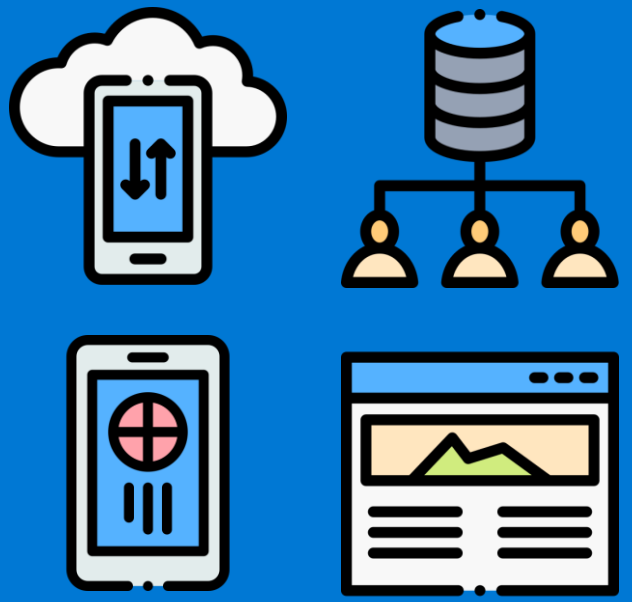
Application Security Maturity Assessment Checklist

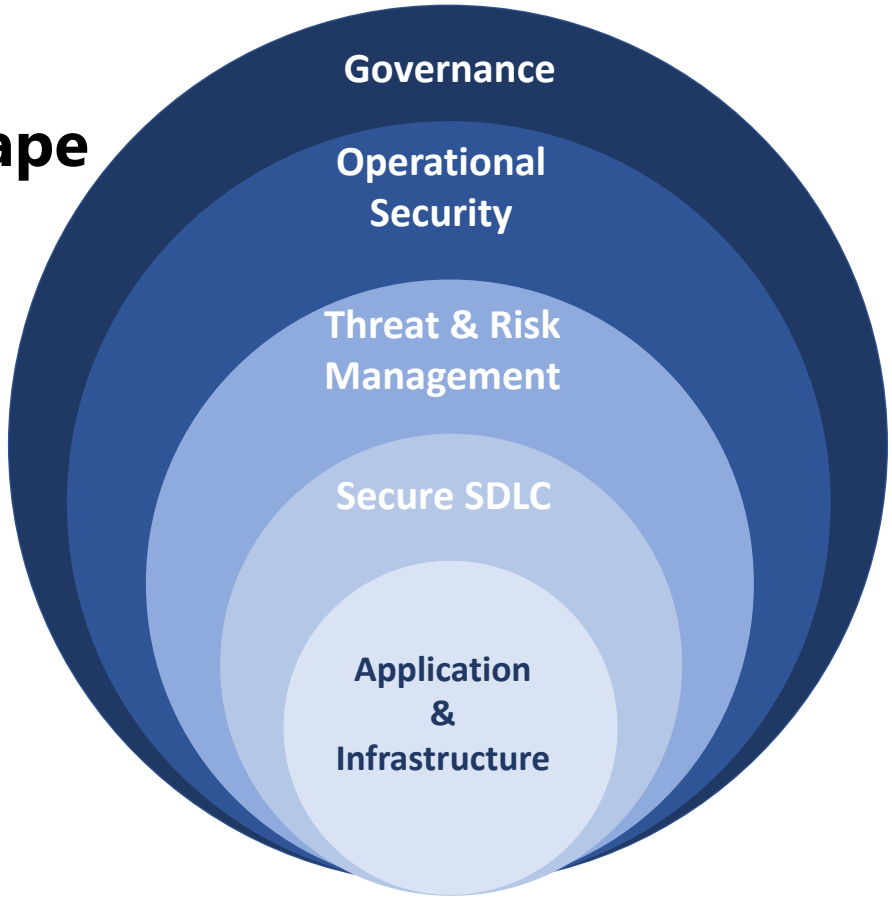


In this presentation, I am going to provide you with the checklist that includes the list of all areas to be considered for the application security maturity assessment. It contains technical, operational, and logical controls.

Application Security Maturity Landscape

These are the five critical components of application security development process. All the components must be coordinated with one another in order to mature the security landscape of your application and application infrastructure.

From the next slide onwards, we are going to deep dive into each pillar and relevant security controls that should or must be implemented in your software development program.



1. Governance

Security governance is a framework of policies, standards, and processes that form a structure for making decisions and defining expectations with respect to software security. The most important combination of policies, standards, and processes is the secure SDLC, which defines how a firm will build security in for both its various SDLCs and for software acquisition.

- ❑ Ensure that there is a software security assurance program
- ❑ Business Stakeholders should be aware of an organization's and application's risk areas
- ❑ Risk categorization must be undertaken
- ❑ Risk categorization must align with critical application activities and the environment used to build an application
- ❑ Organization should have proposed plan to mitigate risk by categories and risk rating
- ❑ An internal team and external vendor must check compliance
- ❑ Policies and procedures must be in place to control application development program end to end
- ❑ An internal audit must be taken place periodically for all procedures, policies and operations
- ❑ A centralized system should be implemented to store audit artifacts for use by different teams
- ❑ Developers must be trained for security awareness and secure coding training
- ❑ Role-specific security training must be provided to each team
- ❑ Security knowledge of developers should be tested periodically to ensure they fully understand the security part of a role in their day-to-day operations
- ❑ Repository for secure development best practices must be stored on centralized location, distributed to relevant teams and must be updated to latest

2. Operational Security

Operational security (OPSEC), also known as procedural security, is a risk management process that encourages managers to view operations from the perspective of an adversary in order to protect sensitive information from falling into the wrong hands.

- ❑ Security incident management and response champion must be created
- ❑ RACI matrix must be created to define incident management and response roles and teams clearly
- ❑ Incident escalation matrix must be created for better communication
- ❑ A security response team must be created which can handle all types of incidents
- ❑ A team must be made aware of a security incident point of contact based on the escalation matrix
- ❑ Stakeholders must be aware of incident handling and response procedures, processes and possible outcomes
- ❑ For every incident, your organization face, root cause analysis must be identified for further recommendations in future
- ❑ Annual incident report matrix must be created to track the flow of the incidents. It is helpful for stakeholders to identify an increase or decrease in incidents
- ❑ Operational security requirements must be defined and documented based on each project
- ❑ Operation team must monitor security incidents in all third-party applications, libraries and all software/hardware components which are being used in application development
- ❑ Security detect, react, and response solutions must be implemented to protect application and infrastructure assets such as WAF, IDS, IPS, EDR, HIDS, Proxy, DDoS solutions, IAM, etc.

2. Operational Security Cont.

Operational security (OPSEC), also known as procedural security, is a risk management process that encourages managers to view operations from the perspective of an adversary in order to protect sensitive information from falling into the wrong hands.

- ❑ An operational team must check the health of the application and supportive components periodically either using the automated tools or manually
- ❑ SOC team must be implemented to identify all threats and risks to the application from internal and external
- ❑ Ensure security notes are transparently delivered upon each release
- ❑ A robust change management process for application development must be implemented
- ❑ A robust patch management process must be implemented for the application development process
- ❑ All software, hardware, logical and physical assets must be maintained, logged and documented to up-to-date

3. Threat and Risk Management

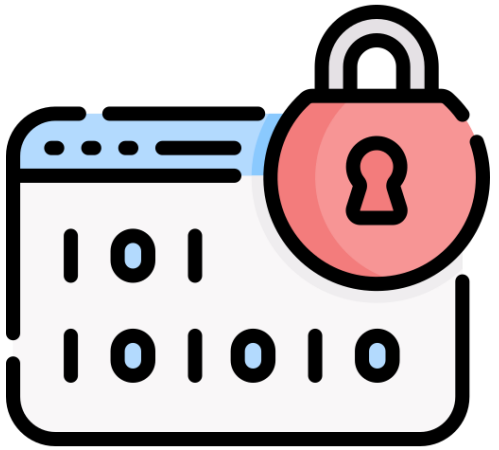
Threat management, or cyber threat management, is a set of procedures often used by cybersecurity professionals to manage the life cycle of a threat to identify and respond to it with speed and accuracy. Architecture and security requirements are internal build of secure software development which are one of the most important factors for application security maturity.

- ❑ A robust threat modelling techniques must be implemented by project champion
- ❑ Document threats to your application environment, document it
- ❑ Classify threats based on their likelihood of attempt and risk severity
- ❑ Threat classification and documentation must consider the type of attackers, likelihood, exploit techniques, exploitation difficulties, technical difficulties, impact area, no. of systems and resources affected by each threat, etc.
- ❑ Stakeholders must be aware of all types of threats
- ❑ Threats from suppliers, contractors and third-party software, libraries must be considered
- ❑ Security requirements should be considered based on various team's activity feedbacks/suggestions such as pentest team, quality assurance team, external users, third-parties
- ❑ Security requirements must be developers by each individual of project teams
- ❑ Security requirements must be taken from industry best practices standards, frameworks and policies
- ❑ RBAC must be implemented to protect application infrastructure from internal and external threats
- ❑ Third-party agreements, services, and data stored in their environment must be audited for security and compliance
- ❑ Audit for security requirements should also be performed
- ❑ Audit must be performed for secure software architecture components
- ❑ Not only penetration testing but red team assessment activity should also be done
- ❑ Threat intelligence and hunting exercise must be conducted on a regular period

4. Secure SDLC

A secure SDLC process ensures that security assurance activities such as penetration testing, code review, and architecture analysis are an integral part of the development effort. The primary advantages of pursuing a secure SDLC approach are: More secure software as security is a continuous concern.

- ❑ Ensure design requirements are recorded in all levels of architecture diagrams
- ❑ Consider application design against known security threats
- ❑ Design requirements must contain security elements such as authentication, authorization, input validation, cryptography, software management, RBAC, insider threats, logging, etc.
- ❑ Each application release should be reviewed against secure design elements before it goes live
- ❑ For design review, processes and procedures must be created and documented for each level of Stakeholders
- ❑ Design review process must include a detailed review of data storage, data in-use, data flow, etc.
- ❑ An organization must record minimum design security criteria for each application component
- ❑ Code segments can be categorized based on critical application functionality or code security sensitiveness. A detailed level of code security review process must be implemented for highly sensitive code
- ❑ A code security review process must utilize manual review and automated scanners along with IDE integrations
- ❑ Continuous code security monitoring approach should be utilized. Use automation wherever it is needed, applicable and possible
- ❑ Security testing must be conducted based on secure design requirements
- ❑ Security testing must be carried by the red team, purple team internally as well as by external vendors



3. S-SDLC Cont.

A secure SDLC process ensures that security assurance activities such as penetration testing, code review, and architecture analysis are an integral part of the development effort. The primary advantages of pursuing a secure SDLC approach are: More secure software as security is a continuous concern.

- ❑ Stakeholders must be well-informed about the security testing results and current status before go-live
- ❑ Utilize continuous security testing using the automation
- ❑ Business logic test cases must be generated by functional testing team as well as security testing team
- ❑ Business logic test cases must be defined based on the usability risk of each application functionality
- ❑ Minimum baseline criteria should exist for security testing
- ❑ A robust and secure credential management policy should be implemented which is not only helpful to store credentials only but also encryption keys and other highly-sensitive data
- ❑ Coding software security (IDE) must be taken into consideration
- ❑ Infrastructure, database and application configuration must be implemented and built securely, and they must be hardened with the industry-leading benchmarks such as CIS controls
- ❑ Dockers and Kubernetes security must be tested, against configuration and usability
- ❑ All the data must be encrypted at rest and in transit

References

- https://securityintelligence.com/posts/what-is-threat-management-common-challenges-and-best-practices/
- https://www.synopsys.com/blogs/software-security/software-security-governance
- https://www.synopsys.com/blogs/software-security/secure-sdlc
- https://digitalguardian.com/blog/what-operational-security-five-step-process-best-practices-and-more