Strategy & Metrics (SM1)

Identify objectives and means of measuring effectiveness of the security program.

Activities

Stream A : Create and Promote

Benefit: *Have a common understanding of an application security baseline.*

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 life-cycle 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.

Stream B : Measure and Improve

Benefit: *Have a set of base metrics to provide insight into software security.*

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.

Strategy & Metrics (SM2)

Establish a unified strategic roadmap for software security within the organization.

Activities

Stream A : Create and Promote

Benefit: *Have an aligned plan and roadmap within the organization.*

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 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.

Stream B : Measure and Improve

Benefit: *A set of concrete objectives has been established to guide your improvement efforts.*

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.

Strategy & Metrics (SM3)

Align security efforts with the relevant organizational indicators and asset values.

Activities

Stream A : Create and Promote

Benefit: *Continuous improvement of your application security efforts.*

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.

Stream B : Measure and Improve

Benefit: *Your application security program is fundamentally driven by objective measures and concrete goals.*

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 life-cycle 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.

Policy & Compliance (PC1)

Identify and document governance and compliance drivers relevant to the organization.

Activities

Stream A : Policy & Standards

Benefit: *Have a common set of policies and standards within your 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.

Stream B : Compliance Management

Benefit: *Have a common understanding of external compliance requirements.*

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.

Policy & Compliance (PC2)

Establish application-specific security and compliance baseline.

Activities

Stream A : Policy & Standards

Benefit: *Have clearly defined evaluation methods to test for adherence to policies and standards.*

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.

Stream B : Compliance Management

Benefit: *Have a standard set of requirements for 3rd party compliance.*

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.

Policy & Compliance (PC3)

Measure adherence to policies, standards, and 3rd-party requirements.

Activities

Stream A : Policy & Standards

Benefit: *Understand your organization’s compliance towards 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.

Stream B : Compliance Management

Benefit: *Have an understanding of your organization’s adherence to 3rd party compliance requirements.*

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.

Education & Guidance (EG1)

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

Activities

Stream A : Training and Awareness

Benefit: *Stakeholders involved in producing software have an appreciation for the difficulty of creating secure software and the value of a secure SDLC.*

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.

Stream B : Organization and Culture

Benefit: *Have a lightweight embedding of software security throughout your organization through security champions.*

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.

Education & Guidance (EG2)

Educate all personnel in the software life-cycle with technology and role-specific guidance on secure development.

Activities

Stream A : Training and Awareness

Benefit: *Stakeholders involved in producing software receive role-specific security training.*

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 SDLC life-cycle, 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. 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.

Stream B : Organization and Culture

Benefit: *Have a central team of software security experts to drive and support your software assurance program.*

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 life-cycle or established security controls.

Education & Guidance (EG3)

Develop in-house training programs facilitated by developers across different teams.

Activities

Stream A : Training and Awareness

Benefit: *Security is an aspect of product quality, addressed throughout development.*

Implement a formal training program requiring anyone involved with the software development life-cycle to complete appropriate role and technology-specific training as part of the on-boarding process. Based on the criticality of the application and user’s role, consider restricting access until the on-boarding 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.

Stream B : Organization and Culture

Benefit: *Software security is a shared and active responsibility among all employees.*

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.

Threat Assessment (TA1)

Consider security explicitly during the software requirements process.

Activities

Stream A : Application Risk Profile

Benefit: *Ability to classify applications according to risk*

As an organization, you want to spend your security budget where it matters. Application risk is a good tool to guide your security spending. A risk classification helps identify which applications can pose a serious threat to the organization if they were attacked or breached. 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. Evaluate the risk based on the set of questions and assign a risk level to each application.

Stream B : Threat Modeling

Benefit: *Basic understanding of potential threats to the solution.*

The purpose of Threat Modeling is to pro-actively identify potential issues in the technical design of the application. A careless setup might lead to important attack vectors in an application that can be exploited to target your organization. Experience shows that architectural design can be an important source of security issues, and the consequences can be significant. The practice of threat modeling includes both eliciting and managing threats. Use known good security practices (or the lack thereof) or a more structured approach such as STRIDE to elicit threats. Threat modeling is often most effective when performed by a group of people, allowing for brainstorming. One of the key challenges in threat modeling is working towards a list of relevant and important threats in an efficient exercise, and avoiding lengthy processes and overly detailed lists of low-relevant threats. Experience helps find a proper balance. 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. Execute threat modeling on important projects (LINK Application Risk Profile) in a best effort mode to identify the most important threats to the application. Existing network diagrams you can annotate during discussion workshops are a good starting point.

Threat Assessment (TA2)

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

Activities

Stream A : Application Risk Profile

Benefit: *Solid understanding of the risk level of an application*

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. Consider using quantitative schemes to 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, build a centralized inventory of risk profiles that use the outcome of the risk evaluations to define the profile. This inventory gives all stakeholders an aligned view of the risk level of an application to assign appropriate priority to security-related activities.

Stream B : Threat Modeling

Benefit: *Improved elicitation and management of threats to the solution.*

Establish a standard approach to perform structured threat modeling to increase the quality and efficiency of threat modeling within your organization, and ensure that the invested effort is useful and well spent. Structured threat modeling takes into account the different actors, assets and flows to identify an extensive list of potential threats to the application. It defines the inputs required to start the activity (e.g., a technical architecture overview and a data flow diagram), the different steps to identify threats, and the formalisms to describe or annotate the threats. You can add mitigating controls to threat models to guide designers in dealing with particular threats. As an organization, define what triggers the execution of threat modeling. For example a change in architecture, or a deployment of an application in a new environment. At the same time, think about ways to support scaling of threat modeling throughout the organization. Feed the output of threat modeling to the defect management process for adequate follow-up. Adopt a weighting system to measure and compare the importance of the different threats. Consider using a tool to manage the treat models of the different applications. Train people to focus on important threats, as one of the challenges in threat modeling is a potential overload of trivial threats. Tools help in identifying potential threats but, in the end, threat modeling requires human intelligence that cannot be easily automated.

Threat Assessment (TA3)

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

Activities

Stream A : Application Risk Profile

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.

Stream B : Threat Modeling

Benefit: *Timely update and qualitative management of application threat*

In a mature setup of threat modeling, regularly (e.g., yearly) review the existing threat models to verify that no new threats are relevant for your applications. Use automated analysis to evaluate the quality and discover gaps and/or patterns in the threat models. These can improve the threat models. Review the threat categories relevant to your organization. When you identify new threat categories, feed this information to the organization to ensure appropriate handling.

Security Requirements (SR1)

Consider security explicitly during the software requirements process.

Activities

Stream A : Software Requirements

Benefit: *You have an understanding of key security requirements.*

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 behaviour and quality of the application (e.g., “Ensure personal data is properly protected in transit”), which will not 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.

Stream B : Supplier Security

Benefit: *You understand the 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. Conduct interviews and 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 organisation, or even organise 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 organisation. 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.

Security Requirements (SR2)

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

Activities

Stream A : Software Requirements

Benefit: *You have specified relevant security requirements in a structured format.*

Security requirements can originate from other sources including policies and legislation, known problems in the application, and intelligence from metrics and feedback (LINK to DM lvl 3). 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), analyse 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).

Stream B : Supplier Security

Benefit: *You structurally assign responsibilities for software security activities.*

Increase your confidence in the capability of your suppliers for software security. Discuss concrete responsibilities and expectations from your suppliers and your own organisation 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.

Security Requirements (SR3)

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

Activities

Stream A : Software Requirements

Benefit: *You have a set of reusable security requirements to improve the overall quality.*

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 organisational 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 DOD, DOR, story description and acceptance criteria.

Stream B : Supplier Security

Benefit: *You align software development practices 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 or 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.

Security Architecture (SA1)

Insert consideration of proactive security guidance into the software design process.

Activities

Stream A : Architecture Design

Benefit: *You get basic security practices right in your software design.*

During design, technical staff on the project 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 design 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 project team with security awareness before this process, and incorporate more security-savvy staff to aid in making design decisions.

Stream B : Technology Management

Benefit: *Risky technologies are identified and replaced*

People often take the path of least resistance in developing, deploying or operating a software solution. New technologies are 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 organisation that you need to manage. Identify the most important technologies, frameworks, tools and integrations being used for each application. Use the solution architect’s knowledge, or study the development and operating environment and artefacts. Then evaluate them for their security quality and raise important issues (LINK TO defect management).

Security Architecture (SA2)

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

Activities

Stream A : Architecture Design

Benefit: *The organisation leverages common security solutions.*

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 evaluat 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 development or an operations 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 development teams to understand how to integrate with the system. Make the guidance available to development or operations teams through training, mentorship, guidelines, and standards. Establish a set of general design patterns representing sound methods of implementing security functionality. You can research them or purchase them, and it is often even 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 development and/or operations teams across the organisation 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 organisation, for instance in case of different development stacks (LINK TO Technology Management). Treat these solutions as actual applications with proper support in case of questions or issues.

Stream B : Technology Management

Benefit: *There is a common agreement on the key technologies to use*

Identify commonly used technologies, frameworks and tools in use across software projects in the organisation, 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 organisation. 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.

Secure Build (SB1)

Build process is repeatable and consistent.

Activities

Stream A : Build Process

Benefit: *Consistent and repeatable builds help developers focus on application-specific issues, and make it possible to automate builds in the future. This reduces the likelihood of human error during builds which can lead to security vulnerabilities.*

Define the build process, breaking it down into a set of clear instuctions to either be followed by a person or an automated tool. The process is complete so that the person or tool can follow it consistently each time and produce the same result. The process definition does not include any secrets (specifically considering those needed during the build process). Use individual credentials that authenticate, authorize, and account to access build tools, and code repositories. Include shared secrets only where you cannot avoid it, managing them with care, preferably via an encrypted password vault. The build process is stored centrally and accessible to any tools or people who might need access. Do not store or distribute multiple copies, some of which may become outdated. Review any build tools routinely, 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.

Stream B : Software Dependencies

Benefit: *You know which production components are at risk from a known vulnerable 3rd party dependencies. Dependencies include 3rd party software dependencies and operating system dependencies. 3rd party dependencies often inculde more dependencies (called transitive dependencies).*

Keep a record of all dependencies used throughout the target production environment. This is sometimes referred to as a Bill of Materials (BOM). In building these records, consider the various locations where dependencies might be specified:

- configuration files
- the project's directory on disk
- package management tool
- code (e.g. via an IDE that supports listing dependencies)

Consider that the different dependencies and aspects of the application may consume entirely different dependencies. For example, if the software package is a web app, cover both the server-side application code and client-side scripts.

The records include the following information about each dependency:

- Where it is used or referenced
- Why it is required
- Version used
- License
- Source information (link to repository, author's name, etc.)

- Open source or proprietary
- Support and maintenance status of the dependency

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

Ensure that providers actively maintain dependencies, and that they deal with security vulnerabilities appropriately. Gain assurance when dealing with open source dependencies, either through agreements with a commercial vendor, or other means, for example, by looking at repository activity, and the developers’ responses to security issues raised by the community.

Secure Build (SB2)

Build process is optimized and fully integrated into the workflow.

Activities

Stream A : Build Process

Benefit: *A fully automated build system allows easy integration of automated security checks at all stages of the build process, and ensures separate but consistent build environments.*

Implement the build process as an automated system, so that builds can be executed repeatedly and consistently. The build process is reliable and does not require developer intervention, further reducing the likelihood of human error. Automation makes it easier to include security checks during the build process. Implement static application security testing (SAST) to run as part of the build process. Refer to guidance in Verification > Security Testing > A3. The use of an automated system to setup the build pipeline increases reliance on the build tools for security, 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. Refer to Implementation > Secure Deployment > B. Sign generated artifacts using a certificate that identifies the organization or business unit that built it, such that its integrity and can be verified later.

Stream B : Software Dependencies

Benefit: *There is an audit trail of all 3rd party dependencies used in development and you know and track their security status at any given time.*

Evaluate dependencies to establish a whitelist of acceptable code dependencies approved for use within a project, team, or the wider organization.

Alternatively, introduce a central repository of approved dependencies that all software must be built from.

Review dependencies regularly to ensure that:

- they remain correctly licensed
- no known and significant vulnerabilities are present
- there is support and active maintenance for the dependency
- there is a good business reason to include the dependency

You may need tools to automate some or all of this process, such as analyzing where the dependency is used, or checking for updates via a package manager. Consider using an automated tool to scan for vulnerable dependencies.

Secure Build (SB3)

Build process helps prevent known defects from entering the production environment.

Activities

Stream A : Build Process

Benefit: *You can enforce a minimal clear security baseline in production. You can automatically deploy compliant applications.#A one sentence description of the activity*

The build process includes automated security checks which break the build if they fail. Static Application Security Testing (SAST), with an appropriate and custom ruleset, is triggered each time the build process executes. Refer to guidance in Verification > Security Testing > A. The organization sets an appropriate threshold for build failure based on the application’s risk appetite. For instance, “High” and “Critical” vulnerabilities, or those with a CVSS score above 7.0. The types of vulnerabilities that the organization consider unacceptable in a build and their typical scores/ratings are considered when setting this threshold. An application with more sensitive functions might have a lower threshold, for instance. Trigger warnings for vulnerabilities below the threshold, and log them to a centralized system to track them and take actions. Put in place a mechanism to bypass this behaviour when a vulnerability has been accepted or mitigated to stop it from breaking the build. Carefully control and approve it, and log all exceptions with a rationale. If any of the security tests like SAST are not carried out successfully, the build fails. If technical limitations prevent the organisation from breaking the build automatically, achieve the same effect via other means, such as a clear policy for the developer not to deploy or execute a build with defects meeting certain criteria. 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. Compare the binary output with that from other equivalent build systems to ensure it hasn’t been tampered with.

Stream B : Software Dependencies

Benefit: *The application’s security level is more indicative of its real security, by consistently assessing its 3rd party dependencies.*

Perform verification tests against dependencies in the same way you do against the target application. Refer to Verification > Security Testing. Depending on the build process maturity level, the discovery of significant issues might cause the build to fail. Log results centrally, triage and validate findings appropriately as described in Implementation > Defect Management. Vulnerable dependencies should be blacklisted and not permitted to be used during builds. Feed findings back to the vendor or open source project, following a set of ethical disclosure guidelines.

Secure Deployment (SD1)

Deployment processes are fully documented.

Activities

Stream A : Deployment Process

Benefit: *Only qualified personnel, different from developers can deploy to production environments.*

Deploy applications via automated processes, or manually by people other than the developers. Developers do not have access to production environments. Choose any tools used during deployment carefully and harden them appropriately. If 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. Handle access to the production credentials and secrets for the tools and engineer conducting the deployment with care - e.g. according to the principle of least privilege, and encrypted at rest with keys held in a trusted platform module (TPM) or hardware security module (HSM). People with access to production have to go through a minimum level of training or certification to ensure competency in this sensitive environment. Refer to Governance > Education & Guidance.

Stream B : Secret Management

Benefit: *Production secrets are adequately protected in a digital vault, inaccessible to developers.*

Version and protect configuration files just like source code. Developers do not have access to secrets or credentials for production environments. Someone responsible for the production environment adds production secrets to configuration files during the deployment process. Do not keep production secrets in configuration files for development or testing environments, as such environments may have a significantly lower security posture. Do not keep secrets in configuration files stored in code repositories. Before deployment, store sensitive credentials and secrets for production systems with encryption-at-rest and appropriate key management. Consider using a purpose-built tool/vault for this data. Handle key management carefully so only personnel with responsibility for production deployments are able to access this data (the principle of least privilege). Encrypt secrets at rest in configuration files during deployment. Manage keys so the application can access the secrets while running, but an attacker who obtains the configuration files alone cannot decipher them.

Secure Deployment (SD2)

Deployment processes include security verification milestones.

Activities

Stream A : Deployment Process

Benefit: *The deployment process is fully or partially automated and can be halted based on the results of integrated security verification tests.*

Fully or partially automate deployment to reduce the need for manual changes on production, and to reduce the chances of human error. Deployments include appropriate automated security checks such as DAST and malware scanning. Notify relevant people of any defects automatically. Stop or reverse the deployment automatically, or as part of a manual approval workflow, if the defect exceeds a certain threshold of severity or risk. Log the results from these tests centrally and take any necessary actions. Account for and audit all deployments. 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.

Stream B : Secret Management

Benefit: *Secrets are dynamically extracted from the digital vault for use in deployment.*

Have an automated process to add credentials and secrets appropriate for the target environment to configuration files during the deployment process. This way, developers and deployers do not see or handle those sensitive values. Make the system used to store and process the secrets and credentials robust from a security perspective. Encrypt secrets at rest and during transport. 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 acceptence test or production environment.

Secure Deployment (SD3)

Deployment process is fully automated and incorporates automated verification of all critical milestones.

Activities

Stream A : Deployment Process

Benefit: *The deployment process automatically validates the integrity of its artifacts.*

The deployment process automatically verifies the integrity of the binaries by checking their signatures against trusted certificates. Sign binaries at build time. This may include binaries developed and built in-house, as well as third-party libraries. Do not deploy binary signatures that 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 organisation’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, do it manually.

Stream B : Secret Management

Benefit: *Secrets are dynamically generated during deployment and a process routinely checks for and mitigates unprotected secrets.*

Where secrets are not predefined or dependant on another system, generate them during the deployment process. Follow appropriate best practices such as using a cryptographically secure pseudorandom number generator if you generate this value randomly. 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, for instance. 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 either remove them or render them non-sensitive. 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.

Defect Management (DM1)

All defects are tracked within each project.

Activities

Stream A : Defect Tracking (Flaws/Bugs/Process)

Benefit: *All software security defects are recorded centrally.*

Track and record all security defects in a central location. This location can be team, project, or organisation-wide. Give defects meaningful categories, and prioritise them based on the risk they pose.

Sources of defects and violations include, but are not limited to, those discovered via:

- Threat assessments
- Developers during self or peer code review
- Static analysis
- Dynamic analysis
- Vulnerability scans
- Penetration testing
- Malware scans
- Public/private vulnerability disclosures (e.g. for 3rd party libraries)
- Bug bounties

Qualify all defects and license violations so the records only contain valid and significant issues. Consider manageability. Void recording duplicate defects, for example, searching for similar issues. Merge duplicates and group similar issues, particularly if you will handle them in the same way. The organisation uses these records to make decisions and resolve defects and violations. Update the records when issues are resolved, tracking vulnerabilities over time. Employ security testing to ensure fixes are effective. Refer to Verification > Security Testing.

Stream B : Metrics and Feedback/Learning

Benefit: *Basic information about defects is shared and used for remediation and training decisions.*

Basic information about defects is calculated, shared, and used to make decisions about remediation. Basic information might include:

- The total number of defects. Tracking this over time shows the effectiveness of resolution efforts.
- The software components the defect resides in, which is indicative of where attention is most required, and where security flaws are most likely to appear in the future.
- 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 organisation understand the software’s risk exposure.
- Outcomes are fed back to the teams involved. This data is used to make decisions about remediation priority and training requirements.

Defects are considered within the wider metrics throughout the oragnisation. See

Defect Management (DM2)

Defect tracking used to influence the deployment process.

Activities

Stream A : Defect Tracking (Flaws/Bugs/Process)

Benefit: *Minimal defect quality gates are enforced throughout the SDLC.*

Define a threshold for defects that require resolution or mitigation. You do not deploy software into production when this threshold is exceeded, until the relevant defects are fixed, or fall below the threshold.

This quality gate may also exist at whatever point the issue is detected. For instance, if you detect an issue by static analysis prior to build, the build system might prevent the software from building. However, deployment to production is always the final and mandatory quality gate.

To set a threshold that is right your organisation, take into consideration:

- the threat model
- the nature of applicable threats (i.e. skill level, motive, level of access, etc)
- how difficult it would be to find and exploit the issue
- potential impact to confidentiality, integrity, availability
- potential impact to the business

Use a well established risk rating methodology consistently across your defect management solution.

The people responsible for application security and key stakeholders reach an agreement regarding an appropriate threshold.

You notify relevant people if a defect exceeds the defined threshold, and take action to resolve or mitigate the issue before deploying the software into a production environment. A satisfactory mitigation is one that reduces the risk below the threshold.

Define processes for dealing with false positive defects, or ones with existing compensating controls. In other words, some defects identified as exceeding the threshold should not prevent the software from being deployed. This might be because they have been misreported, or because you have not considered existing mitigations. Record such defects for accountability purposes.

Consider defects below the threshold deployed into production environments for resolution or mitigation at a later time.

Stream B : Metrics and Feedback/Learning

Benefit: *Advanced defect metrics are calculated and shared and used to taylor the assurance program.*

Calculate and share more advanced metrics. These can include:

- Formal risk ratings that consider likelihood and impact.
- Number of open vulnerabilities above a defined threshold in terms of severity or risk.

- Risk per software component / product / project / business unit.
- Amount of accepted risk.
- Time to detect vulnerabilities.
- Time to resolve vulnerabilities.
- Window of exposure where vulnerabilities are detected on live systems.
- Coverage of software components by verification tests.
- Number of regressions / reopened vulnerabilities.
- The risk metric takes into account the criticality of the asset and the resulting business impact. Defects are mapped to threats to better understand their risk to the organization. See Design > Threat Assessment > B.

Tools such as spreadsheets or dedicated vulnerability tracking software are used to calculate metrics automatically. This makes collecting and acting on metrics a managable exercise.

Make this data accessible to management, information security people, developers, and engineers to inform their decision-making. For example, provide a central dashboard. Metrics should guide remediation efforts and resource allocation.

Security teams are able to report an accurate picture of the organisation’s defect and risk metrics to executive management.

A reliable baseline is established over time for the metrics being collected. Once this baseline is in place, reasonable goals can be set to measure the effectiveness of the overall programme.

Defect Management (DM3)

Defect tracking across multiple components is used to help reduce the number of new defects.

Activities

Stream A : Defect Tracking (Flaws/Bugs/Process)

Benefit: *Quality gates are enforced by a security officer following formal processes.*

An individual or team outside of those responsible for developing and deploying the software (e.g. an information security officer) is responsible for managing known defects and enforcing the defined threshold. Defects that exceed the threshold block or prevent deployment into production, until they are resolved, or fall below the threshold. You can do this automatically in the build and deployment processes, if they are integrated with the defect management system. Alternatively, the deployment process could include a step that requires approval from an information security officer. Keep a list of known defects, with accurate risk rating and categorisations over time (e.g. in-line with new research and changing opinions within the industry, and improving capabilities of some adversaries). This includes managing lists of defects that have been accepted, mitigated, or marked as a false positives. Review the status of accepted and mitigated risks periodically (e.g. to identify a defect that now poses a greater risk because a mitigating control has been changed or removed).

Stream B : Metrics and Feedback/Learning

Benefit: *Defect metrics are enriched with real time information and correlated to detect trends and influence the overall security strategy.*

Add information to defects, such as:

- Category
- CVE / CWE
- Software component
- Business unit
- Exploitability
- Impact
- Risk
- CVSS

For each defect, up-to-date or real-time data on the availability of exploits and hacker activity in the wild is used to contribute to risk scores.

Different metrics and fields are combined to look for trends. Trends across various timespans are identified through graphs and dashboarding. Trends are analysed and the results are used to influence the design and implementation of software and the overall security stategy.

Metrics are used to empower the whole organisation. People and teams all receive the correct information that is relevant to their role(s). Tasks are assigned appropriately and sometimes automatically.

Incident Management (IM1)

Best-effort incident detection and handling

Activities

Stream A : Incident Detection

Benefit: *Ability to detect the most obvious security incidents within a reasonable timeframe*

Available log data (e.g., access logs, application logs, infrastructure logs) are analyzed 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 simple script, looking for suspicious events, is a step forward! If logs from different sources are sent to a dedicated log aggregation system, it might be a good idea to analyze the logs there and employ basic log correlation principles. Even if you don’‘t have a 24/7 incident detection process, unavailability of the responsible person (e.g., due to vacation or illness) shouldn’‘t impact the detection speed and quality significantly. You have a defined and generally known contact point for formal creation of security incidents.

Stream B : Incident Response

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. They keep up with current state of incident handling best practices and forensic tooling. You don’‘t mandate dedicated incident response personnel on this maturity level, but you have defined the participants of the process (usually different roles). There is a known single point of contact for the process and a conscious decision regarding reachability of the participants. When security incidents happen, you document the steps taken. Protect this information from unauthorized access, as necessary.

Incident Management (IM2)

Formal incident management process in place

Activities

Stream A : Incident Detection

Benefit: *Ability to timely detect expected security incidents*

The incident detection process has a dedicated owner and clear documentation accessible to all process stakeholders, and is periodically checked to make sure it is up to date. You 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 collect the logs to a central place, if suitable. Explicit attention is periodically paid to integrity of the analyzed data. If you add a new application, you ensure that the process covers it within reasonable period of time. You detect possible security incidents according to an available checklist. The checklist covers expected attack vectors, and known or expected kill chains. You evaluate it and update it regularly. If you evaluate an event as a security incident with high level of confidence, the responsible staff is notified immediately, even outside business hours. You perform further analysis and start the escalation process.

Stream B : Incident Response

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

Formally establish and document the security incident response process. The 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 mandatory timeframe to do so, if needed), including senior management, Public Relations, Legal, privacy, Human Resources, external (law enforcement) authorities, and customers.

Knowledgeable and properly trained staff is available in and outside of business hours with defined time to act. Keep both hardware and software tools up to date and ready for use anytime. Define a war room.

The process includes a policy for carrying out root cause analysis and its expected outcomes.

Incident Management (IM3)

Mature incident management

Activities

Stream A : Incident Detection

Benefit: *Ability to timely detect unexpected security incidents*

The process documentation includes measures for continuous process improvement. You check the continuity of process improvement (e.g., via tracking of changes).

The checklist for suspicious event detection is correlated at least from: - Sources and knowledge bases external to the company (e.g., new vulnerability announcements affecting the used technologies) - Past security incidents - Threat model outcomes

You use correlation of logs for incident detection for all reasonable incident scenarios. If the log data for incident detection is not available, you document it as a defect, triage and handle it according to the resulting priority / SLA.

The quality of the incident detection does not depend on the time or day of the event. If you do not act upon the security event within a defined time, it triggers further notifications according to a defined escalation path. The efficiency is of the incident is also checked by exercises with defined improvement action points.

Stream B : Incident Response

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

Establish a dedicated incident response team, continuously available and also in charge of the 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. Where sensible, automate procedures. 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.

Environment Management (EM1)

Best-effort patching and hardening

Activities

Stream A : Configuration Hardening

Benefit: *Most evident security configuration is carried out*

You acknowledge the importance of configuration hardening of third-party components your applications are using. You don't have an official process yet. However, at least define the relevant scope for this activity take the first steps. You know and acknowledge the responsibility for hardening of the particular components. Work with publicly available information sources (e.g., open source projects, vendor documentation, blog articles), increasing your know-how from those and implementing at least "low hanging fruit."

Stream B : Patching and Updating

Benefit: *Mitigated prominent issues in third-party code*

Identify applications and third-party application components which need to be updated or patched, including the underlying operating system, application server or third-party code library. Carry out patching activities according to best-effort. However, define the update process at least on a high level (e.g., testing the patches don't break anything). Use opportunities like maintenance windows for best-effort patching. Share knowledge of the patching process for components. Teams cooperate if necessary. You can carry out patching anytime in case of need (e.g., exploit for a third-party component publicly available). You can find out the versions of all components in use to evaluate whether you are affected by a particular public security vulnerability.

Environment Management (EM2)

Formal process with baselines in place

Activities

Stream A : Configuration Hardening

Benefit: *Better efficiency due to established baselines*

Within the scope relevant for this activity, define hardening baselines for particular components. The affected teams know and acknowledge them. This typically leads to a standard way of deploying the affected components over the organization. The baselines have an owner responsible for keeping them up to date (e.g., if new best practices or features are available with new versions) and adapting them according to trustworthy sources. Both new and existing systems are part of the hardening process. In larger environments, derive configuration of all instances from your own master, where there is already common ground work. Consider using automated tools for hardening configuration.

Stream B : Patching and Updating

Benefit: *Reliable handling of third-party code issues*

Define and document the update process across the full stack. You don’t rely on available patches provided by vendors only; you use external sources systematically to gather intelligence about zero day vulnerabilities, and take appropriate risk mitigation steps. There is a guidance for prioritization of particular updates, including concerns important to your organization like the criticality of the application, or severity of security issues. Schedule updates (without necessary relevancy to known issues), e.g. using a patch/upgrade calendar of vendors. If there is a known critical issue while the patch is not available yet, triage and handle this issue (e.g., by finding workarounds, monitoring measures, or even switching off the affected applications).

Environment Management (EM3)

Conformity with continuously improving process enforced

Activities

Stream A : Configuration Hardening

Benefit: *Profound knowledge about state of hardening measures across the organization*

Track and evaluate conformity with the hardening baselines. Triage and handle nonconformities as security findings according to rules and SLAs stemming from the defect management practice. Automated measures ensuring self-healing of critical configuration mistakes and alerting relevant stakeholders are in place if sensible. Verify the validity of the current hardening baselines in the component update process. Incorporate relevant changes in the baselines and in the auditing measures. Periodically audit the continuous improvement process for the baselines and act upon the resulting findings.

Stream B : Patching and Updating

Benefit: *Full visibility into the current patch state over the organization*

You have very good insight (e.g., through a dashboard) of the patching strategy across the organization and full stack. You triage and handle missing updates according to rules and SLAs stemming from the defect management practice. It is guaranteed that patching can take place anytime so that SLAs can be adhered to. If there are applications with worse patch level, the situation is analyzed and corrective actions are performed if reasonable.

Operational Management (OM1)

Foundational Practices

Activities

Stream A : Data Protection

Benefit: *Sensitive data are protected from accidental disclosure*

The organization understands the types and sensitivity of data stored and processed by applications, and maintains 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. The organization protects and handles all data associated with a given application according to protection requirements applying to the most sensitive data stored and processed. The organization implements 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, the organization can focus data protection policies and activities on production.

Stream B : System Decomissioning / Legacy Management

Benefit: *- Reduced operating costs for unused applications, when discovered - Limited reductions in support costs for legacy product versions*

Identification of unused applications occurs on an *ad hoc* basis, either by chance observation, or by occasionally performing a review. When unused applications are identified, findings are processed for further action. If a formal process for decommissioning unused applications has been established, that process is used. The organization manages customer/user migration from older versions of its products individually for each product and customer/user group. When a product version is no longer in use by any customer/user group, support can be discontinued. However, a large number of product versions may remain in active use across the customer/user base, requiring significant developer effort to back-port product fixes.

Operational Management (OM2)

Managed, Responsive Processes

Activities

Stream A : Data Protection

Benefit: - *Increased understanding of the organization’s data landscape - Improved confidentiality, integrity, and availability of data backups*

At this maturity level, Data Protection activities focus on actively managing the organization’s stewardship of data. Technical and administrative controls established as part of this activity serve to protect the confidentiality of sensitive data, and the integrity and availability of all data in the organization’s care, from its initial creation/receipt through the destruction of backups at the end of their retention period.

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

Based on the organization’s Data Protection Policy, the organization establishes processes and procedures for protecting and preserving data throughout their lifetime, whether at rest, being processed, or in transit. Particular attention is given 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. Organization 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.

Stream B : System Decomissioning / Legacy Management

Benefit: - *Reduced attack surface, through elimination of unused configuration in operating environments - Elimination of risks associated with end-of-life software*

As part of decommissioning a system, application, or service, the organization follows an established process for removing all relevant accounts, firewall rules, data, etc. from the operational environment. By removing these unused elements from configuration files, the organization improves the maintainability of its infrastructure-as-code resources. The organization follows 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. The organization engages with customers and user groups for its products at or approaching end of life, to migrate them to supported versions in a timely manner.

Operational Management (OM3)

Active Monitoring and Response

Activities

Stream A : Data Protection

Benefit: *Cost savings realized through automation of monitoring and alerts*

Activities at this maturity level are focused on automating data protection, reducing the organization’‘s 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. The organization implements technical controls to enforce compliance with the Data Protection Policy, and active monitoring is in place to detect attempted or actual violations. The organization may use a variety of available tools for data loss prevention, access control and tracking, or anomalous behavior detection. The organization regularly audits compliance with established administrative controls, and closely monitors performance and operation of automated mechanisms, including backups and record deletions. Monitoring tools quickly detect and report failures in automation, permitting the organization to take timely corrective action. The organization reviews and updates the data catalog regularly, to maintain its accurate reflection of the data landscape. Regular reviews and updates of processes and procedures maintain their alignment with the organization’‘s policies and priorities.

Stream B : System Decomissioning / Legacy Management

Benefit: *- Reduced risks, through eliminating unsupported applications and libraries from operating environments - Minimized product support burden*

The organization regularly evaluates the lifecycle state and support status of every software asset and underlying infrastructure component, and estimates their end-of-life. The organization follows a well-defined process for actively mitigating security risks arising as assets/components approach their end-of-life. The organization regularly reviews and updates its process, to reflect lessons learned. The organization has established a product support plan, providing clear timelines for ending support on older product versions. Product versions in active use are limited to only a small number (e.g., N.x.x and N-1.x.x only). The organization establishes and publicizes timelines for discontinuing support on prior versions, and proactively engages with customers and user groups to prevent disruption of service or support.