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: Vishal Rajesh Mahajan **Student Roll No:** 1

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 priory 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:

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

The model described in “**Evolution of secure development lifecycles and maturity models in practice**” advocates for embedding security practices throughout the Software Development Life Cycle (SDLC). Security is no longer treated as an afterthought or a final step; it becomes an integral part of requirements gathering, design, implementation, and testing.

Key Elements of the Model:

- **Early Planning:** Security requirements are identified at the start of the project. Risk assessments and threat modeling are performed to ensure potential vulnerabilities are considered from the outset.
- **Secure Design and Implementation:** Secure architecture principles are applied during design. Developers follow secure coding standards and receive regular security training to reduce the risk of introducing vulnerabilities.
- **Continuous Testing and Automation:** The model encourages the use of automated tools (static and dynamic code analysis, vulnerability scanning) throughout the build and test phases, enabling rapid identification and remediation of security issues.
- **Maturity and Adaptability:** Organizations regularly assess the maturity of their secure development processes, adapting security measures to fit different SDLC models whether Waterfall, Agile, or DevOps. This ongoing evaluation encourages continuous improvement and ensures security remains aligned with evolving software practices.

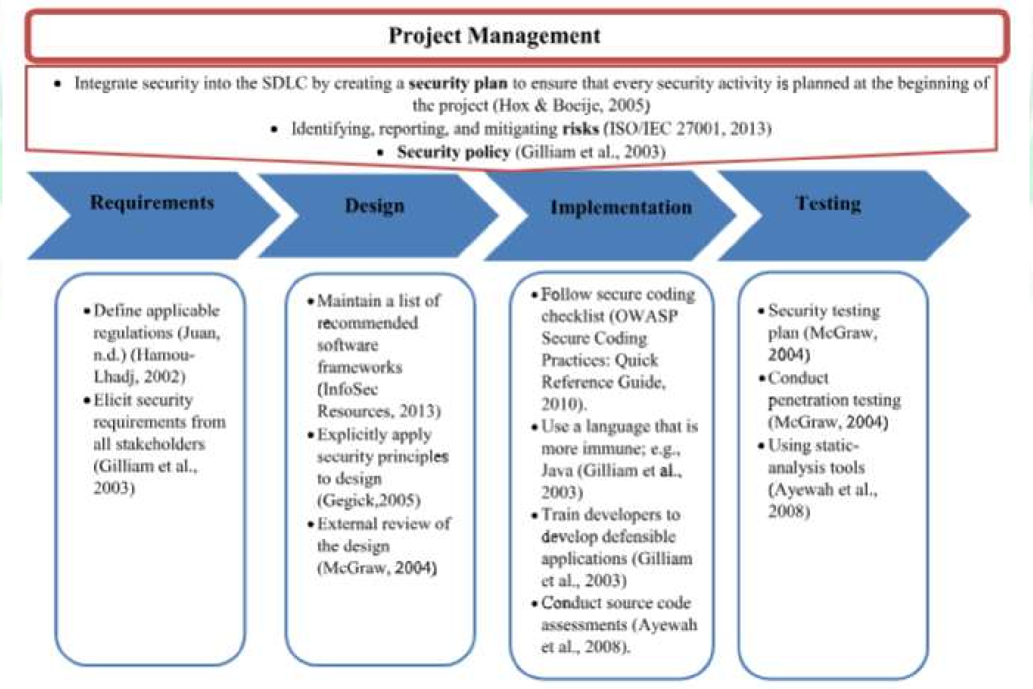
Project Management
• Integrate security into the SDLC by creating a security plan to ensure that every security activity is planned at the beginning of the project (Hox & Boeije, 2005)
• Identifying, reporting, and mitigating risks (ISO/IEC 27001, 2013)
• Security policy (Gilliam et al., 2003)
Requirements
Design
Implementation
Testing
• Define applicable regulations (Juan, n.d.) (Hamou-Lhadj, 2002)
• Elicit security requirements from all stakeholders (Gilliam et al., 2003)
• Maintain a list of recommended software frameworks (InfoSec Resources, 2013)
• Explicitly apply security principles to design (Gegick,2005)
• External review of the design (McGraw, 2004)
• Follow secure coding checklist (OWASP Secure Coding Practices: Quick Reference Guide, 2010).
• Use a language that is more immune; e.g., Java (Gilliam et al., 2003)
• Train developers to develop defensible applications (Gilliam et al., 2003)
• Conduct source code assessments (Ayewah et al., 2008).
• Security testing plan (McGraw, 2004)
• Conduct penetration testing (McGraw, 2004)
• Using static-analysis tools (Ayewah et al., 2008)

Figure 1. Preliminary research model. SDLC, software development life cycles.

[**Source:**https://onlinelibrary.wiley.com/doi/epdf/10.1002/sec.1700]

2. Compare Software Development Life Cycle (SDLC) and Secure SDLC
Answer:

Aspect	**SDLC**	**SSDLC**
Focus	Delivering functional software	Integrating security at every phase
Objective	Meeting business/functional requirements	Protecting against vulnerabilities with security measures
Phases	Planning, analysis, design, implementation, testing, deployment, maintenance	Same phases, but adds threat modeling, secure coding, continual monitoring
Security Placement	Security addressed mainly during testing	Security considered from the start in every stage
Testing	Testing is a separate (often late) phase	Security testing embedded in every phase
Documentation	Focus on functionality/features	Adds threat models and security risk assessments
Speed	Often faster to deliver	May increase cycle time due to security steps
Adaptability	Adaptable to functional/business needs	Can require more effort to update due to robust security controls

[**Source:***https://www.geeksforgeeks.org/ethical-hacking/what-is-secure-software-development-life-cycle-ssdlc/*]

3. Describe how secure coding can be incorporated into the software development process.
Answer:
Secure coding is the practice of writing software in a way that protects it from vulnerabilities and security threats. Integrating secure coding into the Software Development Life Cycle (SDLC) involves embedding security measures throughout each phase, rather than treating security as a separate or last-minute step.

- **Planning and Design:** Early in the process, security requirements are identified alongside functional requirements. Threat modeling is used to analyze potential attack vectors and weaknesses, helping teams design architecture and workflows that minimize risk from the start.

- **Development (Coding) Phase:** Developers adhere to secure coding standards and guidelines, such as those published by OWASP or NIST. This includes validating inputs, managing user authentication properly, avoiding common pitfalls like SQL injection or cross-site scripting, and applying strong cryptography where necessary.

- **Code Reviews and Automated Tools:** Combined manual peer reviews and automated static/dynamic analysis tools (SAST/DAST) are used to detect vulnerabilities or insecure patterns early. These tools help catch bugs that might be missed during manual inspection, enabling timely correction before deployment.

- **Security Testing:** Security-focused testing such as penetration tests or vulnerability scans is integrated alongside functional tests. These tests validate that the application resists attacks and that implemented controls work as intended.

- **Deployment and Maintenance:** Secure coding doesn't end with release. Continuous monitoring, patching vulnerabilities, and securely managing configuration and secrets ensure the software remains protected as threats evolve.

By making secure coding a core part of development processes from the outset and leveraging both human and automated efforts—organizations can significantly reduce software vulnerabilities, decrease remediation costs, and build more trustworthy applications.

[**Source:***http://legitsecurity.com/aspm-knowledge-base/what-is-secure-coding*]

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:

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