Concepts View

CLASP — Comprehensive, Lightweight Application Security Process — is an activity-driven, role-based set of process components whose core contains formalized best practices for building security into your existing or new-start software development lifecycles in a structured, repeatable, and measurable way.

CLASP is the outgrowth of years of extensive field work in which system resources of many development lifecycles were methodically decomposed in order to create a comprehensive set of security requirements. These resulting requirements form the basis of CLASP's best practices which allow organizations to systematically address vulnerabilities that, if exploited, can result in the failure of basic security services — e.g., confidentiality, authentication, and access control.

- **Adaptability of CLASP to Existing Development Processes**
  CLASP is designed to allow you to easily integrate its security-related activities into your existing application development processes. Each CLASP activity is divided into discrete process components and linked to one or more specific project roles. In this way, CLASP provides guidance to project participants — e.g., project managers, security auditors, developers, architects, testers, and others — that is easy to adopt to their way of working; this results in incremental improvements to security that are easily achievable, repeatable, and measurable.
- **CLASP Vulnerability Lexicon**
  CLASP also contains a comprehensive Vulnerability Lexicon that helps development teams avoid/remediate specific designing/coding errors that can lead to exploitable security services. The basis of this Lexicon is a highly flexible taxonomy — i.e., classification structure — that enables development teams to quickly locate Lexicon information from many perspectives: e.g., problem types (i.e., basic causes of vulnerabilities); categories of problem types; exposure periods; avoidance and mitigation periods; consequences of exploited vulnerabilities; affected platforms and programming languages; risk assessment.
- **Automated Analysis Tools**
  Much of the information in the CLASP Vulnerability Lexicon can be enforced through use of automated tools using techniques of static analysis of source code.

Overview of CLASP Process

This section provides an overview of CLASP's structure and of the dependencies between the CLASP process components and is organized as follows:

- CLASP Views
- CLASP Resources
- Vulnerability Use Cases

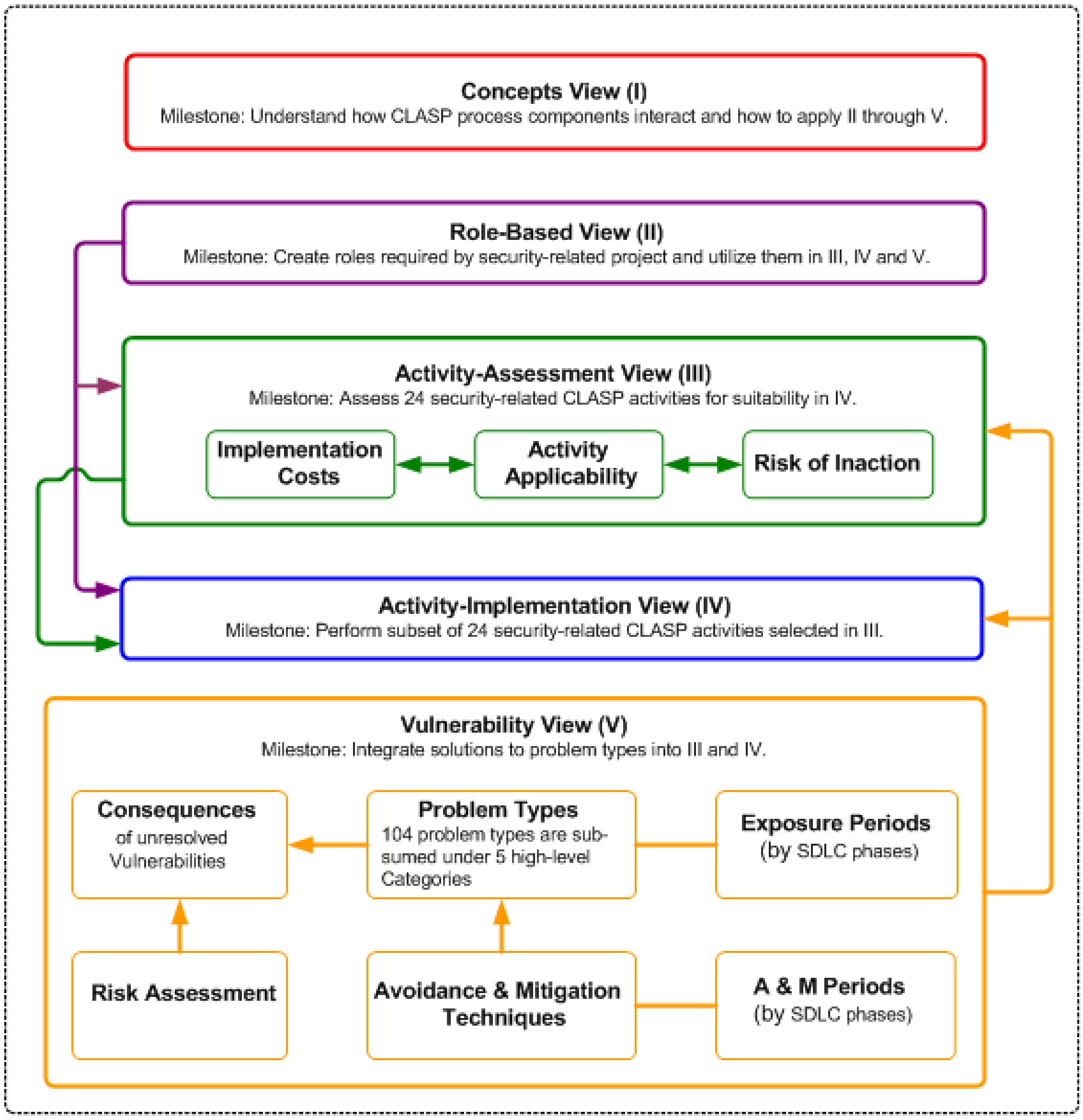
CLASP Views

The CLASP process is presented through five high-level perspectives called CLASP Views. These views are broken down into activities which in turn contain process components. This top-down organization by View > Activity > Process Component allows you to quickly understand the CLASP process, how CLASP pieces interact, and how to apply them to your specific software development lifecycle.

These are the CLASP Views:

- Concepts View
- Role-Based View
- Activity-Assessment View
- Activity-Implementation View
- Vulnerability View

The following figure shows the CLASP Views and their interactions:


Concepts View (I)
Milestone: Understand how CLASP process components interact and how to apply II through V.
Role-Based View (II)
Milestone: Create roles required by security-related project and utilize them in III, IV and V.
Activity-Assessment View (III)
Milestone: Assess 24 security-related CLASP activities for suitability in IV.
Implementation Costs
Activity Applicability
Risk of Inaction
Activity-Implementation View (IV)
Milestone: Perform subset of 24 security-related CLASP activities selected in III.
Vulnerability View (V)
Milestone: Integrate solutions to problem types into III and IV.
Consequences of unresolved Vulnerabilities
Problem Types 104 problem types are sub-sumed under 5 high-level Categories
Exposure Periods (by SDLC phases)
Risk Assessment
Avoidance & Mitigation Techniques
A & M Periods (by SDLC phases)

CLASP Resources

The CLASP process supports planning, implementing and performing security-related software development activities. The CLASP Resources provide access to artifacts that are especially useful if your project is using tools to help automate CLASP process pieces.

This table lists the name and location of CLASP Resources delivered with CLASP and indicates which CLASP Views they can support:

CLASP Resources	**Location**
• Basic Principles in Application Security (all Views)	Resource A
• Example of Basic Principle: Input Validation (all Views)	Resource B
• Example of Basic-Principle Violation: Penetrate-and-Patch Model (all Views)	Resource C
• Core Security Services (all Views; especially III)	Resource D
• Sample Coding Guideline Worksheets (Views II, III & IV) Note: Each worksheet can be pasted into a MS Word document.	Resource E
• System Assessment Worksheets (Views III & IV) Note: Each worksheet can be pasted into a MS Word document.	Resource F
• Sample Road Map: Legacy Projects (View III)	Resource G1
• Sample Road Map: New-Start Projects (View III)	Resource G2
• Creating the Process Engineering Plan (View III)	Resource H
• Forming the Process Engineering Team (View III)	Resource I
• Glossary of Security Terms (all Views)	Resource J

Vulnerability Use Cases

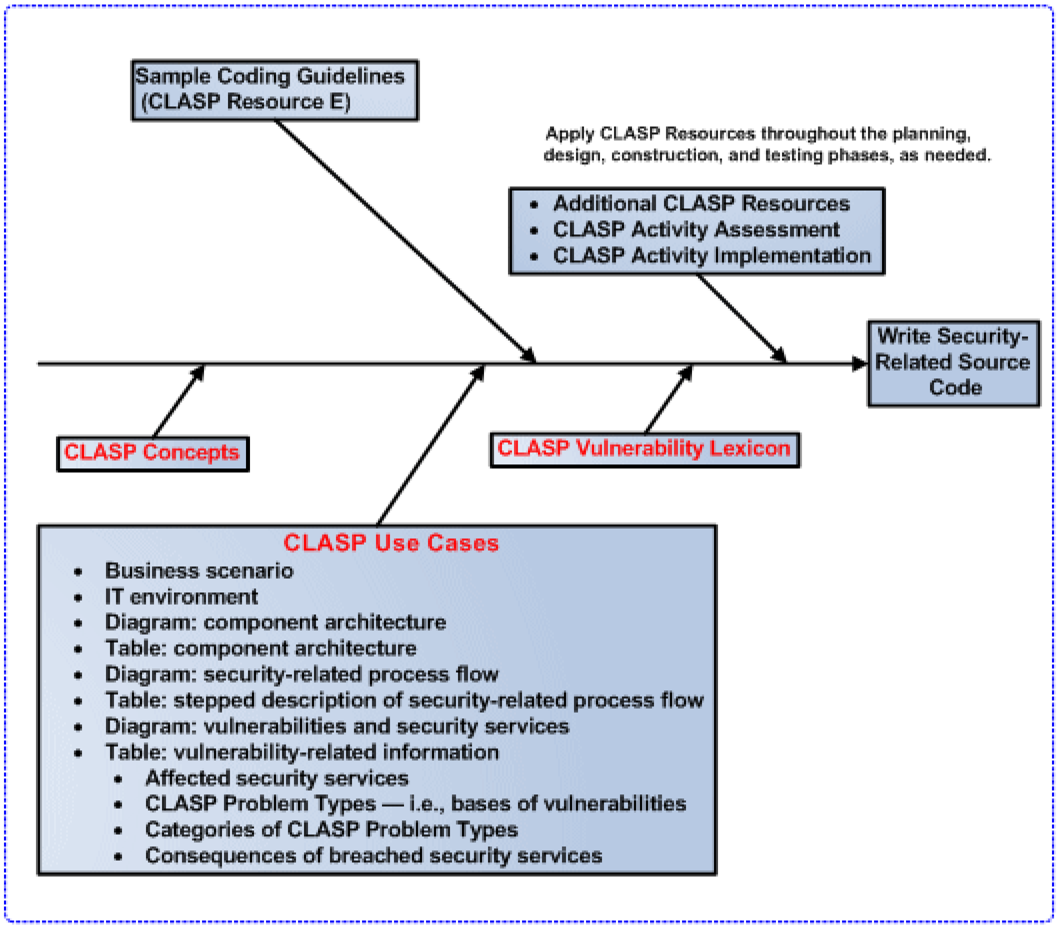
The CLASP Vulnerability Use Cases depict conditions under which security services can become vulnerable in software applications. The Use Cases provide CLASP users with easy-to-understand, specific examples of the cause-and-effect relationship between security-unaware design/source coding and possible resulting vulnerabilities in basic security services — e.g., authentication authorization, confidentiality, availability, accountability, and non-repudiation.

The CLASP Vulnerability Use Cases are based on the following common component architectures:

- Monolithic UNIX
- Monolithic mainframe
- Distributed architecture (HTTP[S] & TCP/IP)

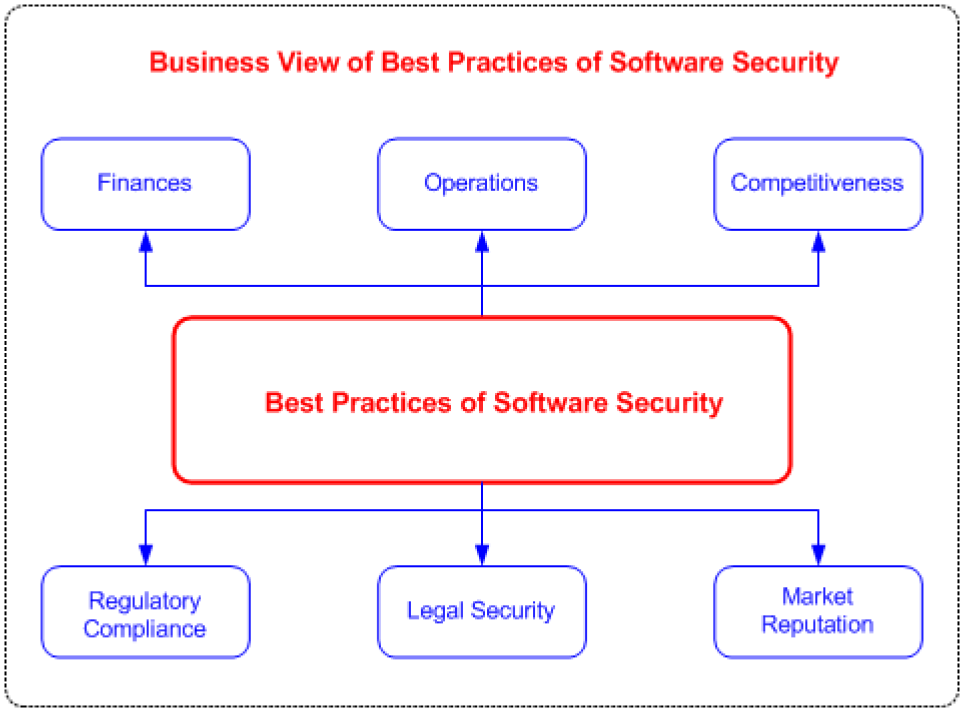
It is recommended to understand the CLASP Use Cases as a bridge from the Concepts View of CLASP to the Vulnerability Lexicon (in the Vulnerability View) since they provide specific examples of security services becoming vulnerable in software applications

The following diagram depicts a recommended position of the Use Cases within the CLASP process:


Sample Coding Guidelines (CLASP Resource E)
Apply CLASP Resources throughout the planning, design, construction, and testing phases, as needed.
• Additional CLASP Resources
• CLASP Activity Assessment
• CLASP Activity Implementation
Write Security-Related Source Code
CLASP Concepts
CLASP Vulnerability Lexicon
CLASP Use Cases
• Business scenario
• IT environment
• Diagram: component architecture
• Table: component architecture
• Diagram: security-related process flow
• Table: stepped description of security-related process flow
• Diagram: vulnerabilities and security services
• Table: vulnerability-related information
• Affected security services
• CLASP Problem Types — i.e., bases of vulnerabilities
• Categories of CLASP Problem Types
• Consequences of breached security services

CLASP Best Practices

If security vulnerabilities built into your applications' source code survive into production, they can become corporate liabilities with broad and severe business impact on your organization. In view of the consequences of exploited security vulnerabilities, there is no reasonable alternative to using best practices of application security as early as possible in — and throughout — your software development lifecycle.


Business View of Best Practices of Software Security
Finances
Operations
Competitiveness
Best Practices of Software Security
Regulatory Compliance
Legal Security
Market Reputation

To be effective, best practices of software application security must have a reliable process to guide a development team in creating and deploying a software application that is as resistant as possible to security vulnerabilities.

Within a software development project, the CLASP Best Practices are the basis of all security-related software development activities — whether planning, designing or implementing — including the use of all tools and techniques that support CLASP.

These are the CLASP Best Practices:

- **Institute awareness programs.**
  Essential security concepts and techniques may be foreign to your organization's software developers and others involved in application development and deployment. So it is imperative at the outset to educate everyone involved. It is critical that project managers — as the driving force behind most application development or upgrade projects — consider security to be an important project goal, both through training and accountability. Awareness programs can be readily implemented, using external expert resources, as appropriate, and deliver a high return by helping to ensure that other activities promoting secure software will be implemented effectively.
- **Perform application assessments.**
  While it's true that you cannot test security into an application, application testing and assessments should still be a central component of your overall security strategy. Assessments

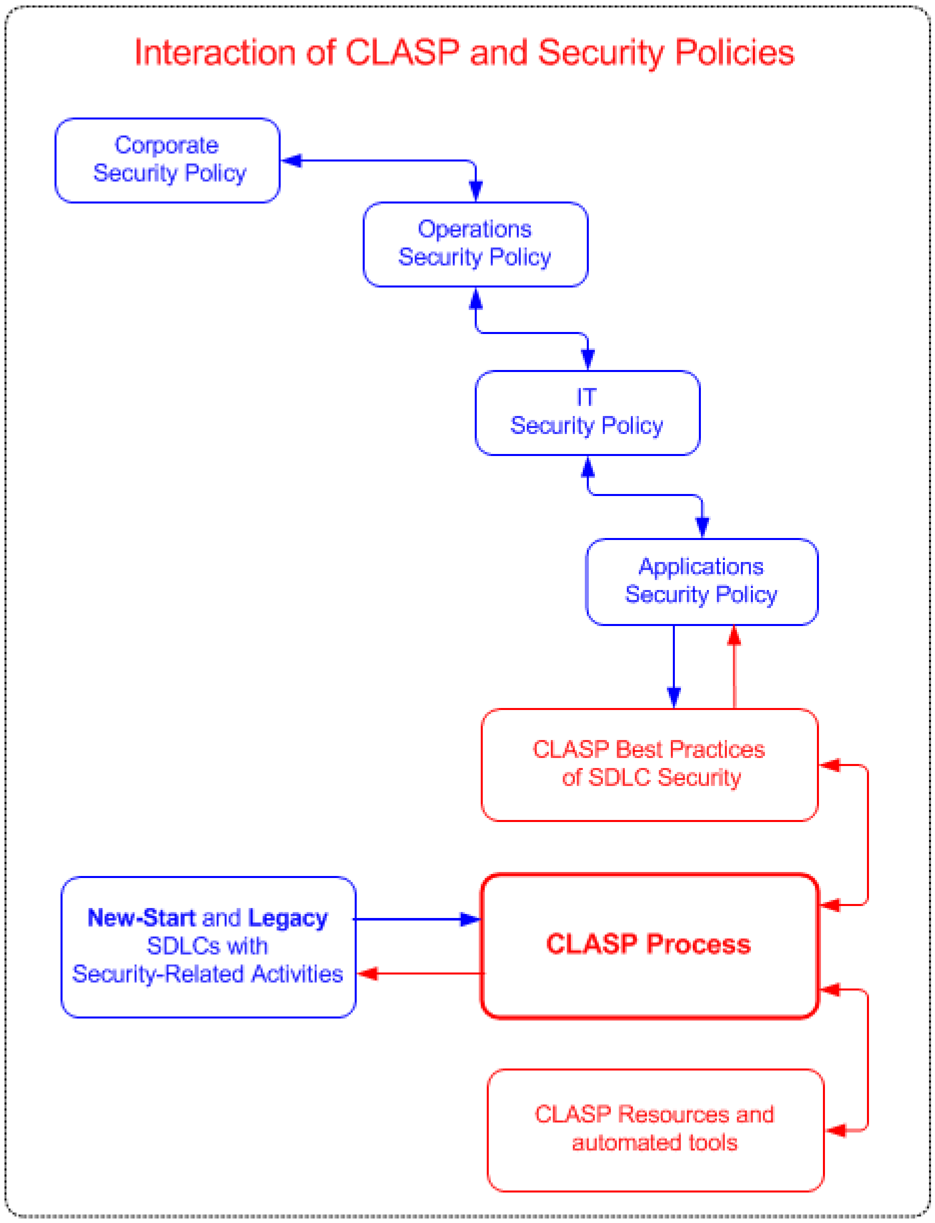
— particularly automated tests — can find security problems not detected during code or implementation reviews, find security risks introduced by the operational environment, and act as a defense-in-depth mechanism by catching failures in design, specification or implementation. Test and assessment functions are typically owned by a test analyst or by the QA organization but can span the entire life cycle.

- **Capture security requirements.**
  Ensure that security requirements have the same level of "citizenship" as all other "must haves." It's easy for application architects and project managers to focus on functionality when defining requirements, since they support the greater purpose of the application to deliver value to the organization. Security considerations can easily go by the wayside. So it is crucial that security requirements be an explicit part of any application development effort. Among the factors to be considered:
  - An understanding of how applications will be used, and how they might be misused or attacked.
  - The assets (data and services) that the application will access or provide, and what level of protection is appropriate given your organization's appetite for risk, regulations you are subject to, and the potential impact on your reputation should an application be exploited.
  - The architecture of the application and probable attack vectors.
  - Potential compensating controls, and their cost and effectiveness.
- **Implement secure development practices.**
  Defined security activities, artifacts, guidelines and continuous reinforcement should become part of your organization's overall culture.
- **Build vulnerability remediation procedures.**
  It is especially important in the context of application updates and enhancements to define which steps will be taken to identify, assess, prioritize and remediate vulnerabilities.
- **Define and monitor metrics.**
  You cannot manage what you cannot measure. Unfortunately, implementing an effective metrics monitoring effort can be a difficult undertaking. Despite this, metrics are an essential element of your overall application security effort. They are crucial in assessing the current security posture of your organization, help focus attention on the most critical vulnerabilities, and reveal how well — or poorly — your investments in improved security are performing.
- **Publish operational security guidelines.**
  Security does not end when an application is completed and deployed in a production environment. Making the most out of existing network and operational security investments requires that you inform and educate those tasked with monitoring and managing the security of running systems with advice and guidance on the security requirements your application demands, and how best to make use of the capabilities you've built into your application.

CLASP and Security Policies

CLASP is a field-proven, comprehensive SDLC process guide that derives from years of cooperation with development teams in resolving security-related issues. CLASP not only implements best practices of application security but also — due to its experience past and present in the field — continually refines application-security best practices.

As a result, a high-level view of CLASP can also help increase awareness of the importance of implementing application security on these organizational levels, from the bottom up: > best practices of application-security > application-security policy > IT security policy > operations security policy > corporate security policy.


Interaction of CLASP and Security Policies
Corporate Security Policy
Operations Security Policy
IT Security Policy
Applications Security Policy
CLASP Best Practices of SDLC Security
New-Start and Legacy SDLCs with Security-Related Activities
CLASP Process
CLASP Resources and automated tools

What is a Security Vulnerability?

CLASP defines a security vulnerability as a flaw in a software environment — especially in an application — that allows an attacker to assume privileges within the user's system, utilize and regulate its operation, compromise the data it contains, and/or assume trust not granted to the attacker.

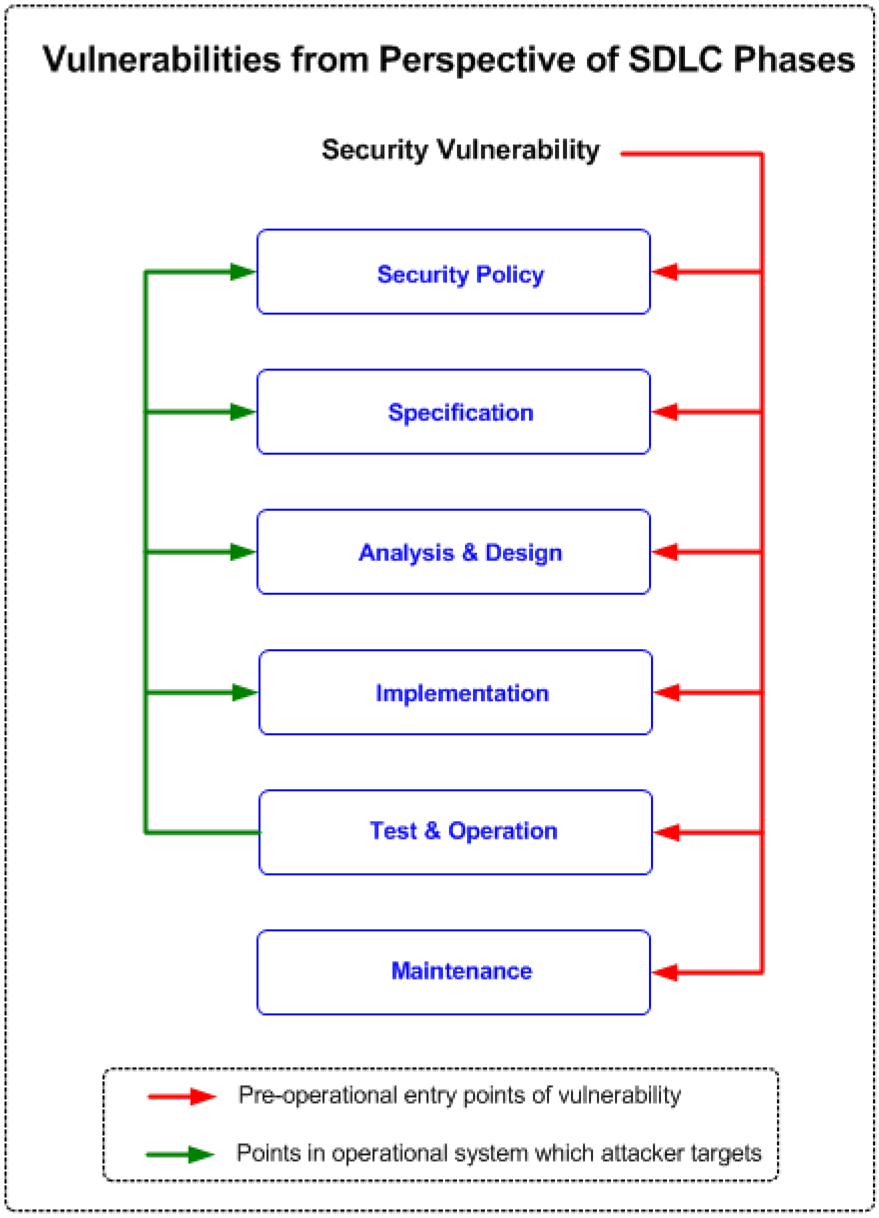
A security vulnerability occurs in a software application when any part of it allows a breach of the security policy governing it.

CLASP identifies 104 underlying problem types that form the basis of security vulnerabilities in application source code. An individual problem type in itself is often not a security vulnerability; frequently it is a combination of problems that create a security condition leading to a vulnerability in the source code. CLASP divides the 104 problem types into 5 high-level categories. Each problem type may have more than one parent category.

CLASP defines a consequence of an exploited or exploitable vulnerability as a failure in one or more of these basic security services:

- Authorization (resource access control)
- Confidentiality (of data or other resources)
- Authentication (identity establishment and integrity)
- Availability (denial of service)
- Accountability
- Non-repudiation

The following figure shows in which phases of the software development lifecycle a security-related vulnerability can occur and also which points in an operational system an attack can target.

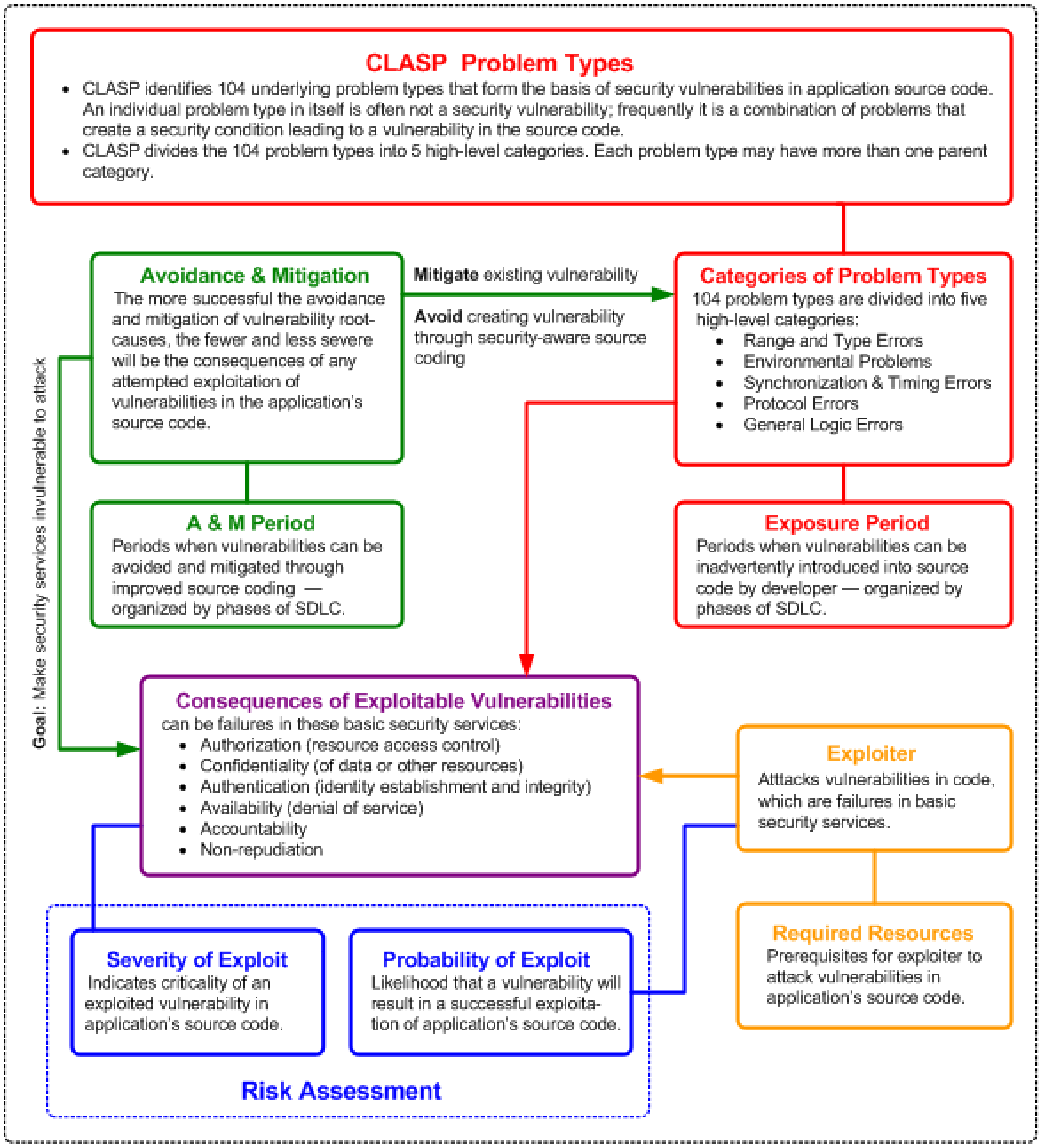

Vulnerabilities from Perspective of SDLC Phases
Security Vulnerability
Security Policy
Specification
Analysis & Design
Implementation
Test & Operation
Maintenance
Pre-operational entry points of vulnerability
Points in operational system which attacker targets

Overview of CLASP Taxonomy

The CLASP taxonomy is a high-level classification of the CLASP process, divided into the following classes for better evaluation and resolution of security vulnerabilities in source code:

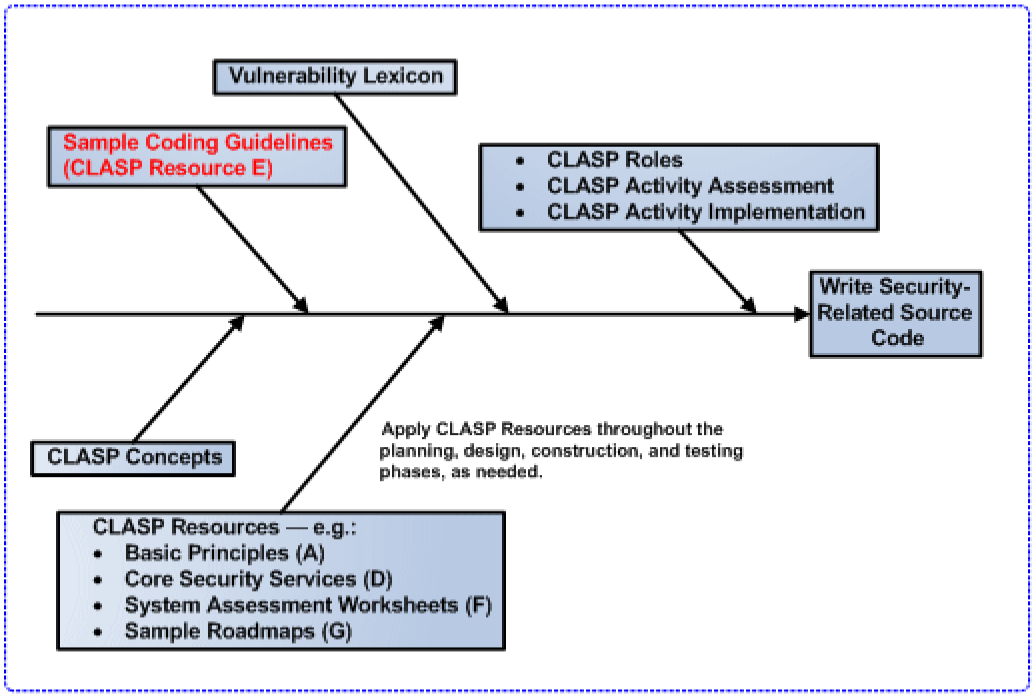
- **Problem types** underlying security-related vulnerabilities.
- **Categories** into which the problem types are divided for diagnostic and resolution purposes.
- **Exposure periods** (i.e., SDLC phases) in which vulnerabilities can be inadvertently introduced into application source code.
- **Consequences** of exploited vulnerabilities for basic security services.
- **Platforms** and programming languages which may be affected by a vulnerability.
- **Resources** required for attack against vulnerabilities.
- **Risk assessment** of exploitable/exploited vulnerabilities.
- **Avoidance and mitigation periods** (i.e., SDLC phases) in which preventative measures and countermeasures can be applied.

The following figure illustrates the CLASP taxonomy and the relationship between its parts:


CLASP Problem Types
CLASP identifies 104 underlying problem types that form the basis of security vulnerabilities in application source code. An individual problem type in itself is often not a security vulnerability; frequently it is a combination of problems that create a security condition leading to a vulnerability in the source code.
CLASP divides the 104 problem types into 5 high-level categories. Each problem type may have more than one parent category.
Avoidance & Mitigation
The more successful the avoidance and mitigation of vulnerability root-causes, the fewer and less severe will be the consequences of any attempted exploitation of vulnerabilities in the application's source code.
Mitigate existing vulnerability
Avoid creating vulnerability through security-aware source coding
Categories of Problem Types
104 problem types are divided into five high-level categories:
Range and Type Errors
Environmental Problems
Synchronization & Timing Errors
Protocol Errors
General Logic Errors
A & M Period
Periods when vulnerabilities can be avoided and mitigated through improved source coding — organized by phases of SDLC.
Exposure Period
Periods when vulnerabilities can be inadvertently introduced into source code by developer — organized by phases of SDLC.
Goal: Make security services invulnerable to attack
Consequences of Exploitable Vulnerabilities
can be failures in these basic security services:
Authorization (resource access control)
Confidentiality (of data or other resources)
Authentication (identity establishment and integrity)
Availability (denial of service)
Accountability
Non-repudiation
Exploiter
Attacks vulnerabilities in code, which are failures in basic security services.
Required Resources
Prerequisites for exploiter to attack vulnerabilities in application's source code.
Severity of Exploit
Indicates criticality of an exploited vulnerability in application's source code.
Probability of Exploit
Likelihood that a vulnerability will result in a successful exploitation of application's source code.
Risk Assessment

Applying CLASP Components

This page describes a possible sequence for applying CLASP components, using the Sample Coding Guidelines (CLASP Resource E) as a basis.


Vulnerability Lexicon
Sample Coding Guidelines (CLASP Resource E)
• CLASP Roles
• CLASP Activity Assessment
• CLASP Activity Implementation
Write Security-Related Source Code
CLASP Concepts
Apply CLASP Resources throughout the planning, design, construction, and testing phases, as needed.
CLASP Resources — e.g.:
• Basic Principles (A)
• Core Security Services (D)
• System Assessment Worksheets (F)
• Sample Roadmaps (G)

The steps below describe a possible sequence for applying CLASP components depicted in the figure above:

- Read the CLASP Concepts View to gain an overview of the CLASP process.
- In the Concepts View, pay special attention to the page Description of CLASP Process. This page contains a diagram showing, among other things, the location of the 104 CLASP problem types (i.e., basic causes of vulnerabilities), the five high-level, source-code-related categories by which they are organized, and the consequences of exploitable security vulnerabilities for security services.
- Read the CLASP Sample Coding Guidelines thoroughly and select a subset of them relevant to your specific software development project. These guidelines contain a set of security-related coding standards to be applied to your project.
- Apply the remaining CLASP Resources throughout the planning, design, construction, and testing process, as needed.
- Use the Sample Coding Guidelines to select a subset of the 104 CLASP problem types (i.e., basic causes of vulnerabilities) — located in the CLASP Vulnerability View — which are most important to your project.
- Familiarize yourself with the CLASP Role-Based View, which provides an overview of the project roles associated with applying the selected subset of Sample Coding Guidelines, and assign

these guidelines to your relevant project personnel — e.g., designer, security auditor, implementers.

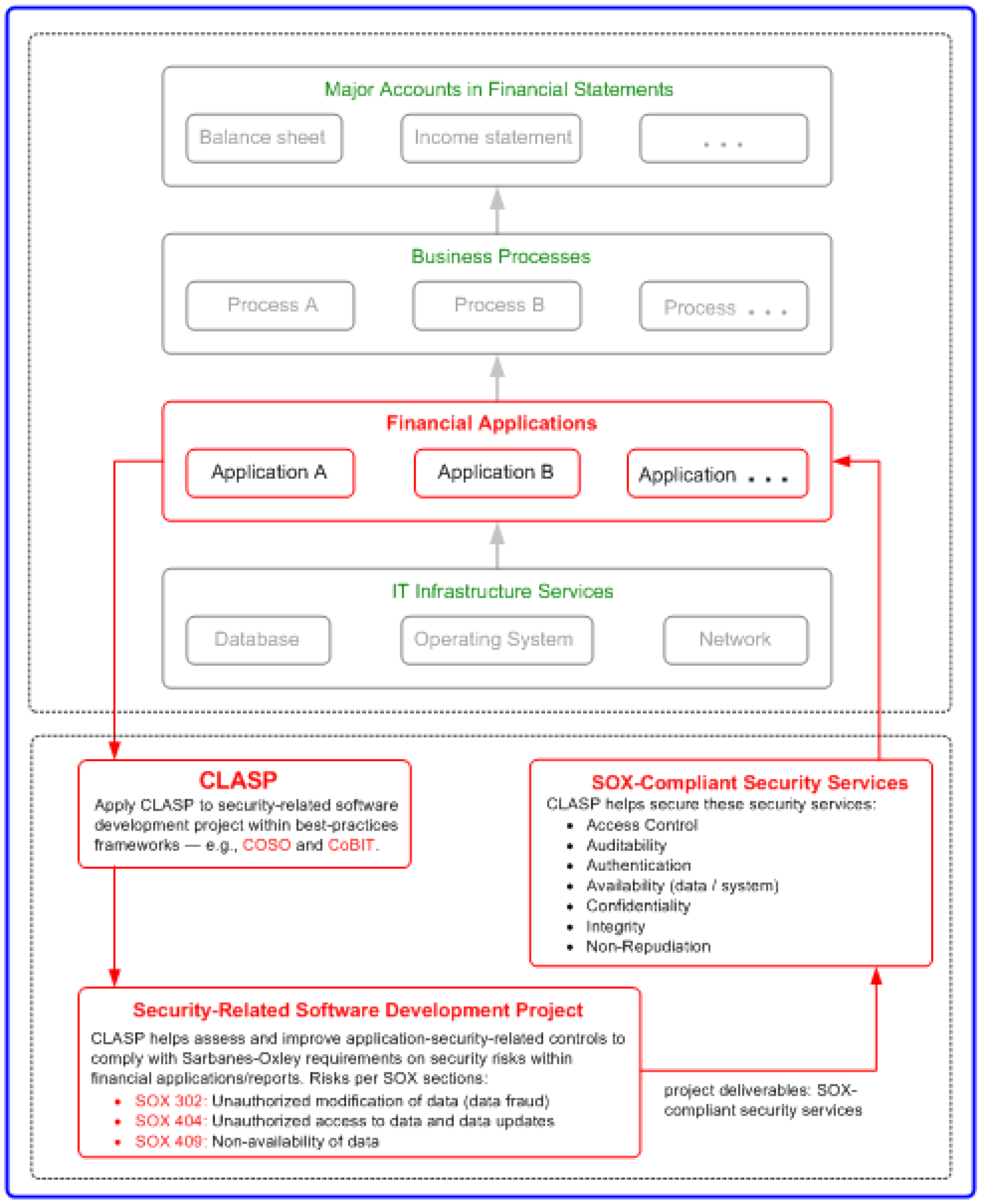
- Consider the subset of vulnerabilities selected in part though the Sample Coding Guidelines when using the Activity-Assessment View to assess and select the desired subset of 24 activities contained in the Activity-Implementation View.

CLASP and IT Internal Controls

A significant number of the internal controls required by the Sarbanes-Oxley Act for assurance of accurate and reliable corporate financial reporting are located in the IT area.

The figure below shows how CLASP can help secure the IT internal controls that are necessary to assure the integrity of data in financial applications within the scope of security-related software development projects.

This figure centers on Sarbanes-Oxley sections 302, 404 and 409:


Major Accounts in Financial Statements
Balance sheet
Income statement
. . .
Business Processes
Process A
Process B
Process . . .
Financial Applications
Application A
Application B
Application . . .
IT Infrastructure Services
Database
Operating System
Network
CLASP
Apply CLASP to security-related software development project within best-practices frameworks — e.g., COSO and CoBIT.
SOX-Compliant Security Services
CLASP helps secure these security services:
Access Control
Auditability
Authentication
Availability (data / system)
Confidentiality
Integrity
Non-Repudiation
Security-Related Software Development Project
CLASP helps assess and improve application-security-related controls to comply with Sarbanes-Oxley requirements on security risks within financial applications/reports. Risks per SOX sections:
SOX 302: Unauthorized modification of data (data fraud)
SOX 404: Unauthorized access to data and data updates
SOX 409: Non-availability of data
project deliverables: SOX-compliant security services