**Governance (1)**

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

**Strategy & Metrics**

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

**Governance (long)**

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.

**Construction (2)**

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

**Verification (3)**

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

**Operations (4)**

Operations entails the processes and activities related to how an organization manages release of software 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.

SM1

**Objective**

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 3 and no more

than 7 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.</content>

</activity>

<activity number="B" id="SM1B">

<title>Build and maintain assurance program

roadmap</title>

<content>Understanding the main business risks to the

organization, evaluate the current performance of the organization against each the twelve

Practices. Assign a score for each Practice from 1, 2, or 3 based on the corresponding Objective if

the organization passes all the cumulative success metrics. If no success metrics are being met,

assign a score of 0 to the Practice. 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 3 months to review progress on

activities, performance against success metrics and other business drivers that may require program

changes.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Is there a software security assurance program already

in place?</entry>

<entry>Do most of the business stakeholders understand your

organizationís risk profile?</entry>

<entry>Is most of your development staff aware of future plans

for the assurance program?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Concrete list of the most critical business-level risks

caused by software</entry>

<entry>Tailored roadmap that addresses the security needs for

your organization with minimal overhead</entry>

<entry>Organization-wide understanding of how the assurance

program will grow over time</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;80% of stakeholders briefed on business risk profile

in past 6 months</entry>

<entry>&gt;80% of staff briefed on assurance program roadmap in

past 3 months</entry>

<entry>&gt;1 assurance program strategy session in past 3

months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout and maintenance of business risk

profile</entry>

<entry>Quarterly evaluation of assurance program</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Developers (1 day/yr)</entry>

<entry>Architects (4 days/yr)</entry>

<entry>Managers (4 days/yr)</entry>

<entry>Business Owners (4 days/yr)</entry>

<entry>QA Testers (1 day/yr)</entry>

<entry>Security Auditor (4 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 1</entry>

<entry>Threat Assessment - 1</entry>

<entry>Security Requirements - 2</entry>

</related>

</level>

<level number="2" id="SM2">

<objective>Measure relative value of data and software assets and

choose risk tolerance</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SM2A">

<title>Classify data and applications based on business

risk</title>

<content>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 organizationís 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.</content>

</activity>

<activity number="B" id="SM2B">

<title>Establish and measure per-classification security

goals</title>

<content>With a classification scheme for the

organizationís application portfolio in place, direct security goals and assurance program roadmap

choices can be made more granular. The assurance programís 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 organizationís

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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most of your applications and resources categorized

by risk?</entry>

<entry>Are risk ratings used to tailor the required assurance

activities?</entry>

<entry>Does most of the organization know about whatís required

based on risk ratings?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Customized assurance plans per project based on core

value to the business</entry>

<entry>Organization-wide understanding of security-relevance of

data and application assets</entry>

<entry>Better informed stakeholders with respect to

understanding and accepting risks</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;90% applications and data assets evaluated for risk

classification in past 12 months</entry>

<entry>&gt;80% of staff briefed on relevant application and

data risk ratings in past 6 months</entry>

<entry>&gt;80% of staff briefed on relevant assurance program

roadmap in past 3 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout or license of application and data risk

categorization scheme</entry>

<entry>Program overhead from more granular roadmap

planning</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (2 days/yr)</entry>

<entry>Managers (2 days/yr)</entry>

<entry>Business Owners (2 days/yr)</entry>

<entry>Security Auditor (2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Threat Assessment - 2</entry>

<entry>Design Review - 2</entry>

</related>

</level>

<level number="3" id="SM3">

<objective>Align security expenditure with relevant business

indicators and asset value</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SM3A">

<title>Conduct periodic industry-wide cost

comparisons</title>

<content>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 presented to stakeholders in order to better align the assurance program with the

business.</content>

</activity>

<activity number="B" id="SM3B">

<title>Collect metrics for historic security

spend</title>

<content>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 organizationís 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 affect on assurance program

roadmap.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Is per-project data for cost of assurance activities

collected?</entry>

<entry>Does your organization regularly compare your security

spend with other organizations?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Information to make informed case-by-case decisions on

security expenditures</entry>

<entry>Estimates of past loss due to security issues</entry>

<entry>Per-project consideration of security expense versus

loss potential</entry>

<entry>Industry-wide due diligence with regard to

security</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of projects reporting security costs in past 3

months</entry>

<entry>&gt;1 industry-wide cost comparison in past 1

year</entry>

<entry>&gt;1 historic security spend evaluation in past 1

year</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout or license industry intelligence on security

programs</entry>

<entry>Program overhead from cost estimation, tracking, and

evaluation</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

<entry>Business Owners (1 days/yr)</entry>

<entry>Security Auditor (1 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Issue Management - 1</entry>

</related>

</level>

</security-practice>

<security-practice number="2" id="PC" business-function="governance">

<title>Policy &amp; Compliance</title>

<abbreviation>PC</abbreviation>

<description-short>Policy &amp; Compliance involves setting up a security and

compliance control and audit framework throughout an organization to achieve increased assurance in

software under construction and in operation.</description-short>

<description>The Policy &amp; 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 thatís 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 organizationís 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.</description>

<level number="1" id="PC1">

<objective>Understand relevant governance and compliance drivers to

the organization</objective>

<activities>

<title>Activities</title>

<activity number="A" id="PC1A">

<title>Identify and monitor external compliance

drivers</title>

<content>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 organizationís 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 updated 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.</content>

</activity>

<activity number="B" id="PC1B">

<title>Build and maintain compliance guidelines</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most project stakeholders know their projectís

compliance status?</entry>

<entry>Are compliance requirements specifically considered by

project teams?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Increased assurance for handling third-party audit with

positive outcome</entry>

<entry>Alignment of internal resources based on priority of

compliance requirements</entry>

<entry>Timely discovery of evolving regulatory requirements

that affect your organization</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;1 compliance discovery meeting in past 6

months</entry>

<entry>Compliance checklist completed and updated within past 6

months</entry>

<entry>&gt;1 compliance review meeting with stakeholders in

past 6 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Initial creation and ongoing maintenance of compliance

checklist</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (2 days/yr)</entry>

<entry>Business Owners (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Strategy &amp; Metrics - 1</entry>

</related>

</level>

<level number="2" id="PC2">

<objective>Establish security and compliance baseline and understand

per-project risks</objective>

<activities>

<title>Activities</title>

<activity number="A" id="PC2A">

<title>Build policies and standards for security and

compliance</title>

<content>Beginning with a current compliance guidelines,

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.</content>

</activity>

<activity number="B" id="PC2B">

<title>Establish project audit practice</title>

<content>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, subsequent 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Does the organization utilize a set of policies and

standards to control software development?</entry>

<entry>Are project teams able to request an audit for

compliance with policies and standards?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Awareness for project teams regarding expectations for

both security and compliance</entry>

<entry>Business owners that better understand specific

compliance risks in their product lines</entry>

<entry>Optimized approach for efficiently meeting compliance

with opportunistic security improvement</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;75% of staff briefed on policies and standards in

past 6 months</entry>

<entry>&gt;80% stakeholders aware of compliance status against

policies and standards</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Internal standards buildout or license</entry>

<entry>Per-project overhead from compliance with internal

standards and audit</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

<entry>Security Auditors (2 days/project/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 1 &amp; 3</entry>

<entry>Strategy &amp; Metrics - 2</entry>

<entry>Security Requirements - 1 &amp; 3</entry>

<entry>Secure Architecture - 3</entry>

<entry>Implementation Review - 3</entry>

<entry>Design Review - 3</entry>

<entry>Environment Hardening - 3</entry>

</related>

</level>

<level number="3" id="PC3">

<objective>Require compliance and measure projects against

organization-wide policies and standards</objective>

<activities>

<title>Activities</title>

<activity number="A" id="PC3A">

<title>Create compliance gates for projects</title>

<content>Once an organization has established internal

standards for security, the next level of enforcement is to set particular points in the project

life-cycle 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 QA. 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.</content>

</activity>

<activity number="B" id="PC3B">

<title>Adopt solution for audit data collection</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are projects periodically audited to ensure a baseline

of compliance with policies and standards?</entry>

<entry>Does the organization systematically use audits to

collect and control compliance evidence?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Organization-level visibility of accepted risks due to

non-compliance</entry>

<entry>Concrete assurance for compliance at the project

level</entry>

<entry>Accurate tracking of past project compliance

history</entry>

<entry>Efficient audit process leveraging tools to cut manual

effort</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% projects in compliance with policies and

standards as seen by audit</entry>

<entry>&lt;50% time per audit as compared to manual</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout or license tools to automate audit against

internal standards</entry>

<entry>Ongoing maintenance of audit gates and exception

process</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1 days/yr)</entry>

<entry>Architects (1 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 3</entry>

<entry>Implementation Review - 2</entry>

<entry>Security Testing - 2</entry>

</related>

</level>

</security-practice>

<security-practice number="3" id="EG" business-function="governance">

<title>Education &amp; Guidance</title>

<abbreviation>EG</abbreviation>

<description-short>Education &amp; Guidance involves increasing security

knowledge amongst personnel in software development through training and guidance on security

topics relevant to individual job functions.</description-short>

<description>The Education &amp; 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 the 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.</description>

<level number="1" id="EG1">

<objective>Offer development staff access to resources around the

topics of secure programming and operations</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EG1A">

<title>Conduct technical security awareness

training</title>

<content>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. Additional coverage of commonplace software vulnerabilities is also desirable such

as a Top 10 list appropriate to the software being developed (web applications, embedded devices,

client-server applications, back-end transaction systems, etc.). Wherever possible, use code

samples and lab exercises in the specific programming 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.</content>

</activity>

<activity number="B" id="EG1B">

<title>Build and maintain technical guidelines</title>

<content>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 proprietary 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Have most developers been given high-level security

awareness training?</entry>

<entry>Does each project team have access to secure development

best practices and guidance?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Increased developer awareness on the most common

problems at the code level</entry>

<entry>Maintain software with rudimentary security best-

practices in place</entry>

<entry>Set baseline for security know-how among technical

staff</entry>

<entry>Enable qualitative security checks for baseline security

knowledge</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% development staff briefed on security issues

within past 1 year</entry>

<entry>&gt;75% senior development/architect staff briefed on

security issues within past 1 year</entry>

<entry>Launch technical guidance within 3 months of first

training</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Training course buildout or license</entry>

<entry>Ongoing maintenance of technical guidance</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Developers (1-2 days/yr)</entry>

<entry>Architects (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Security Requirements - 1</entry>

<entry>Secure Architecture - 1</entry>

</related>

</level>

<level number="2" id="EG2">

<objective>Educate all personnel in the software life-cycle with role-

specific guidance on secure development</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EG2A">

<title>Conduct role-specific application security

training</title>

<content>Conduct security training for staff that

highlights application security in the context of each roleís 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.</content>

</activity>

<activity number="B" id="EG2B">

<title>Utilize security coaches to enhance project

teams</title>

<content>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 otherís area of expertise and

route questions accordingly for efficiency. While coaches can be used at any point in the software

life-cycle, 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most roles in the development process given role-

specific training and guidance?</entry>

<entry>Are most stakeholders able to pull in security coaches

for use on projects?</entry>

</assessment>

<results>

<title>Results</title>

<entry>End-to-end awareness of the issues that leads to

security vulnerabilities at the product, design, and code levels</entry>

<entry>Build plans to remediate vulnerabilities and design

flaws in ongoing projects</entry>

<entry>Enable qualitative security checkpoints at requirements,

design, and development stages</entry>

<entry>Deeper understanding of security issues encourages more

proactive security planning</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;60% development staff trained within past 1

year</entry>

<entry>&gt;50% management/analyst staff trained within past 1

year</entry>

<entry>&gt;80% senior development/architect staff trained

within past 1 year</entry>

<entry>&gt;3.0 Likert on usefulness of training courses</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Training library build-out or license</entry>

<entry>Security-savvy staff for hands-on coaching</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (2 days/yr)</entry>

<entry>Architects (2 days/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Business Owners (1-2 days/yr)</entry>

<entry>QA Testers (1-2 days/yr)</entry>

<entry>Security Auditors (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Issue Management - 1</entry>

<entry>Design Review - 2</entry>

<entry>Secure Architecture - 2</entry>

</related>

</level>

<level number="3" id="EG3">

<objective>Mandate comprehensive security training and certify

personnel for baseline knowledge</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EG3A">

<title>Create formal application security support

portal</title>

<content>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, life-cycle stage, etc. Project teams

creating software should align themselves early in product development to the specific guidelines

that they will follow. In product assessments, the list of applicable guidelines and product-

related discussions should be used as audit criteria.</content>

</activity>

<activity number="B" id="EG3B">

<title>Establish role-based

examination/certification</title>

<content>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 annually, 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 an 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Is security-related guidance centrally controlled and

consistently distributed throughout the organization?</entry>

<entry>Are most people tested to ensure a baseline skill-set

for secure development and operations practices?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Efficient remediation of vulnerabilities in both ongoing

and legacy code bases</entry>

<entry>Quickly understand and mitigate against new attacks and

threats</entry>

<entry>Judge security-savvy of staff and measure against a

common standard</entry>

<entry>Establish fair incentives toward security

awareness</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% staff certified within past 1 year</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Certification examination build-out or license </entry>

<entry>Ongoing maintenance and change control for application

security support portal</entry>

<entry>Human-resources and overhead cost for implementing

employee certification</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

<entry>QA Testers (1 day/yr)</entry>

<entry>Security Auditors (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2 &amp; 3</entry>

</related>

</level>

</security-practice>

<security-practice number="4" id="TA" business-function="construction">

<title>Threat Assessment</title>

<abbreviation>TA</abbreviation>

<description-short>Threat Assessment involves accurately identifying and

characterizing potential attacks upon an organizationís software in order to better understand the

risks and facilitate risk management.</description-short>

<description>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 organizationís current performance

against known threats.</description>

<level number="1" id="TA1">

<objective>Identify and understand high-level threats to the

organization and individual projects</objective>

<activities>

<title>Activities</title>

<activity number="A" id="TA1A">

<title>Build and maintain application-specific threat

models</title>

<content>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 Microsoftís 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. Brainstorm 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.</content>

</activity>

<activity number="B" id="TA1B">

<title>Develop attacker profile from software

architecture</title>

<content>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 susceptible 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most projects in your organization consider and

document likely threats?</entry>

<entry>Does your organization understand and document the types

of attackers it faces?</entry>

</assessment>

<results>

<title>Results</title>

<entry>High-level understanding of factors that may lead to

negative outcomes</entry>

<entry>Increased awareness of threats amongst project

teams</entry>

<entry>Inventory of threats for your organization</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of project stakeholders briefed on the threat

models of relevant projects within past 12 months</entry>

<entry>&gt;75% of project stakeholders briefed on attacker

profiles for relevant architectures</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout and maintenance of project artifacts for threat

models</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Business Owners (1 day/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Security Auditors (2 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Strategy &amp; Metrics - 1</entry>

<entry>Security Requirements - 2</entry>

</related>

</level>

<level number="2" id="TA2">

<objective>Increase accuracy of threat assessment and improve

granularity of per-project understanding</objective>

<activities>

<title>Activities</title>

<activity number="A" id="TA2A">

<title>Build and maintain abuse-case models per

project</title>

<content>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 isnít 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 ìnoî or ìnotî 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.</content>

</activity>

<activity number="B" id="TA2B">

<title>Adopt a weighting system for measurement of

threats</title>

<content>Based on the established attacker profiles,

identify a rating system to allow relative comparison 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 5 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 life-cycle. Once built

for a project team, it should be updated during design of new features or refactoring

efforts.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do project teams regularly analyze functional

requirements for likely abuses?</entry>

<entry>Do project teams use a method of rating threats for

relative comparison?</entry>

<entry>Are stakeholders aware of relevant threats and

ratings?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Granular understanding of likely threats to individual

projects</entry>

<entry>Framework for better tradeoff decisions within project

teams</entry>

<entry>Ability to prioritize development efforts within a

project team based on risk weighting</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;75% of project teams with identified and rated

threats</entry>

<entry>&gt;75% of project stakeholders briefed on threat and

abuse models of relevant projects within past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Project overhead from maintenance of threat models and

attacker profiles</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Security Auditor (1 day/yr)</entry>

<entry>Business Owner (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Strategy &amp; Metrics - 2</entry>

<entry>Secure Architecture - 2</entry>

</related>

</level>

<level number="3" id="TA3">

<objective>Concretely tie compensating controls to each threat against

internal and third-party software</objective>

<activities>

<title>Activities</title>

<activity number="A" id="TA3A">

<title>Explicitly evaluate risk from third-party

components</title>

<content>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 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.</content>

</activity>

<activity number="B" id="TA3B">

<title>Elaborate threat models with compensating

controls</title>

<content>Conduct an assessment to formally identify

factors that directly prevent preconditions for compromise represented by the threat models. These

mitigating factors are the compensating 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 life-cycle, 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do project teams specifically consider risk from

external software?</entry>

<entry>Are all protection mechanisms and controls captured and

mapped back to threats?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Deeper consideration of full threat profile for each

software project</entry>

<entry>Detailed mapping of assurance features to established

threats against each software project</entry>

<entry>Artifacts to document due diligence based on business

function of each software project</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of project teams with updated threat models

prior to every implementation cycle</entry>

<entry>&gt;80% of project teams with updated inventory of

third-party components prior to every release</entry>

<entry>&gt;50% of all security incidents identified a priori by

threat models in past 12 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Project overhead from maintenance of detailed threat

models and expanded attacker profiles</entry>

<entry>Discovery of all third-party dependencies</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Business Owners (1 day/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Security Auditors (2 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Security Requirements - 2 &amp; 3</entry>

</related>

</level>

</security-practice>

<security-practice number="5" id="SR" business-function="construction">

<title>Security Requirements</title>

<abbreviation>SR</abbreviation>

<description-short>Security Requirements involves promoting the inclusion of

security-related requirements during the software development process in order to specify correct

functionality from inception.</description-short>

<description>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 organization 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.</description>

<level number="1" id="SR1">

<objective>Consider security explicitly during the software

requirements process</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SR1A">

<title>Derive security requirements from business

functionality</title>

<content>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.</content>

</activity>

<activity number="B" id="SR1B">

<title>Evaluate security and compliance guidance for

requirements</title>

<content>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 softwareís

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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most project teams specify some security requirements

during development?</entry>

<entry>Do project teams pull requirements from best-practices

and compliance guidance?</entry>

</assessment>

<results>

<title>Results</title>

<entry>High-level alignment of development effort with business

risks</entry>

<entry>Ad hoc capturing of industry best-practices for security

as explicit requirements</entry>

<entry>Awareness amongst stakeholders of measures being taken

to mitigate risk from software</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of project teams with explicitly defined

security requirements</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Project overhead from addition of security requirements

to each development cycle</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Security Auditor (2 days/yr)</entry>

<entry>Business Owner (1 days/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 1</entry>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Design Review - 1</entry>

<entry>Implementation Review - 1</entry>

<entry>Security Testing - 1</entry>

</related>

</level>

<level number="2" id="SR2">

<objective>Increase granularity of security requirements derived from

business logic and known risks</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SR2A">

<title>Build an access control matrix for resources and

capabilities</title>

<content>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.</content>

</activity>

<activity number="B" id="SR2B">

<title>Specify security requirements based on known

risks</title>

<content>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, implementation

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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most stakeholders reviewing access control matrices

for relevant projects?</entry>

<entry>Are project teams specifying requirements based on

feedback from other security activities?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Detailed understanding of attack scenarios against

business logic</entry>

<entry>Prioritized development effort for security features

based on likely attacks</entry>

<entry>More educated decision-making for tradeoffs between

features and security efforts</entry>

<entry>Stakeholders that can better avoid functional

requirements that inherently have security flaws</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;75% of all projects with updated abuse-case models

within past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Project overhead from buildout and maintenance of abuse-

case models</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Security Auditor (2 days/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Architects (2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Threat Assessment - 1 &amp; 3</entry>

<entry>Strategy &amp; Metrics - 1</entry>

</related>

</level>

<level number="3" id="SR3">

<objective>Mandate security requirements process for all software

projects and third-party dependencies</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SR3A">

<title>Build security requirements into supplier

agreements</title>

<content>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, Implementation 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.</content>

</activity>

<activity number="B" id="SR3B">

<title>Expand audit program for security

requirements</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most stakeholders reviewing vendor agreements for

security requirements?</entry>

<entry>Are the security requirements specified by project teams

being audited?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Formally set baseline for security expectations from

external code</entry>

<entry>Centralized information on security effort undertaken by

each project team</entry>

<entry>Ability to align resources to projects based on

application risk and desired security requirements</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of projects passing security requirements audit

in past 6 months</entry>

<entry>&gt;80% of vendor agreements analyzed for contractual

security requirements in past 12 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Increased cost from outsourced development from

additional security requirements</entry>

<entry>Ongoing project overhead from release gates for security

requirements</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Security Auditor (2 days/yr)</entry>

<entry>Managers (2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Threat Assessment - 3</entry>

<entry>Policy &amp; Compliance - 2</entry>

</related>

</level>

</security-practice>

<security-practice number="6" id="SA" business-function="construction">

<title>Secure Architecture</title>

<abbreviation>SA</abbreviation>

<description-short>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.</description-short>

<description>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 less risk of

vulnerabilities.</description>

<level number="1" id="SA1">

<objective>Insert consideration of proactive security guidance into

the software design process</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SA1A">

<title>Maintain list of recommended software

frameworks</title>

<content>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.</content>

</activity>

<activity number="B" id="SA1B">

<title>Explicitly apply security principles to

design</title>

<content>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 aide in making design decisions.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are project teams provided with a list of recommended

third-party components?</entry>

<entry>Are most project teams aware of secure design principles

and applying them?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Ad hoc prevention of unexpected dependencies and one-off

implementation choices</entry>

<entry>Stakeholders aware of increased project risk due to

libraries and frameworks chosen</entry>

<entry>Established protocol within development for proactively

applying security mechanisms to a design</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;80% of development staff briefed on software

framework recommendations in past 1 year</entry>

<entry>&gt;50% of projects self-reporting application of

security principles to design</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout, maintenance, and awareness of software

framework recommendations</entry>

<entry>Ongoing project overhead from analysis and application

of security principles</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Architects (2-4 days/yr)</entry>

<entry>Developers (2-4 days/yr)</entry>

<entry>Security Auditors (2-4 days/yr)</entry>

<entry>Managers (2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 1</entry>

</related>

</level>

<level number="2" id="SA2">

<objective>Direct the software design process toward known-secure

services and secure-by-default designs</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SA2A">

<title>Identify and promote security services and

infrastructure</title>

<content>Organizations should identify shared

infrastructure or services with security functionality. These will typically include single-sign-on

services, corporate directory systems, access control or entitlements services, and authentication

systems. By collecting and evaluating reusable systems, assemble a list of such resources and

categorize them by the security mechanism they fulfill. It is also helpful to consider each

resource in terms of why a development team would want to integrate with it, i.e. the benefits of

using the shared resource. If multiple resources exist in each category, an organization should

select and standardize on one or more shared service per category. Because future software

development will rely on these selected services, each should be thoroughly audited to ensure the

baseline security posture is understood. For each selected service, design guidance should be

created for development teams to understand how to integrate with the system. After such guidance

is assembled, it should be made available to development teams through training, mentorship,

guidelines, and standards. The benefits of doing this include promotion of known-secure systems,

simplified security guidance for project design teams, and clearer paths to building assurance

around the applications utilizing the shared security services.</content>

</activity>

<activity number="B" id="SA2B">

<title>Identify security design patterns from

architecture</title>

<content>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 organizations 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 organizationís

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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do you advertise shared security services with guidance

for project teams?</entry>

<entry>Are project teams provided with prescriptive design

patterns based on their application architecture?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Detailed mapping of assets to user roles to encourage

better compartmentalization in design</entry>

<entry>Reusable design building blocks for provision of

security protections and functionality</entry>

<entry>Increased confidence for software projects from use of

established design techniques for security</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of projects with updated permission matrix in

past 6 months</entry>

<entry>&gt;80% of project teams briefed on applicable security

patterns in past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout or license of applicable security

patterns</entry>

<entry>Ongoing project overhead from maintenance of permission

matrix</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (2-4 days/yr)</entry>

<entry>Developers (1-2 days/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

<entry>Security Auditors (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 1</entry>

</related>

</level>

<level number="3" id="SA3">

<objective>Formally control the software design process and validate

utilization of secure components</objective>

<activities>

<title>Activities</title>

<activity number="A" id="SA3A">

<title>Establish formal reference architectures and

platforms</title>

<content>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 speeding 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.</content>

</activity>

<activity number="B" id="SA3B">

<title>Validate usage of frameworks, patterns, and

platforms</title>

<content>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 organizationís 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are project teams building software from centrally

controlled platforms and frameworks?</entry>

<entry>Are project teams being audited for usage of secure

architecture components?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Customized application development platforms that

provide built-in security protections</entry>

<entry>Organization-wide expectations for proactive security

effort in development</entry>

<entry>Stakeholders better able to make tradeoff decisions

based on business need for secure design</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of active projects using reference

platforms</entry>

<entry>&gt;80% of projects reporting framework, pattern, and

platform usage feedback in past 6 months</entry>

<entry>&gt;3.0 Likert on usefulness of guidance/platforms

reported by project teams</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout or license of reference platform(s)</entry>

<entry>Ongoing maintenance and support of reference

platforms</entry>

<entry>Ongoing project overhead from usage validation during

audit</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Managers (1 day/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

<entry>Architects (3-4 days/yr)</entry>

<entry>Developers (2-3 days/yr)</entry>

<entry>Security Auditors (2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Design Review - 3</entry>

<entry>Implementation Review - 3</entry>

<entry>Security Testing - 3</entry>

</related>

</level>

</security-practice>

<security-practice number="7" id="DR" business-function="verification">

<title>Design Review</title>

<abbreviation>DR</abbreviation>

<description-short>Design Review involves inspection of the artifacts created

from the design process to ensure provision of adequate security mechanisms and adherence to an

organizationís expectations for security.</description-short>

<description>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.</description>

<level number="1" id="DR1">

<objective>Support ad hoc reviews of software design to ensure

baseline mitigations for known risks</objective>

<activities>

<title>Activities</title>

<activity number="A" id="DR1A">

<title>Identify software attack surface</title>

<content>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 needs to

be updated during the design phase when additions or changes are made to the edge system

interfaces.</content>

</activity>

<activity number="B" id="DR1B">

<title>Analyze design against known security

requirements</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do project teams document the attack perimeter of

software designs?</entry>

<entry>Do project teams check software designs against known

security risks?</entry>

</assessment>

<results>

<title>Results</title>

<entry>High-level understanding of security implications from

perimeter architecture</entry>

<entry>Enable development teams to self-check designs for

security best-practices</entry>

<entry>Lightweight process for conducting project-level design

reviews</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of projects with updated attack surface analysis

in past 12 months</entry>

<entry>&gt;50% of projects with updated security requirements

design-level analysis in past 12 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout and maintenance of architecture diagrams for

each project</entry>

<entry>Ongoing project overhead from attack surface and

security requirement design inspection</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Architects (2-3 days/yr)</entry>

<entry>Developers (1-2 days/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Security Auditor (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Security Requirements - 1</entry>

</related>

</level>

<level number="2" id="DR2">

<objective>Offer assessment services to review software design against

comprehensive best practices for security</objective>

<activities>

<title>Activities</title>

<activity number="A" id="DR2A">

<title>Inspect for complete provision of security

mechanisms</title>

<content>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.</content>

</activity>

<activity number="B" id="DR2B">

<title>Deploy design review service for project

teams</title>

<content>Institute a process whereby project

stakeholders can request an 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most project teams specifically analyze design

elements for security mechanisms?</entry>

<entry>Are most project stakeholders aware of how to obtain a

formal design review?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Formally offered assessment service to consistently

review architecture for security</entry>

<entry>Pinpoint security flaws in maintenance-mode and legacy

systems</entry>

<entry>Deeper understanding amongst project stakeholders on how

the software provides assurance protections</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of stakeholders briefed on status of review

requests in past 6 months</entry>

<entry>&gt;75% of projects undergoing design review in past 12

months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout, training, and maintenance of design review

team</entry>

<entry>Ongoing project overhead from review activities</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1-2 days/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Security Auditors (2-3 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 2</entry>

<entry>Strategy &amp; Metrics - 2</entry>

</related>

</level>

<level number="3" id="DR3">

<objective>Require assessments and validate artifacts to develop

detailed understanding of protection mechanisms</objective>

<activities>

<title>Activities</title>

<activity number="A" id="DR3A">

<title>Develop data-flow diagrams for sensitive

resources</title>

<content>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.</content>

</activity>

<activity number="B" id="DR3B">

<title>Establish release gates for design review</title>

<content>Having established a consistent design review

program, the next step of enforcement is to set a particular point in the software development

life-cycle where a project cannot pass until an design review is conducted and findings are

reviewed and accepted. In order to accomplish this, a baseline level of expectations should be set,

e.g. no projects with any high-severity findings will be allowed to pass and all other findings

must be accepted by the business owner. Generally, design reviews should occur toward the end of

the design phase to aide early detection of security issues, but it must occur before releases can

be made from the project team. For legacy systems or inactive projects, an exception process should

be created to allow those projects to continue operations, but with an explicitly assigned time-

frame for each to be reviewed to illuminate any hidden vulnerabilities in the existing systems.

Exceptions for should be limited to no more than 20% of all projects.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Does the design review process incorporate detailed

data-level analysis?</entry>

<entry>Does routine project audit require a baseline for design

review results?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Granular view of weak points in a system design to

encourage better compartmentalization</entry>

<entry>Organization-level awareness of project standing against

baseline security expectations for architecture</entry>

<entry>Comparisons between projects for efficiency and progress

toward mitigating known flaws</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of projects with updated data-flow diagrams in

past 6 months</entry>

<entry>&gt;75% of projects passing design review audit in past

6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing project overhead from maintenance of data-flow

diagrams</entry>

<entry>Organization overhead from project delays caused by

failed design review audits</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (2 days/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Business Owners (1-2 days/yr)</entry>

<entry>Security Auditors (2-3 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Secure Architecture - 3</entry>

<entry>Implementation Review - 3</entry>

</related>

</level>

</security-practice>

<security-practice number="8" id="IR" business-function="verification">

<title>Implementation review</title>

<abbreviation>IR</abbreviation>

<description-short>Implementation Review involves assessment of an

organizationís source code to aid vulnerability discovery and related mitigation activities as well

as establish a baseline for secure coding expectations.</description-short>

<description>The Implementation Review (IR) Practice is focused on inspection

of software at the source code 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.</description>

<level number="1" id="IR1">

<objective>Opportunistically find basic code-level vulnerabilities and

other high-risk security issues</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IR1A">

<title>Create review checklists from known security

requirements</title>

<content>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 checklist 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.</content>

</activity>

<activity number="B" id="IR1B">

<title>Perform point-review of high-risk code</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most project teams have review checklists based on

common problems?</entry>

<entry>Are project teams generally performing review of

selected high-risk code?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Inspection for common code vulnerabilities that lead to

likely discovery or attack</entry>

<entry>Lightweight review for coding errors that lead to severe

security impact</entry>

<entry>Basic code-level due diligence for security

assurance</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;80% of project teams briefed on relevant

implementation review checklists in past 6 months</entry>

<entry>&gt;50% of project teams performing implementation

review on high-risk code in past 6 months</entry>

<entry>&gt;3.0 Likert on usefulness of implementation review

checklists reported by developers</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout or license of implementation review

checklists</entry>

<entry>Ongoing project overhead from implementation review

activities of high-risk code</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Developers (2-4 days/yr)</entry>

<entry>Architects (1-2 days/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Security Requirements - 1</entry>

</related>

</level>

<level number="2" id="IR2">

<objective>Make implementation review during development more accurate

and efficient through automation</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IR2A">

<title>Utilize automated code analysis tools</title>

<content>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 organizationís architecture 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.</content>

</activity>

<activity number="B" id="IR2B">

<title>Integrate code analysis into development

process</title>

<content>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 projectís 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. If 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 release, these must be reviewed and accepted by project stakeholders.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Can most project teams access automated code analysis

tools to find security problems?</entry>

<entry>Do most stakeholders consistently require and review

results from implementation reviews?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Development enabled to consistently self-check for code-

level security vulnerabilities</entry>

<entry>Routine analysis results to compile historic data on

per-team secure coding habits</entry>

<entry>Stakeholders aware of unmitigated vulnerabilities to

support better tradeoff analysis</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of projects with implementation review and

stakeholder sign-off in past 6 months</entry>

<entry>&gt;80% of projects with access to automated

implementation review results in past 1 month</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Research and selection of code analysis solution</entry>

<entry>Initial cost and maintenance of automation

integration</entry>

<entry>Ongoing project overhead from automated implementation

review and mitigation</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1-2 days/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Security Auditors (3-4 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

<level number="3" id="IR3">

<objective>Mandate comprehensive implementation review process to

discover language-level and application-specific risks</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IR3A">

<title>Customize code analysis for application-specific

concerns</title>

<content>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.</content>

</activity>

<activity number="B" id="IR3B">

<title>Establish release gates for implementation

review</title>

<content>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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do project teams utilize automation to check code

against application-specific coding standards?</entry>

<entry>Does routine project audit require a baseline for

implementation review results prior to release?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Increased confidence in accuracy and applicability of

code analysis results</entry>

<entry>Organization-wide baseline for secure coding

expectations</entry>

<entry>Project teams with an objective goal for judging code-

level security</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of projects using code analysis

customizations</entry>

<entry>&gt;75% of projects passing implementation review audit

in past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout and maintenance of custom implementation review

checks</entry>

<entry>Ongoing project overhead from implementation review

audit</entry>

<entry>Organization overhead from project delays caused by

failed implementation review audits</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1 day/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Security Auditors (1-2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Secure Architecture - 3</entry>

</related>

</level>

</security-practice>

<security-practice number="9" id="ST" business-function="verification">

<title>Security Testing</title>

<abbreviation>ST</abbreviation>

<description-short>Security Testing involves testing the organizationís

software in its runtime environment in order to both discover vulnerabilities and establish a

minimum standard for software releases.</description-short>

<description>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.</description>

<level number="1" id="ST1">

<objective>Establish process to perform basic security tests based on

implementation and software requirements</objective>

<activities>

<title>Activities</title>

<activity number="A" id="ST1A">

<title>Derive test cases from known security

requirements</title>

<content>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 teamís 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.</content>

</activity>

<activity number="B" id="ST1B">

<title>Conduct penetration testing on software

releases</title>

<content>Using the set of security test cases identified

for each project, penetration testing should be conducted to evaluate the systemís 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 operations, 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are projects specifying some security tests based on

requirements?</entry>

<entry>Do most projects perform penetration tests prior to

release?</entry>

<entry>Are most stakeholders aware of the security test status

prior to release?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Independent verification of expected security mechanisms

surrounding critical business functions</entry>

<entry>High-level due diligence toward security testing</entry>

</results>

<entry>Ad hoc growth of a security test suite for each software

project</entry>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of projects specifying security test cases in

past 12 months</entry>

<entry>&gt;50% of stakeholders briefed on project status

against security tests in past 6 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Buildout or license of security test cases</entry>

<entry>Ongoing project overhead from maintenance and evaluation

of security test cases</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>QA Testers (1-2 days/yr)</entry>

<entry>Security Auditor (1-2 days/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Security Requirements - 1</entry>

</related>

</level>

<level number="2" id="ST2">

<objective>Make security testing during development more complete and

efficient through automation</objective>

<activities>

<title>Activities</title>

<activity number="A" id="ST2A">

<title>Utilize automated security testing tools</title>

<content>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

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.</content>

</activity>

<activity number="B" id="ST2B">

<title>Integrate security testing into development

process</title>

<content>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 non-existent

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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are projects using automation to evaluate security test

cases?</entry>

<entry>Do most projects follow a consistent process to evaluate

and report on security tests to stakeholders?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Deeper and more consistent verification of software

functionality for security</entry>

<entry>Development teams enabled to self-check and correct

problems before release</entry>

<entry>Stakeholders better aware of open vulnerabilities when

making risk acceptance decisions</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of projects with security testing and

stakeholder sign-off in past 6 months</entry>

<entry>&gt;80% of projects with access to automated security

testing results in past 1 month</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Research and selection of automated security testing

solution</entry>

<entry>Initial cost and maintenance of automation

integration</entry>

<entry>Ongoing project overhead from automated security testing

and mitigation</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1 days/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Security Auditors (2 days/yr)</entry>

<entry>QA Testers (3-4 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

<level number="3" id="ST3">

<objective>Require application-specific security testing to ensure

baseline security before deployment</objective>

<activities>

<title>Activities</title>

<activity number="A" id="ST3A">

<title>Employ application-specific security testing

automation</title>

<content>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.</content>

</activity>

<activity number="B" id="ST3B">

<title>Establish release gates for security

testing</title>

<content>To prevent software from being released with

easily found security bugs, a particular point in the software development life-cycle should be

identified as a checkpoint where an established set of security test cases must pass in order to

make a release from the project. This establishes a baseline for the kinds of security tests all

projects are expected to pass. Since adding too many test cases initially can result in an overhead

cost bubble, begin by choosing one or two security issues and include a wide variety of test cases

for each with the expectation that no project may pass if any test fails. Over time, this baseline

should be improved by selecting additional security issues and adding a variety of corresponding

test cases. Generally, this security testing checkpoint should occur toward the end of the

implementation or testing, but must occur before release. For legacy systems or inactive projects,

an exception process should be created to allow those projects to continue operations, but with an

explicitly assigned timeframe for mitigation of findings. Exceptions should be limited to no more

that 20% of all projects.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are security test cases comprehensively generated for

application-specific logic?</entry>

<entry>Do routine project audits demand minimum standard

results from security testing?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Organization-wide baseline for expected application

performance against attacks</entry>

<entry>Customized security test suites to improve accuracy of

automated analysis</entry>

<entry>Project teams aware of objective goals for attack

resistance</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of projects using security testing

customizations</entry>

<entry>&gt;75% of projects passing all security tests in past 6

months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Buildout and maintenance of customizations to security

testing automation</entry>

<entry>Ongoing project overhead from security testing audit

process</entry>

<entry>Organization overhead from project delays caused by

failed security testing audits</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1 day/yr)</entry>

<entry>Developers (1 day/yr)</entry>

<entry>Security Auditors (1-2 days/yr)</entry>

<entry>QA Testers (1-2 days/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

<entry>Secure Architecture - 3</entry>

</related>

</level>

</security-practice>

<security-practice number="10" id="IM" business-function="operations">

<title>Issue Management</title>

<abbreviation>IM</abbreviation>

<description-short>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.</description-short>

<description>The Issue Management (IM) Practice is focused on the processes

within an organization with respect to handling vulnerability reports and operational incidents. By

having these processes in place, an organizationís 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 vulnerability reports

to collect detailed metrics and other root-cause information to feedback into the organizationís

downstream behavior.</description>

<level number="1" id="IM1">

<objective>Understand high-level plan for responding to vulnerability

reports or incidents</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IM1A">

<title>Identify point of contact for security

issues</title>

<content>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 issue

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.</content>

</activity>

<activity number="B" id="IM1B">

<title>Create informal security response team(s)</title>

<content>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 vulnerability 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do most projects have a point of contact for security

issues?</entry>

<entry>Does your organization have an assigned security

response team?</entry>

<entry>Are most project teams aware of their security point(s)

of contact and response team(s)?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Lightweight process in place to handle high-priority

vulnerabilities or incidents</entry>

<entry>Framework for stakeholder notification and reporting of

events with security impact</entry>

<entry>High-level due diligence for handling security

issues</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of the organization briefed on closest security

point of contact in past 6 months</entry>

<entry>&gt;1 meeting of security response team and points of

contact in past 12 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Ongoing variable project overhead from staff filling the

security point of contact roles</entry>

<entry>Identification of appropriate security response

team</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Security Auditors (1 day/yr)</entry>

<entry>Architects (1 day/yr)</entry>

<entry>Managers (1 day/yr)</entry>

<entry>Business Owners (1 day/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Education &amp; Guidance - 2</entry>

<entry>Strategy &amp; Metrics - 3</entry>

</related>

</level>

<level number="2" id="IM2">

<objective>Elaborate expectations for response process to improve

consistency and communications</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IM2A">

<title>Establish consistent incident response

process</title>

<content>Extending from the informal security response

team, explicitly document the organizationís 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 vulnerability 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.</content>

</activity>

<activity number="B" id="IM2B">

<title>Adopt a security issue disclosure process</title>

<content>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 organizationís software. By adopting and externally

posting the expected process with timeframes 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Does the organization utilize a consistent process for

incident reporting and handling?</entry>

<entry>Are most project stakeholders aware of relevant security

disclosures related to their software projects?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Communications plan for dealing with vulnerability

reports from third-parties</entry>

<entry>Clear process for releasing security patches to software

operators</entry>

<entry>Formal process for tracking, handling, and internally

communicating about incidents</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of project teams briefed on incident response

process in past 6 months</entry>

<entry>&gt;80% of stakeholders briefed on security issue

disclosures in past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing organization overhead from incident response

process</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Security Auditors (3-5 days/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Business Owners (1-2 days/yr)</entry>

<entry>Support/Operators (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

<level number="3" id="IM3">

<objective>Improve analysis and data gathering within response process

for feedback into proactive planning</objective>

<activities>

<title>Activities</title>

<activity number="A" id="IM3A">

<title>Conduct root cause analysis for incidents</title>

<content>Though potentially time consuming, the incident

response process should be augmented to include additional analysis to identify the key, underlying

security failures. These root causes can be technical problems such as code-level vulnerabilities,

configuration errors, etc. or they can be people/process problems such as social engineering,

failure to follow procedures, etc. Once a root cause is identified for an incident, it should be

used as a tool to find other potential weaknesses in the organization where an analogous incident

could have occurred. For each identified weakness additional recommendations for proactive

mitigations should be communicated as part of closing out the original incident response effort.

Any recommendations based on root cause analysis should be reviewed by management and relevant

business stakeholders in order to either schedule mitigation activities or note the accepted

risks.</content>

</activity>

<activity number="B" id="IM3B">

<title>Collect per-incident metrics</title>

<content>By having a centralized process to handle all

compromise and high-priority vulnerability reports, an organization is enabled to take measurements

of trends over time to determine impact and efficiency of initiatives for security assurance.

Records of past incidents should be stored and reviewed at least every 6 months. Group similar

incidents and simply tally the overall count for each type of problem. Additional measurements to

take from the incidents include frequency of software projects affected by incidents, system

downtime and cost from loss of use, human resources taken in handling and cleanup of the incident,

estimates of long-term costs such as regulatory fines or brand damage, etc. For root causes that

were technical problems in nature, it is also helpful to identify what kind of proactive, review,

or operational practice might have detected it earlier or lessened the damage. This information is

concrete feedback into the program planning process since it represents the real security impact

that the organization has felt over time.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most incidents inspected for root causes to generate

further recommendations?</entry>

<entry>Do most projects consistently collect and report data

and metrics related to incidents?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Detailed feedback for organizational improvement after

each incident</entry>

<entry>Rough cost estimation from vulnerabilities and

compromises</entry>

<entry>Stakeholders better able to make tradeoff decisions

based on historic incident trends</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of incidents documented with root causes and

further recommendations in past 6 months</entry>

<entry>&gt;80% of incidents collated for metrics in the past 6

months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing organization overhead from conducting deeper

research and analysis of incidents</entry>

<entry>Ongoing organization overhead from collection and review

of incident metrics</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Security Auditors (3 days/yr)</entry>

<entry>Managers (2 days/yr)</entry>

<entry>Business Owners (2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Strategy &amp; Metrics - 3</entry>

</related>

</level>

</security-practice>

<security-practice number="11" id="EH" business-function="operations">

<title>Environment Hardening</title>

<abbreviation>EH</abbreviation>

<description-short>Environment Hardening involves implementing controls for

the operating environment surrounding an organizationís software to bolster the security posture of

applications that have been deployed.</description-short>

<description>The Environment Hardening (EH) Practice is focused on building

assurance for the runtime environment that hosts the organizationís 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.</description>

<level number="1" id="EH1">

<objective>Understand baseline operational environment for

applications and software components</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EH1A">

<title>Maintain operational environment

specification</title>

<content>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 should 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 6 months for active projects or more

often if changes are being made to the software design or the expected operating

environment.</content>

</activity>

<activity number="B" id="EH1B">

<title>Identify and install critical security upgrades

and patches</title>

<content>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

organizationís 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do the majority of projects document some requirements

for the operational environment?</entry>

<entry>Do most projects check for security updates to third-

party software components?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Clear understanding of operational expectations within

the development team </entry>

<entry>High-priority risks from underlying infrastructure

mitigated on a well-understood timeline</entry>

<entry>Software operators with a high-level plan for security-

critical maintenance of infrastructure</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% project with updated operational environment

specification in past 6 months</entry>

<entry>&gt;50% of projects with updated list of relevant

critical security patches in past 6 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Ongoing project overhead from buildout and maintenance

of operational environment specification</entry>

<entry>Ongoing project overhead from monitoring and installing

critical security updates</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Developers (1-2 day/yr)</entry>

<entry>Architects (1-2 day/yr)</entry>

<entry>Managers (2-4 day/yr)</entry>

<entry>Support/Operators (3-4 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Operational Enablement - 2</entry>

</related>

</level>

<level number="2" id="EH2">

<objective>Improve confidence in application operations by hardening

the operating environment</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EH2A">

<title>Establish routine patch management

process</title>

<content>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 2 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.</content>

</activity>

<activity number="B" id="EH2B">

<title>Monitor baseline environment configuration

status</title>

<content>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

organizationís needs. Typical selection criteria includes ease of deployment and customization,

applicability to the organizationís 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 environmentís 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Is a consistent process used to apply upgrades and

patches to critical dependencies?</entry>

<entry>Do most project leverage automation to check application

and environment health?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Granular verification of security characteristics of

systems in operations</entry>

<entry>Formal expectations on timelines for infrastructure risk

mitigation</entry>

<entry>Stakeholders consistently aware of current operations

status of software projects</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of project teams briefed on patch management

process in past 12 months</entry>

<entry>&gt;80% of stakeholders aware of current patch status in

past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing organization overhead from patch management and

monitoring</entry>

<entry>Buildout or license of infrastructure monitoring

tools</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Architects (1-2 days/yr)</entry>

<entry>Developers (1-2 days/yr)</entry>

<entry>Business Owners (1-2 days/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Support/Operators (3-4 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

<level number="3" id="EH3">

<objective>Validate application health and status of operational

environment against known best practices</objective>

<activities>

<title>Activities</title>

<activity number="A" id="EH3A">

<title>Identify and deploy relevant operations

protection tools</title>

<content>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.</content>

</activity>

<activity number="B" id="EH3B">

<title>Expand audit program for environment

configuration</title>

<content>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 softwareís architecture type. Audits for infrastructure can occur at any point

after a projectís initial release and deployment, but should occur at least every 6 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are stakeholders aware of options for additional tools

to protect software while running in operations?</entry>

<entry>Does routine audit check most projects for baseline

environment health?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Reinforced operational environment with layered checks

for security</entry>

<entry>Established and measured goals for operational

maintenance and performance</entry>

<entry>Reduced likelihood of successful attack via flaws in

external dependencies</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of stakeholders briefed on relevant operations

protection tools in past 6 months</entry>

<entry>&gt;75% of projects passing infrastructure audits in

past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Research and selection of operations protection

solutions</entry>

<entry>Buildout or license of operations protections

tools</entry>

<entry>Ongoing operations overhead from maintenance of

protection tools</entry>

<entry>Ongoing project overhead from infrastructure-related

audits</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Business Owners (1 day/yr)</entry>

<entry>Managers (1-2 days/yr)</entry>

<entry>Support/Operators (3-4 days)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Policy &amp; Compliance - 2</entry>

</related>

</level>

</security-practice>

<security-practice number="12" id="OE" business-function="deployment">

<title>Operational Enablement</title>

<abbreviation>OE</abbreviation>

<description-short>Operational Enablement involves identifying and capturing

security-relevant information needed by an operator to properly configure, deploy, and run an

organizationís software.</description-short>

<description>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 impactful 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.</description>

<level number="1" id="OE1">

<objective>Enable communications between development teams and

operators for critical security-relevant data</objective>

<activities>

<title>Activities</title>

<activity number="A" id="OE1A">

<title>Capture critical security information for

operations</title>

<content>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 operations 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 6

months.</content>

</activity>

<activity number="B" id="OE1B">

<title>Document procedures for typical application

alerts</title>

<content>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 reviewed 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Do you deliver security notes with the majority of

software releases?</entry>

<entry>Are security-related alerts and error conditions

documented for most projects?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Ad hoc improvements to software security posture through

better understanding of correct operations</entry>

<entry>Operators and users aware of their role in ensuring

secure operations</entry>

<entry>Improved communications between software developers and

users for security-critical information</entry>

</results>

<metrics>

<title>Success Metrics</title>

<entry>&gt;50% of projects with updated operations security

information in past 6 months</entry>

<entry>&gt;50% of projects with operational procedures for

events updated in past 6 months</entry>

</metrics>

<costs>

<title>Costs</title>

<entry>Ongoing project overhead from maintenance of operations

security information</entry>

<entry>Ongoing project overhead from maintenance of critical

operating procedures</entry>

</costs>

<personnel>

<title>Personnel</title>

<entry>Developers (1-2 days/yr)</entry>

<entry>Architects (1-2 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

<entry>Support/Operators (1 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

<level number="2" id="OE2">

<objective>Improve expectations for continuous secure operations

through provision of detailed procedures</objective>

<activities>

<title>Activities</title>

<activity number="A" id="OE2A">

<title>Create per-release change management

procedures</title>

<content>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 operations. 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

operations.</content>

</activity>

<activity number="B" id="OE2B">

<title>Maintain formal operational security

guides</title>

<content>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 operations

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 operations that

would enhance security. The operational security guide, after initial creation, should be reviewed

by project teams and updated with each release.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most project utilizing a change management process

thatís well understood?</entry>

<entry>Do project teams deliver an operational security guide

with each product release?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Detailed guidance for security-relevant changes

delivered with software releases</entry>

<entry>Updated information repository on secure operating

procedures per application</entry>

<entry>Alignment of operations expectations among developers,

operators, and users.</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;50% of projects with updated change management

procedures in past 6 months</entry>

<entry>&gt;80% of stakeholders briefed on status of operational

security guides in past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing project overhead from maintenance of change

management procedures</entry>

<entry>Ongoing project overhead from maintenance of operational

security guides</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1-2 days/yr)</entry>

<entry>Architects (1-2 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

<entry>Support/Operators (1 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry>Environment Hardening - 1</entry>

</related>

</level>

<level number="3" id="OE3">

<objective>Mandate communication of security information and validate

artifacts for completeness</objective>

<activities>

<title>Activities</title>

<activity number="A" id="OE3A">

<title>Expand audit program for operational

information</title>

<content>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.</content>

</activity>

<activity number="B" id="OE3B">

<title>Perform code signing for application

components</title>

<content>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 operations 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.</content>

</activity>

</activities>

<assessment>

<title>Assessment</title>

<entry>Are most projects being audited to check each release

for appropriate operational security information?</entry>

<entry>Is code signing routinely performed on software

components using a consistent process?</entry>

</assessment>

<results>

<title>Results</title>

<entry>Organization-wide understanding of expectations for

security-relevant documentation</entry>

<entry>Stakeholders better able to make tradeoff decisions

based on feedback from deployment and operations</entry>

<entry>Operators and/or users able to independently verify

integrity of software releases</entry>

</results>

<metrics>

<title>Addíl Success Metrics</title>

<entry>&gt;80% of projects with updated operational security

guide in last 6 months</entry>

<entry>&gt;80% of stakeholders briefed on code signing options

and status in past 6 months</entry>

</metrics>

<costs>

<title>Addíl Costs</title>

<entry>Ongoing project overhead from audit of operational

guides</entry>

<entry>Ongoing organization overhead from management of code

signing credentials</entry>

<entry>Ongoing project overhead from identification and signing

of code modules.</entry>

</costs>

<personnel>

<title>Addíl Personnel</title>

<entry>Developers (1 days/yr)</entry>

<entry>Architects (1 days/yr)</entry>

<entry>Managers (1 days/yr)</entry>

<entry>Security Auditors (1-2 days/yr)</entry>

</personnel>

<related>

<title>Related Levels</title>

<entry> </entry>

</related>

</level>

</security-practice>

</security-practices>

<section id="front-matter">

<title>Software Assurance Maturity Model</title>

<subtitle>A guide to building security into software development</subtitle>

<version>Version - 1.0</version>

<content><heading-bullets>For the latest version and additional info, please see the

project web site at</heading-bullets> http://www.opensamm.org<heading1>Acknowledgements</heading1>

The Software Assurance Maturity Model (SAMM) was originally developed, designed, and written 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. This document is

currently maintained and updated through the OpenSAMM Project led by Pravir Chandra. Since the

initial release of SAMM, this project has become part of the Open Web Application Security Project

(OWASP). Thanks also go to many supporting organizations that are listed on back cover.

<heading1>Contributors &amp; Reviewers</heading1> This work would not be possible without the

support of many individual reviewers and experts that offered contributions and critical feedback.

They are (in alphabetical order):<bullet>Fabio Arciniegas</bullet> <bullet>Matt Bartoldus</bullet>

<bullet>Sebastien Deleersnyder</bullet> <bullet>Jonathan Carter</bullet> <bullet>Darren

Challey</bullet> <bullet>Brian Chess</bullet> <bullet>Dinis Cruz</bullet> <bullet>Justin

Derry</bullet> <bullet>Bart De Win</bullet> <bullet>James McGovern</bullet> <bullet>Matteo

Meucci</bullet> <bullet>Jeff Payne</bullet> <bullet>Gunnar Peterson</bullet> <bullet>Jeff

Piper</bullet> <bullet>Andy Steingruebl</bullet> <bullet>John Steven</bullet> <bullet>Chad

Thunberg</bullet> <bullet>Colin Watson</bullet> <bullet>Jeff Williams</bullet> This is an OWASP

ProjectThe Open Web Application Security Project (OWASP) is a worldwide free and open community

focused on improving the security of application software. Our mission is to make application

security ìvisible,î so that people and organizations can make informed decisions about application

security risks. Every- one is free to participate in OWASP and all of our materials are available

under a free and open software license. The OWASP Foundation is a 501(c)3 not-for-profit

charitable organization that ensures the ongoing availability and support for our work. Visit OWASP

online at http://www.owasp.org.<heading1>License</heading1> This work is licensed under the

Creative Commons Attribution-Share Alike 3.0 License. To view a copy of this license, visit

http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 171 Second

Street, Suite 300, San Francisco, California, 94105, USA.</content>

</section>

<section id="executive-summary">

<title>Executive Summary</title>

<content>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:

<bullet><emph>Evaluating an organizationís existing software security practices</emph></bullet>

<bullet><emph>Building a balanced software security assurance program in well-defined

iterations</emph></bullet> <bullet><emph>Demonstrating concrete improvements to a security

assurance program</emph></bullet> <bullet><emph>Defining and measuring security-related activities

throughout an organization</emph></bullet> 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:

<bullet><emph>An organizationís behavior changes slowly over time</emph> - A successful software

security program should be specified in small iterations that deliver tangible assurance gains

while incrementally working toward long-term goals.</bullet> <bullet><emph>There is no single

recipe that works for all organizations</emph> - 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.</bullet> <bullet><emph>Guidance related to security activities must be

prescriptive</emph> - 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.</bullet> 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.</content>

</section>

<section-split number="1" id="understanding-the-model">

<title>Understanding the Model</title>

<subtitle>A view of the big picture</subtitle>

<content>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.</content>

</section-split>

<section id="business-functions">

<title>Business Functions</title>

<content><emph>At the highest level, SAMM defines four critical Business

Functions.</emph> Each Business Function (listed below) 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. <emph>For each

Business Function, SAMM defines three Security Practices.</emph> Each Security Practice (listed

opposite) is an area of security-related activities that build assurance for the related Business

Function. So overall, there are twelve Security Practices that are the independent silos for

improvement that map underneath the Business Functions of software development. <emph>For each

Security Practice, SAMM defines three Maturity Levels as Objectives.</emph> 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.<heading2>Maturity Levels</heading2> Each of the twelve Security Practices has three

defined Maturity Levels and an implicit starting point at zero. The details for each level differs

between the Practices, but they generally represent:<bullet>0 Implicit starting point representing

the activities in the Practice being unfulfilled</bullet><bullet>1 Initial understanding and ad hoc

provision of Security Practice</bullet><bullet>2 Increase efficiency and/or effectiveness of the

Security Practice</bullet><bullet>3 Comprehensive mastery of the Security Practice at

scale</bullet><heading2>Notation</heading2> Throughout this document, the following capitalized

terms will be reserved words that refer to the SAMM components defined in this section. If these

terms appear without capitalization, they should be interpreted based on the their context:

<bullet>Business Function also as Function</bullet> <bullet>Security Practice also as

Practice</bullet> <bullet>Maturity Level also as Level, Objective</bullet></content>

</section>

<section-split number="2" id="applying-the-model">

<title>Applying the Model</title>

<subtitle>Putting it all to work</subtitle>

<content>This section covers several important and useful applications of SAMM. Given

the core design of the model itself, an organization can use SAMM as a benchmark to measure its

security assurance program and create a scorecard. Using scorecards, an organization can

demonstrate improvement through iterations of developing an assurance program. And most

importantly, an organization can use SAMM roadmap templates to guide the build-out or improvement

of a security assurance initiative.</content>

</section-split>

<section id="using-maturity">

<title>Using the Maturity Levels</title>

<content>Each of the twelve Security Practices have three Maturity Levels. Each Level

has several components that specify the critical factors for understanding and achieving the stated

Level. Beyond that, these prescriptive details make it possible to use the definitions of the

Security Practices even outside the context of using SAMM to build a software assurance

program.<heading1>Objective</heading1> The Objective is a general statement that captures the

assurance goal of attaining the associated Level. As the Levels increase for a given Practice, the

Objectives characterize more sophisticated goals in terms of building assurance for software

development and operations. <heading1>Activities</heading1> The Activities are core requisites for

attaining the Level. Some are meant to be performed organization-wide and some correspond to

actions for individual project teams. In either case, the Activities capture the core security

function and organizations are free to determine how they fulfill the Activities.

<heading1>Results</heading1> The Results characterize capabilities and deliverables obtained by

achieving the given Level. In some cases these are specified concretely and in others, a more

qualitative statement is made about increased capability. <heading1>Success Metrics</heading1> The

Success Metrics specify example measurements that can be used to check if an organization is

performing at the given Level. Data collection and management is left to the choice of each

organization, but recommended data sources and thresholds are provided. <heading1>Costs</heading1>

The Costs are qualitative statements about the expenses incurred by an organization attaining the

given Level. While specific values will vary for each organizations, these are meant to provide an

idea of the one-time and ongoing costs associated with operating at a particular

Level.<heading1>Personnel</heading1> These properties of a Level indicate the estimated ongoing

overhead in terms of human resources for operating at the given Level.

<bullet><emph>Developers</emph> - Individuals performing detailed design and implementation of the

software</bullet> <bullet><emph>Architects</emph> - Individuals performing high-level design work

and large scale system engineering</bullet> <bullet><emph>Managers</emph> - Individuals performing

day-to-day management of development staff</bullet> <bullet><emph>QA Testers</emph> - Individuals

performing quality assurance testing and pre-release verification of software</bullet>

<bullet><emph>Security Auditors</emph> - Individuals with technical security knowledge related to

software being produced</bullet> <bullet><emph>Business Owners</emph> - Individuals performing key

decision making on software and its business requirements</bullet> <bullet><emph>Support

Operations</emph> - Individuals performing customer support or direct technical operations

support</bullet> <heading1>Related Levels</heading1> The Related Levels are references to Levels

within other Practices that have some potential overlaps depending upon the organizationís

structure and progress in building an assurance program. Functionally, these indicate synergies or

optimizations in Activity implementation if the Related Level is also a goal or already in

place.</content>

</section>

<section id="conducting-assessments">

<title>Conducting Assessments</title>

<content>By measuring an organization against the defined Security Practices, an

overall picture of built-in security assurance activities is created. This type of assessment is

useful for understanding the breadth of security activities currently in place at an organization.

Further, it enables that organization to then utilize SAMM to create a future roadmap for iterative

improvement. The process of conducting an assessment is simply evaluating an organization to

determine the Maturity Level at which it is performing, The extent to which an organizationís

performance is checked will usually vary according to the drivers behind the assessment, but in

general, there are two recommended styles: <bullet><emph>Lightweight</emph> - The assessment

worksheets for each Practice are evaluated and scores are assigned based on answers. This type of

assessment is usually sufficient for an organization that is trying to map their existing assurance

program into SAMM and just wants to get a quick picture of where they stand.</bullet>

<bullet><emph>Detailed</emph> - After completion of the assessment worksheets, additional audit

work is performed to check the organization to ensure the Activities prescribed by each Practice

are in place. Additionally since each Practice also specifies Success Metrics, that data should be

collected to ensure that the organization is performing as expected.</bullet> Scoring an

organization using the assessment worksheets is straightforward. After answering the questions,

evaluate the answer column to determine the Level. It is indicated by affirmative answers on all

questions above the markers to the right of the answer column. Existing 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. For such cases, the organizationís score should be

annotated with a ì+î symbol to indicate thereís 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 ì1+î

score. Likewise, an organization performing all Activities for a Security Practice, including some

beyond the scope of SAMM, would be given a "3+" score.Operational Enablement</content>

</section>

<section id="creating-scorecards">

<title>Creating Scorecards</title>

<content>Based on the scores assigned to each Security Practice, an organization can

create a scorecard to capture those values. Functionally, a scorecard can be the simple set of 12

scores for a particular time. However, selecting a time interval over which to generate a scorecard

facilitates understanding of overall changes in the assurance program during the time frame. Using

interval scorecards is encouraged for several situations: <bullet><emph>Gap analysis</emph> -

Capturing scores from detailed assessments versus expected performance levels</bullet>

<bullet><emph>Demonstrating improvement</emph> - Capturing scores from before and after an

iteration of assurance program build-out</bullet> <bullet><emph>Ongoing measurement</emph> -

Capturing scores over consistent time frames for an assurance program that is already in

place</bullet> The figure on the right shows an example scorecard for how an organizationís

assurance program changed over the course of one year. If that organization had also saved the data

about where they were planning on being at the end of the year, that would be another interesting

data set to plot since it would help show the extent to which the plans had to change over the

year.</content>

</section>

<section id="building-programs">

<title>Building Assurance Programs</title>

<content>One of the main uses of SAMM is to help organizations build software

security assurance programs. That process is straightforward, and generally begins with an

assessment if the organization is already performing some security assurance activities. Several

roadmap templates for common types of organizations are provided. Thus, many organizations can

choose an appropriate match and then tailor the roadmap template to their needs. For other types of

organizations, it may be necessary to build a custom roadmap. Roadmaps (pictured to the right)

consist of phases (the vertical bars) in which several Practices are each improved by one Level.

Therefore, building a roadmap entails selection of which Practices to improve in each planned

phase. Organizations are free to plan into the future as far as they wish, but are encouraged to

iterate based on business drivers and organization-specific information to ensure the assurance

goals are commensurate with their business goals and risk tolerance. After a roadmap is

established, the build-out of an assurance program is simple. An organization begins an improvement

phases and works to achieve the stated Levels by performing the prescribed Activities. At the end

of the phase, the roadmap should be adjusted based on what was actually accomplished, and then the

next phase can begin.</content>

</section>

<section id="roadmap-isv">

<title>Independent Software Vendor</title>

<subtitle>Roadmap template</subtitle>

<content><heading1>Rationale</heading1> An Independent Software Vendor involves the

core business function of building and selling software components and applications. Initial

drivers to limit common vulnerabilities affecting customers and users leads to early concentration

on Implementation Review and Security Testing activities. Shifting toward more proactive prevention

of security errors in product specification, an organization adds activities for Security

Requirements over time. Also, to minimize the impact from any discovered security issues, the

organization ramps up Issue Management activities over time. As the organization matures, knowledge

transfer activities from Operational Enablement are added to better inform customers and users

about secure operation of the software. <heading1>Additional Considerations</heading1>

<heading2>Outsourced Development</heading2> For organizations using external development resources,

restrictions on code access typically leads to prioritization of Security Requirements activities

instead of Implementation Review activities. Additionally, advancing Threat Assessment in earlier

phases would allow the organization to better clarify security needs to the outsourced developers.

Since expertise on software configuration will generally be strongest within the outsourced group,

contracts should be constructed to account for the activities related to Operational Enablement.

<heading2>Internet-Connected Applications</heading2> Organizations building applications that use

online resources have additional risks from the core internet-facing infrastructure that hosts the

internet-facing systems. To account for this risk, organizations should add activities from

Environment Hardening to their roadmaps. <heading2>Drivers and Embedded Development</heading2> For

organizations building low-level drivers or software for embedded systems, security vulnerabilities

in software design can be more damaging and costly to repair. Therefore, roadmaps should be

modified to emphasize Secure Architecture and Design Review activities in earlier phases.

<heading2>Organizations Grown by Acquisition</heading2> In an organization grown by acquisition,

there can often be several project teams following different development models with varying

degrees of security-related activities incorporated. An organization such as this may require a

separate roadmap for each division or project team to account for varying starting points as well

as project-specific concerns if a variety of software types are being developed.</content>

</section>

<section id="roadmap-osp">

<title>Online Service Provider</title>

<subtitle>Roadmap template</subtitle>

<content><heading1>Rationale</heading1> An Online Services Provider involves the core

business function of building web applications and other network-accessible interfaces. Initial

drivers to validate the overall soundness of design without stifling innovation lead to early

concentration on Design Review and Security Testing activities. Since critical systems will be

network-facing, Environment Hardening activities are also added early and ramped over time to

account for risks from the hosted environment. Though it can vary based on the core business of the

organizations, Policy &amp; Compliance activities should be started early and then advanced

according to the criticality of external compliance drivers. As the organization matures,

activities from Threat Assessment, Security Requirements, and Secure Architecture are slowly added

to help bolster proactive security after some baseline expectations for security have been

established. <heading1>Additional Considerations</heading1> <heading2>Outsourced

Development</heading2> For organizations using external development resources, restrictions on code

access typically leads to prioritization of Security Requirements activities instead of

Implementation Review activities. Additionally, advancing Threat Assessment in earlier phases would

allow the organization to better clarify security needs to the outsourced developers. Since

expertise on software configuration will generally be strongest within the outsourced group,

contracts should be constructed to account for the activities related to Operational Enablement.

<heading2>Online Payment Processing</heading2> Organizations required to be in compliance with the

Payment Card Industry Data Security Standard (PCI-DSS) or other online payment standards should

place activities from Policy &amp; Compliance in earlier phases of the roadmap. This allows the

organization to opportunistically establish activities that ensure compliance and enable the future

roadmap to be tailored accordingly. <heading2>Web Services Platforms</heading2> For organizations

building web services platforms, design errors can carry additional risks and be more costly to

mitigate. Therefore, activities from Threat Assessment, Security Requirements, and Secure

Architecture should be placed in earlier phases of the roadmap. <heading2>Organizations Grown by

Acquisition</heading2> In an organization grown by acquisition, there can often be several project

teams following different development models with varying degrees of security-related activities

incorporated. An organization such as this may require a separate roadmap for each division or

project team to account for varying starting points as well as project-specific concerns if a

variety of software types are being developed.</content>

</section>

<section id="roadmap-fso">

<title>Financial Services Organization</title>

<subtitle>Roadmap template</subtitle>

<content><heading1>Rationale</heading1> A Financial Services Organization involves

the core business function of building systems to support financial transactions and processing. In

general, this implies a greater concentration of internal and back-end systems that interface with

disparate external data providers. Initially, effort is focused on improving the Practices related

to Governance since these are critical services that set the baseline for the assurance program and

help meet compliance requirements for the organization. Since building secure and reliable software

proactively is an overall goal, Practices within Construction are started early on and ramped up

sharply as the program matures. Verification activities are also ramped up smoothly over the course

of the roadmap to handle legacy systems without creating unrealistic expectations. Additionally,

this helps ensure enough cycles are spent building out more proactive Practices. Since a financial

services organization often operates the software they build, focus is given to the Practices

within Operations during the middle of the roadmap after some initial Governance is in place but

before heavy focus is given to the proactive Construction Practices. <heading1>Additional

Considerations</heading1> <heading2>Outsourced Development</heading2> For organizations using

external development resources, restrictions on code access typically leads to prioritization of

Security Requirements activities instead of Implementation Review activities. Additionally,

advancing Threat Assessment in earlier phases would allow the organization to better clarify

security needs to the outsourced developers. Since expertise on software configuration will

generally be strongest within the outsourced group, contracts should be constructed to account for

the activities related to Operational Enablement. <heading2>Web Services Platforms</heading2> For

organizations building web services platforms, design errors can carry additional risks and be more

costly to mitigate. Therefore, activities from Threat Assessment, Security Requirements, and Secure

Architecture should be placed in earlier phases of the roadmap. <heading2>Organizations Grown by

Acquisition</heading2> In an organization grown by acquisition, there can often be several project

teams following different development models with varying degrees of security-related activities

incorporated. An organization such as this may require a separate roadmap for each division or

project team to account for varying starting points as well as project-specific concerns if a

variety of software types are being developed.</content>

</section>

<section id="roadmap-go">

<title>Government Organization</title>

<subtitle>Roadmap template</subtitle>

<content><heading1>Rationale</heading1> A Government Organization involves the core

business function of being a state-affiliated organization that builds software to support public

sector projects. Initially, Governance Practices are established, generally to get an idea of the

overall compliance burden for the organization in context of the concrete roadmap for improvement.

Because of risks of public exposure and the quantity of legacy code generally in place, early

emphasis is given to Security Testing within the Verification Practices and later the more involved

Implementation Review or Design Review Practices are developed. Similar emphasis is placed on the

Construction and Operations Practices. This helps establish the organizationís management of

vulnerabilities and moves toward bolstering the security posture of the operating environment. At

the same time, proactive security activities under Construction are built up to help prevent new

issues in software under development. <heading1>Additional Considerations</heading1>

<heading2>Outsourced Development</heading2> For organizations using external development resources,

restrictions on code access typically leads to prioritization of Security Requirements activities

instead of Implementation Review activities. Additionally, advancing Threat Assessment in earlier

phases would allow the organization to better clarify security needs to the outsourced developers.

Since expertise on software configuration will generally be strongest within the outsourced group,

contracts should be constructed to account for the activities related to Operational Enablement.

<heading2>Web Services Platforms</heading2> For organizations building web services platforms,

design errors can carry additional risks and be more costly to mitigate. Therefore, activities from

Threat Assessment, Security Requirements, and Secure Architecture should be placed in earlier

phases of the roadmap. <heading2>Regulatory Compliance</heading2> For organizations under heavy

regulations that affect business processes, the build-out of the Policy &amp; Compliance Practice

should be adjusted to accommodate external drivers. Likewise, organizations under a lighter

compliance load should take the opportunity to push back build-out of that Practice in favor of

others.</content>

</section>

<section-split number="3" id="security-practices">

<title>The Security Practices</title>

<subtitle>An explanation of the details</subtitle>

<content>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.</content>

</section-split>

<section-split number="4" id="case-studies">

<title>Case Studies</title>

<subtitle>A walkthrough of example scenarios</subtitle>

<content>This section features a selection of scenarios in which the application of

SAMM is explained in the context of a specific business case. Using the roadmap templates as a

guide, the case studies tell the story of how an organization might adapt best practices and take

into account organization-specific risks when building a security assurance program.</content>

</section-split>

<misc-text>

<entry>Contents</entry>

<entry>I would like to...</entry>

<entry>Assess existing software assurance practices</entry>

<entry>Build a strategic roadmap for an organization</entry>

<entry>Implement or perform security activities</entry>

<entry>Description of Security Practices</entry>

<entry>Activities overview</entry>

<entry>Assessment worksheet</entry>

<entry> read </entry>

<entry> skim</entry>

<entry>assessment scores</entry>

</misc-text>

<level-data>

<level-value id="0" value="0.0"/>

<level-value id="0+" value="0.5"/>

<level-value id="1" value="1.0"/>

<level-value id="1+" value="1.5"/>

<level-value id="2" value="2.0"/>

<level-value id="2+" value="2.5"/>

<level-value id="3" value="3.0"/>

<level-value id="3+" value="3.5"/>

</level-data>

</SAMM>