SSDLC Maturity Model

1-Security Policies and Standards

The security policies and standards guide every aspect of secure software development. These standards and policies should be accessible throughout the organization and should encompass application security and secure coding practices.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	Security standards either do not currently exist or are still in the process of being developed		
2	The engineering teams have access to published security standards for their use.	InfoSec publishes security standards in a central location, making them accessible to all teams.	• The InfoSec team verifies that the standards are available. • Engineering team leads showcase their understanding of the standards and verify their accessibility
3	The engineering teams have formally acknowledged the security standards.	A record indicating that all engineering team members have formally agreed to the published security standard	• Submit a confirmation via a tool (e.g. Workday or email or via slack in writing)

4	Adherence to the security policy and standards is consistently measured and reported.	Evidence of regular assessments measuring the teams' compliance with security policies and standards	• Evidence document is available for each policy & compliance standard (e.g. Threat Model is posted, secure code scanning report available etc)
5	Review and optimize the security policy and standards, as well as the compliance verification and reporting process, regularly (at least annually)	Evidence of the proposal and implementation of newer or optimized: 1. Policies and standards documentation 2. Compliance verification methods	• Schedule a meeting to propose and implement newer or optimized policies and standards, along with methods for compliance verification. • Record the meeting minutes.

2-Security Roles and Culture

Security roles and cultural practices not only clarify the responsibilities of security teams but also enable the assignment of security champion roles to existing software engineers within the engineering teams. This approach integrates security expertise into the engineering teams

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	Security roles and responsibilities are either not defined or are currently being established.		

2	Security roles and responsibilities have been established.	• Security champions have been designated within each engineering team. • Security engineers have been assigned to engineering teams, and they are familiar with the structure of the engineering teams. Similarly, the engineering teams understand the InfoSec team structure and are aware of the role of application security engineers.	• The names of security champions have been documented, such as on a Confluence page. • Security engineers have been introduced to engineering team.
3	Security roles and responsibilities have been published, and they actively contribute to enhancing the organization's security posture	• Formal documentation of security roles and responsibilities has been created. • A formal security community of practice has been established. • The engineering team actively works to enhance the security posture and collaborates with the security engineering team.	• A published RACI chart of security roles and responsibilities. • “Community of practice” communication channel e.g. slack channel and regular meetings with security champions
4	Engineering teams contribute to the maturity of roles and culture by participating in various security meetings and engaging in security discussions on Slack channels.	Evidence of participation in security meetings or security discussion in the slack channel. Security Champions have been formally allocated time to perform their security roles and contribute in the security culture.	• Schedule monthly or quarterly meetings for security champions • Initiate topics of conversation in the slack channel to enhance security awareness

5	The security knowledge base and processes for all security aspects are regularly reviewed by executives and updated as needed.	Evidence that security champions actively engage in the ongoing enhancement of security maturity and practices. Evidence of events that enhance security culture and improve the overall security posture of processes, such as bug bounty programs or conferences	• Schedule quarterly executive-level meetings for security champions and application security engineers to explore new and optimized approaches to strengthening and improving security culture

3-Security Training

The security training practice ensures that an appropriate level of security training is available to enhance security awareness within the engineering teams, AppSec teams and Security Champions.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	Security training does not currently exist or is in the process of being developed.		
2	Basic application security training is available and has been completed by the members of the engineering team	• Basic AppSec training is available. • Engineering teams have completed the training • New starters complete the training as part of the on boarding process.	• Training must be made available by the InfoSec team. • Engineering team members confirm that they have completed the training (e.g. via Workday) • Evidence that training is part of onboarding process for new starters.

3	Security champions have completed more in-depth and advanced training necessary for their roles.	• Security champions training is available. • 100% Security champions have completed the training • New security champions complete training as part of onboarding process.	• Security champions confirm that training have been completed (e.g. via Workday)
4	Security champions and senior software engineers have completed the advanced security training	Evidence that all Security Champions and Senior Software Engineers have completed the advanced Security training.	• Communicate the advanced training details and the target audience formally via email to ensure completion of the training.
5	The engineering and security teams contribute to the topics and content of the security training courses, ensuring that the courses are current and address the latest requirements	Evidence of feedback and improvement suggestions collected from the engineering teams by the security training team.	• Schedule meetings to explore new training topics or discuss them in the "Security Centric meetings" and seek approval for the new topics from the "Director of InfoSec "

4-Application Asset Inventory

Application asset inventory practices focus on creating a catalog of all application assets.
The catalog for source code repositories is GitHub; for third-party binary libraries (packages) and open-source libraries, it is JFrog Artifactory or Nexus Repository etc; for public or private APIs and endpoints, it is Noname

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	The application asset inventory does not currently exist or is in the process of being created or defined.		

2	The application asset inventory exists and includes identification of the owner for each asset, covering internal assets while excluding external or third-party dependencies.	• The application asset inventory is accessible from a central location. • The engineering team can confirm the accuracy of the inventory at any time.	• A formal inventory of applications should include essential information, such as the asset name, asset owner, asset URL (e.g., GitHub repository URL), and the engineering team associated with the asset, among other details.
3	Application asset inventory must include all internal as well as external assets (e.g. 3rd party dependencies)	• Third party dependencies are included in the inventory. • Any changes to the asset should be updated in the inventory. • A documented procedure for transferring ownership of the asset from one owner to another. • Third party assets have been risk profiled.	• Clearly identify third party assets. • Third party assets include risk profile. • A procedure exist to reflect any change in the asset inventory
4	Only approved 3rd party assets are used.	Evidence from the SCA scanning that no 3rd party asset (library) is used that has any Critical or High violation.	• Conduct regular SCA scans to ensure that the SCA scan report contains no critical or high-level violations.

5	The asset inventory format/tool is assessed according to evolving requirements. The asset inventory is regularly reviewed and updated, including scans for any vulnerable assets.	Evidence of a documented standard process for maintaining the inventory and evaluating, at least annually, whether the tool/format for asset inventory meets evolving requirements, with provisions for making changes if necessary.	• Perform the Asset inventory tools' evaluation as per the "documented standard process"

5-Application Architecture Assessment

The application architecture assessment practice focuses on identifying security vulnerabilities in new application architectures as well as in modifications to existing ones.

According to the 'shift-left' principle, this practice should ideally commence during the requirements elicitation phase or the architecture design phase of the SDLC.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	Threat identification for application architectures is either not conducted or is carried out on an ad hoc, best-effort basis.		
2	The security architecture review for each application architecture is conducted without a standardized format.	Application architectures are reviewed in collaboration with security engineers or security champions and documented without a standardized format.	• Identified threats and their mitigations can be recorded in the comments section of the Jira tickets.

3	A formal Threat Modeling (TM) process is conducted for each application architecture using a standardized TM format/template.	• A formal TM format has been published by the InfoSec team. • The InfoSec team has published TM guidelines that include a list of potential threats. • AppSec engineers conduct Threat Modeling using the formal TM template in collaboration with the engineering team.	• A TM document is published for each application architecture.
4	• Application architectures and their Threat models are kept in a central space for reference. • Threat Modeling process is iterative and each Threat Model is reviewed again at least annually. • Security Architecture review and Threat Modeling is conducted by the Architects with minimum involvement from the security engineers.	• All Security architecture assessment and Threat Models are posted at a centrally accessible space. • Risk assessment of identified threats is performed (i.e. the severity of threat, likelihood of occurrence and impact of the threat are determined)	• Post the list of common threats that are used as a baseline for all security architecture assessment and threat modeling.
5	Security Architecture review and threat modeling practice is continually reviewed and improved. Threat modeling template is also reviewed and improved as per the changing needs.	• Update the Threat Modeling reference list of potential threats. • Updated and newer security architecture review template and practice should be posted and communicated.	• Review and then post the updated list of common threats that are used as a baseline for all security architecture assessment and threat modeling. This list of common threats should evolve overtime.

6-Source Code and Build (With SAST and SCA)

The source code and build practices ensure adherence to secure coding principles by identifying and addressing source code vulnerabilities, including those in third-party dependencies, using SAST and SCA tools. A threshold is implemented as a security gate.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	A secure coding standard exists and is applied on a best-effort basis. Manual code reviews (potentially through pull requests or another code review tool) are conducted to identify and rectify any violations of this coding standard.		
2	The engineering teams have adapted the available SAST and SCA tools, and vulnerabilities are being regularly mitigated.	All source code repositories are scanned using SAST and SCA tools, and there are no critical or high vulnerabilities remaining for mitigation.	• SAST and SCA tools' vulnerability reports showing the criteria is met. • Tickets to mitigate the vulnerabilities are completed.
3	SAST and SCA scanning are integrated into the CI/CD pipeline to automate the scanning process.	• All source code repositories are build via CICD pipeline. • Quality gates are implemented in the pipeline to prevent the inclusion of any critical or high vulnerability.	• Quality gate are implemented in the CI/CD pipeline to fail the build pipeline in case of any Critical or High vulnerability.

4	For all code repositories, SAST and SCA scanning is integrated with **quality gate** in the CI/CD pipeline for main branch and merge block for Pull request in case of critical vulnerabilities with clear exception policy defined.	Evidence that there is a quality gate implemented in the CI/CD pipeline that fails the “build pipeline” in case of any Critical/High vulnerabilities found by SAST/SCA scan. 100% of the source-code repositories should have implemented this quality gate in their pipeline.	• Publish an example of CICD pipeline (e.g. Jenkins or GitHbuActions workflow) that implements the **quality gate that demonstrate** “how to incorporate that workflow in the pipeline workflow”.
5	Code scanning tools are configured for applications to optimize and enhance SAST & SCA scans. Different SAST/SCA scanning tools are evaluated/adapted annually to enhance the scanning capabilities.	• There is evidence that SAST and SCA scanning have been tuned and incrementally improved based on feedback from both engineering and security teams. • Evidence of reduction in false positives based on tuning and feedback provided.	• The InfoSec team should publish a guideline document detailing how the tools can be configured differently to achieve improved scanning results. • Engineering teams are supposed to continue code scanning as per the new configuration and observe the improvement.

7-Secure Deployment

Secure deployment practice ensures that security of application is not compromised during deployment. Deployment should be automated via pipeline while the pipelines should not have an secrets.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**

1	Deployment process is not documented. There are minimal or no measures in place to secure secrets in production, and the development and production environments are not properly separated during deployment. For example, development roles are used to deploy in the production environment, and vice versa.		
2	Deployment process is documented. Secrets' scanning is implemented in all repositories.	• Well defined/documented deployment process. • Vault is used to store production secrets. • Regularly scan the code repositories to identify any hard-coded credentials or other secrets.	• Never store secrets (such as credentials or keys) in pipeline parameters or configuration files on production servers. • Use a digital vault to store secrets.
3	The deployment process is automated using a CI/CD secret manager for deployment secrets and a dynamic injection procedure for runtime secrets in every environment (development, production, staging). As a result, no secrets are stored in Git under any circumstances.	• Evidence (e.g. demo) automated deployment process with dynamically injected secrets during deployment process from vault or hardened storage. • Repository scan report showing no secrets hardcoded in source code.	• Automate the deployment process and inject secrets dynamically during deployment process from vault or hardened storage.

4	For all code repositories, the “deployment pipeline” should have : 1. Automated DAST scanning in the pipeline at the ephemeral environment that fail the pipeline in case of any Critical/High violations 2. Automated SAST scanning that fail the pipeline in case of any secrets /keys/ passwords found in plain text in the source code.	• If we try to deploy a code with critical vulnerabilities it will block the build or deploy, until unless we fix it or make an exception. • PR marge will be blocked with critical vulnerabilities it will block the build or deploy, until unless we fix it or make an exception.	• Enable pull request checks and blocks for SAST, SCA, and secret scanning, and incorporate the SAST scanning check in the CI/CD deployment workflow.
5	Teams regularly review and improve the security of the deployment process, focusing on key management, vault management, and secret detection where possible.	There is evidence of improvements in the deployment process, such as enhancements in key management, vault management, and secret detection. Additionally, a comparative analysis of different tools and techniques has been conducted to identify enhancements for secure deployment beyond the existing process.	• AppSec engineers should conduct a thorough review of the deployment procedures and CI/CD pipelines at least annually to identify potential security improvements.

8-Dynamic Application Security Testing (DAST)

The Dynamic Application Security Testing (DAST) practice ensures that all applications in production are continuously scanned at run time, generating alerts for the relevant teams if any vulnerabilities are detected.

Level	Description	Criteria / Evidence	Recommendation How to meet criteria

1	Production application scanning (DAST) is either not in place or in the process of being implemented, leading to a reliance on periodic penetration testing, which slows down the release cycle.		
2	DAST Scanning has been integrated in application runtime while the scanning is performed on adhoc manner. The reported vulnerabilities are also remediated on adhoc bases.	There is evidence of alerting and remediation efforts, such as vulnerability reports.	• Relevant assets from the asset inventory are integrated into the DAST tools. • Scanning of apps in production is scheduled on regular bases. • Reported vulnerabilities are remediated on best effort bases.
3	All production applications and APIs are scanned using the DAST tool.	100% of the applications are regularly scanned and the vulnerability report shows that less than 10% of the vulnerabilities (DAST) are open at any time.	• Automate the DAST scanning and alerting • Security engineers ensure engagement with the relevant engineering teams to promptly address and fix vulnerabilities.
4	DAST in implemented in CI/CD prior to production deployment in an ephemeral environment in fully automated fashion.	70% of the repositories have included DAST in their CI/CD for major builds	• Conduct automated DAST in every major build prior to deployment, significantly reducing the need for manual penetration testing, which is a resource-intensive effort, to perhaps twice a year.

5	DAST scanning is continually optimized, reducing false positives and improved scan speed.	Evidence of the tuning of scanning tools or migrating to better tools with comparative analysis.	• App Sec Engineers assist the engineering teams to continually improve the procedure to block the release of insecure application to the production environment.

9-Manual Security Testing (Pentest)

Security testing is conducted when the application is deployed in the QA/staging environment (open beta). During this phase, penetration testing is performed, and QA verifies that all mitigation controls identified during threat modeling are effectively guarding against the corresponding threats.

Level	**Description**	**Criteria / Evidence**	**Recommendation How to meet criteria**
1	Minimal or no security testing is performed in the staging (Dev/QA/non-production) environment.		
2	Manual security testing by the QA team and penetration testing by the Red Team (potentially using tools like Burp Suite) are conducted in the staging (Dev/QA/non-production) environment on an ad-hoc basis.	• A new API or service collects PII or PCI data • Significant Infrastructure changes • New authentication protocols are used for the applications • A pen-testing report is generated after pen-testing. • A manual black box testing report is generated, or the results of the manual testing are documented.	• Penetration testing is conducted before the release of an entire application or a specific feature of an application. • Manual black box testing is performed by QA personnel to validate that all security requirements and mitigation controls identified during threat modeling are in place.

3	Pen test is mandatory whenever • A new API or service collects PII or PCI data • Significant Infrastructure changes • New authentication protocols are used for the applications • The Application Security Engineer determines other factors that may require pen testing	A formal report is generated to confirm that all high and critical threats/vulnerabilities have been mitigated.	• QA personnel are informed during and after TM exercise about the identified threats and corresponding mitigation controls and they test those mitigation controls.
4	All security functionality (aka mitigation control) e.g. TLS, MFA, input-validation, injection-attack-validation etc have been tested by the QA or pen-testing team.	There is evidence of a pen-testing report that confirms which mitigation controls have been tested and the results of those tests.	• The pen-testing team should prepare a set of common tests against the list of important mitigation controls they plan to assess, and subsequently share the pen-testing report corresponding to those common tests.
5	The pen-testing team and QA, potentially in collaboration with the security team, review the overall QA and pen-testing procedures to identify improvements, including ways to optimize or automate testing.	There is evidence of improvement suggestions made in the QA and pen-testing processes.	• Continuously explore newer penetration tests that the pen-testing team can conduct to identify threats and vulnerabilities.