- Does your company have a process which defines the propagation of security vulnerabilities to your customers?
- Do your project teams provide a security degradation guidance model?
- Does your company have a process which defines the propagation of security updates to end users?

RESULTS

- Company has knowledge about current security risk of their products
- Company has process and infrastructure for deploying security information and updates to end user
- Security is guaranteed because all patches and workarounds are deployed to customers/end users

SUCCESS METRICS

- Information channels to end user are all known
- Updated guideline for developers at least two times per year
- Implemented process for distributing patches/updates/degradation (>80% automated)
- >80% of the developers are skilled regarding security
- Training for developers at least two times a year
- Trainings for operators and managers at least once per year

COSTS

- Ongoing project overhead due to providing degredation models
- Ongoing organization overhead from maintaining an update/patch process
- Ongoing organization overhead from maintaining an update channel

PERSONNEL

- Developers
- Architects
- Managers
- Business Owners
- Security Auditor
- Support/Operators

RELATED LEVELS

- None

Sponsors

We would like to thank the following sponsors who have donated funds to the SAMM project in the past: