

**St. Francis Institute of Technology
(An Autonomous Institution)**

AICTE Approved | Affiliated to University of Mumbai
A+ Grade by NAAC: CMPN, EXTC, INFT NBA Accredited: ISO 9001:2015 Certified

Department of Information Technology

A.Y. 2025-2026
Class: BE-IT A/B, Semester: VII
Subject: Secure Application Development Lab

Student Name: Durva Kadam

Student Roll No: 23

Experiment – 2 : Study of Secure Software Development Life Cycle (SDLC)

Aim: To study secure Software Development Life Cycle (SDLC).

Objectives: After study of this experiment, the student will be able to

- Understand different steps of SDLC .
- Identify and learn different standards of cyber security.

Lab objective mapped: ITL703.2 : To **understand** the methodologies and standards for developing secure code

Prerequisite: Basic Knowledge of different stages SDLC

Requirements: Personal Computer, Windows operating system browser, Internet Connection etc.

Pre-Experiment Theory:

What is Secure SDLC and Why is it important ?

Secure System Development Life Cycle (SecSDLC) is defined as the set of procedures that are executed in a sequence in the Software Development Life Cycle (SDLC). It is designed such that it can help developers to create software and applications in a way that reduces the security risks at later stages significantly from the start. The Secure System Development Life Cycle (SecSDLC) is similar to Software Development Life Cycle (SDLC), but they differ in terms of the activities that are carried out in each phase of the cycle. SecSDLC eliminates security vulnerabilities. Its process involves identification of certain threats and the risks they impose on a system as well as the needed implementation of security controls to counter, remove and manage the risks involved. Whereas, in the SDLC process, the focus is mainly on the designs and implementations of an information system.

Phases involved in SecSDLC are:

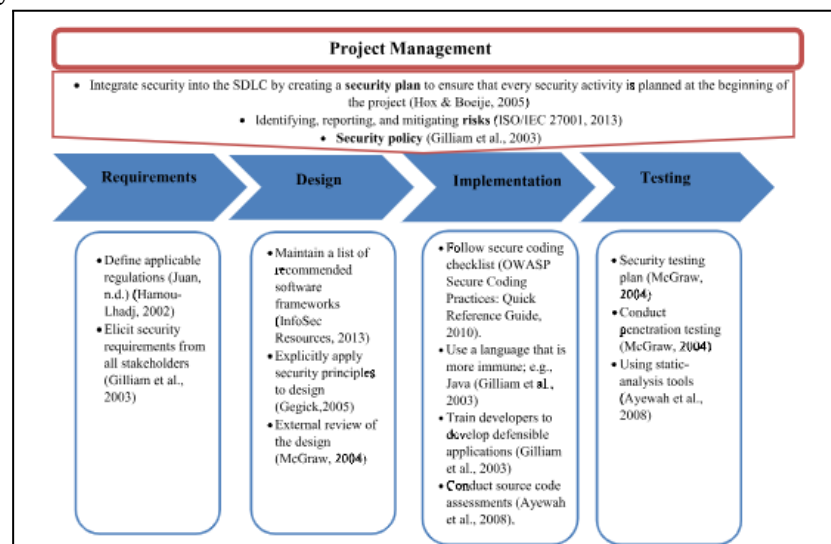
- **System Investigation:** This process is started by the officials/directives working at the top-level management in the organization. The objectives and goals of the project are considered priority in order to execute this process. An Information Security Policy is defined which contains the descriptions of

security applications and programs installed along with their implementations in organization's system.

- **System Analysis:** In this phase, detailed document analysis of the documents from the System Investigation phase are done. Already existing security policies, applications and software are analyzed in order to check for different flaws and vulnerabilities in the system. Upcoming threat possibilities are also analyzed. Risk management comes under this process only.
- **Logical Design:** The Logical Design phase deals with the development of tools and following blueprints that are involved in various information security policies, their applications and software. Backup and recovery policies are also drafted in order to prevent future losses. In case of any disaster, the steps to take in business are also planned. The decision to outsource the company project is decided in this phase. It is analyzed whether the project can be completed in the company itself or it needs to be sent to another company for the specific task.
- **Physical Design:** The technical teams acquire the tools and blueprints needed for the implementation of the software and application of the system security. During this phase, different solutions are investigated for any unforeseen issues, which may be encountered in the future. They are analyzed and written down in order to cover most of the vulnerabilities that were missed during the analysis phase.
- **Implementation:** The solution decided in earlier phases is made final whether the project is in-house or outsourced. The proper documentation is provided of the product in order to meet the requirements specified for the project to be met. Implementation and integration process of the project are carried out with the help of various teams aggressively testing whether the product meets the system requirements specified in the system documentation.
- **Maintenance:** After the implementation of the security program, it must be ensured that it is functioning properly and is managed accordingly. The security program must be kept up to date accordingly in order to counter new threats that can be left unseen at the time of design.

Procedure:

- Study the case study from references and discuss the proposed model to integrate security in SDLC.



Project Management Phase

Integrate security into the SDLC: This phase emphasizes the importance of creating a comprehensive security plan right from the inception of the project. This plan should outline all security activities to be carried out throughout the development process.

Identifying, reporting, and mitigating risks: This involves proactively identifying potential security risks, establishing procedures for reporting them, and implementing strategies to mitigate these risks.

Security policy: A well-defined security policy is crucial. It should outline the organization's security objectives, roles and responsibilities, and the standards to be followed.

Requirements Phase

Define applicable regulations: This phase involves identifying and understanding all relevant regulations and standards (like ISO/IEC 27001) that the software must comply with.

Elicit security requirements from all stakeholders: Gathering security requirements is not just the responsibility of the security team. It involves input from various stakeholders, including end-users, to ensure a holistic approach.

Design Phase

Maintain a list of recommended software frameworks: This phase focuses on selecting secure software frameworks and libraries that adhere to established security best practices.

Explicitly apply security principles to design: Security should be an integral part of the design process. This involves incorporating security controls and mechanisms into the software architecture.

External review of the design: An independent review of the design by security experts can help identify potential vulnerabilities and weaknesses.

Implementation Phase

Follow secure coding checklist (OWASP Secure Coding Practices: Quick Reference Guide, 2010): Developers should follow established secure coding guidelines to minimize vulnerabilities.

Use a language that is more secure, e.g., Java: Choosing programming languages with strong security features can be beneficial.

Train developers to develop defensible applications: Regular training on secure coding practices is essential to build a security-conscious development team.

Conduct source code assessments: Analyzing the code for vulnerabilities using static analysis tools is a crucial step in this phase.



Testing Phase

Security testing plan (McGraw, 2004): A dedicated security testing plan should be in place to identify and address security flaws.

Conduct penetration testing: Simulating real-world attacks can help uncover vulnerabilities that might have been missed by other testing methods.

Using static analysis tools (Ayewah et al., 2008): These tools can automatically analyze code for potential security weaknesses.

- Compare Software Development Life Cycle (SDLC) and Secure SDLC

Parameter	SDLC	Secure SDLC
Focus	Functional and performance aspects of the software	Security integration throughout the development process
Requirements Gathering	Focus on gathering functional and non-functional requirements	Includes security requirements, threat modeling, and compliance needs
Design	Architectural and design specifications for functionality	Incorporates secure design principles, risk analysis, and security controls
Implementation	Writing code based on design specifications	Applies secure coding practices, performs security-focused code reviews
Testing	Verifies functionality and performance	Verifies functionality and performance
Deployment	Installation and configuration in production environment	Ensures secure configurations and applies security hardening
Maintenance	Fixes bugs and adds features	Focuses on patch management and incident response
		

- Describe how secure coding can be incorporated into the software development process.

ANS:

Incorporating secure coding into the software development process involves integrating security practices and principles at various stages of development to prevent vulnerabilities and ensure robust protection. Here's a detailed approach to incorporating secure coding:

1. Training and Awareness

Developer Training: Provide regular training sessions for developers on secure coding practices, common vulnerabilities (e.g., OWASP Top Ten), and secure development principles.

2. Secure Design Principles

Adopt Secure Design: Apply secure design principles such as least privilege, defense in depth, and secure defaults to the overall architecture and design of the software.

Threat Modeling: Conduct threat modeling sessions to identify potential security threats and design countermeasures early in the development process.

3. Secure Coding Standards

Coding Guidelines: Establish and enforce secure coding standards and guidelines for developers. These should address common vulnerabilities and secure coding practices specific to the programming languages and technologies used.

Code Reviews: Implement peer code reviews with a focus on security to catch vulnerabilities early. Use both manual review and automated tools to check for adherence to secure coding practices.

4. Input Validation and Output Encoding

Input Validation: Validate all input data to ensure it meets expected formats and ranges before processing. This helps prevent injection attacks and other data-related vulnerabilities.

Output Encoding: Encode output data to prevent injection of malicious content into web pages or other output channels. This includes proper escaping of HTML, SQL, and other data formats.

5. Authentication and Authorization

Strong Authentication: Implement strong authentication mechanisms, such as multi-factor authentication (MFA), to ensure that only authorized users can access the system.

Authorization Controls: Enforce proper authorization checks to ensure that users can only access resources and perform actions they are permitted to.

6. Secure Error Handling

Avoid Information Leakage: Ensure error messages do not expose sensitive information that could be exploited by attackers. Implement generic error messages for end-users and detailed logs for developers.

Exception Handling: Handle exceptions properly to avoid application crashes and potential information leaks.

7. Data Protection

Data Encryption: Encrypt sensitive data both in transit (using protocols like TLS) and at rest (using strong encryption algorithms).

Secure Storage: Store sensitive information, such as passwords and encryption keys, securely using appropriate hashing algorithms and key management practices.

8. Session Management

Secure Session Handling: Implement secure session management practices, including secure cookie attributes, session timeouts, and proper session invalidation on logout.

Session Fixation Prevention: Ensure that session identifiers are regenerated upon login to prevent session fixation attacks.

9. Dependency Management

Regular Updates: Keep third-party libraries and dependencies up-to-date with the latest security patches and updates.

Dependency Scanning: Use tools to scan for known vulnerabilities in dependencies and libraries used within the project.

Post-Experiments Exercise:

Extended Theory: Nil

Post Experimental Exercise:

Questions:

- List the major types of coding errors and their root causes.
- Describe good software development practices and explain how they affect application security.

Conclusion:

- Write what was performed in the experiment.
- Write the significance of the topic studied in the experiment.

References:

- Case study: <https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.1700>
- <https://snyk.io/learn/secure-sdlc/>
- <https://www.checkpoint.com/cyber-hub/cloud-security/what-is-secure-coding/>
- <https://snyk.io/learn/secure-coding-practices/>