PRESENTING

Table of Contents

Overview

What is OWASP SAMM?

SAMM stands for Software Assurance Maturity Model.

Our mission is to provide an effective and measurable way for all types of organizations to analyze and improve their software security posture. We want to raise awareness and educate organizations on how to design, develop, and deploy secure software through our self-assessment model. SAMM supports the complete software lifecycle and is technology and process agnostic. We built SAMM to be evolutive and risk-driven in nature, as there is no single recipe that works for all organizations.

The three main characteristics of SAMM are:

- Measurable: Defined maturity levels across security practices
- Actionable: Clear pathways for improving maturity levels
- Versatile: Technology, process, and organization agnostic

The OWASP SAMM community is powered by security knowledgeable volunteers from businesses and educational organizations. The global community works to create freely-available articles, methodologies, documentation, tools, and technologies.

The OWASP SAMM Model

SAMM is a prescriptive model, an open framework which is simple to use, fully defined, and measurable. The solution details are easy enough to follow even for non-security personnel. It helps organizations analyze their current software security practices, build a security program in defined iterations, show progressive improvements in secure practices, define, and measure security-related activities.

SAMM was defined with flexibility in mind so that small, medium, and large organizations using any style of development can customize and adopt it. It provides a means of knowing where your organization is on its journey towards software assurance and understanding what is recommended to move to the next level of maturity.

SAMM does not insist that all organizations achieve the maximum maturity level in every category. Each organization can determine the target maturity level for each Security Practice that is the best fit and adapt the templates provided for their specific needs.

OWASP SAMM structure

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

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

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

For each security practice, SAMM defines two streams. Each stream has an objective to be reached, and this objective can be reached in increasing levels of maturity. Streams align and link the activities in the practice over the different maturity levels.

The structure and setup of the SAMM model support

- the assessment of the organization’s current software security posture
- the definition of the organization’s target
- the definition of an implementation roadmap to get there
- prescriptive advice on how to implement particular activities

Overview of Business Functions

The SAMM model contains five business functions with three security practices each. This section provides a high level overview of each of the business functions.

Governance

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

Strategy & Metrics This practice forms the basis of your secure software activities by building an overall plan.

Policy & Compliance This practice drives the adherence to internal and external standards and regulations.

Education & Guidance This practice focuses on increasing the knowledge in the organization regarding secure software.

Design

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

Threat Assessment This practice focuses on identifying potential threats in applications.

Security Requirements This practice focuses on defining appropriate security requirements for your software and your software suppliers.

Security Architecture The security architecture practice focuses on managing architectural risks for the software solution.

Implementation

Implementation is focused on the processes and activities related to how an organization builds and deploys software components and its related defects.

Activities within the Implementation function have the most impact on the daily life of developers. The joint goal is to ship reliably working software with minimum defects.

Secure Build This practice focuses on creating a consistently repeatable build process and accounting for the security of application dependencies.

Secure Deployment This practice focuses on increasing the security of software deployments to the production environment and the supporting secrets.

Defect Management This practice focuses on managing security defects in software and their associated metrics.

Verification

Verification focuses 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.

Architecture Assessment This practice focuses on validating the security and compliance of the software and supporting infrastructure architecture.

Requirements-driven Testing This practice focuses on using both positive (control verification) and negative (misuse/abuse testing) security tests based on requirements (user stories).

Security Testing This practice focuses on the detection and resolution of basic security issues through automation, allowing manual testing to focus on more complex attack vectors.

Operations

The Operations Business Function encompasses those activities necessary to ensure confidentiality, integrity, and availability are maintained throughout the operational lifetime of an application and its associated data. Increased maturity with regard to this Business Function provides greater assurance that the organization is resilient in the face of operational disruptions, and responsive to changes in the operational landscape.

Incident Management This practice addresses activities carried out improve the organization's detection of, and response to, security incidents.

Environment Management This practice describes proactive activities carried out to improve and maintain the security of the environments in which the organization's applications operate.

Operational Management This practice focuses on operational support activities required to maintain security throughout the product lifecycle.

Overview of the Security Practices

This section presents the fifteen security practices in more detail, in order to understand their objectives and the types of activities they contain.

Governance

This business function contains the following three security practices.

Strategy & Metrics

Software assurance entails many different activities and concerns. Without an overall plan, you might be spending a lot of effort to build in security, while in fact your efforts may be unaligned, disproportional or even counterproductive. The goal of the Strategy & Metrics (SM) practice is to build an efficient and effective plan for realizing your software security objectives within your organization.

A software security program, that selects and prioritizes activities of the rest of the model, serves as the foundation for your efforts. The practice works on building the plan, maintaining and disseminating it.

At the same time, you want to keep track of your security posture and program improvements. A metrics-driven approach is included to ensure an accurate view on your activities. To measure is to know.

Streams

- Create and Promote: This stream is about creating and promoting an application security roadmap to set the objectives of the enterprise on this topic and increase alignment among stakeholders.
- Measure and Improve: This stream aims to drive the validity, relevance, and improvement of the application security roadmap through measurements of performance within the organization.

Policy & Compliance

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

A driving theme for improvement within this practice is describing organization's standards and 3rd party obligations as application requirements, enabling efficient and automated audits that may be leveraged within the SDLC and continuously demonstrate that all expectations are met.

In a sophisticated form, provision of this practice entails an 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.

Streams

- Policy & Standards: This stream focuses on maintaining policies and standards and providing them to support integration into the SDLC.
- Compliance Management: This stream focuses on identifying and providing compliance requirements to support integration into the SDLC.

Education & Guidance

The Education & Guidance (EG) practice focuses on arming personnel involved in the software lifecycle with knowledge and resources to design, develop, and deploy secure software. With improved access to information, project teams can 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 and increasing their security awareness, either through instructor-led sessions or computer-based modules. As an organization progresses, it builds a broad base of training starting with developers and moving to other roles, culminating with the addition of role-based training to ensure applicability and effectiveness.

In addition to training, this practice also requires the organization to make a significant investment in improving organizational culture to promote application security through collaboration between teams. Collaboration tools and increased transparency between technologies and tools support this approach to improve the security of the applications.

Streams

- Training and Awareness: Training and awareness focuses on increasing the overall knowledge around software security anmong the different stakeholders within the organization.
- Organization and Culture: Organization and culture focuses on promoting the culture of application security within the organization as an important success factor of an SDLC project.

Design

This business function contains the following three security practices.

Threat Assessment

The Threat Assessment (TA) practice focuses on identifying and understanding of 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 application risk profiles, an organization improves over time. Ultimately, a sophisticated organization would maintain this information in a way that is tightly coupled to the compensating factors and pass-through risks from external entities. This provides greater breadth of understanding for potential downstream impacts from security issues while keeping a close watch on the organization’s current performance against known threats.

Streams

- Application Risk Profile: An application risk profile helps to identify which applications can pose a serious threat to the organization if they were attacked or breached.
- Threat Modeling: Threat modeling is intended to help software development teams understand what risks exist in what is being built, what could go wrong, and how we the risks can be mitigated or remediated.

Security Requirements

The Security Requirements (SR) practice focuses on security requirements that are important in the context of secure software. A first type deals with typical software-related requirements, to specify objectives and expectations to protect the service and data at the core of the application. A second type deals with requirements relative to supplier organizations that are part of the development context of the application, in particular for outsourced development. It is important to streamline the expectations in terms of secure development because outsourced development can have significant impact on the security of the application. The security of 3rd party (technical) libraries is part of the software supply chains stream (see Secure Build), and it is not included in this practice.

Streams

- Software Requirements: Software requirements specify objectives and expectations to protect the service and data at the core of the application.
- Supplier Security: Supplier security deals with requirements that are relative to supplier organizations within the development context of the application, in particular for outsourced development.

Security Architecture

The Security Architecture (SA) practice focuses on the security linked to components and technology you deal with during the architectural design of your software. Secure Architecture Design looks at the selection and composition of components that form the foundation of your solution, focusing on its security properties. Technology management looks at the security of supporting technologies used during development, deployment and operations, such as development stacks and tooling, deployment tooling, and operating systems and tooling.

Streams

- Architecture Design: The design of a software architecture can significantly impact the security posture of a software and the use of good security practices will improve the overall design.
- Technology Management: Technologies and frameworks are the cornerstones of any software solution. The security properties of these must be looked into to ensure an appropriate security level and to anticipate any potential issues herein.

Implementation

This business function contains the following three security practices.

Secure Build

The Secure Build (SB) practice emphasises the importance of building software in a standardised, repeatable manner, and of doing so using secure components, including 3rd party software dependencies.

The first stream focuses on removing any subjectivity from the build process by striving for full automation. An automated build pipeline can include additional automated security checks such as SAST and DAST to gain further assurance and flag security regressions early by failing the build, for example.

The second stream acknowledges the prevalence of software dependencies in modern applications. It aims to identify them and track their security status in order to contain the impact of their insecurity on an otherwise secure application. In an advanced form, it applies similar security checks to software dependencies as to the application itself.

Streams

- Build Process: A consistent build process ensures the software you are deploying is predictable and directly linked to the source code. Furthermore, you can take advantage of the software build process for various security activities.
- Software Dependencies: External libraries are a significant part of modern software. The activities in this stream help create a view of external libraries and ensure their security strength is adequate.

Secure Deployment

One of the final stages in delivering secure software is ensuring the security and integrity of developed applications are not compromised during deployment. The Secure Deployment (SD) practice focuses on this. To this end, the practice’s first stream focuses on removing manual error by automating the deployment process as much as possible, and making its success contingent upon the outcomes of integrated security verification checks. It also fosters Separation of Duties by making adequately trained, non-developers responsible for deployment.

The second stream goes beyond the mechanics of deployment, and focuses on protecting the privacy and integrity of sensitive data, such as passwords, tokens, and other secrets, required for applications to operate in production environments. In its simplest form, suitable production secrets are moved from repositories and configuration files into adequately managed digital vaults. In more advanced forms, secrets are dynamically generated at deployment time and routine processes detect and mitigate the presence of any unprotected secrets in the environment.

Streams

- Deployment Process: A repeatable and consistent deployment process ensures you only deploy correct software artifacts to production. It also paves the way for representative test environments prior to production.
- Secret Management: As the secure execution of any software system requires credentials, this stream ensures a proper handling of these sensitive data elements within the organization’s environment.

Defect Management

The Defect Management (DM) practice focuses on collecting, recording, and analyzing software security defects and enriching them with information to drive metrics-based decisions.

The practice's first stream deals with the process of handling and managing defects to ensure released software has a given assurance level. The second stream focuses on enriching the information about the defects and deriving metrics to guide decisions about the security of individual projects and of the security assurance program as a whole.

In a sophisticated form, the practice requires formalised, independent defect management and real-time, correlated information to detect trends and influence security strategy.

Streams

- Defect Tracking: Defect tracking manages the collection and follow-up of all potential issues in a piece of software, from architectural flaws to coding issues and run-time vulnerabilities.
- Metrics and Feedback: Defect tracking can drive the improvement of security activities within the organization through metrics and feedback.

Verification

This business function contains the following three security practices.

Architecture Assessment

The Architecture Assessment (AA) practice ensures that the application and infrastructure architecture adequately meets all relevant security and compliance requirements, and sufficiently mitigates identified security threats. The first stream focuses on verifying that the security and compliance requirements identified in the Policy & Compliance and Security Requirements practices are met, first in an ad-hoc manner, then more systematically for each interface in the system. The second stream reviews the architecture, first for mitigations against typical threats, then against the specific threats identified in the Threat Assessment practice.

In its more advanced form, the practice formalizes the security architecture review process, continuously evaluates the effectiveness of the architecture's security controls, their scalability and strategic alignment. Identified weaknesses and possible improvements are fed back to the Security Architecture practice to improve reference architectures.

Streams

- Architecture Validation: Architecture validation confirms the security of the software and supporting architecture by identifying application and infrastructure architecture components and verifying their provision of security objectives and requirements.
- Architecture Mitigation: Architecture mitigation focuses on ensuring that all threats identified during Threat Assessment are adequately mitigated, and existing reference architectures updated to address any unhandled threats.

Requirements-driven Testing

The goal of the Requirements-driven Testing (RT) practice is to ensure that the implemented security controls operate as expected and satisfy the project's stated security requirements. It does so by incrementally building a set of security test and regression cases and executing them regularly.

A key aspect of this practice is its attention to both positive and negative testing. The former verifies that the application's security controls satisfy stated security requirements and validates their correct functioning. These requirements are typically functional in nature. Negative testing addresses the quality of the implementation of the security controls and aims to detect unexpected design flaws and implementation bugs through misuse and abuse testing. In its more advanced forms, the practice promotes security stress testing, such as denial of service, and strives to continuously improve application security by consistently automating security unit tests and creating security regression tests for all bugs identified and fixed.

Although both the Requirements-driven Testing and Security Testing practices are concerned with security testing, the former focuses on verifying the correct implementation of security requirements, while the latter aims to uncover technical implementation weaknesses in an application, irrespective of requirements.

Streams

- Control Verification: Control Verification validates that security controls and requirements are met through

testing derived from requirements, and prevents the introduction of bugs into later releases through regression testing.

- Misuse/Abuse Testing: Misuse/Abuse Testing leverages fuzzing, misuse/abuse cases, and the identification of any functionality or resources in the software that can be abused in order to identify weaknesses in features to attack an application.

Security Testing

The Security Testing (ST) practice leverages the fact that, while automated security testing is fast and scales well to numerous applications, in-depth testing based on good knowledge of an application and its business logic is often only possible via slower, manual expert security testing. Each stream therefore has one approach at its core.

The first stream focuses on establishing a common security baseline to automatically detect so-called “low hanging fruit”. Progressively customize the automated tests for each application and increase their frequency of execution to detect more bugs and regressions earlier, as close as possible to their inception. The more bugs the automated processes can detect, the more time experts have to use their knowledge and creativity to focus on more complex attack vectors and ensure in-depth application testing in the second stream. As manual review is slow and hard to scale, reviewers prioritize testing components based on their risk, recent relevant changes, or upcoming major releases. Organizations can also access external expertise by participating in bug bounty programs, for example.

Unlike the Requirements-driven testing practice which focuses on verifying that applications correctly implement their requirements, the goal of this practice is to uncover technical and business-logic weaknesses in application and make them visible to management and business stakeholders, irrespective of requirements.

Streams

- Scalable Baseline: Scalable baseline focuses on the use of application-specific automated testing tools that integrate security validation into the build and deploy process. The goal of this stream is to favor width (a broad spectrum of applications) over depth of testing.
- Deep Understanding: Deep understanding focuses on performing manual security testing of high-risk components, using complex attack vectors with the goal of making advanced security testing an integral part of the development process. The goal of this stream is to favor testing depth (testing rigour) over testing width (the portfolio of applications).

Operations

This business function contains the following three security practices.

Incident Management

Once your organization has applications in operation, you’re likely to face security incidents. In this model, we define a security incident as a breach, or the threat of an imminent breach, of at least one asset’s security goals, whether due to malicious or negligent behavior. Examples of security incidents might include: a successful Denial of Service (DoS) attack against a cloud application, an application user accessing private data of another by abusing a security vulnerability, or an attacker modifying application source code. The Incident Management (IM) practice focuses on dealing with these in your organization.

Historically, many security incidents have been detected months, or even years, after the initial breach. During the “dwell time” before an incident is detected, significant damage can occur, increasing the difficulty of recovery. Our first activity stream, Incident Detection, focuses on decreasing that dwell time.

Once you have identified that you’re suffering from a security incident, it’s essential to respond in a disciplined, thorough manner to limit the damage, and return to normal operations as efficiently as possible. This is the focus of our second stream.

Streams

- Incident Detection: Incident Detection refers to the process of determining an identified security-relevant event is, in fact, a security incident. The activities in this stream focus on the organization’s ability to identify security incidents when they occur, and to initiate appropriate incident response activities.
- Incident Response: Incident Response starts the moment you acknowledge and verify the existence of a security incident. Your goal is to act in a coordinated and efficient way so that further damage is limited

as much as possible. The activities in this stream focus on the organization’s ability to respond appropriately and effectively to reported security incidents.

Environment Management

The organization’s work on application security doesn’t end once the application becomes operational. New security features and patches are regularly released for the various elements of the technology stack you’re using, until they become obsolete or are no longer supported.

Most of the technologies in any application stack are not secure by default. This is frequently intentional, to enhance backwards compatibility or ease of setup. For this reason, ensuring the secure operation of the organization’s technology stack requires the consistent application of secure baseline configurations to all components. The Environment Management (EM) practice focuses on keeping your environment clean and secure.

Vulnerabilities are discovered throughout the lifecycles of the technologies on which your organization relies, and new versions addressing them are released on various schedules. This makes it essential to monitor vulnerability reports and perform orderly, timely patching across all affected systems.

Streams

- Configuration Hardening: The activities in this stream focus on the organization’s management of security-related configurations in all elements of the technology stack. The emphasis is on those elements (e.g., operating systems, containers, frameworks, services, appliances, and libraries) obtained from third parties, because their architecture and design are not under the organization’s control.
- Patching and Updating: The activities in this stream focus on the organization’s handling of patches and updates for all elements of the technology stack. For software developed by the organization, these activities are concerned with delivering patches and updates to customers, as well as applying them to organization-managed solutions (e.g., software as a service). For third-party elements, these activities are concerned with the organization’s timely application of updates and patches received.

Operational Management

The Operational Management (OM) practice focuses on activities to ensure security is maintained throughout operational support functions. Although these functions are not performed directly by an application, the overall security of the application and its data depends on their proper performance. Deploying an application on an unsupported operating system with unpatched vulnerabilities, or failing to store backup media securely, can make the protections built into that application irrelevant.

The functions covered by this practice include, but are not limited to: system provisioning, administration, and decommissioning; database provisioning and administration; and data backup, restore, and archival.

Streams

- Data Protection: The activities in this stream focus on ensuring the organization properly protects data in all aspects of their creation, handling, storage, and processing.
- System Decomissioning / Legacy Management: From the perspective of the organization as a consumer of resources, the activities in this stream focus on the identification, management, and tracking of systems, applications, application dependencies, and services that are no longer used, have reached end of life, or are no longer actively developed or supported. Removal of unused systems and services improves manageability of the environment and reduces the organization’s attack surface, while affording direct and indirect cost savings (e.g., reduced license count, reduced logging volume, or reduced analyst effort).

Overview of Security Activities

This section provides further details at the level of individual activities. Per activity, we discuss the benefit, provide a more detailed description, assessment questions and linked quality criteria.

Governance

Strategy & Metrics (SM1)

The goal of this practice level is to identify objectives and means of measuring effectiveness of the security program.

Activities

[Stream A] Identify the organization’s risk appetite

Benefit: *Common understanding of your organization’s security posture*

Understand, based on application risk exposure, what threats exist or may exist, as well as how tolerant executive leadership is of these risks. This understanding is a key component of determining software security assurance priorities. To ascertain these threats, interview business owners and stakeholders and document drivers specific to industries where the organization operates as well as drivers specific to the organization. Gathered information includes worst-case scenarios that could impact the organization, as well as opportunities where an optimized software development lifecycle and more secure applications could provide a market-differentiator or create additional opportunities.

Gathered information provides a baseline for the organization to develop and promote its application security program. Items in the program are prioritized to address threats and opportunities most important to the organization. The baseline is split into several risk factors and drivers linked directly to the organization’s priorities and used to help build a risk profile of each custom-developed application by documenting how they can impact the organization if they are compromised.

The baseline and individual risk factors should be published and made available to application development teams to ensure a more transparent process of creating application risk profiles and incorporating the organization’s priorities into the program. Additionally, these goals should provide a set of objectives which should be used to ensure all application security program enhancements provide direct support of the organization’s current and future needs.

Assessment Question(s)

Do you understand the enterprise-wide risk appetite for your applications ?

- No
- Yes, it covers general risks
- Yes, it covers organization-specific risks
- Yes, it covers risks and opportunities

Quality Criteria:

- You capture the risk appetite of your organization’s executive leadership
- The organization’s leadership vet and approve the set of risks
- You identify the main business and technical threats to your assets and data
- You document risks and store them in an accessible location

[Stream B] Define basic security metrics

Benefit: *Basic insights into your AppSec program’s effectiveness and efficiency*

Define and document metrics to evaluate the effectiveness and efficiency of the application security program. This way improvements are measurable and you can use them to secure future support and funding for the program. Considering the dynamic nature of most development environments, metrics should be comprised of measurements in the following categories

- Effort metrics measure the effort spent on security. For example training hours, time spent performing code reviews, and number of applications scanned for vulnerabilities.
- Result metrics measure the results of security efforts. Examples include number of unpatched security defects and number of security incidents involving application vulnerabilities.
- Environment metrics measure the environment where security efforts take place. Examples include number of applications or lines of code as a measure of difficulty or complexity.

Each measure by itself is useful for a specific purpose, but a combination of two or three metrics together helps explain spikes in metrics trends. For example, a spike in a total number of vulnerabilities may be caused by the organization on-boarding several new applications that have not been previously exposed to the implemented application security mechanisms. Alternatively, an increase in the environment metrics without a corresponding increase in the effort or result could be an indicator of a mature and efficient security program.

While identifying metrics, it’s always recommended to stick to the metrics that meet several criteria

- Consistently Measured
- Inexpensive to gather

- Expressed as a cardinal number or a percentage
- Expressed as a unit of measure

Document metrics and include descriptions of best and most efficient methods for gathering data, as well as recommended methods for combining individual measures into meaningful metrics. For example, a number of applications and a total number of defects across all applications may not be useful by themselves but, when combined as a number of outstanding high-severity defects per application, they provide a more actionable metric.

Assessment Question(s)

Do you use a set of metrics to measure the effectiveness and efficiency of the application security program across applications?

- No
- Yes, for one metrics category
- Yes, for two metrics categories
- Yes, for all three metrics categories

Quality Criteria:

- You document each metric, including a description of the sources, measurement coverage, and guidance on how to use it to explain application security trends
- Metrics include measures of efforts, results, and the environment measurement categories
- Most of the metrics are frequently measured, easy or inexpensive to gather, and expressed as a cardinal number or a percentage
- Application security and development teams publish metrics

Strategy & Metrics (SM2)

The goal of this practice level is to establish a unified strategic roadmap for software security within the organization.

Activities

[Stream A] Define the security strategy

Benefit: *Available and agreed up roadmap of your AppSec program*

Based on the magnitude of assets, threats, and risk tolerance, develop a security strategic plan and budget to address business priorities around application security. The plan covers 1 to 3 years and includes milestones consistent with the organization’s business drivers and risks. It provides tactical and strategic initiatives and follows a roadmap that makes its alignment with business priorities and needs visible.

In the roadmap, you reach a balance between changes requiring financial expenditures, changes of processes and procedures, and changes impacting the organization’s culture. This balance helps accomplish multiple milestones concurrently and without overloading or exhausting available resources or development teams. The milestones are frequent enough to help monitor program success and trigger timely roadmap adjustments.

For the program to be successful, the application security team obtains buy-in from the organization’s stakeholders and application development teams. A published plan is available to anyone who is required to support or participate in its implementation.

Assessment Question(s)

Do you have a strategic plan for application security and use it to make decisions?

- No
- Yes, we review it annually
- Yes, we consult the plan before making significant decisions
- Yes, we consult the plan often, and it is aligned with our application security strategy

Quality Criteria:

- The plan reflects the organization’s business priorities and risk appetite
- The plan includes measurable milestones and a budget
- The plan is consistent with the organization’s business drivers and risks
- The plan lays out a roadmap for strategic and tactical initiatives

- You have buy-in from stakeholders, including development teams

[Stream B] Set strategic KPIs

Benefit: *Transparency on your AppSec program's performance*

Once the organization has defined its application security metrics, collect enough information to establish realistic goals. Test identified metrics to ensure you can gather data consistently and efficiently over a short period. After the initial testing period, the organization should have enough information to commit to goals and objectives expressed through Key Performance Indicators (KPIs).

While several measurements are useful for monitoring the information security program and its effectiveness, KPIs are comprised of the most meaningful and effective metrics. Aim to remove volatility common in application development environments from KPIs to reduce chances of unfavorable numbers resulting from temporary or misleading individual measurements. Base KPIs on metrics considered valuable not only to Information Security professionals but also to individuals responsible for the overall success of the application, and organization's leadership. View KPIs as definitive indicators of the success of the whole program and consider them actionable.

Fully document KPIs and distribute them to the teams contributing to the success of the program as well as organization's leadership. Ideally, include a brief explanation of the information sources for each KPI and the meaning if the numbers are high or low. Include short and long-term goals, and ranges for unacceptable measurements requiring immediate intervention. Share action plans with application security and application development teams to ensure full transparency in understanding of the organization's objectives and goals.

Assessment Question(s)

Did you define Key Perfomance Indicators (KPI) from available application security metrics?

- No
- Yes, for some of the metrics
- Yes, for at least half of the metrics
- Yes, for most or all of the metrics

Quality Criteria:

- You defined KPIs after gathering enough information to establish realistic objectives
- You developed KPIs with the buy-in from the leadership and teams responsible for application security
- KPIs are available to the application teams and include acceptability thresholds and guidance in case teams need to take action
- Success of the application security program is clearly visible based on defined KPIs

Strategy & Metrics (SM3)

The goal of this practice level is to align security efforts with the relevant organizational indicators and asset values.

Activities

[Stream A] Align security and business strategies

Benefit: *Continuous AppSec program alignment with the organization's business goals*

You review the application security plan periodically for ongoing applicability and support of the organization's evolving needs and future growth. To do this, you repeat the steps from the first two maturity levels of this Security Practice at least annually. The goal is for the plan to always support the current and future needs of the organization, which ensures the program is aligned with the business.

In addition to reviewing the business drivers, the organization closely monitors the success of the implementation of each of the roadmap milestones. You evaluate the success of the milestones based on a wide range of criteria, including completeness and efficiency of the implementation, budget considerations, and any cultural impacts or changes resulting from the initiative. You review missed or unsatisfactory milestones and evaluate possible changes to the overall program.

The organization develops dashboards and measurements for management and teams responsible for software development to monitor the implementation of the roadmap. These dashboards are detailed enough to identify individual projects and initiatives and provide a clear understanding of whether the program is successful and aligned with the organization's needs.

Assessment Question(s)

Do you regularly review and update the Strategic Plan for Application Security?

- No
- Yes, but review is ad-hoc
- Yes, we review it at regular times
- Yes, we review it at least annually

Quality Criteria:

- You review and update the plan in response to significant changes in the business environment, the organization, or its risk appetite
- Plan update steps include reviewing the plan with all the stakeholders and updating the business drivers and strategies
- You adjust the plan and roadmap based on lessons learned from completed roadmap activities
- You publish progress information on roadmap activities, making sure they are available to all stakeholders

[Stream B] Drive the security program through metrics

Benefit: *Continuous improvement of your program according to results*

Define guidelines for influencing the Application Security program based on the KPIs and other application security metrics. These guidelines combine the maturity of the application development process and procedures with different metrics to make the program more efficient. The following examples show a relationship between measurements and ways of evolving and improving application security

- Focus on maturity of the development lifecycle makes the relative cost per defect lower by applying security proactively.
- Monitoring the balance between effort, result, and environment metrics improves the program's efficiency and justifies additional automation and other methods for improving the overall application security baselines.
- Individual Security Practices could provide indicators of success or failure of individual application security initiatives.
- Effort metrics helps ensure application security work is directed at the more relevant and important technologies and disciplines.

When defining the overall metrics strategy, keep the end-goal in mind and define what decisions can be made as a result of changes in KPIs and metrics as soon as possible, to help guide development of metrics.

Assessment Question(s)

Do you update the Application Security strategy and roadmap based on application security metrics and KPIs?

- No
- Yes, but review is ad-hoc
- Yes, we review it at regular times
- Yes, we review it at least annually

Quality Criteria:

- You review KPIs at least yearly for their efficiency and effectiveness
- KPIs and application security metrics trigger most of the changes to the application security strategy

Policy & Compliance (PC1)

The goal of this practice level is to identify and document governance and compliance drivers relevant to the organization.

Activities

[Stream A] Define policies and standards

Benefit: *Clear expectation of minimum security level in the organization*

Develop a library of policies and standards to govern all aspects of software development in the organization. Policies and standards are based on existing industry standards and appropriate for the

organization’s industry. Due to the full range of technology-specific limitations and best practices, review proposed standards with the various product teams. With the overarching objective of increasing security of the applications and computing infrastructure, invite product teams to offer feedback on any aspects of the standards that would not be feasible or cost-effective to implement, as well as opportunities for standards to go further with little effort on the product teams.

For policies, emphasize high-level definitions and aspects of application security that do not depend on specific technology or hosting environment. Focus on broader objectives of the organization to protect the integrity of its computing environment, safety and privacy of the data, and maturity of the software development life-cycles. For larger organizations, policies may qualify specific requirements based on data classification or application functionality, but should not be detailed enough to offer technology-specific guidance.

For standards, incorporate requirements set forth by policies, and focus on technology-specific implementation guidance intended to capture and take advantage of the security features of different programming languages and frameworks. Standards require input from senior developers and architects considered experts in various technologies in use by the organization. Create them in a format that allows for periodic updates. Label or tag individual requirements with the policy or a 3rd party requirement, to make maintenance and audits easier and more efficient.

Assessment Question(s)

Do you have and apply a common set of policies and standards throughout your organization?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have adapted existing standards appropriate for the organization’s industry to account for domain-specific considerations
- Your standards are aligned with your policies and incorporate technology-specific implementation guidance

[Stream B] Identify compliance requirements

Benefit: *Security policies and standards aligned with external compliance drivers*

Create a comprehensive list of all compliance requirements, including any triggers that could help determine which applications are in scope. Compliance requirements may be considered in scope based on factors such as geographic location, types of data, or contractual obligations with clients or business partners. Review each identified compliance requirement with the appropriate experts and legal, to ensure the obligation is understood. Since many compliance obligations vary in applicability based on how the data is processed, stored, or transmitted across the computing environment, compliance drivers should always indicate opportunities for lowering the overall compliance burden by changing how the data is handled.

Evaluate publishing a compliance matrix to help identify which factors could put an application in scope for a specific regulatory requirement. Have the matrix indicate which compliance requirements are applicable at the organization level and do not depend on individual applications. The matrix provides at least a basic understanding of useful compliance requirements to review obligations around different applications.

Since many compliance standards are focused around security best-practices, many compliance requirements may already be a part of the Policy and Standards library published by the organization. Therefore, once you review compliance requirements, map them to any applicable existing policies and standards. Whenever there are discrepancies, update the policies and standards to include organization-wide compliance requirements. Then, begin creating compliance-specific standards only applicable to individual compliance requirements. The goal is to have a compliance matrix that indicates which policies and standards have more detailed information about compliance requirements, as well as ensure individual policies and standards reference applicable compliance requirements.

Assessment Question(s)

Do you have a complete picture of your external compliance obligations?

- No
- Yes, for some applications
- Yes, for at least half of the applications

- Yes, for most or all of the applications

Quality Criteria:

- You have identified all sources of external compliance obligations
- You have captured and reconciled compliance obligations from all sources

Policy & Compliance (PC2)

The goal of this practice level is to establish application-specific security and compliance baseline.

Activities

[Stream A] Develop test procedures

Benefit: *Common understanding of how to reach compliance with security policies for product teams*

To assist with the ongoing implementation and verification of compliance with policies and standards, develop application security and appropriate test scripts related to each applicable requirement. Organize these documents into libraries and make them available to all application teams in formats most conducive for inclusion into each application. Clearly label the documents and link them to the policies and standards they represent, to assist with the ongoing updates and maintenance. Version policies and standards and include detailed change logs with each iterative update to make ongoing inclusion into different products' SDLC easier.

Write application security requirements in a format consistent with the existing requirements management processes. You may need more than one version catering to different development methodologies or technologies. The goal is to make it easy for various product teams to incorporate policies and standards into their existing development life-cycles needing minimal interpretation of requirements.

Test scripts help reinforce application security requirements through clear expectations of application functionality, and guide automated or manual testing efforts that may already be part of the development process. These efforts not only help each team establish the current state of compliance with existing policies and standards, but also ensure compliance as applications continue to change.

Assessment Question(s)

Do you publish the organization's policies as test scripts or run-books for easy interpretation by development teams?

- No
- Yes, some content
- Yes, at least half of the content
- Yes, most or all of the content

Quality Criteria:

- You create verification checklists and test scripts where applicable, aligned with the policy's requirements and the implementation guidance in the associated standards
- You create versions adapted to each development methodology and technology the organization uses

[Stream B] Standardize policy and compliance requirements

Benefit: *Common understanding how to reach compliance with external compliance drivers for product teams*

Develop a library of application requirements and test scripts to establish and verify regulatory compliance of applications. Some of these are tied to individual compliance requirements like PCI or GDPR, while others are more general in nature and address global compliance requirements such as ISO. The library is available to all application development teams. It includes guidance for determining all applicable requirements including considerations for reducing the compliance burden and scope. Implement a process to periodically re-assess each application's compliance requirements. Re-assessment includes reviewing all application functionality and opportunities to reduce scope to lower the overall cost of compliance.

Requirements include enough information for developers to understand functional and non-functional requirements of the different compliance obligations. They include references to policies and standards, and provide explicit references to regulations. If there are questions about the implementation of a particular requirement, the original text of the regulation can help interpret the intent more accurately. Each requirement includes a set of test scripts for verifying compliance. In addition to assisting QA with

compliance verification, these can help clarify compliance requirements for developers and make the compliance process transparent. Requirements have a format that allows importing them into individual requirements repositories. further clarify compliance requirements for developers and ensure the process of achieving compliance is fully transparent.

Assessment Question(s)

Do you have a standard set of security requirements and verification procedures addressing the organization’s external compliance obligations?

- No
- Yes, for some obligations
- Yes, for at least half of the obligations
- Yes, for most or all of the obligations

Quality Criteria:

- You map each external compliance obligation to a well-defined set of application requirements
- You define verification procedures, including automated tests, to verify compliance with compliance-related requirements

Policy & Compliance (PC3)

The goal of this practice level is to measure adherence to policies, standards, and 3rd-party requirements.

Activities

[Stream A] Measure compliance to policies and standards

Benefit: *Understanding of your organization’s compliance with policies and standards*

Develop a program to measure each application’s compliance with existing policies and standards. Mandatory requirements should be motivated and reported consistently across all teams. Whenever possible, tie compliance status into automated testing and report with each version. Compliance reporting includes the version of policies and standards and appropriate code coverage factors.

Encourage non-compliant teams to review available resources such as security requirements and test scripts, to ensure non-compliance is not a result of inadequate guidance. Forward issues resulting from insufficient guidance to the teams responsible for publishing application requirements and test scripts, to include them in the future releases. Escalate issues resulting from the inability to meet policies and standards the teams that handle application security risks.

Assessment Question(s)

Do you regularly report on policy and standard compliance, and use that information to guide compliance improvement efforts?

- No
- Yes, but reporting is ad-hoc
- Yes, we report at regular times
- Yes, we report at least annually

Quality Criteria:

- You have procedures (automated, if possible) to regularly generate compliance reports
- You deliver compliance reports to all relevant stakeholders
- Stakeholders use the reported compliance status information to identify areas for improvement

[Stream B] Measure compliance to external requirements

Benefit: *Understanding of your organization’s compliance with external compliance drivers*

Develop a program for measuring and reporting on the status of compliance between different applications. Application requirements and test scripts help determine the status of compliance. Leverage testing automation to promptly detect compliance regressions in frequently updated applications and ensure compliance is maintained through the different application versions. Whenever fully automated testing is not possible, QA, Internal Audit, or Information Security teams assess compliance periodically through a combination of manual testing and interview.

While full compliance is always the ultimate goal, include tracking remediation actions and periodic updates in the program. Review compliance remediation activities periodically to check teams are making appropriate progress, and that remediation strategies will be effective in achieving compliance. To further improve the process, develop a series of standard reports and compliance scorecards. These help individual teams understand the current state of compliance, and the organization manage assistance for remediating compliance gaps more effectively.

Review compliance gaps requiring significant expenses or development with the subject-matter experts and compare them against the cost of reducing the application's functionality, minimizing scope or eliminating the compliance requirement. longterm compliance gaps require management approval and a formal compliance risk acceptance, so they receive appropriate attention and scrutiny from the organization's leadership.

Assessment Question(s)

Do you regularly report on adherence to external compliance obligations and use that information to guide efforts to close compliance gaps?

- No
- Yes, but reporting is ad-hoc
- Yes, we report at regular times
- Yes, we report at least annually

Quality Criteria:

- You have established, well-defined compliance metrics
- You measure and report on applications' compliance metrics regularly
- Stakeholders use the reported compliance status information to identify compliance gaps and prioritize gap remediation efforts

Education & Guidance (EG1)

The goal of this practice level is to offer staff access to resources around the topics of secure development and deployment.

Activities

[Stream A] Train all stakeholders for awareness

Benefit: *Basic security awareness for all relevant employees*

Conduct security awareness training for all roles currently involved in the management, development, testing, or auditing of the software. The goal is to increase the awareness of application security threats and risks, security best practices, and secure software design principles. Develop training internally or procure it externally. Ideally, deliver training in person so participants can have discussions as a team, but Computer-Based Training (CBT) is also an option.

Course content should include a range of topics relevant to application security and privacy, while remaining accessible to a non-technical audience. Suitable concepts are secure design principles including Least Privilege, Defense-in-Depth, Fail Secure (Safe), Complete Mediation, Session Management, Open Design, and Psychological Acceptability. Additionally, the training should include references to any organization-wide standards, policies, and procedures defined to improve application security. The OWASP Top 10 vulnerabilities should be covered at a high level.

Training is mandatory for all employees and contractors involved with software development and includes an auditable sign-off to demonstrate compliance. Consider incorporating innovative ways of delivery (such as gamification) to maximize its effectiveness and combat desensitization.

Assessment Question(s)

Do you require employees involved with application development to take SDLC training?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- Training is repeatable, consistent, and available to anyone involved with software development lifecycle
- Training includes the latest OWASP Top 10 if appropriate and includes concepts such as Least Privilege, Defense-in-Depth, Fail Secure (Safe), Complete Mediation, Session Management, Open Design, and Psychological Acceptability
- Training requires a sign-off or an acknowledgement from attendees
- You have updated the training in the last 12 months
- Training is required during employees' onboarding process

[Stream B] Identify security champions

Benefit: *Basic embedding of security in the development organization*

Implement a program where each software development team has a member considered a "Security Champion" who is the liaison between Information Security and developers. Depending on the size and structure of the team the "Security Champion" may be a software developer, tester, or a product manager. The "Security Champion" has a set number of hours per week for Information Security related activities. They participate in periodic briefings to increase awareness and expertise in different security disciplines. "Security Champions" have additional training to help develop these roles as Software Security subject-matter experts. You may need to customize the way you create and support "Security Champions" for cultural reasons.

The goals of the position are to increase effectiveness and efficiency of application security and compliance and to strengthen the relationship between various teams and Information Security. To achieve these objectives, "Security Champions" assist with researching, verifying, and prioritizing security and compliance related software defects. They are involved in all Risk Assessments, Threat Assessments, and Architectural Reviews to help identify opportunities to remediate security defects by making the architecture of the application more resilient and reducing the attack threat surface.

In addition to assisting Information Security, "Security Champions" provide periodic reviews of all security-related issues for the project team so everyone is aware of the problems and any current and future remediation efforts. These reviews are leveraged to help brainstorm solutions to more complex problems by engaging the entire development team.

Assessment Question(s)

Have you identified a Security Champion for each development team?

- No
- Yes, for some teams
- Yes, for at least half of the teams
- Yes, for most or all of the teams

Quality Criteria:

- Security Champions receive appropriate training
- Application Security and Development teams receive periodic briefings from Security Champions on the overall status of security initiatives and fixes
- The Security Champion reviews the results of external testing before adding to the application backlog

Education & Guidance (EG2)

The goal of this practice level is to educate all personnel in the software lifecycle with technology and role-specific guidance on secure development.

Activities

[Stream A] Customize security training

Benefit: *Relevant employee roles trained according to their specific role*

Conduct instructor-led or CBT security training specific to the organization's roles and technologies, starting with the core development team. The organization customizes training for product managers, software developers, testers, and security auditors, based on each group's technical needs.

- Product managers train on topics related to SAMM business functions and security practices, with emphasis on security requirements, threat modeling, and defect tracking.
- Developers train on coding standards and best practices for the technologies they work with to ensure the training directly benefits application security. They have a solid technical understanding of the OWASP Top 10 vulnerabilities, or similar weaknesses relevant to the technologies and frameworks used

(e.g. mobile), and the most common remediation strategies for each issue.
- Testers train on the different testing tools and best practices for technologies used in the organization, and in tools that identify security defects.
- Security auditors train on the software development lifecycle, application security mechanisms used in the organization, and the process for submitting security defects for remediation.
- Security Champions train on security topics from various phases of the SDLC. They receive the same training as developers and testers, but also understand threat modeling and secure design, as well as security tools and technologies that can be integrated into the build environment.

Include all training content from the Maturity Level 1 activities of this stream and additional role-specific and technology-specific content. Eliminate unnecessary aspects of the training.

Ideally, identify a subject-matter expert in each technology to assist with procuring or developing the training content and updating it regularly. The training consists of demonstrations of vulnerability exploitation using intentionally weakened applications, such as WebGoat or Juice Shop. Include results of the previous penetration as examples of vulnerabilities and implemented remediation strategies. Ask a penetration tester to assist with developing examples of vulnerability exploitation demonstrations.

Training is mandatory for all employees and contractors involved with software development, and includes an auditable sign-off to demonstrate compliance. Whenever possible, training should also include a test to ensure understanding, not just compliance. Update and deliver training annually to include changes in the organization, technology, and trends. Poll training participants to evaluate the quality and relevance of the training. Gather suggestions of other information relevant to their work or environments.

Assessment Question(s)

Is training customized for individual roles such as developers, testers, or security champions?

- No
- Yes, for some of the training
- Yes, for at least half of the training
- Yes, for most or all of the training

Quality Criteria:

- Training includes all topics from maturity level 1, and adds more specific tools, techniques, and demonstrations
- Training is mandatory for all employees and contractors
- Training includes input from in-house SMEs and trainees
- Training includes demonstrations of tools and techniques developed in-house
- You use feedback to enhance and make future training more relevant

[Stream B] Implement centers of excellence

Benefit: *Specific security best practices tailored to the organization*

The organization implements a formal secure coding center of excellence, with architects and senior developers representing the different business units and technology stacks. The team has an official charter and defines standards and best practices to improve software development practices. The goal is to mitigate the way velocity of change in technology, programming languages, and development frameworks and libraries makes it difficult for Information Security professionals to be fully informed of all the technical nuances that impact security. Even developers often struggle keeping up with all the changes and new tools intended to make software development faster, better, and safer.

This ensures all current programming efforts follow industry’s best practices and organization’s development and implementation standards include all critical configuration settings. It helps identify, train, and support “Product Champions”, responsible for assisting different teams with implementing tools that automate, streamline, or improve various aspects of the SDLC. It identifies development teams with higher maturity levels within their SDLC and the practices and tools that enable these achievements, with the goal of replicating them to other teams.

The group provides subject matter expertise, helping information security teams evaluate tools and solutions to improve application security, ensuring these tools are not only useful but also compatible with the way different teams develop applications. Teams looking to make significant architectural changes to their software consult with this group to avoid adversely impacting the SDLC lifecycle or established security controls.

Assessment Question(s)

Does the organization have a Secure Software Center of Excellence (SSCE)?

- No
- Yes, we started implementing it
- Yes, for part of the organization
- Yes, for the entire organization

Quality Criteria:

- The SSCE has a charter defining its role in the organization
- Development teams review all significant architectural changes with the SSCE
- The SSCE publishes SDLC standards and guidelines related to Application Security
- Product Champions are responsible for promoting the use of specific security tools

Education & Guidance (EG3)

The goal of this practice level is to develop in-house training programs facilitated by developers across different teams.

Activities

[Stream A] Standardize security guidance

Benefit: *Adequate security knowledge of all employees ensured prior to working on critical tasks*

Implement a formal training program requiring anyone involved with the software development lifecycle to complete appropriate role and technology-specific training as part of the onboarding process. Based on the criticality of the application and user's role, consider restricting access until the onboarding training has been completed. While the organization may source some modules externally, the program is facilitated and managed in-house and includes content specific to the organization going beyond general security best practices. The program has a defined curriculum, checks participation, and tests understanding and competence. The training consists of a combination of industry best practices and organization's internal standards, including training on specific systems used by the organization.

In addition to issues directly related to security, the organization includes other standards to the program, such as code complexity, code documentation, naming convention, and other process-related disciplines. This training minimizes issues resulting from employees following practices incorporated outside the organization and ensures continuity in the style and competency of the code.

To facilitate progress monitoring and successful completion of each training module the organization has a learning management platform or another centralized portal with similar functionality. Employees can monitor their progress and have access to all training resources even after they complete initial training.

Review issues resulting from employees not following established standards, policies, procedures, or security best practices at least annually to gauge the effectiveness of the training and ensure it covers all issues relevant to the organization. Update the training periodically and train employees on any changes and most prevalent security deficiencies.

Assessment Question(s)

Have you implemented a Learning Management System or equivalent to track employee training and certification processes?

- No
- Yes, for some of the training
- Yes, for at least half of the training
- Yes, for most or all of the training

Quality Criteria:

- A Learning Management System (LMS) is used to track trainings and certifications
- Training is based on internal standards, policies, and procedures
- You use certification programs or attendance records to determine access to development systems and resources

[Stream B] Establish a security community

Benefit: *Collective development of security know-how among all product teams*

Security is the responsibility of all employees, not just the Information Security team. Deploy communication and knowledge sharing platforms to help developers build communities around different technologies, tools, and programming languages. In these communities employees share information, discuss challenges with other developers, and search the knowledge base for answers to previously discussed issues.

Form communities around roles and responsibilities and enable developers and engineers from different teams and business units to communicate freely and benefit from each other's expertise. Encourage participation, set up a program to promote those who help the most people as thought leaders, and have management recognize them. In addition to improving application security, this platform may help identify future members of the Secure Software Center of Excellence, or 'Security Champions' based on their expertise and willingness to help others.

The Secure Software Center of Excellence and Application Security teams review the information portal regularly for insights into the new and upcoming technologies, as well as opportunities to assist the development community with new initiatives, tools, programs, and training resources. Use the portal to disseminate information about new standards, tools, and resources to all developers for the continued improvement of SDLC maturity and application security.

Assessment Question(s)

Is there a centralized portal where developers and application security professionals from different teams and business units are able to communicate and share information?

- No
- Yes, we started implementing it
- Yes, for part of the organization
- Yes, for the entire organization

Quality Criteria:

- The organization promotes use of a single portal across different teams and business units
- The portal is used for timely information such as notification of security incidents, tool updates, architectural standard changes, and other related announcements
- The portal is widely recognized by developers and architects as a centralized repository of the organization-specific application security information
- All content is considered persistent and searchable
- The portal provides access to application-specific security metrics

Design

Threat Assessment (TA1)

The goal of this practice level is to consider security explicitly during the software requirements process.

Activities

[Stream A] Perform application risk assessments

Benefit: *Ability to classify applications according to risk*

Use a simple method to evaluate the application risk per application, estimating the potential business impact that it poses for the organization in case of an attack. To achieve this, evaluate the impact of a breach in the confidentiality, integrity and availability of the data or service. Consider using a set of 5-10 questions to understand important application characteristics, such as whether the application processes financial data, whether it is internet facing, or whether privacy-related data is involved. The application risk profile tells you whether these factors are applicable and if they could significatly impact the organization.

Next, use a scheme to classify applications according to this risk. A simple, qualitative scheme (e.g. high/medium/low) that translates these characteristics into a value is often effective. It is important to use these values to represent and compare the risk of different applications against each other. Mature highly risk-driven organizations might make use of more quantitative risk schemes. Don't invent a new risk scheme if your organization already has one that works well.

Assessment Question(s)

Do you classify applications according to business risk based on a simple and predefined set of questions?

- No

- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- An agreed-upon risk classification exists
- The application team understands the risk classification
- The risk classification covers critical aspects of business risks the organization is facing
- The organization has an inventory for the applications in scope

[Stream B] Perform basic threat modeling

Benefit: *Identification of architectural design flaws in your applications*

Threat modeling is a structured activity for identifying, evaluating, and managing system threats, architectural design flaws, and recommended security mitigations. It is typically done as part of the design phase or as part of a security assessment.

Threat modeling is a team exercise, including product owners, architects, security champions, and security testers. At this maturity level, expose teams and stakeholders to threat modeling to increase security awareness and to create a shared vision on the security of the system.

At maturity level 1, you perform threat modeling ad-hoc for high-risk applications and use simple threat checklists, such as STRIDE. Avoid lengthy workshops and overly detailed lists of low-relevant threats. Perform threat modeling iteratively to align to more iterative development paradigms. If you add new functionality to an existing application, look only into the newly added functions instead of trying to cover the entire scope. A good starting point is the existing diagrams that you annotate during discussion workshops. Always make sure to persist the outcome of a threat modeling discussion for later use.

Your most important tool to start threat modeling is a whiteboard, smartboard, or a piece of paper. Aim for security awareness, a simple process, and actionable outcomes that you agree upon with your team.

Assessment Question(s)

Do you identify and manage architectural design flaws with threat modeling?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- You perform threat modeling for high-risk applications
- You use simple threat checklists, such as STRIDE
- You persist the outcome of a threat model for later use

Threat Assessment (TA2)

The goal of this practice level is to increase granularity of security requirements derived from business logic and known risks.

Activities

[Stream A] Inventorize risk profiles

Benefit: *Solid understanding of the risk level of your application portfolio*

The goal of this activity is to thoroughly understand the risk level of all applications within the organization, to focus the effort of your software assurance activities where it really matters.

From a risk evaluation perspective, the basic set of questions is not enough to thoroughly evaluate the risk of all applications. Create an extensive and standardized way to evaluate the risk of the application, among others via their impact on information security (confidentiality, integrity and availability of data). Next to security, you also want to evaluate the privacy risk of the application. Understand the data that the application processes and what potential privacy violations are relevant. Finally, study the impact that this application has on other applications within the organization (e.g., the application might be modifying data that was considered read-only in another context). Evaluate all applications within the organization,

including all existing and legacy ones.

Leverage business impact analysis to quantify and classify application risk. A simple qualitative scheme (such as high/medium/low) is not enough to effectively manage and compare applications on an enterprise-wide level.

Based on this input, Security Officers leverage the classification to define the risk profile to build a centralized inventory of risk profiles and manage accountability. This inventory gives Product Owners, Managers, and other organizational stakeholders an aligned view of the risk level of an application in order to assign appropriate priority to security-related activities.

Assessment Question(s)

Do you use centralized and quantified application risk profiles to evaluate business risk?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- The application risk profile is in line with the organizational risk standard
- The application risk profile covers impact to security and privacy
- You validate the quality of the risk profile manually and/or automatically
- The application risk profiles are stored in a central inventory

[Stream B] Standardize and scale threat modeling

Benefit: *Clear expectations of the quality of threat modeling activites*

Use a standardized threat modeling methodology for your organization and align this on your application risk levels. Think about ways to support the scaling of threat modeling throughout the organization.

Train your architects, security champions, and other stakeholders on how to do practical threat modeling. Threat modeling requires understanding, clear playbooks and templates, organization-specific examples, and experience, which is hard to automate.

Your threat modeling methodology includes at least diagramming, threat identification, design flaw mitigations, and how to validate your threat model artifacts. Your threat model diagram allows a detailed understanding of the environment and the mechanics of the application. You discover threats to your application with checklists, such as STRIDE or more organization-specific threats. For identified design flaws (ranked according to risk for your organization), you add mitigating controls to support stakeholders in dealing with particular threats. Define what triggers updating a threat model, for example, a technology change or deployment of an application in a new environment.

Feed the output of threat modeling to the defect management process for adequate follow-up. Capture the threat modeling artifacts with tools that are used by your application teams.

Assessment Question(s)

Do you use a standard methodology, aligned on your application risk levels?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You train your architects, security champions, and other stakeholders on how to do practical threat modeling
- Your threat modeling methodology includes at least diagramming, threat identification, design flaw mitigations, and how to validate your threat model artifacts
- Changes in the application or business context trigger a review of the relevant threat models
- You capture the threat modeling artifacts with tools that are used by your application teams

Threat Assessment (TA3)

The goal of this practice level is to mandate security requirements process for all software projects and third-party dependencies.

Activities

[Stream A] Periodic review of risk profiles

Benefit: *Timely update of the application classification in case of changes*

The application portfolio of an organization changes, as well as the conditions and constraints in which an application lives (e.g., driven by the company strategy). Periodically review the risk inventory to ensure correctness of the risk evaluations of the different applications.

Have a periodic review at an enterprise-wide level. Also, as your enterprise matures in software assurance, stimulate teams to continuously question which changes in conditions might impact the risk profile. For instance, an internal application might become exposed to the internet by a business decision. This should trigger the teams to rerun the risk evaluation and update the application risk profile accordingly.

In a mature implementation of this practice, train and continuously update teams on lessons learned and best practices from these risk evaluations. This leads to a better execution and a more accurate representation of the application risk profile.

Assessment Question(s)

Do you regularly review and update the risk profiles for your applications?

- No
- Yes, sporadically
- Yes, upon change of the application
- Yes, at least annually

Quality Criteria:

- The organizational risk standard considers historical feedback to improve the evaluation method
- Significant changes in the application or business context trigger a review of the relevant risk profiles

[Stream B] Optimize threat modeling

Benefit: *Assurance of continuous improvement of threat modeling activities*

Threat modeling is integrated into your SDLC and has become part of the developer security culture. Reusable risk patterns, comprising of related threat libraries, design flaws, and security mitigations, are created and improved, based on the organization’s threat models. You regularly (e.g., yearly) review the existing threat models to verify that no new threats are relevant for your applications.

You optimize your threat modeling methodology. You capture lessons learned from threat models and use these to improve your threat modeling methodology. You review the threat categories relevant to your organization and update your methodology appropriately. From time to time, you evaluate the quality of your threat models independently.

You automate parts of your threat modeling process with threat modeling tools. You integrate your threat modeling tools with other security tools, such as security verification tools and risk tracking tools. You consider “threat modeling as code” practices to integrate threat modeling artifacts with application code.

Assessment Question(s)

Do you regularly review and update the threat modeling methodology for your applications?

- No
- Yes, but review is ad-hoc
- Yes, we review it at regular times
- Yes, we review it at least annually

Quality Criteria:

- The threat model methodology considers historical feedback for improvement
- You regularly (e.g., yearly) review the existing threat models to verify that no new threats are relevant for your applications
- You automate parts of your threat modeling process with threat modeling tools

Security Requirements (SR1)

The goal of this practice level is to consider security explicitly during the software requirements process.

Activities

[Stream A] Identify security requirements

Benefit: *Understanding of key security requirements during development*

Perform a review of the functional requirements of the software project. Identify relevant security requirements (i.e. expectations) for this functionality by reasoning on the desired confidentiality, integrity or availability of the service or data offered by the software project. Requirements state the objective (e.g., “personal data for the registration process should be transferred and stored securely”), but not the actual measure to achieve the objective (e.g., “use TLSv1.2 for secure transfer”).

At the same time, review the functionality from an attacker perspective to understand how it could be misused. This way you can identify extra protective requirements for the software project at hand.

Security objectives can relate to specific security functionality you need to add to the application (e.g., “Identify the user of the application at all times”) or to the overall application quality and behavior (e.g., “Ensure personal data is properly protected in transit”), which does not necessarily lead to new functionality. Follow good practices for writing security requirements. Make them specific, measurable, actionable, relevant and time-bound (SMART). Beware of adding requirements too general-purpose to not relate to the application at hand (e.g., The application should protect against the OWASP Top 10). While they can be true, they don’t add value to the discussion.

Assessment Question(s)

Do project teams specify security requirements during development?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Teams derive security requirements from functional requirements and customer or organization concerns
- Security requirements are specific, measurable, and reasonable
- Security requirements are in line with the organizational baseline

[Stream B] Perform vendor assessments

Benefit: *Transparency of security practices of your software suppliers*

The security competences and habits of the expernal suppliers involved in the development of your software can have a significant impact on the security posture of the final product. Consequently, it is important to know and evaluate your suppliers on this front.

Carry out a vendor assessment to understand the strengths and weaknesses of your suppliers. Use a basic checklist or conduct interviews to review their typical practices and deliveries. This gives you an idea of how they organize themselves and elements to evaluate whether you need to take additional measures to mitigate potential risks. Ideally, speak to different roles in the organization, or even set up a small maturity evaluation to this end. Strong suppliers will run their own software assurance program and will be able to answer most of your questions. If suppliers have weak competences in software security, discuss with them how and to what extent they plan to work on this and evaluate whether this is enough for your organization. A software supplier might be working on a low-risk project, but this could change.

It is important that your suppliers understand and align to the risk appetite and are able to meet your requirements in that area. Make what you expect from them explicit and discuss this clearly.

Assessment Question(s)

Do stakeholders review vendor collaborations for security requirements and methodology?

- No
- Yes, some of the time

- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- You consider including specific security requirements, activities, and processes when creating third-party agreements
- A vendor questionnaire is available and used to assess the strengths and weaknesses of your suppliers

Security Requirements (SR2)

The goal of this practice level is to increase granularity of security requirements derived from business logic and known risks.

Activities

[Stream A] Standardize and integrate security requirements

Benefit: *Alignment of security requirements with other types of requirements*

Security requirements can originate from other sources including policies and legislation, known problems within the application, and intelligence from metrics and feedback. At this level, a more systematic elicitation of security requirements must be achieved by analysing different sources of such requirements. Ensure that appropriate input is received from these sources to help the elicitation of requirements. For example, organize interviews or brainstorm sessions (e.g., in the case of policy and legislation), analyze historical logs or vulnerability systems.

Use a structured notation of security requirements across applications and an appropriate formalism that integrates well with how you specify other (functional) requirements for the project. This could mean, for example, extending analysis documents, writing user stories, etc.

When requirements are specified, it is important to ensure that these requirements are taken into account during product development. Setup a mechanism to stimulate or force project teams to meet these requirements in the product. For example, annotate requirements with priorities, or influence the handling of requirements to enforce sufficient security appetite (while balancing against other non-functional requirements).

Assessment Question(s)

Do you define, structure, and include prioritization in the artifacts of the security requirements gathering process?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- Security requirements take into consideration domain specific knowledge when applying policies and guidance to product development
- Domain experts are involved in the requirements definition process
- You have an agreed upon structured notation for security requirements
- Development teams have a security champion dedicated to reviewing security requirements and outcomes

[Stream B] Discuss security responsabilities with suppliers

Benefit: *Clearly defined security responsibilities of your software suppliers*

Increase your confidence in the capability of your suppliers for software security. Discuss concrete responsibilities and expectations from your suppliers and your own organization and establish a contract with the supplier. The responsibilities can be specific quality requirements or particular tasks, and minimal service can be detailed in a Service Level Agreement (SLA). A quality requirement example is that they will deliver software that is protected against the OWASP Top 10, and in case issues are detected, these will be fixed. A task example is that they have to perform continuous static code analysis, or perform an independent penetration test before a major release. The agreement stipulates liabilities and caps in case an important issue arises.

Once you have implemented this for a few suppliers, work towards a standard agreement for suppliers that forms the basis of your negotiations. You can deviate from this standard agreement on a case by case basis, but it will help you to ensure you do not overlook important topics.

Assessment Question(s)

Do vendors meet the security responsibilities and quality measures of service level agreements defined by the organization?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- You discuss security requirements with the vendor when creating vendor agreements
- Vendor agreements provide specific guidance on security defect remediation within an agreed upon timeframe
- The organization has a templated agreement of responsibilities and service levels for key vendor security processes
- You measure key performance indicators

Security Requirements (SR3)

The goal of this practice level is to mandate security requirements process for all software projects and third-party dependencies.

Activities

[Stream A] Develop a security requirements framework

Benefit: *Efficient and effective handling of security requirements in your organization*

Setup a security requirements framework to help projects elicit an appropriate and complete requirements set for their project. This framework considers the different types of requirements and sources of requirements. It should be adapted to the organizational habits and culture, and provide effective methodology and guidance in the elicitation and formation of requirements.

The framework helps project teams increase the efficiency and effectiveness of requirements engineering. It can provide a categorisation of common requirements and a number of reusable requirements. Do remember that, while thoughtless copying is ineffective, the fact of having potential relevant requirements to reason about is often productive.

The framework also gives clear guidance on the quality of requirements and formalizes how to describe them. For user stories, for instance, concrete guidance can explain what to describe in the definition of done, definition of ready, story description, and acceptance criteria.

Assessment Question(s)

Do you use a standard requirements framework to streamline the elicitation of security requirements?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- A security requirements framework is available for project teams
- The framework is categorized by common requirements and standards-based requirements
- The framework gives clear guidance on the quality of requirements and how to describe them
- The framework is adaptable to specific business requirements

[Stream B] Align security methodology with suppliers

Benefit: *Alignment of software development practices with suppliers to limit security risks*

The best way to minimize the risk of issues in software is to align maximally and integrate closely between

the different parties. From a process perspective, this means using similar development paradigms and introducing regular milestones to ensure proper alignment and qualitative progress. From a tools perspective, this might mean using similar build, verification and deployment environments, and sharing other supporting tools (e.g. requirements, architecture tools, or code repositories).

In case suppliers cannot meet the objectives that you have set, implement compensating controls so that, overall, you meet your objectives. Execute extra activities (e.g., threat modelling before starting the actual implementation cycle) or implement extra tooling (e.g., 3rd party library analysis at solution intake). The more suppliers deviate from your requirements, the more work will be required to compensate.

Assessment Question(s)

Are vendors aligned with standard security controls and software development tools and processes that the organization utilizes?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- The vendor has a secure SDLC that includes secure build, secure deployment, defect management, and incident management that align with those used in your organization
- You verify the solution meets quality and security objectives before every major release
- When standard verification processes are not available, you use compensating controls such as software composition analysis and independent penetration testing

Security Architecture (SA1)

The goal of this practice level is to insert consideration of proactive security guidance into the software design process.

Activities

[Stream A] Adhere to basic security principles

Benefit: *Sets of security basic principles available to product teams*

During design, technical staff on the product team use a short checklist of security principles. 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.

For perimeter interfaces, the team considers each principle in the context of the overall system and identify features that can be added to bolster security at each such interface. Limit these such that they only take a small amount of extra effort beyond the normal implementation cost of functional requirements. Note anything larger, and schedule it for future releases.

Train each product team with security awareness before this process, and incorporate more security-savvy staff to aid in making design decisions.

Assessment Question(s)

Do teams use security principles during design?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have an agreed upon checklist of security principles
- You store your checklist in an accessible location
- Relevant stakeholders understand security principles

[Stream B] Identify tools and technologies

Benefit: *Transparency of technologies that introduce security risk*

People often take the path of least resistance in developing, deploying or operating a software solution. New technologies are often included when they can facilitate or speed up the effort or enable the solution to scale better. These new technologies might, however, introduce new risks to the organization that you need to manage.

Identify the most important technologies, frameworks, tools and integrations being used for each application. Use the knowledge of the architect to study the development and operating environment as well as artefacts. Then evaluate them for their security quality and raise important findings to be managed.

Assessment Question(s)

Do you evaluate the security quality of important technologies used for development?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have a list of the most important technologies used in or in support of each application
- You identify and track technological risks
- You ensure the risks to these technologies are in line with the organizational baseline

Security Architecture (SA2)

The goal of this practice level is to direct the software design process toward known secure services and secure-by-default designs.

Activities

[Stream A] Provide preferred security solutions

Benefit: *Reusable security services available for product teams*

Identify shared infrastructure or services with security functionality. These typically include single-sign-on services, access control or entitlements services, logging and monitoring services or application-level firewalling. Collect and evaluate reusable systems to assemble a list of such resources and categorize them by the security mechanism they fulfill. Consider each resource in terms of why a product team would want to integrate with it, i.e. the benefits of using the shared resource.

If multiple resources exist in each category, select and standardize on one or more shared service per category. Because future software development will rely on these services, review each thoroughly to ensure understanding of the baseline security posture. For each selected service, create design guidance for product teams to understand how to integrate with the system. Make the guidance available through training, mentorship, guidelines, and standards.

Establish a set of best practices representing sound methods of implementing security functionality. You can research them or purchase them, and it is often more effective if you customize them so they are more specific to your organization. Example patterns include a single-sign-on subsystem, a cross-tier delegation model, a separation-of-duties authorization model, a centralized logging pattern, etc.

These patterns can originate from specific projects or applications, but make sure you share them between different teams across the organization for efficient and consistent application of appropriate security solutions.

To increase adoption of these patterns, link them to the shared security services, or implement them into actual component solutions that can be easily integrated into an application during development. Support the key technologies within the organization, for instance in case of different development stacks. Treat these solutions as actual applications with proper support in case of questions or issues.

Assessment Question(s)

Do you use shared security services during design?

- No
- Yes, for some applications

- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have a documented list of reusable security services, available to relevant stakeholders
- You have reviewed the baseline security posture for each selected service
- Your designers are trained to integrate each selected service following available guidance

[Stream B] Promote preferred tools and technologies

Benefit: *Technologies with appropriate security level available to product teams*

Identify commonly used technologies, frameworks and tools in use across software projects in the organization, whereby you focus on capturing the high-level technologies.

Create a list and share it across the development organization as recommended technologies. When selecting them, consider incident history, track record for responding to vulnerabilities, appropriateness of functionality for the organization, excessive complexity in usage of the third-party component, and sufficient knowledge within the organization.

Senior developers and architects create this list, including input from managers and security auditors. Share this list of recommended components with the development organization. Ultimately, the goal is to provide well-known defaults for project teams. Perform a periodic review of these technologies for security and appropriateness.

Assessment Question(s)

Do you have a list of recommended technologies for the organization?

- No
- Yes, for some of the technology domains
- Yes, for at least half of the technology domains
- Yes, for most or all of the technology domains

Quality Criteria:

- The list is based on technologies used in the software portfolio
- Lead architects and developers review and approve the list
- You share the list across the organization
- You review and update the list at least yearly

Security Architecture (SA3)

The goal of this practice level is to formally control the software design process and validate utilization of secure components.

Activities

[Stream A] Build reference architectures

Benefit: *Full transparency of quality and usability of centrally provided security solutions*

Build a set of reference architectures that select and combine a verified set of security components to ensure a proper design of security. Reference platforms have advantages in terms of shortening audit and security-related reviews, increasing efficiency in development, and lowering maintenance overhead. Continuously maintain and improve the reference architecture based on new insights in the organization and within the community. Have architects, senior developers and other technical stakeholders participate in design and creation of reference platforms. After creation, teams maintain ongoing support and updates.

Reference architectures may materialize into a set of software libraries and tools upon which project teams build their software. They serve as a starting point that standardizes the configuration-driven, security-by-default security approach. You can bootstrap the framework by selecting a particular project early in the lifecycle and having security-savvy staff work with them to build the security functionality in a generic way so that it can be extracted from the project and used elsewhere in the organization.

Monitor weaknesses or gaps in the set of security solutions available in your organization continuously in the context of discussions on architecture, development, or operations. This serves as an input to improve the appropriateness and effectiveness of the reference architectures that you have in place.

Assessment Question(s)

Do you base your design on available reference architectures?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have one or more approved reference architectures documented and available to stakeholders
- You improve the reference architectures continuously based on insights and best practices
- You provide a set of components, libraries, and tools to implement each reference architecture

[Stream B] Enforce the use of recommended technologies

Benefit: *Limited attack surface due to usage of vetted technologies*

For all proprietary development (in-house or acquired), impose and monitor the use of standardized technology. Depending on your organization, either implement these restrictions into build or deployment tools, by means of after-the-fact automated analysis of application artefacts (e.g., source code, configuration files or deployment artefacts), or periodically review focusing on the correct use of these frameworks.

Verify several factors with project teams. Identify use of non-recommended technologies to determine if there are gaps in recommendations versus the organization’s needs. Examine unused or incorrectly used design patterns and reference platform modules to determine if updates are needed. Additionally, implement functionality in the reference platforms as the organization evolves and project teams request it.

Assessment Question(s)

Do you enforce the use of recommended technologies within the organization?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You monitor applications regularly for the correct use of the recommended technologies
- You solve violations against the list accoranding to organizational policies
- You take action if the number of violations falls outside the yearly objectives

Implementation

Secure Build (SB1)

The goal of this practice level is to build process is repeatable and consistent.

Activities

[Stream A] Define a consistent build process

Benefit: *Limited risk of human error during build process minimizing security issues*

Define the build process, breaking it down into a set of clear instructions to either be followed by a person or an automated tool. The build process definition describes the whole process end-to-end so that the person or tool can follow it consistently each time and produce the same result. The definition is stored centrally and accessible to any tools or people. Avoid storing multiple copies as they may become unaligned and outdated.

The process definition does not include any secrets (specifically considering those needed during the build process).

Review any build tools, ensuring that they are actively maintained by vendors and up-to-date with security patches. Harden each tool’s configuration so that it is aligned with vendor guidelines and industry best practices.

Determine a value for each generated artifact that can be later used to verify its integrity, such as a

signature or a hash. Protect this value and, if the artifact is signed, the private signing certificate.

Ensure that build tools are routinely patched and properly hardened.

Assessment Question(s)

Is your full build process formally described?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have enough information to recreate the build processes
- Your build documentation up to date
- Your build documentation is stored in an accessible location
- Produced artifact checksums are created during build to support later verification
- You harden the tools that are used within the build process

[Stream B] Identify application dependencies

Benefit: *Available information on known security issues in dependencies*

Keep a record of all dependencies used throughout the target production environment. This is sometimes referred to as a Bill of Materials (BOM). Consider that different components of the application may consume entirely different dependencies. For example, if the software package is a web application, cover both the server-side application code and client-side scripts. In building these records, consider the various locations where dependencies might be specified such as configuration files, the project's directory on disk, a package management tool or the actual code (e.g. via an IDE that supports listing dependencies).

Gather the following information about each dependency:

- Where it is used or referenced
- Version used
- License
- Source information (link to repository, author's name, etc.)
- Support and maintenance status of the dependency

Check the records to discover any dependencies with known vulnerabilities and update or replace them accordingly.

Assessment Question(s)

Do you have solid knowledge about dependencies you're relying on?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have a current bill of materials (BOM) for every application
- You can quickly find out which applications are affected by a particular CVE
- You have analyzed, addressed, and documented findings from dependencies at least once in the last three months

Secure Build (SB2)

The goal of this practice level is to build process is optimized and fully integrated into the workflow.

Activities

[Stream A] Automate the build process

Benefit: *Efficient build process with integrated security tools*

Automate the build process so that builds can be executed consistently anytime. The build process shouldn’t typically require any intervention, further reducing the likelihood of human error.

The use of an automated system increases reliance on security of the build tooling and makes hardening and maintaining the toolset even more critical. Pay particular attention to the interfaces of those tools, such as web-based portals and how they can be locked-down. The exposure of a build tool to the network could allow a malicious actor to tamper with the integrity of the process. This might, for example, allow malicious code to be built into software.

The automated process may require access to credentials and secrets required to build the software, such as the code signing certificate or access to repositories. Handle these with care. Sign generated artifacts using a certificate that identifies the organization or business unit that built it, so you can verify its integrity.

Finally, add appropriate automated security checks (e.g. using SAST tools) in the pipeline to leverage the automation for security benefit.

Assessment Question(s)

Is the build process fully automated?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- The build process itself doesn’t require any human interaction
- Your build tools are hardened as per best practice and vendor guidance
- You encrypt the secrets required by the build tools and control access based on the principle of least privilege

[Stream B] Review application dependencies for security

Benefit: *Full transparency of known security issues in dependencies*

Evaluate used dependencies and establish a list of acceptable ones approved for use within a project, team, or the wider organization according to a defined set of criteria.

Introduce a central repository of dependencies that all software can be built from.

Review used dependencies regularly to ensure that:

- they remain correctly licensed
- no known and significant vulnerabilities impacting your applications are present
- the dependency is still actively supported and maintained
- you are using a current version
- there is a valid reason to include the dependency

React timely and appropriately to non-conformities by handling these as defects. Consider using an automated tool to scan for vulnerable dependencies and assign the identified issues to the respective development teams.

Assessment Question(s)

Do you handle 3rd party dependency risk by a formal process?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You keep a list of approved dependencies that meet predefined criteria
- You automatically evaluate dependencies for new CVEs and alert responsible staff
- You automatically detect and alert to license changes with possible impact on legal application usage
- You track and alert to usage of unmaintained dependencies
- You reliably detect and remove unnecessary dependencies from the software

Secure Build (SB3)

The goal of this practice level is to build process helps prevent known defects from entering the production environment.

Activities

[Stream A] Enforce a security baseline during build

Benefit: *Assurance that you build software complying with a security baseline*

Define security checks suitable to be carried out during the build process, as well as minimum criteria for passing the build - these might differ according to the risk profiles of various applications. Include the respective security checks in the build and enforce breaking the build process in case the predefined criteria are not met. Trigger warnings for issues below the threshold and log these to a centralized system to track them and take actions. If sensible, implement an exception mechanism to bypass this behavior if the risk of a particular vulnerability has been accepted or mitigated. However, ensure these cases are explicitely approved first and log their occurence together with a rationale.

If technical limitations prevent the organization from breaking the build automatically, ensure the same effect via other measures, such as a clear policy and regular audit.

Handle code signing on a separate centralized server which does not expose the certificate to the system executing the build. Where possible, use a deterministic method that outputs byte-for-byte reproducible artifacts.

Assessment Question(s)

Do you enforce automated security checks in your build processes?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Builds fail if the application doesn’t meet a predefined security baseline
- You have a maximum accepted severity for vulnerabilties
- You log warnings and failures in a centralized system
- You select and configure tools to evaluate each application against its security requirements at least once a year

[Stream B] Test application dependencies

Benefit: *Handling of security issues in dependencies comparable to those in your own code*

Maintain a whitelist of approved dependencies and versions, and ensure that the build process fails upon a presence of dependency not being on the list. Include a sign-off process for handling exceptions to this rule if sensible.

Perform security verification activites against dependencies on the whitelist in a comparable way to the target applications themselves (esp. using SAST and analyzing transitive dependencies). Ensure that these checks also aim to identify possible backdoors or easter eggs in the dependencies. Establish vulnerability disclosure processes with the dependency authors including SLAs for fixing issues. In case enforcing SLAs is not realistic (e.g. with open source vulnerabilities), ensure that the most probable cases are expected and you are able to implement compensating measures in a timely manner. Implement regression tests for the fixes to identified issues.

Track all identified issues and their state using your defect tracking system. Integrate your build pipeline with this system to enable failing the build whenever the included dependencies contain issues above a defined criticality level.

Assessment Question(s)

Do you prevent build of software if it’s affected by vulnerabilities in dependencies?

- No
- Yes, for some applications

- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Your build system is connected to a system for tracking 3rd party dependency risk, causing build to fail unless the vulnerability is evaluated to be a false positive or the risk is explicitly accepted
- You scan your dependencies using a static analysis tool
- You report findings back to dependency authors using an established responsible disclosure process
- Using a new dependency not evaluated for security risks causes the build to fail

Secure Deployment (SD1)

The goal of this practice level is to deployment processes are fully documented.

Activities

[Stream A] Use a repeatable deployment process

Benefit: *Limited risk of human error during deployment process minimizing security issues*

Define the deployment process over all stages, breaking it down into a set of clear instructions to either be followed by a person or an automated tooling. The deployment process definition should describe the whole process end-to-end so that it can be consistently followed each time to produce the same result. The definition is stored centrally and accessible to all relevant personnel. Do not store or distribute multiple copies, some of which may become outdated.

Deploy applications to production either using an automated process, or manually by personnel other than the developers. Ensure that developers do not need direct access to the production environment for application deployment.

Review any deployment tools, ensuring that they are actively maintained by vendors and up to date with security patches. Harden each tool’s configuration so that it is aligned with vendor guidelines and industry best practices. Given that most of these tools require access to the production environment, their security is extremely critical. Ensure the integrity of the tools themselves and the workflows they follow, and configure access rules to these tools according to the least privilege principle.

Have personnel with access to the production environment go through at least a minimum level of training or certification to ensure their competency in this matter.

Assessment Question(s)

Do you use repeatable deployment processes?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have enough information to run the deployment processes
- Your deployment documentation up to date
- Your deployment documentation is accessible to relevant stakeholders
- You ensure that only defined qualified personnel can trigger a deployment
- You harden the tools that are used within the deployment process

[Stream B] Protect application secrets in configuration and code

Benefit: *Defined and limited access to your production secrets*

Developers should not have access to secrets or credentials for production environments. Have a mechanism in place to adequately protect production secrets, for instance by (i) having specific persons adding them to relevant configuration files upon deployment (the separation of duty principle) or (ii) by encrypting the production secrets contained in the configuration files upfront.

Do not use production secrets in configuration files for development or testing environments, as such environments may have a significantly lower security posture. Similarly, do not keep secrets unprotected in configuration files stored in code repositories.

Store sensitive credentials and secrets for production systems with encryption-at-rest at all times. Consider using a purpose-built tool for this. Handle key management carefully so only personnel with responsibility for production deployments are able to access this data.

Assessment Question(s)

Do you limit access to application secrets according to the least privilege principle?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You store production secrets protected in a secured location
- Developers do not have access to production secrets
- Production secrets are not available in non-production environments

Secure Deployment (SD2)

The goal of this practice level is to deployment processes include security verification milestones.

Activities

[Stream A] Automate deployment and integrate security checks

Benefit: *Efficient deployment process with integrated security tools*

Automate the deployment process to cover various stages, so that no manual configuration steps are needed and the risk of isolated human errors is eliminated. Ensure and verify that the deployment is consistent over all stages.

Integrate automated security checks in your deployment process, e.g. using Dynamic Analysis Security Testing (DAST) and vulnerability scanning tools. Also, verify the integrity of the deployed artefacts where this makes sense. Log the results from these tests centrally and take any necessary actions. Ensure that in case any defects are detected, relevant personnel is notified automaticaly. In case any issues exceeding predefined criticality are identified, stop or reverse the deployment either automatically, or introduce a separate manual approval workflow so that this decision is recorded, containing an explanation for the exception.

Account for and audit all deployments to all stages. Have a system in place to record each deployment, including information about who conducted it, the software version that was deployed, and any relevant variables specific to the deploy.

Assessment Question(s)

Are deployment processes automated and employing security checks?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Deployment processes are automated on all stages
- Deployment includes automated security testing procedures
- You alert responsible staff to identified vulnerabilities
- You have logs available for your past deployments for a defined period of time

[Stream B] Include application secrets during deployment

Benefit: *Detection of potential leakage of production secrets*

Have an automated process to add credentials and secrets to configuration files during the deployment process to respective stages. This way, developers and deployers do not see or handle those sensitive values.

Implement checks that detect the presence of secrets in code repositories and files, and run them periodically. Configure tools to look for known strings and unknown high entropy strings. In systems such as code repositories, where there is a history, include the versions in the checks. Mark potential secrets you discover as sensitive values, and remove them where appropriate. If you cannot remove them from a historic file in a code repository, for example, you may need to refresh the value on the system that consumes the secret. This way, if an attacker discovers the secret, it will not be useful to them.

Make the system used to store and process the secrets and credentials robust from a security perspective. Encrypt all secrets at rest and in transit. Users who configure this system and the secrets it contains are subject to the principle of least privilege. For example, a developer might need to manage the secrets for a development environment, but not a user acceptance test or production environment.

Assessment Question(s)

Do you inject production secrets into configuration files during deployment?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Source code files no longer contain active application secrets
- Under normal circumstances, no humans access secrets during deployment procedures
- You log and alert to any abnormal access to secrets

Secure Deployment (SD3)

The goal of this practice level is to deployment process is fully automated and incorporates automated verification of all critical milestones.

Activities

[Stream A] Verify the integrity of deployment artifacts

Benefit: *Assured integrity of artifacts being deployed to production*

Take advantage of binaries being signed at the build time and include automatic verification of the integrity of software being deployed by checking their signatures against trusted certificates. This may include binaries developed and built in-house, as well as third-party artifacts. Do not deploy artifacts if their signatures cannot be verified, including those with invalid or expired certificates.

If the list of trusted certificates includes third-party developers, check them periodically, and keep them in line with the organization’s wider governance surrounding trusted third-party suppliers.

Manually approve the deployment at least once during an automated deployment. Whenever a human check is significantly more accurate than an automated one during the deployment process, go for this option.

Assessment Question(s)

Do you consistently validate the integrity of deployed artifacts?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You prevent or roll back deployment if you detect an integrity breach
- The verification is done against signatures created during the build time
- If checking of signatures is not possible (e.g. externally build software), you introduce compensating measures

[Stream B] Enforce lifecycle management of application secrets

Benefit: *Minimized possibility and timely detection of production secret abuse*

Implement lifecycle management for production secrets, and ensure the generation of new secrets as much as possible, and for every application instance. The use of secrets per application instance ensures that unexpected application behavior can be traced back and properly analyzed. Tools can help in automatically and seemlessly updating the secrets in all relevant places upon change.

Ensure that all access to secrets (both reading and writing) is logged in a central infrastructure. Review these logs regularly to identify unexpected behavior and perform proper analysis to understand why this happened. Feed issues and root causes into the defect management practice to make sure that the organization will resolve any unacceptable situations.

Assessment Question(s)

Do you practice proper lifecycle management for application secrets?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You generate and synchronize secrets using a vetted solution
- Secrets are different between different application instances
- Secrets are regularly updated

Defect Management (DM1)

The goal of this practice level is to all defects are tracked within each project.

Activities

[Stream A] Track security defects centrally

Benefit: *Transparency of known security defects impacting particular applications*

Introduce a common definition / understanding of a security defect and define the most common ways of identifying these. These typically include, but are not limited to:

- Threat assessments
- Penetration tests
- Output from static and dynamic analysis scanning tools
- Responsible disclosure processes or bug bounties

Foster a culture of transparency and avoid blaming any teams for introducing or identifying security defects. Record and track all security defects in a defined location. This location doesn’t necessarily have to centralized for the whole organization, however ensure that you’re able to get an overview of all defects affecting a particular application at any single point in time. Define and apply access rules for the tracked security defects to mitigate the risk of leakage and abuse of this information.

Introduce at least rudimentary qualitative classificiation of security defects so that you are able to prioritize fixing efforts accordingly. Strive for limiting duplication of information and presence of false positives to increase the trustworthiness of the process.

Assessment Question(s)

Do you track all known security defects in accessible locations?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You can easily get an overview of all security defects impacting one application
- You have at least a rudimentary classification scheme in place
- The process includes a strategy for handling false positives and duplicate entries
- The defect management system covers defects from various sources and activities

[Stream B] Define basic defect metrics

Benefit: *Identification of quick wins derived from available defect information*

Once per defined period of time (typically at least once per year), go over your both resolved and still open recorded security defects in every team and extract basic metrics from the available data. These might include:

- The total number of defects versus total number of verification activities. This could give you an idea whether you're looking for defects with an adequate intensity and quality.
- The software components the defects reside in. This is indicative of where attention might be most required, and where security flaws might be more likely to appear in the future again.
- The type or category of the defect, which suggests areas where the development team need further training.
- The severity of the defect, which can help the team understand the software's risk exposure.

Identify and carry out sensible quick win activities which you can derive from the newly acquired knowledge. These might include things like a knowledge sharing session about one particular vulnerability type or carrying out / automating a security scan.

Assessment Question(s)

Do you use basic metrics about recorded security defects to carry out quick win improvement activities?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You analyzed your recorded metrics at least once in the last year
- At least basic information about this initiative is recorded and available
- You have identified and carried out at least one quick win activity based on the data

Defect Management (DM2)

The goal of this practice level is to defect tracking used to influence the deployment process.

Activities

[Stream A] Rate and track security defects

Benefit: *Consistent classification of security defects with clear expecations of their handling*

Introduce and apply a well defined rating methodology for your security defects consistently across the whole organization, based on the probability and expected impact of the defect being exploited. This will allow you to identify applications which need higher attention and investments. In case you don't store the information about security defects centrally, ensure that you're still able to easily pull the information from all sources and get an overview about "hot spots" needing your attention.

Introduce SLAs for timely fixing of security defects according to their criticality rating and centrally monitor and regularly report on SLA breaches. Define a process for cases where it's not feasible or economical to fix a defect within the time defined by the SLAs. This should at least ensure that all relevant stakeholders have a solid understanding of the imposed risk. If suitable, employ compensating controls for these cases.

Even if you don't have any formal SLAs for fixing low severity defects, ensure that responsible teams still get a regular overview about issues affecting their applications and understand how particular issues affect or amplify each other.

Assessment Question(s)

Do you keep an overview of the state of security defects across the organization?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- A single severity scheme is applied to all defects across the organization
- The scheme includes SLAs for fixing particular severity classes
- You regularly report compliance to SLAs

[Stream B] Define advanced defect metrics

Benefit: *Improved learning from security defects in your organization*

Define, collect and calculate unified metrics across the whole organization. These might include:

- Total amount of verification activites and identified defects.
- Types and severities of identified defects.
- Time to detect and time to resolve defects.
- Windows of exposure of defects being present on live systems.
- Number of regressions / reopened vulnerabilities.
- Coverage of verification activities for particular software components.
- Amount of accepted risk.
- Ratio of security incidents caused due to unknown or undocumented security defects.

Generate a regular (e.g. monthly) report for a suitable audience. This would typically reach audience like managers and security officer and engineers. Use the information in the report as an input for your security strategy, e.g. improving trainings or security verification activities.

Share the most prominent or interesting technical details about security defects including the fixing strategy to other teams once these defects are fixed, e.g. in a regular knowledge sharing meeting. This will help scale the learning effect from defects to the whole organization and limit their occurence in the future.

Assessment Question(s)

Do you improve your security assurance program upon standardized metrics?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You document metrics for defect classification and categorization and keep them up to date
- Executive management regularly receives information about defects and has acted upon it in the last year
- You regularly share technical details about security defects among teams

Defect Management (DM3)

The goal of this practice level is to defect tracking across multiple components is used to help reduce the number of new defects.

Activities

[Stream A] Enforce an SLA for defect management

Benefit: *Assurance that security defects are handled within predefined SLAs*

Implement an automated alerting on security defects if the fix time breaches the defined SLAs. Ensure that these defects are automatically transfered into the risk management process and rated by a consistent quantitative methodology. Evaluate how particular defects influence / amplify each other not only on the level of separate teams, but on the level of the whole organization. Use the knowledge of the full kill chain to prioritize, introduce and track compensating controls mitigating the respective business risks.

Integrate your defect management system with the automated tooling introduced by other practices, e.g.:

- Build and Deployment: Fail the build / deployment process if security defects above certain severity affect the final artifact, unless someone explicitely signs off the exception.
- Monitoring: If possible, ensure that abuse of the security defect in production environment is recognized and alerted.

Assessment Question(s)

Do you enforce SLAs for fixing security defects?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You automatically alert of SLA breaches and transfer respective defects to the risk management process
- You integrate relevant tooling (e.g. monitoring, build, deployment) with the defect management system

[Stream B] Use metrics to improve the security strategy

Benefit: *Optimized security strategy based on defect information*

Regularly (at least once per year) revisit the defect management metrics you're collecting and compare the effort needed to collect and track these to the expected outcomes. Make knowledgeable decision about removing metrics which don't deliver the overall expected value. Wherever possible, include and automate verification activities for the quality of the collected data and ensure sustainable improvement if any differences are detected.

Aggregate the data with your threat intelligence and incident management metrics and use the results as input for other initiatives over the whole organization, such as:

- Planning security trainings for various personnel
- Improvement of security verification activities for both internally and externally develeoped collected
- Supply chain management, e.g. carrying out security audits of partner organizations
- Monitoring of attacks against your infrastructure and applications
- Investing in security infrastructure or compensating controls
- Staffing your security team and setting up the security budget

Assessment Question(s)

Do you regularly evaluate the effectiveness of your security metrics so that its input helps drive your security strategy?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have analyzed the effectivenss of the security metrics at least once in the last year
- Where possible, you verify the correctness of the data automatically
- The metrics is aggregated with other sources like threat intelligence or incident management
- You derived at least one strategic activity from the metrics in the last year

Verification

Architecture Assessment (AA1)

The goal of this practice level is to review the architecture to ensure baseline mitigations are in place for typical risks.

Activities

[Stream A] Assess application architecture

Benefit: *Understanding of high-level architecture and sensible security measures*

Create a view of the overall architecture and examine it for the correct provision of general security mechanisms such as authentication, authorization, user and rights management, secure communication, data protection, key management and log management. Also consider the support for privacy. Do this based

on project artifacts such as architecture or design documents, or interviews with business owners and technical staff. Also consider the infrastructure components - these are all the systems, components and libraries (including SDKs) that are not specific to the application, but provide direct support to use or manage the application(s) in the organisation.

Note any security-related functionality in the architecture and review its correct provision. Do this in an ad-hoc manner, from the point of view of anonymous users, authorized users, and specific application roles.

Assessment Question(s)

Do you review the application architecture for key security objectives on an ad-hoc basis?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have an agreed upon model of the overall software architecture
- You include components, interfaces, and integrations in the architecture model
- You verify the correct provision of general security mechanisms
- You log missing security controls as defects

[Stream B] Evaluate architecture for typical threats

Benefit: *Assures that the architecture protects against typical security threats.*

Review the architecture for typical security threats. Security-savvy technical staff conduct this analysis with input from architects, developers, managers, and business owners as needed, to ensure the architecture addresses all common threats which development teams lacking specialised security expertise may have overlooked.

Typical threats in an architecture can relate to incorrect assumptions in, or overly reliance on, the provisioning of security mechanisms such as authentication, authorization, user and rights management, secure communication, data protection, key management and log management. Threats, on the other hand, can also relate to known limitations of, or issues in, technological components or frameworks that are part of the solution and for which insufficient mitigation has been put in place.

Assessment Question(s)

Do you review the application architecture for mitigations of typical threats on an ad-hoc basis?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have an agreed upon model of the overall software architecture
- Security savvy staff conduct the review
- You consider different types of threats, including insider and data-related one

Architecture Assessment (AA2)

The goal of this practice level is to review the complete provision of security mechanisms in the architecture.

Activities

[Stream A] Verify the application architecture for security methodically

Benefit: *Consistent architecture review process across your organization*

Verify that the solution architecture addresses all identified security and compliance requirements. For each interface in the application, iterate through the list of security and compliance requirements and analyze the architecture for their provision. Also perform an interaction or data flow analysis to ensure that the requirements are adequately addressed over different components. Elaborate the analysis to show the design-level features that address each requirement.

Perform this type of analysis on both internal interfaces, e.g. between tiers, as well as external ones, e.g. those comprising the attack surface. Also identify and validate important design decisions made as part of the architecture, in particular when they deviate from the available shared security solutions in the organization. Finally, update the findings based on changes made during the development cycle, and note any requirements that are not clearly provided at the design level as assessment findings.

Assessment Question(s)

Do you regularly review the security mechanisms of your architecture?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You review compliance with internal and external requirements
- You systematically review each interface in the system
- You use a formalized review method and structured validation
- You log missing security mechanisms as defects

[Stream B] Structurally verify the architecture for identified threats

Benefit: *All identified threats to the application are adequately handled.*

Systematically review each threat identified during the Threat Assessment activities and examine how the architecture mitigates them. Use a standardised process for analyzing system architectures and the flow of data within them. This is typically linked to the threat model used (e.g. STRIDE) in order to identify the relevant security objectives which address each type of threat. For each threat, identify the design-level features of the architecture which counter it and assess their effectiveness in doing so.

Where available, review architectural decision records to understand the architectural constraints and tradeoffs made during design. Take their impact into consideration along with any security assumptions on which the safe operation of the system relies and re-evaluate them.

Enrich your previously created threat model such that each threat and its estimated impact are linked to the corresponding counter measure. Produce a mapping document, or dashboard in a specialized tool, to make the information available and visible to the relevant stakeholders.

Assessment Question(s)

Do you regularly evaluate the threats to your architecture?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You systematically review each threat identified in the Threat Assessment
- Trained or experienced people lead review exercise
- You identify mitigating design-level features for each identified threat
- You log unhandled threats as defects

Architecture Assessment (AA3)

The goal of this practice level is to review the architecture effectiveness and feedback results to improve the security architecture.

Activities

[Stream A] Verify the effectiveness of security components

Benefit: *Assurance of effectivness of architecture controls*

Review the effectiveness of the architecture components and their provided security mechanisms in terms of alignment with the overall strategy of the organization, and scrutinize the degree of availability, scalability

and enterprise-readiness of the chosen security solutions. While tactical choices for a particular application can make sense in specific contexts, it is important to keep an eye on the bigger picture and ensure future readiness of the designed solution.

Feed any findings back into defect management to trigger further improvements to the architecture.

Assessment Question(s)

Do you regularly review the effectiveness of the security controls?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You evaluate the preventive, detective, and response capabilities of security controls
- You evaluate the strategy alignment, appropriate support, and scalability of security controls
- You evaluate the effectiveness at least yearly
- You log identified shortcomings as defects

[Stream B] Feed review results back to improve reference architectures

Benefit: *Continuous improvement of enterprise architecture based on architecture reviews*

As an organization, you can further improve your software security posture by understanding which threats remain unaddressed in the software architectures and adapting your tactics to prevent this. Formalize a process to use recurring architecture findings as a trigger to identify the causes of gaps in the security assessment and deal with them. Feed findings back to the Design phase by creating, or updating relevant reference architectures, existing security solutions, or organisation design principles and patterns.

Assessment Question(s)

Do you regularly update your reference architectures based on architecture assessment findings?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You assess your architectures in a standardized, documented manner
- You use recurring findings to trigger a review of reference architectures
- You independently review the quality of the architecture assessments on an ad-hoc basis
- You use reference architecture updates to trigger reviews of relevant shared solutions, in a risk-based manner

Requirements-driven Testing (RT1)

The goal of this practice level is to opportunistically find basic vulnerabilities and other security issues.

Activities

[Stream A] Test the effectiveness of security controls

Benefit: *Verified effectiveness of your standard security controls*

Conduct security tests to verify that the standard software security controls operate as expected. At a high level, this means testing the correct functioning of the confidentiality, integrity, and availability controls of the data as well as the service. Security tests at least include testing for authentication, access control, input validation, encoding, and escaping data and encryption controls. The test objective is to validate that the security controls are correctly implemented.

The security testing validates the relevant software security controls. Perform control-verification security tests manually or with tools, each time the application changes its use of the controls. Techniques such as feature toggles and A/B testing can be used to progressively expose features to broader audiences as they are sufficiently validated. Software control verification is mandatory for all software that is part of the SAMM

program.

Assessment Question(s)

Do you test applications for the correct functioning of standard security controls?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- Security testing at least verifies the implementation of authentication, access control, input validation, encoding and escaping data, and encryption controls
- Security testing executes whenever the application changes its use of the controls

[Stream B] Perform fuzz testing

Benefit: *Insight into behaviour of your applications when dealing with unexpected input*

Perform fuzzing, sending random or malformed data to the test subject in an attempt to make it crash. Fuzz testing or Fuzzing is a Black Box software testing technique, which consists of finding implementation bugs using automated malformed or semi-malformed data injection. Cover at least a minimum fuzzing for vulnerabilities against the main input parameters of the application.

The advantage of fuzz testing is the simplicity of the test design, and its lack of preconceptions about system behavior. The stochastic approach results in bugs that human eyes or structured testing would often miss. It is also one of the few means of assessing the quality of a closed system (such as a SIP phone). The simplicity of fuzzing a target is offset by the difficulty in accurately detecting and triaging crashes. Favour existing fuzzing tools and frameworks to leverage their supporting tooling.

Assessment Question(s)

Do you test applications using randomization or fuzzing techniques?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Testing covers most or all of the application’s main input parameters
- You record and inspect all application crashes for security impact on a best-effort basis

Requirements-driven Testing (RT2)

The goal of this practice level is to perform implementation review to discover application-specific risks against the security requirements.

Activities

[Stream A] Define and run security test cases from requirements

Benefit: *Integration of security requirements into test scenarios*

From the security requirements, identify and implement a set of security test cases to check the software for correct functionality. To have a successful testing program, you must know the testing objectives, specified by the security requirements.

Derive security test cases for the applications in scope from the security requirements created as part of the “Security Requirements” SAMM security practice. To validate security requirements with security tests, security requirements are function-driven and highlight the expected functionality (the what) and, implicitly, the implementation (the how). These requirements are also referred to as “positive requirements”, since they state the expected functionality that can be validated through security tests. Examples of positive requirements include “the application will lockout the user after six failed login attempts” or “passwords need to be a minimum of six alphanumeric characters”. The validation of positive requirements consists of asserting the expected functionality. You can do it re-creating the testing conditions and running the test

according to predefined inputs. Show the results as as a fail or pass condition.

Often, it is most effective to use the project team's time to build application-specific test cases, and publicly available resources or purchased knowledge bases to select applicable general test cases for security. Relevant development, security, and quality assurance staff review candidate test cases for applicability, efficacy, and feasibility. Derive the test cases during the requirements and/or design phase of the functionality. Testing the security requirements is part of the functional testing of the software.

Assessment Question(s)

Do you consistently write and execute test scripts to verify the functionality of security requirements?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- You tailor tests to each application and assert expected security functionality
- You capture test results as a pass or fail condition
- Tests use a standardized framework or DSL

[Stream B] Define and run security abuse cases from requirements

Benefit: *Detection of application business logic flaws*

Misuse and abuse cases describe unintended and malicious use scenarios of the application, describing how an attacker could do this. Create misuse and abuse cases to misuse or exploit the weaknesses of controls in software features to attack an application. Use abuse-case models for an application to serve as fuel for identification of concrete security tests that directly or indirectly exploit the abuse scenarios.

Abuse of functionality, sometimes referred to as a "business logic attack", depends on the design and implementation of application functions and features. An example is using a password reset flow to enumerate accounts. As part of business logic testing, identify the business rules that are important for the application and turn them into experiments to verify whether the application properly enforces the business rule. For example, on a stock trading application, is the attacker allowed to start a trade at the beginning of the day and lock in a price, hold the transaction open until the end of the day, then complete the sale if the stock price has risen or cancel if the price dropped?

Assessment Question(s)

Do you create abuse cases from functional requirements and use them to drive security tests?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- Important business functionality has corresponding abuse cases
- You build abuse stories around relevant personas with well-defined motivations and characteristics
- You capture identified weaknesses as security requirements

Requirements-driven Testing (RT3)

The goal of this practice level is to maintain the application security level after bug fixes, changes or during maintenance.

Activities

[Stream A] Automate security requirements testing

Benefit: *Timely and reliable detection of violations to security requirements*

Write and automate regression tests for all identified (and fixed) bugs to ensure that these become a test harness preventing similar issues being introduced during later releases. Security unit tests should verify dynamically (i.e., at run time) that the components function as expected and should validate that code

changes are properly implemented.

A good practice for developers is to build security test cases as a generic security test suite that is part of the existing unit testing framework. A generic security test suite might include security test cases to validate both positive and negative requirements for security controls such as Identity, Authentication & Access Control, Input Validation & Encoding, User and Session Management, Error and Exception Handling, Encryption, and Auditing and Logging. Verify the correct execution of the security tests as early as possible. If feasible for example, consider the passing of security tests as part of merge requirements before allowing new code to enter the main code base. Alternatively, consider their passing a requirement for validating a build.

For security functional tests, use unit level tests for the functionality of security controls at the software component level, such as functions, methods, or classes. For example, a test case could check input and output validation (e.g., variable sanitation) and boundary checks for variables by asserting the expected functionality of the component.

Assessment Question(s)

Do you automatically test applications for security regressions?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You consistently write tests for all identified bugs (possibly exceeding a pre-defined severity threshhold)
- You collect security tests in a test suite that is part of the existing unit testing framework

[Stream B] Perform security stress testing

Benefit: *Transparency of resilience against denial of service attacks*

Applications are particularly susceptible to denial of service attacks. Perform denial of service and security stress testing against them in controlled conditiions, preferably on application acceptance environments.

Load testing tools generate synthetic traffic, allowing you to test the application’s performance under heavy load. One important test is how many requests per second an application can handle while remaining within its performance requirements. Testing from a single IP address is still useful as it gives an indication of how many requests an attacker must generate to impact the application.

Denial of service attacks typically result in application resource starvation or exhaustion. To determine if any resources can be used to create a denial of service, analyze each application resource to see how it can be exhausted. Prioritise actions unauthenticated user can do. Complement overall denial of service tests with security stress tests to perform actions or create conditions which cause delays, disruptions, or failures of the application under test.

Assessment Question(s)

Do you perform denial of service and security stress testing?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- Stress tests target specific application resources (e.g. memory exhaustion by saving large amounts of data to a user session)
- You design tests around relevant personas with well-defined capabilities (knowledge, resources)
- You feed the results back to the Design practices

Security Testing (ST1)

The goal of this practice level is to perform security testing (both manual and tool based) to discover security defects.

Activities

[Stream A] Perform automated security testing

Benefit: *Detection of common easy-to-find vulnerabilities*

Use automated static and dynamic security test tools for software, resulting in more efficient security testing and higher quality results. Progressively increase the frequency of security tests and extend code coverage.

Application security testing can be performed statically, by inspecting an application's source code without running it, or dynamically by simply observing the application's behavior in response to various input conditions. The former approach is often referred to as Static Application Security Testing (SAST), the latter as Dynamic Application Security Testing (DAST). A hybrid approach, known as Interactive Application Security Testing (IAST), combines the strengths of both approaches (at the cost of additional overhead) by dynamically testing automatically instrumented applications, allowing accurate monitoring of the application's internal state in response to external input.

Many security vulnerabilties are very hard to detect without carefully inspecting the source code. While this is ideally performed by expert or peer review, it is a slow and expensive task. Although "noisier" and frequently less accurate than expert-led reviews, automated SAST tools are cheaper, much faster, and more consitent than humans. A number of commercial and free tools are able to efficiently detect sufficiently important bugs and vulnerabilities in large code bases.

Dynamic testing does not require application source code, making it ideal for cases where source code is not available. It also identifies concrete instances of vulnerabilities. Due to its "black-box" approach , without instrumentation, it is more likely to uncover shallow bugs. Dynamic testing tools need a large source of test data whose manual test generation is prohibitive. Many tools exist which generate suitable test data automatically, leading to more efficient security testing and higher quality results.

Select appropriate tools based on several factors, including depth and accuracy of inspection, robustness and accuracy of security test cases, available integrations with other tools, usage and cost model, etc. When selecting tools, use input from security-savvy technical staff as well as developers and development managers and review results with stakeholders.

Assessment Question(s)

Do you scan applications with automated security testing tools?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- You dynamically generate inputs for security tests using automated tools
- You choose the security testing tools to fit the organization's architecture and technology stack, and balance depth and accuracy of inspection with usability of findings to the organization

[Stream B] Test high risk application components manually

Benefit: *Detection of manually identifable vulnerabilities in critical components*

Perform selective manual security testing, possibly using a combination of static and dynamic analysis tools to guide or focus the review, in order to more thoroughly analyze parts of the application, as an attacker. Automated tools are effective at finding various types of vulnerabilities but can never replace expert manual review.

Code-level vulnerabilities in security-critical parts of software can have dramatically increased impact so project teams 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, and input validators and data parsers. Teams can combine code-level metrics and focused automated scans to determine where best to focus their efforts. In practice, the activity can take many forms including pair programming and peer review, time-boxed security "pushes" involving the entire development team, or spontaneous independent reviews by members of a specialised security group.

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

Assessment Question(s)

Do you manually review the security quality of selected high-risk components?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- Criteria exist to help the reviewer focus on high-risk components
- Qualified personnel conduct reviews following documented guidelines
- You address findings in accordance with the organization’s defect management policy

Security Testing (ST2)

The goal of this practice level is to make security testing during development more complete and efficient through automation complemented with regular manual security penetration tests.

Activities

[Stream A] Develop application-specific security test cases

Benefit: *Detection of organization-specific easy-to-find vulnerabilites*

Increase the effectiveness of automated security testing tools by tuning and customizing them for your particular technology stacks and applications. Automated security testing tools have 2 important characteristics: Their false positive rate, i.e. the non-existent bugs and vulnerabilities they incorrectly report; their false negative rate, i.e. actual bugs and vulnerabilities which they fail to detect. As you mature in your use of automated testing tools, you strive to minimize their false positive and false negative rates. This maximizes the time development teams spend reviewing and addressing real security issues in their applications, and reduces the friction typically associated with using untuned automated security analysis tools.

Start by disabling tool support for technologies and frameworks you do not use and target specific versions where possible. This will increase execution speed and reduce the number spurious results reported. Rely on security tool champions or shared security teams to pilot the tools in coordination with a select group of motivated development teams. This will identify likely false positive findings to ignore or remove from the tools’ output. Identify specific security issues and anti-patterns and favor the best tool for detecting them.

Leverage available tool features to take application-specific and organizational coding styles, as well as technical standards into account. Many automated static analysis tools allow users to write their own rules or customize default analysis rules to the specific software interfaces in the project under test for improved accuracy and depth of coverage. For example, potentially dangerous input (aka tainted) can be marked as safe after it flows through a designated custom sanitization method.

Strategically, it is better to reliably detect a limited subset of security issues via automated tooling, and incrementally extend coverage than attempting to detect all known issues immediately. Once the tools have been sufficiently tuned, they can be made available to a more development teams. It is important to continuously monitor their perceived efficacy among development teams. In more advanced forms, machine learning techniques can be adopted to identify and automatically filter out likely false positives at scale.

Assessment Question(s)

Do you customize the automated security tools to your applications and technology stacks?

- No
- Yes, some of them
- Yes, at least half of them
- Yes, most or all of them

Quality Criteria:

- You tune and select tool features which match your application or technology stack
- You minimize false positives by silencing or automatically filter irrelevant warnings or low probability findings
- You minimize false negatives by leverage tool extensions or DSLs to customize tools for your application or organizational standards

[Stream B] Establish a penetration testing process

Benefit: *Understanding of application resilience from black-box perspective*

Using the set of security test cases identified for each project, conduct manual penetration testing to evaluate the system's performance against each case. Generally, this happens during the testing phase prior to release and includes both static and dynamic manual penetration testing. In cases where software cannot be realistically tested outside of production, use of techniques such as blue-green deployments or A/B testing can allow ring-fenced security testing in production.

Penetration testing cases include both application-specific tests to check soundness of business logic, and common vulnerability tests to check the design and implementation. Once specified, security-savvy quality assurance or development staff can execute security test cases. The central software security group monitors first-time execution of security test cases for a project team to assist and coach the team security champions.

Many organizations offer "Bug Bunty" programmes which invite security researchers to find vulnerabilties in applications and report them responsibly in exchange for rewards. The approach allows organizations to access a bigger pool of talent, especially those lacking sufficient internal capacity or requiring the additional assurance.

Prior to release or mass deployment, stakeholders review results of security tests and accept the risks indicated by failing security tests at release time. Establish a concrete timeline to address the gaps over time. Spread the knowledge of manual security testing and the results across the development team to improve security knowledge and awareness inside the organization.

Assessment Question(s)

Do you perform penetration testing for your applications at regular intervals?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- Penetration testing uses application-specific security test cases to evaluate security
- Penetration testing looks for both technical and logical issues in the application
- Stakeholders review the test results and handle them in accordance with the organization's risk management
- Qualified personnnel performs penetration testing

Security Testing (ST3)

The goal of this practice level is to embed security testing as part of the development and deployment processes.

Activities

[Stream A] Integrate security testing tools in the delivery pipeline

Benefit: *Identification of automatically identifiable vulnerabilities in earliest possible stages*

Projects within the organization routinely run automated security tests and review results during development. Configure security testing tools to automatically run as part of the build and deploy process to make this scalable with low overhead. Inspect findings 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. 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, lowering risk of release delays due to security concerns or forced acceptance of risk to meet project deadlines.

Make the results of automated and manual security tests visible via dashboards, and routinely present them to management and business stakeholders (e.g. before each release) for review. If there are unaddressed findings that remain as accepted risks for the release, stakeholders and development managers work

together to establish a concrete timeframe for addressing them. Continuously review and improve the quality of the security tests.

Consider and implement security test correlation tools to automate the matching and merging of test results from dynamic, static, and interactive application scanners into one central dashboard, providing direct input towards Defect Management. Spread the knowledge of the created security tests and the results across the development team to improve security knowledge and awareness inside the organization.

Assessment Question(s)

Do you integrate automated security testing into the build and deploy process?

- No
- Yes, some of it
- Yes, at least half of it
- Yes, most or all of it

Quality Criteria:

- Management and business stakeholders track and review test results throughout the development cycle
- You merge test results into a central dashboard and feed them into defect management

[Stream B] Establish continuous, scalable security verification

Benefit: *Identification of manually identifiable security issues in earliest possible stages*

Integrate security testing in parallel to all other development activities, including requirement analysis, software design and construction.

With the multiplicity of security tools running at every phase of development, remediating security issues at a designated stage (such as pre-release testing) is no longer appropriate or desirable. Security issues must be quickly triaged and fixes planned in a tradeoff between risk and cost of remediation. Continuously striving to detect issues earlier in the development lifecycle, via specific, low-friction automated tests integrated into development tools and build processes, lowers the cost of remediation thereby increasing the likelihood of issues being quickly resolved.

Proactively improve the security testing effort integrated into the development process by adequately propagating the results of other security test activities. For example, if a security penetration test identifies issues with session management, any changes to session management should trigger explicit security tests before pushing the changes to production.

Security champions and the central secure software group continuously review results from automated and manual security tests during development, including these results as part of the security awareness trainings for the development teams. Integrate lessons learned in overall playbooks to improve security testing as part of the organization development. If there are unaddressed findings that remain as accepted risks for the release, stakeholders and development managers should work together to establish a concrete timeframe for addressing them.

Assessment Question(s)

Do you use the results of security testing to improve the development lifecycle?

- No
- Yes, but we improve it ad-hoc
- Yes, we we improve it at regular times
- Yes, we improve it at least annually

Quality Criteria:

- You use results from other security activities to improve integrated security testing during development
- You review test results and incorporate them into security awareness training and security testing playbooks
- Stakeholders review the test results and handle them in accordance with the organization’s risk management

Operations

Incident Management (IM1)

The goal of this practice level is to best-effort incident detection and handling

Activities

[Stream A] Use best-effort incident detection

Benefit: *Ability to detect the most obvious security incidents*

Analyze available log data (e.g., access logs, application logs, infrastructure logs), to detect possible security incidents in accordance with known log data retention periods.

In small setups, you can do this manually with the help of common command-line tools. With larger log volumes, employ automation techniques. Even a `cron` job, running a simple script to look for suspicious events, is a step forward!

If you send logs from different sources to a dedicated log aggregation system, analyze the logs there and employ basic log correlation principles.

Even if you don’t have a 24/7 incident detection process, ensure that unavailability of the responsible person (e.g., due to vacation or illness) doesn’t significantly impact detection speed or quality.

Establish and share points of contact for formal creation of security incidents.

Assessment Question(s)

Do you analyze log data for security incidents periodically?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You have a contact point for the creation of security incidents
- You analyze data in accordance with the log data retention periods
- The frequency of this analysis is aligned with the criticality of your applications

[Stream B] Create an incident response plan

Benefit: *Ability to efficiently solve most common security incidents*

The first step is to recognize the incident response competence as such, and define a responsible owner. Provide them the time and resources they need to keep up with current state of incident handling best practices and forensic tooling.

At this level of maturity, you may not have established a dedicated incident response team, but you have defined the participants of the process (usually different roles). Assign a single point of contact for the process, known to all relevant stakeholders. Ensure that the point of contact knows how to reach each participant, and define on-call responsibilities for those who have them.

When security incidents happen, document all actions taken. Protect this information from unauthorized access.

Assessment Question(s)

Do you respond to detected incidents?

- No
- Yes, for some incidents
- Yes, for at least half of the incidents
- Yes, for most or all of the incidents

Quality Criteria:

- You have a defined person or role for incident handling
- You document security incidents

Incident Management (IM2)

The goal of this practice level is to formal incident management process in place

Activities

[Stream A] Define an incident detection process

Benefit: *Timely and consistent detection of expected security incidents*

Establish a dedicated owner for the incident detection process, make clear documentation accessible to all process stakeholders, and ensure it is regularly reviewed and updated as necessary. Ensure employees responsible for incident detection follow this process (e.g., using training).

The process typically relies on a high degree of automation, collecting and correlating log data from different sources, including application logs. You may aggregate logs in a central place, if suitable. Periodically verify the integrity of analyzed data. If you add a new application, ensure the process covers it within a reasonable period of time.

Detect possible security incidents using an available checklist. The checklist should cover expected attack vectors and known or expected kill chains. Evaluate and update it regularly.

When you determine an event is a security incident (with sufficiently high confidence), notify responsible staff immediately, even outside business hours. Perform further analysis, as appropriate, and start the escalation process.

Assessment Question(s)

Do you follow a documented process for incident detection?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- The process has a dedicated owner
- You store process documentation in an accessible location
- The process considers an escalation path for further analysis
- You train employees responsible for incident detection in this process
- You have a checklist of potential attacks to simplify incident detection

[Stream B] Define and incident response process

Benefit: *Understanding and efficient handling of most security incidents*

Establish and document the formal security incident response process. Ensure documentation includes information like:

- most probable/common scenarios of security incidents and high-level instructions for handling them; for such scenarios, also use public knowledge about possibly relevant third-party incidents
- rules for triaging each incident
- rules for involvement of different stakeholders including senior management, Public Relations, Legal, privacy, Human Resources, external (law enforcement) authorities, and customers; specify mandatory timeframe to do so, if needed
- the process for performing root-cause analysis and documentation of its results

Ensure a knowledgeable and properly trained incident response team is available both during and outside of business hours. Define timelines for action and a war room. Keep hardware and software tools up to date and ready for use anytime.

Assessment Question(s)

Do you use a repeatable process for incident handling?

- No
- Yes, for some incident types
- Yes, for at least half of the incident types
- Yes, for most or all of the incident types

Quality Criteria:

- You have an agreed upon incident classification
- The process considers Root Case Analysis for high severity incidents
- Employees responsible for incident response are trained in this process
- Forensic analysis tooling is available

Incident Management (IM3)

The goal of this practice level is to mature incident management

Activities

[Stream A] Improve the incident detection process

Benefit: *Ability to timely detect security incidents*

Ensure process documentation includes measures for continuous process improvement. Check the continuity of process improvement (e.g., via tracking of changes).

Ensure the checklist for suspicious event detection is correlated at least from (i) sources and knowledge bases external to the company (e.g., new vulnerability announcements affecting the used technologies), (ii) past security incidents, and (iii) threat model outcomes.

Use correlation of logs for incident detection for all reasonable incident scenarios. If the log data for incident detection is not available, document its absence as a defect, triage and handle it according to your established Defect Management process.

The quality of the incident detection does not depend on the time or day of the event. If security events are not acknowledged and resolved within a specified time (e.g., 20 minutes), ensure further notifications are generated according to an established escalation path.

Assessment Question(s)

Do you review and update the incident detection process regularly?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You perform reviews at least annually
- You update the checklist of potential attacks with external and internal data

[Stream B] Establish an incident response team

Benefit: *Efficient incident response independent of time, location, or type of incident*

Establish a dedicated incident response team, continuously available and responsible for continuous process improvement with the help of regular RCAs. For distributed organizations, define and document logistics rules for all relevant locations if sensible.

Document detailed incident response procedures and keep them up to date. Automate procedures where appropriate. Keep all resources necessary for these procedures (e.g., separate communicating infrastructure or reliable external location) ready to use. Detect and correct unavailability of these resources in a timely manner.

Carry out incident and emergency exercises are regularly. Use the results for process improvement.

Define, gather, evaluate, and act upon metrics on the incident response process, including its continuous improvement.

Assessment Question(s)

Do you have a dedicated incident response team available?

- No

- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- The team performs Root Cause Analysis for all security incidents unless there is a specific reason not to do so
- You review and update the response process at least annually

Environment Management (EM1)

The goal of this practice level is to best-effort patching and hardening

Activities

[Stream A] Use best-effort hardening

Benefit: *Hardened basic configuration settings of your components*

Understanding the importance of securing the technology stacks you're using, apply secure configuration to stack elements, based on readily available guidance (e.g., open source projects, vendor documentation, blog articles). When your teams develop configuration guidance for their applications, based on trial-and-error and information gathered by team members, encourage them to share their learnings across the organization.

Identify key elements of common technology stacks, and establish configuration standards for those, based on teams' experiences of what works.

At this level of maturity, you don't yet have a formal process for managing configuration baselines. Configurations may not be applied consistently across applications and deployments, and monitoring of conformance is likely absent.

Assessment Question(s)

Do you harden configurations for key components of your technology stacks?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- You have identified the key components in each technology stack used
- You have an established configuration standard for each key component

[Stream B] Practice best-effort patching

Benefit: *Mitigation of well-known issues in third-party components*

Identify applications and third-party components which need to be updated or patched, including underlying operating systems, application servers, and third-party code libraries.

At this level of maturity, your identification and patching activities are best-effort and *ad hoc*, without a managed process for tracking component versions, available updates, and patch status. However, high-level requirements for patching activities (e.g., testing patches before pushing to production) may exist, and product teams are achieving best-effort compliance with those requirements.

Except for critical security updates (e.g., an exploit for a third-party component has been publicly released), teams leverage maintenance windows established for other purposes to apply component patches. For software developed by the organization, component patches are delivered to customers and organization-managed solutions only as part of feature releases.

Teams share their awareness of available updates, and their experiences with patching, on an *ad hoc* basis. Ensure teams can determine the versions of all components in use, to evaluate whether their products are affected by a security vulnerability when notified. However, the process for generating and maintaining component lists may require significant analyst effort.

Assessment Question(s)

Do you identify and patch vulnerable components?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- You have an up-to-date list of components, including version information
- You regularly review public sources for vulnerabilities related to your components

Environment Management (EM2)

The goal of this practice level is to formal process with baselines in place

Activities

[Stream A] Establish hardening baselines

Benefit: *Consistent hardening of technology stack components in your organization*

Establsh configuration hardening baselines for all components in each technology stack used. To assist with consistent application of the hardening baselines, develop configuration guides for the components. Require product teams to apply configuration baselines to all new systems, and to existing systems when practicable.

Place hardening baselines and configuration guides under change management, and assign an owner to each. Owners have ongoing responsibility to keep them up-to-date, based on evolving best practices or changes to the relevant components (e.g., version updates, new features).

In larger environments, derive configurations of instances from a locally maintained master, with relevant configuration baselines applied. Employ automated tools for hardening configurations.

Assessment Question(s)

Do you have hardening baselines for your components?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- You have assigned an owner for each baseline
- The owner keeps their assigned baselines up to date
- You store baselines in an accessible location
- You train employees responsible for configurations in these baselines

[Stream B] Formalize patch management

Benefit: *Consistent and proactive patching of technology stack components*

Develop and follow a well-defined process for managing patches to application components across the technology stacks in use. Ensure processes include regular schedules for applying vendor updates, aligned with vendor update calendars (e.g., Microsoft Patch Tuesday). For software developed by the organization, deliver releases to customers and organization-managed solutions on a regular basis (e.g., monthly), regardless of whether you are including new features.

Create guidance for prioritizing component patching, reflecting your risk tolerance and management objectives. Consider operational factors (e.g., criticality of the application, severity of the vulnerabilities addressed) in determining priorities for testing and applying patches.

In the event receive a notification for a critical vulnerability in a component, while no patch is yet available, triage and handle the situation as a risk management issue (e.g., implement compensating controls, obtain customer risk acceptance, or disable affected applications/features).

Assessment Question(s)

Do you follow an established process for updating components of your technology stacks?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- The process includes vendor information for third-party patches
- The process considers external sources to gather information about zero day attacks, and includes appropriate risk mitigation steps
- The process includes guidance for prioritizing component updates

Environment Management (EM3)

The goal of this practice level is to conformity with continuously improving process enforced

Activities

[Stream A] Perform continuous configuration monitoring

Benefit: *Clear view on component configurations to avoid non-conformities*

Actively monitor the security configurations of deployed technology stacks, performing regular checks against established baselines. Ensure results of configuration checks are readily available, through published reports and dashboards.

When you detect non-conforming configurations, treat each occurrence as a security finding, and manage corrective actions within your established Defect Management practice.

Further gains may be realized using automated measures, such as “self-healing” configurations and security information and event management (SIEM) alerts.

As part of the process for updating components (e.g., new releases, vendor patches), review corresponding baselines and configuration guides, updating them as needed to maintain their relevance and accuracy. Review other baselines and configuration guides at least annually.

Periodically review your baseline management process, incorporating feedback and lessons learned from teams applying and maintaining configuration baselines and configuration guides.

Assessment Question(s)

Do you monitor and enforce conformity with hardening baselines?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- You perform conformity checks regularly, preferably using automation
- You store conformity check results in an accessible location
- You follow an established process to address reported non-conformities
- You review each baseline at least annually, and update it when required

[Stream B] Enforce timely patch management

Benefit: *Clear view on component patch state to avoid non-conformities*

Develop and use management dashboards/reports to track compliance with patching processes and SLAs, across the portfolio. Ensure dependency management and application packaging processes can support applying component-level patches at any time, to meet required SLAs.

Treat missed updates as security-related product defects, and manage their triage and correction in accordance with your established Defect Management practice.

Don’t rely on routine notifications from component vendors to learn about vulnerabilities and associated patches. Monitor a variety of external threat intelligence sources, to learn about zero day vulnerabilities; handle those affecting your applications as risk management issues.

Assessment Question(s)

Do you regularly evaluate components and review patch level status?

- No
- Yes, for some components
- Yes, for at least half of the components
- Yes, for most or all of the components

Quality Criteria:

- You update the list with components and versions
- You identify and update missing updates according to existing SLA
- You review and update the process based on feedback from the people who perform patching

Operational Management (OM1)

The goal of this practice level is to foundational Practices

Activities

[Stream A] Organize basic data protections

Benefit: *Understanding of sensitivity of processed data with derived quick-win measures*

Understand the types and sensitivity of data stored and processed by your applications, and maintain awareness of the fate of processed data (e.g., backups, sharing with external partners). At this level of maturity, the information gathered may be captured in varying forms and different places; no organization-wide data catalog is assumed to exist. Protect and handle all data associated with a given application according to protection requirements applying to the most sensitive data stored and processed.

Implement basic controls, to prevent propagation of unsanitized sensitive data from production environments to lower environments. By ensuring unsanitized production data are never propagated to lower (non-production) environments, you can focus data protection policies and activities on production.

Assessment Question(s)

Do you protect and handle information according to protection requirements for data stored and processed on each application?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You know the data elements processed and stored by each application
- You know the type and sensitivity level of each identified data element
- You have controls to prevent propagation of unsanitized sensitive data from production to lower environments

[Stream B] Identify unused applications

Benefit: *Identification of unused of software assets or components*

Identify unused applications on an *ad hoc* basis, either by chance observation, or by occasionally performing a review. When you identify unused applications, process those findings for further action. If you have established a formal process for decommissioning unused applications, ensure teams are aware of and use it.

Manage customer/user migration from older versions of your products for each product and customer/user group. When a product version is no longer in use by any customer/user group, discontinue support for that version. However, at this level of maturity you may have a large number of product versions in active use across the customer/user base, requiring significant developer effort to back-port product fixes.

Assessment Question(s)

Do you identify and remove systems, applications, application dependencies, or services that are no longer used, have reached end of life, or are no longer actively developed or supported?

- No
- Yes, for some applications
- Yes, for at least half of the applications
- Yes, for most or all of the applications

Quality Criteria:

- You do not use unsupported applications or dependencies
- You manage customer/user migration from older versions for each product and customer/user group

Operational Management (OM2)

The goal of this practice level is to managed, Responsive Processes

Activities

[Stream A] Establish a data catalog

Benefit: *Standardized handling of different classes of sensitive data*

At this maturity level, Data Protection activities focus on actively managing your stewardship of data. Establish technical and administrative controls to protect the confidentiality of sensitive data, and the integrity and availability of all data in your care, from its initial creation/receipt through the destruction of backups at the end of their retention period.

Identify the data stored, processed, and transmitted by applications, and capture information regarding their types, sensitivity (classification) levels, and storage location(s) in your data catalog. Clearly identify records or data elements subject to specific regulation. Establishing a single source of truth regarding the data you work with supports finer-grained selection of controls for their protection. Collecting this information enhances the accuracy, timeliness, and efficiency of your responses to data-related queries (e.g., from auditors, incident response teams, or customers), and supports threat modeling and compliance activities.

Based on your Data Protection Policy, establish processes and procedures for protecting and preserving data throughout their lifetime, whether at rest, while being processed, or in transit. Pay particular attention to the handling and protection of sensitive data outside the active processing system, including, but not limited to: storage, retention, and destruction of backups; and the labeling, encryption, and physical protection of offline storage media. Your processes and procedures cover the implementation of all controls adopted to comply with regulatory, contractual, or other restrictions on storage locations, personnel access, and other factors.

Assessment Question(s)

Do you maintain a data catalog, including types, sensitivity levels, and processing and storage locations?

- No
- Yes, for some of our data
- Yes, for at least half of our data
- Yes, for most or all of our data

Quality Criteria:

- The data catalog is stored in an accessible location
- You know which data elements are subject to specific regulation
- You have controls for protecting and preserving data throughout its lifetime
- You have retention requirements for data, and you destroy backups in a timely manner after the relevant retention period ends

[Stream B] Formalize decommissioning process

Benefit: *Standardized decommisioning process decreasing the risk of forgetting components*

As part of decommissioning a system, application, or service, follow an established process for removing all relevant accounts, firewall rules, data, etc. from the operational environment. By removing these unused elements from configuration files, you improve the maintainability of infrastructure-as-code resources.

Follow a consistent process for timely replacement or upgrade of third-party applications, or application dependencies (e.g., operating system, utility applications, libraries), that have reached end of life.

Engage with customers and user groups for your products at or approaching end of life, to migrate them to supported versions in a timely manner.

Assessment Question(s)

Do you follow an established process for removing all associated resources, as part of decommissioning of unused systems, applications, application dependencies, or services?

- No
- Yes, some of the time
- Yes, at least half of the time
- Yes, most or all of the time

Quality Criteria:

- You document the status of support for all released versions of your products, in an accessible location
- The process includes replacement or upgrade of third-party applications, or application dependencies, that have reached end of life
- Operating environments do not contain orphaned accounts, firewall rules, or other configuration artifacts

Operational Management (OM3)

The goal of this practice level is to active Monitoring and Response

Activities

[Stream A] Respond to data breaches

Benefit: *Technically enforced compliance with your data protection policy*

Activities at this maturity level are focused on automating data protection, reducing your reliance on human effort to assess and manage compliance with policies. There is a focus on feedback mechanisms and proactive reviews, to identify and act on opportunities for process improvement.

Implement technical controls to enforce compliance with your Data Protection Policy, and put monitoring in place to detect attempted or actual violations. You may use a variety of available tools for data loss prevention, access control and tracking, or anomalous behavior detection.

Regularly audit compliance with established administrative controls, and closely monitor performance and operation of automated mechanisms, including backups and record deletions. Monitoring tools quickly detect and report failures in automation, permitting you to take timely corrective action.

Reviews and update the data catalog regularly, to maintain its accurate reflection of your data landscape. Regular reviews and updates of processes and procedures maintain their alignment with your policies and priorities.

Assessment Question(s)

Do you regularly review and update the data catalog and your data protection policies and procedures?

- No
- Yes, we do it when requested
- Yes, we do it every few years
- Yes, we do it at least annually

Quality Criteria:

- You have automated monitoring to detect attempted or actual violations of the Data Protection Policy
- You have tools for data loss prevention, access control and tracking, or anomalous behavior detection
- You periodically audit the operation of automated mechanisms, including backups and record deletions

[Stream B] Review application lifecycle state regularly

Benefit: *Full visibility into lifecycle of all software assets*

Regularly evaluate the lifecycle state and support status of every software asset and underlying infrastructure component, and estimate their end-of-life. Follow a well-defined process for actively mitigating security risks arising as assets/components approach their end-of-life. Regularly review and update your process, to reflect lessons learned.

Establish a product support plan, providing clear timelines for ending support on older product versions. Limit product versions in active use to only a small number (e.g., N.x.x and N-1.x.x only). Establish and publicize timelines for discontinuing support on prior versions, and proactively engage with customers and user groups to prevent disruption of service or support.

Assessment Question(s)

Do you regularly evaluate the lifecycle state and support status of every software asset and underlying infrastructure component, and estimate their end of life?

- No
- Yes, for some of the assets
- Yes, for at least half of the assets
- Yes, for most or all of the assets

Quality Criteria:

- Your end of life management process is agreed upon
- You inform customers and user groups of product timelines to prevent disruption of service or support
- You review the process at least annually

Credits

The OWASP SAMM project is led by Sebastien Deleersnyder & Bart De Win with contributions from:

- Sebastian Arriada
- Chris Cooper
- Patricia Duarte
- John DiLeo
- John Ellingsworth
- Bryan Glass
- Daniel Kefer
- John Kennedy
- Nessim Kisserli
- Yan Kravchenko
- Hardik Parekh

Sponsors

By sponsoring SAMM, you support a Flagship OWASP project. The OWASP Flagship designation is given to projects that have demonstrated strategic value to OWASP and application security as a whole.

Don’t hesitate to contact us.

splunk> TOREON

Proceeds

All proceeds from the sponsorship support the mission of the OWASP Foundation and the further development of SAMM, funding

- marketing & PR support
- technical editing & UX support
- website development and hosting
- SAMM participation in the Open Security Summit
- core team summits
- tooling for the SAMM Benchmark project

Sponsorship levels

Sponsorship level	Supporter	Ambassador	Leader
Price (in USD)	1000	2500	5000

Sponsorship benefits

By sponsoring SAMM, you get

- visibility during the next SAMM Summit
- recognition on our website and the v 2.0 release of SAMM

Website benefits: listing on	Supporter	Ambassador	Leader
owaspsamm.org website home page with logo			x
owaspsamm.org website supporters page with logo		x	x
OWASP SAMM wiki project page (supporters tab) with logo		x	x
OWASP SAMM wiki project page (supporters tab) with name	x	x	x
OWASP SAMM GitHub page with logo	x	x	x

Join as a sponsor

1. Select your level.
2. We draw a sponsorship contract and provide you with an invoice.
3. Upon payment, we activate your benefits for 1 year and for SAMM release 2.0 publications.

License