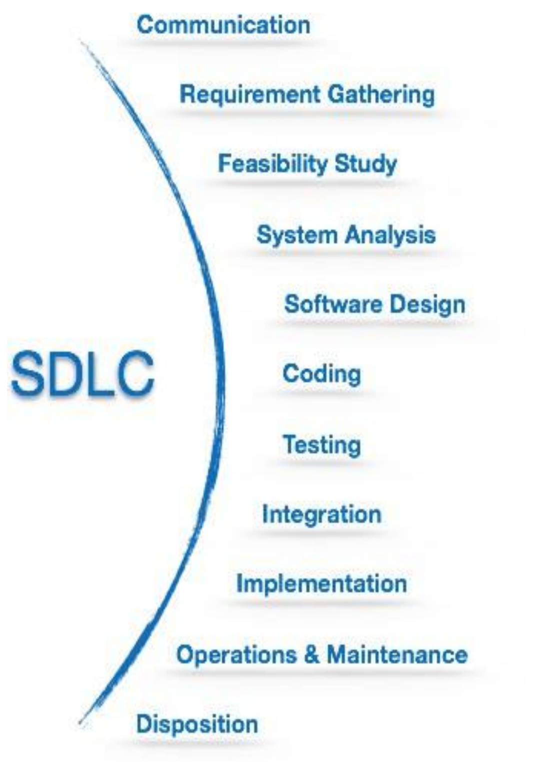
Secure System SDLC

Abstract:

The system development life cycle (SDLC) is a formal way of ensuring that
adequate security controls and requirements are implemented in a new system or application. Integrating technologies and practices into the development of new system and application deployments provides an opportunity to design security into the solution on the front end of the process, rather than retrofitting it after the solution is deployed.
To achieve this integration, the SDLC process for system and application deployments should be clearly outlined, with defined and enforced checkpoints that incorporate security reviews prior to moving to the next project phase. Without formally implementing the SDLC and generating the requisite deliverables, it is much more difficult to effectively manage the development process and ensure that security-related issues are adequately addressed.



What is Secure SDLC?

A software development life cycle (SDLC) is a framework that defines the process used by organizations to build an application from its inception to its decommission. Over the years, multiple standard SDLC models have been proposed (waterfall, iterative, agile, etc.) and used in various ways to fit individual circumstances. It is, however, safe to say that in general, SDLCs include the following phases:

- Planning and requirements
- Architecture and design
- Test planning
- Coding
- Testing and results

- Release and maintenance

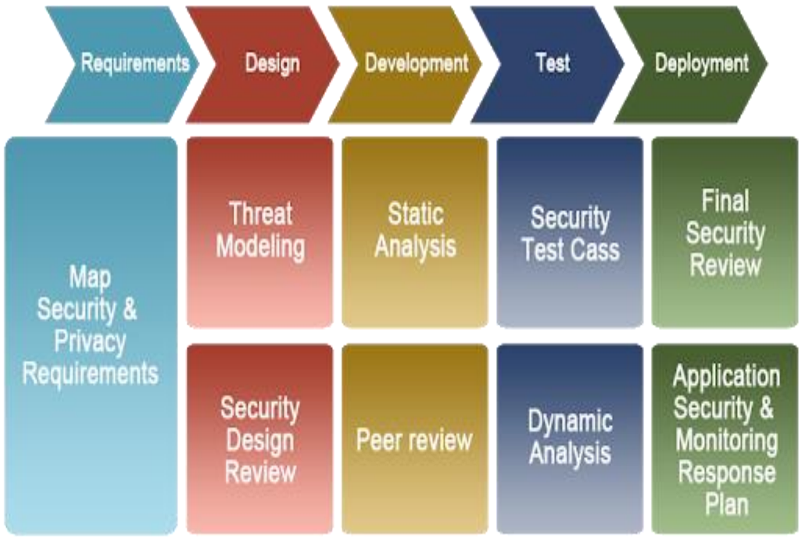
In the past, it was common practice to perform security-related activities only as part of testing. This after-the-fact technique usually resulted in a high number of issues discovered too late (or not discovered at all). It is a far better practice to integrate activities across the SDLC to help discover and reduce vulnerabilities early, effectively building security in.

It is in this spirit that the concept of the secure SDLC arises. 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.
- Awareness of security considerations by stakeholders.
- Early detection of flaws in the system.
- Cost reduction as a result of early detection and resolution of issues.
- Overall reduction of intrinsic business risks for the organization.

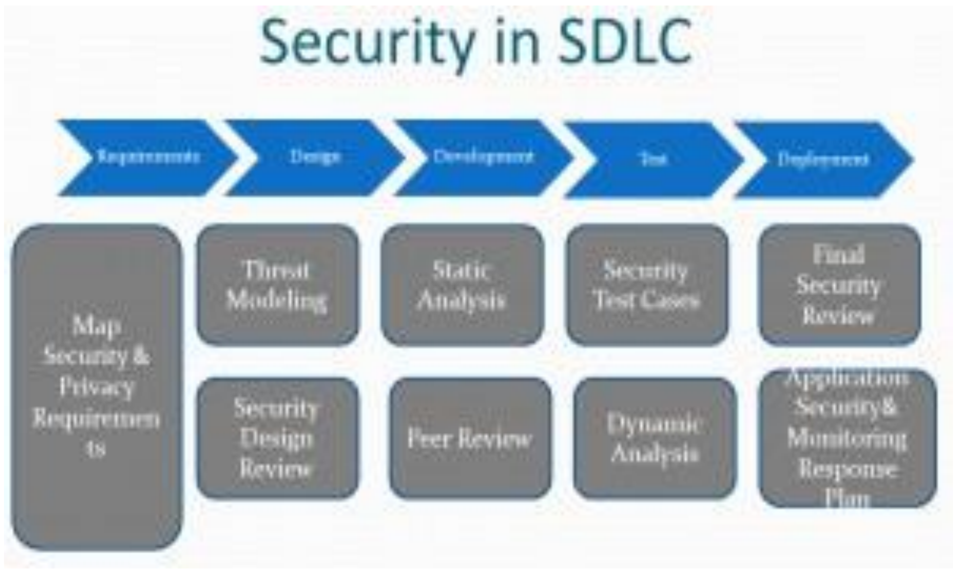
The Secure Systems Development Lifecycle (SSDLC) defines security requirements and tasks that must be considered and addressed within every system, project or application that is created or updated to address a business need. The SSDLC is used to ensure that security is adequately considered and built into each phase of every system development lifecycle (SDLC).

The SSDLC toolkit was developed to assist project, systems, and application teams in collecting the appropriate artifacts and documentation to fulfill the security tasks in the SSDLC standard (NYS-S13-001). The security tasks within the SSDLC are easily mapped back to the phases in most SDLC and should be used as a guideline to initiation of the security tasks.



Why is it important?

Systems and applications change over time to adjust to ever changing business, regulatory and statutory requirement. Security is a requirement that must be included within every phase of a systems development life cycle. Per NYS Information Security Policy, (NYS-P03-002), a secure SDLC must be utilized in the development of all State Entities (SE) applications and systems. This includes applications and systems developed for SEs. Agency program staff are ultimately responsible for maintaining system documentation as defined by the SSDLC standard.



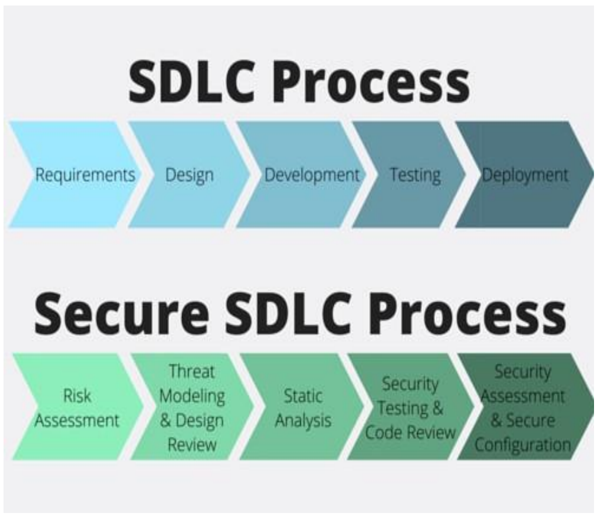
How does a secure SDLC work?

Generally speaking, a secure SDLC is set up by adding security-related activities to an existing development process. For example, writing security requirements alongside the collection of functional requirements, or performing an architecture risk analysis during the design phase of the SDLC.
Many secure SDLC models have been proposed. Here are a couple of them:

- MS Security Development Lifecycle (MS SDL): One of the first of its kind, the MS SDL was proposed by Microsoft in association with the phases of a classic SDLC.
- NIST 800-64: Provides security considerations within the SDLC. Standards were developed by the National Institute of Standards and Technology to be observed by US federal agencies.

Security System Development Life Cycle

Today, security of software applications and databases has become as important as the software and data itself. Security forms a major aspect of the business development process.
Security System Development Life Cycle is defined as the series of processes and procedures in the software development cycle, designed to enable development teams create software and applications in a manner that significantly reduces security risks, eliminate security vulnerabilities and reducing costs. The process, like the traditional Systems Development Life Cycle, is divided into a number of phases.



Systems Investigation

Directives normally emanating from top level management initiates this investigative process. The overall objective, goal and budget of the project are brought into perspective. An Information Security Policy is defined which details the various security programs and their implementation plans within the organisation. The various related teams: managers, employees and contractors are established. Their respective detailed goals and objectives are set up including any unforeseen issues that arise or not previously covered.

Systems Analysis

In the System Analysis phase, detailed document analysis, of the documents from the investigative phase, is done. Existing security policies, software and applications are analysed and assessed. Current threats, new risks and their associated internal controls are evaluated. During the systems analysis phase, the process of Risk Management commences. Risk Management is the series of processes that identify and evaluate current and future risks and vulnerabilities.

Logical Design

The Logical Design phase involves the development of tools and blueprints of the various information security policies. Backup and recovery processes and details of the organisation's incidence response actions are laid out. Details of business response action to disaster are carefully planned. The decision as to whether the project is developed in-house or outsourced, is reached during this phase.

Physical Design

This is the point at which the technical teams move into action. The information security technology that will be needed for the implementation of all blueprints and analysis, detailed during the logical design phase, are evaluated and acquired. During this phase, alternative solutions investigated for any unforeseen issues which may arise, are analysed, and mapped out.
All the different teams at this point issue their stamp of approval of all processes and the green light is given to proceed.

Implementation

The security solutions decided and approved are acquired, whether built in-house or outsourced. Adequate documentation is provided on specifications of the product to ensure project specifications are met. Their implementation and integration processes are rolled out, with various teams carrying out intensive testing to ensure that the solutions meet the requirements outlined in the various blueprints and policies.

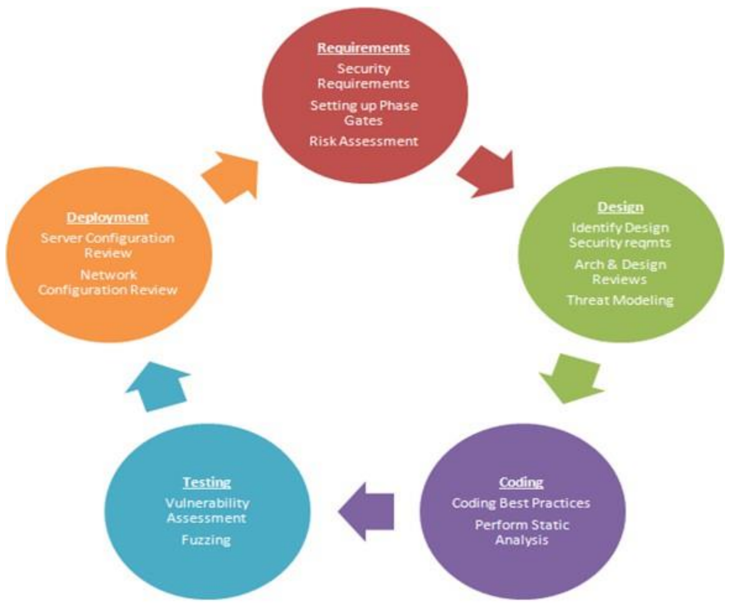
Deployment:

The deployment phase is where the system or application’s security features are developed, configured, and enabled. Use the program specifications to describe the program logic and processing requirements. Program specifications are developed as part of the development phase prior to the commencement of programming.

These specifications provide the thought process required to determine the steps to code the programs.

Breaking Down the System Development Life Cycle and Its Phases:

What is the difference between the system development life cycle and the software development life cycle? The system development life cycle involves end-to-end people, processes, and technology deployments, which includes software, infrastructure and change management. The software development life cycle focuses exclusively on software components, such as development planning, technical architecture, software quality testing and the actual deployment of the software. Put simply, the system development life cycle is more holistic and comprehensive.



Sample S-SDLC

The SDLC typically reflects the phased activities described below.

Project Initiation:

Prepare a formal project request to initiate all system development and integration activities. The request should include the project objectives, users of the system or application, criticality in terms of confidentiality, integrity and availability, and key time frames for completion.

Analysis

Perform a feasibility study to determine whether the project request should be approved for development. The feasibility study should incorporate:

- Assessment of the impact on the existing environment;
- Staff development and resource requirements;
- Project development cost analysis;
- Program maintenance costs;
- Evaluation of alternative project implementation approaches, such as build versus buy and outsourcing;
- Description of the proposed solution approach;
- Risks associated with the proposed solution;
- Benefit analysis including cost reduction, error reduction, new customers and improved customer service.

- Information security teams should initiate their own involvement with the project during this phase to ensure that appropriate security concerns have been incorporated into the feasibility study.

Business and Operational Requirements Specifications

Develop business and operational requirements specifications to ensure that the project requirements necessary to support business objectives are understood.
Users and development teams generally lead this process. Business requirements should address:

- Data that is required to support the system or application and how it relates to other data;
- Frequency of use of the system or application;
- Required response time for online processing;
- Function relations and dependencies on other components;
- Identification of applicable legal or regulatory requirements or constraints; and
- Anticipated life span of the system or application.
- Operational requirements should address:
- Security requirements;
- Contingency requirements;
- Distributed and centralized processing requirements;
- Data input strategy and assumed responsibility;
- Data retention requirements;
- Output distribution requirements;
- Expected transaction volumes, including project transaction growth; and
- Critical system performance requirements.

This document should also describe the type of development activity that the project represents. Common project types include maintenance, enhancement, new system, and emergency change. Criteria should be defined for when a development activity may be assigned to these categories.

Information security teams should be involved throughout the business and operational requirements phase to ensure that security concerns are properly addressed and reflected in the requirements document. The risk assessment methodology is largely performed during this phase, providing early security perspectives to the project team.

Functional Specifications

Transpose the business and operational requirements into functional requirements to reflect the anticipated user experience associated with the system or application. Functional specifications reflect the user's perspective that has been translated into the preliminary design. For maintenance and enhancement activities, the focus is to document what is changing using a before/after description.

Functional specifications should include:

- Data flow diagrams — tracing of data through all its processing points;
- Data definitions — definition of data, data relationships and naming conventions;
- Screen definitions — definition of input fields, range checks, etc.;
- Inputs — source of input, type of data and description of data;

- Report definitions — description of reports, data contained in each report, how data values are derived and users that utilize specific reports;
- Control and security requirements — input edit requirements, audit log trails for critical data from the point of origin to the point of disposition, audit log trails for use of privileges and identification of critical processing areas;
- System interface requirements — interaction points between this system and other systems, anticipated inputs and outputs, response time expectations, and other intersystem dependencies;
- Backup, restart and recovery — frequency of backup, rationale behind backup, backup retention requirements, restart requirements specifying how the application should be restarted and recovery requirements;
- Contingency requirements — analysis to determine how long the application can be unavailable before the business is affected and identification of data sets, software and other items that need to be restored at an off-site processing center;
- Hardware requirements — communication requirements, disk space, processing equipment, etc.;
- Service-level requirements — uptime requirements, required response times, critical windows, deadlines for input, deadlines for report distribution, etc.;
- Capacity requirements — transaction volumes, expected growth, etc.;
- Conversion requirements — method used for creating data on the new system, method for reconciling data during conversion, cut-over requirements, and process for verifying converted data.

During the functional specifications process, information security teams should generally play a supportive role, supporting the project team's effort to capture the preliminary design and functional description of the system or application.
Functional specifications should include security-related information such as technical features (e.g., access controls) and operational practices (e.g., awareness and training). Information security teams should review and provide feedback on this document prior to the detailed design phase.

Detailed Design Specifications

Develop detailed design specifications that translate functional specifications into a logical and physical design. Detailed design specifications are developed during the design phase of the SDLC and describe how the system or application is designed to satisfy the requirements documented in the functional specifications. Detailed design specifications should include the following:

- Database requirements — relationship between data elements;
- File requirements — description of each file, file access methods, list of fields within a record, data attributes and anticipated number of records;
- System flow diagram — sequential flow of programs that are executed, their relationships to inputs and outputs, and security provisions between programs;
- Program requirements — programs used and their purpose, description of formulas and calculations, and interrelationships between programs;
- System operations requirements — job streams, including purpose of processing, job names used and restart/recovery procedures;
- Error handling requirements;
- Backup/recovery procedures;
- System startup and shutdown procedures;

- Screen designs — fields in each screen, purpose of each field, description of how each screen is triggered, logical flow of screens specifying screens that pull other screens, input checks performed, and description of error messages;
- Report designs — data contained in each field of each report and a definition of how each data result is derived; and
- Security design — description of access control mechanisms, audit log provisions, user authentication and encryption provisions.

During the detailed design phase, once again, information security teams should support the project team's effort to design the system to achieve the desired solution. Security professionals should participate in project meetings for major design reviews, including a security design review, and at the request of the project team. As part of the detailed design process, information security teams should assess whether security requirements have been adequately addressed and whether adequate testing plans are in place. They should also review the detailed design specifications prior to the next phase.

Development

The development phase is where the system or application's security features are developed, configured and enabled. Use the program specifications to describe the program logic and processing requirements. Program specifications are developed as part of the development phase prior to the commencement of programming. These specifications provide the thought process required to determine the steps to code the programs.
Information security teams should retain the right to perform source code reviews for critical aspects of the system or application, including user authentication, authorization and financial transactions. Source code reviews should have an enhanced focus on code provided by third parties, including offshore development organizations.

Unit Testing

Unit testing aims to identify program problems within a standalone environment. Unit test criteria should include:

- File updating, merging and sorting;
- All decision logic;
- All system or application interfaces (integration testing);
- Invalid transactions and their error handling routines;
- Restart/recovery routines;
- Stress testing;
- Error conditions; and
- Page counters and overflow headers.

System Testing

During the system testing phase, all program development for the project is completed and testing is performed to ensure that all functionality works as required. The system test environment is typically shared among all programmers with strictly controlled changes to the environment. System test criteria should include:

- Verification that all functionality is performed as specified by the functional and design specifications;
- Program interfaces;

- Other system interfaces;
- Restart and recovery procedures;
- Transaction validation and rejection;
- Transaction processing cycles;
- System or application performance criteria;
- System or application output generation;
- Stress testing;
- Error handling;
- Input/output verification;
- Procedures and restrictions regarding the use of production data;
- Completeness and accuracy of audit log trails;
- Security testing (e.g., authentication, authorization);
- System or application security testing (e.g., ethical hacking); and
- Code reviews of critical sections of code and code developed externally.

Where possible, system or application security testing should be executed using an automated testing tool. This will support the creation of test harnesses and procedures that can be used for regression testing during future enhancements.
During the system testing phase, information security teams should be heavily involved in reviewing the security tests being written by the project/test team and validating the security testing results. Security teams may also elect to perform a penetration test to validate that the development team did not overlook common security vulnerabilities.

Parallel Test Plan

Where an existing system or application is in place, parallel testing ensures that the functions within a simulated production environment are equivalent to the existing process.

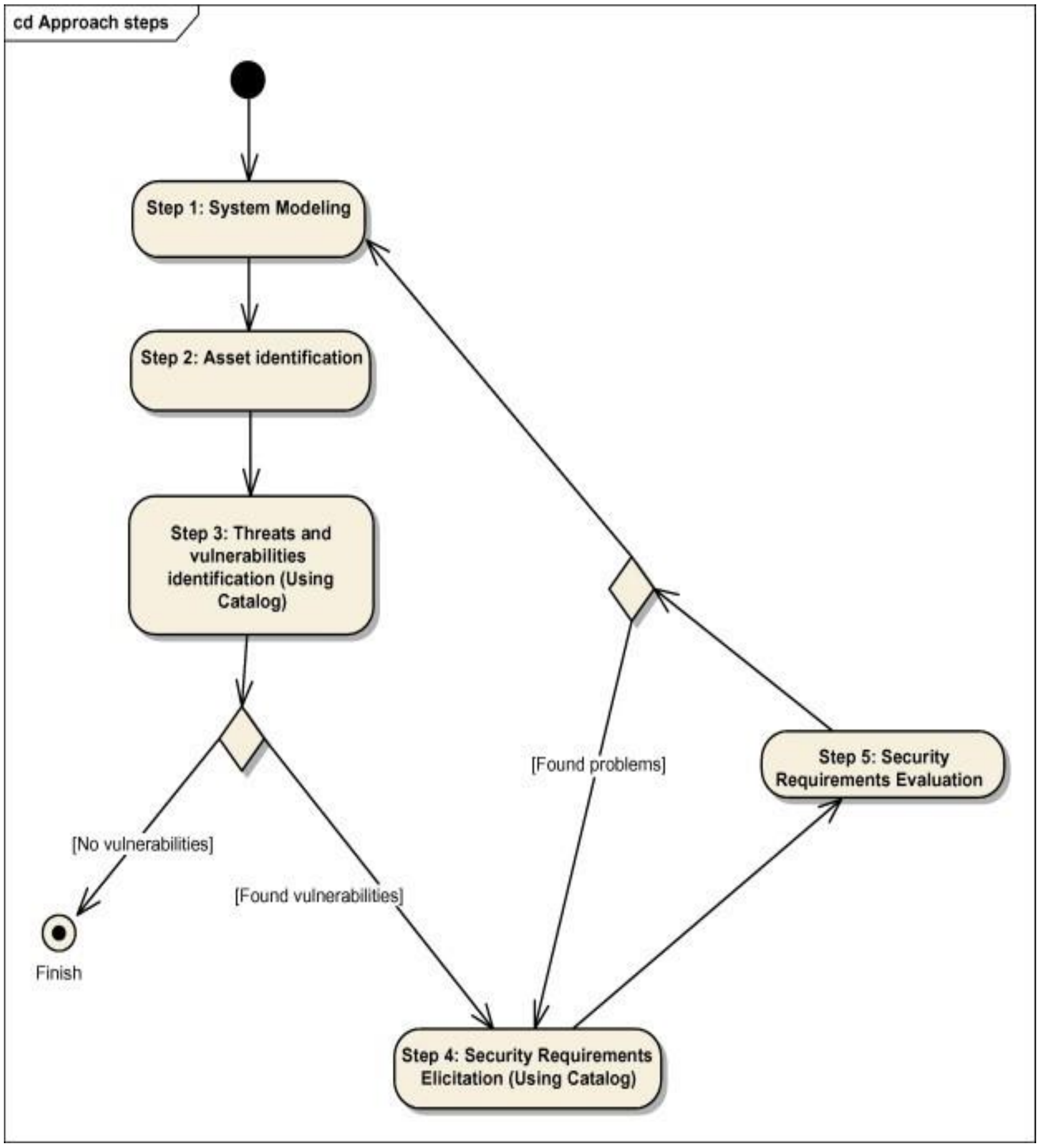
Cutover/Installation Plan

The cutover/installation plan documents the transition from an old system or application to a new one. This plan should address any migration of production data that has not been performed. It should also address the installation activities and coordination with system users. Fallback procedures should be defined in the event of an erroneous transition.

Post-Implementation Review

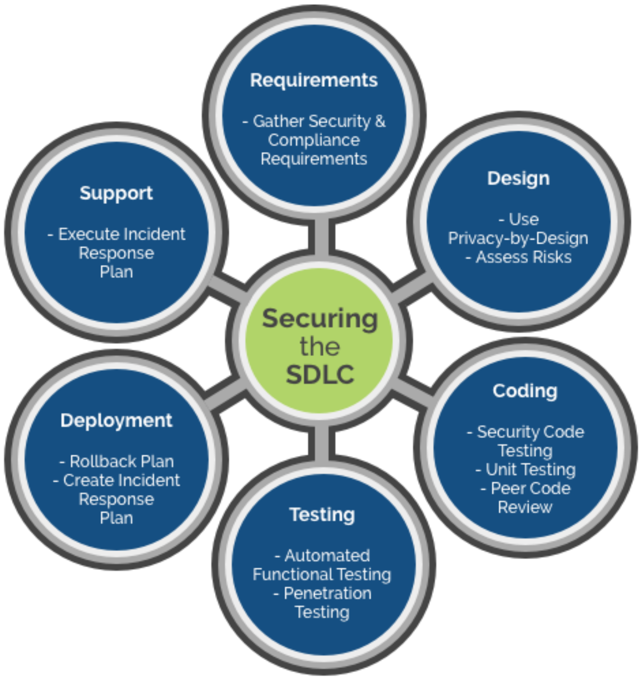
A post-implementation review ensures that the system or application is operating at a satisfactory level. This review involves soliciting user feedback on the overall effectiveness of the project and achievement of the requirements, timelines, etc. This information provides valuable insight for future projects and identifies potential shortcomings in the SDLC.

Security teams should participate in the post-implementation review to confirm that the security capabilities deployed are satisfactory. At this time, the documentation of all security decisions made in support of the system or application is finalized and variances to the existing security policies and standards are noted. Where variances are permitted on a temporary basis, tracking is initiated to ensure that variances are resolved in accordance with an agreed-upon schedule.

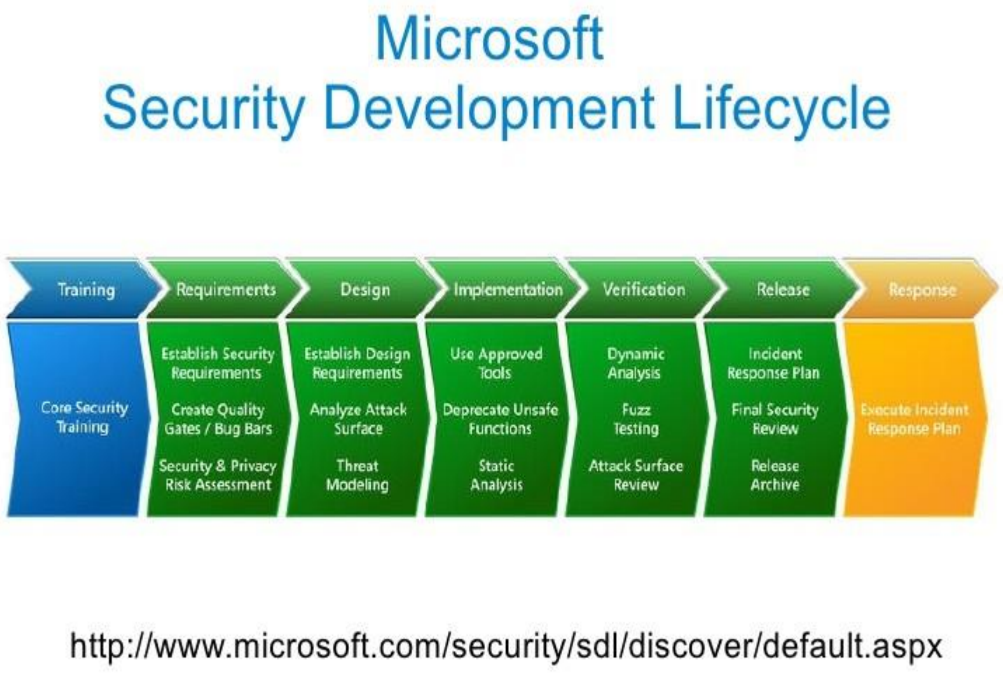

cd Approach steps
Step 1: System Modeling
Step 2: Asset identification
Step 3: Threats and vulnerabilities identification (Using Catalog)
[No vulnerabilities]
Finish
[Found vulnerabilities]
[Found problems]
Step 5: Security Requirements Evaluation
Step 4: Security Requirements Elicitation (Using Catalog)

Conformance and Defect Tracking

The project management process should ensure conformance with all aspects of the SDLC. In this context, conformance refers to ensuring that the documents itemized above are created and then reviewed and approved prior to the project moving on to the next phase of the SDLC. Any modifications to a document, once approved, should be reviewed and all impacted groups should agree on the change.
Defect checking tools should be used to monitor and track identified defects during all testing phases. This provides the basis for making informed decisions regarding the status and resolution of any defects.



Example:



Conclusion:

The Secure SDLC ensures that project development is sufficiently integrated to provide adequate security in the resulting system or application. The SDLC should be documented, and project development activities should conform to them; all should be guided by written standards and procedures for each phase. These standards should address design, programming, testing, implementation, documentation and maintenance and be flexible while incorporating security checkpoints to validate the adequacy of controls within the system or application.