Activity-Assessment View

For organizations that have never formally dealt with software-security issues, the numerous activities defined in CLASP may look quite formidable. Yet there is no need for an organization to implement all of the activities defined by CLASP. It is perfectly reasonable to add activities one at a time, focusing on ones that are the most appropriate and have the most benefit-for-cost.

The purpose of the Activity-Assessment View section is to lessen the burden on a project manager and his process engineering team by giving guidance to help assess the appropriateness of CLASP activities. We do this by providing the following information for each activity:

- *Information on activity applicability*. For example, some activities are only applicable when building applications that will use a back-end database. Other activities are not appropriate for maintaining legacy software that wasn't designed with security in mind.
- *A discussion of risks associated with omitting the activity*. This includes a rating of the overall impact of the activity, relative to other CLASP activities.
- *An indication of implementation cost* — in terms of both the frequency of the activity and the man-hours per iteration. Currently, the man-hour estimates are only rough approximations based on limited experience deploying CLASP and similar activities.We note where an activity will contain steps that are not critical to completing the task but can help provide higher assurance levels. Where appropriate, we discuss the availability of automation technologies for activities that would otherwise be performed manually.

The 24 CLASP activities to be assessed by the project manager and process engineering team are detailed below.

CLASP also has an impact on several key traditional software engineering activities, such as requirements specification. CLASP does not materially change the steps within such activities. Instead, it recommends extensions to common artifacts and provides implementation guidance for security-specific content.

Table: Roles and Related Activities

The following table relates the security-related project roles to the 24 CLASP activities to be assessed.

CLASP Activity	Related Project Role
Institute security awareness program	• Project Manager
Monitor security metrics	• Project Manager
Specify operational environment	• Owner: Requirements Specifier • Key Contributor: Architect
Identify global security policy	• Requirements Specifier
Identify resources and trust boundaries	• Owner: Architect • Key Contributor: Requirements Specifier
Identify user roles and resource capabilities	• Owner: Architect • Key Contributor: Requirements Specifier
Document security-relevant requirements	• Owner: Requirements Specifier • Key Contributor: Architect
Detail misuse cases	• Owner: Requirements Specifier • Key Contributor: Stakeholder
Identify attack surface	• Designer
Apply security principles to design	• Designer
Research and assess security posture of technology solutions	• Owner: Designer • Key Contributor: Component Vendor
Annotate class designs with security properties	• Designer
Specify database security configuration	• Database Designer
Perform security analysis of system requirements and design (threat modeling)	• Security Auditor
Integrate security analysis into source management process	• Integrator
Implement interface contracts	• implementer
Implement and elaborate resource policies and security technologies	• implementer
Address reported security issues	• Owner: Designer • Fault Reporter
Perform source-level security review	• Owner: Security Auditor • Key Contributor: implementer; Designer
Identify, implement and perform security tests	• Test Analyst
Verify security attributes of resources	• Tester
Perform code signing	• Integrator

CLASP Activity	Related Project Role
Build operational security guide	• Owner: Integrator • Key Contributor: Designer; Architect; implementer
Manage security issue disclosure process	• Owner: Project Manager • Key Contributor: Designer

Institute security awareness program

Purpose:	• Ensure project members consider security to be an important project goal through training and accountability. • Ensure project members have enough exposure to security to deal with it effectively.
Owner:	Project Manager
Key contributors:	
Applicability:	All projects
Relative impact:	Very high
Risks in omission:	• Other activities promoting more secure software are less likely to be applied effectively. • Accountability for mistakes is not reasonable.
Activity frequency:	Ongoing
Approximate man hours:	• 160 hours for instituting programs. • 4 hours up-front per person. • 1 hour per month per person for maintenance.

Monitor security metrics

Purpose:	• Gauge the likely security posture of the ongoing development effort. • Enforce accountability for inadequate security.
Owner:	Project Manager
Key contributors:	
Applicability:	All projects
Relative impact:	High
Risks in omission:	No concrete basis for measuring the effectiveness of security efforts.
Activity frequency:	Weekly or monthly.
Approximate man hours:	• 160 hours for instituting programs. • 2 to 4 hours per iteration for manual collection. • 1 with automating tools.

Specify operational environment

Purpose:	Document assumptions and requirements about the operating environment so that the impact on security can be assessed.
Owner:	Requirements Specifier
Key contributors:	Architect
Applicability:	All projects
Relative impact:	Medium
Risks in omission:	• Risks specific to the deployment environment may be overlooked in design. • May not properly communicate to users the design decisions with security impact.
Activity frequency:	Generally, once per iteration.
Approximate man hours:	• 20 man hours in the first iteration. • < 4 hours per iteration in maintenance.

Identify global security policy

Purpose:	• Provide default baseline product-security business requirements. • Provide a means of comparing the security posture of different products across an organization.
Owner:	Requirements Specifier
Key contributors:	
Applicability:	Most appropriate for larger organizations with many developmental efforts that are to be held to the same standard but can easily be effective in any organization developing software.
Relative impact:	Low
Risks in omission:	• Wider organizational security requirements may not be understood — such as compliance to standards. • Difficult to make meaningful comparisons in security posture among projects.
Activity frequency:	Generally, once per project.
Approximate man hours:	• 120 man hours to identify organizational requirements. • 40 hours per project to incorporate requirements.

Identify resources and trust boundaries

Purpose:	Provide a structured foundation for understanding the security requirements of a system.
Owner:	Architect
Key contributors:	Requirements Specifier
Applicability:	All projects
Relative impact:	High
Risks in omission:	• Design process will consider these items intuitively, and overlook important resources. That is, the design process becomes much more *ad hoc*. • Intuitive consideration is still an application of this activity, without the benefit of structure or documentation. Not performing the activity at all leads to inability to perform other CLASP design activities, thereby pushing the cost of initial security assurance to more expensive parts of the lifecycle.
Activity frequency:	Generally, once per iteration.
Approximate man hours:	• Usually 8 hours in the first iteration. • < 3 hours in subsequent iterations.

Identify user roles and resource capabilities

Purpose:	Define system roles and the capabilities/resources that the role can access.
Owner:	Architect
Key contributors:	Requirements Specifier
Applicability:	All projects
Relative impact:	Medium
Risks in omission:	• Access control mechanisms are more likely to be underspecified. • Identified protection mechanisms on resources may not adequately protect all capabilities.
Activity frequency:	Usually, once per iteration.
Approximate man hours:	Dependent on the number of resources, but generally less than 80 hours in the initial iteration; then proportional based on significant changes and additions in each iteration — usually less than 10 hours.

Document security-relevant requirements

Purpose:	Document business-level and functional requirements for security.
Owner:	Requirements Specifier
Key contributors:	Architect
Applicability:	All projects, particularly new application development but also legacy systems.
Relative impact:	Very High
Risks in omission:	• Security services for system resources are extremely likely to be addressed in an ad-hoc manner and have significant gaps as a result.
Activity frequency:	As needed, at least once per iteration.
Approximate man hours:	• If using capabilities, generally up to 120 man hours, depending on the number of capabilities. • If using resources, up to 80 man hours, depending on the level of detail of requirement specification.

Detail misuse cases

Purpose:	• Communicate potential risks to stakeholder. • Communicate rationale for security-relevant decisions to stakeholder.
Owner:	Requirements Specifier
Key contributors:	Stakeholder
Applicability:	Best suited only to organizations that already apply use cases extensively.
Relative impact:	Low
Risks in omission:	Customers will not understand the system security risks and requirements of the project adequately through design and implementation, which can potentially lead to increased security exposure.
Activity frequency:	As required, typically occurring multiple times per iteration and most frequently in Inception and Elaboration iterations.
Approximate man hours:	Generally, one hour per misuse case that is changed per iteration.

Identify attack surface

Purpose:	Specify all entry points to a program in a structured way to facilitate analysis.
Owner:	Designer
Key contributors:	
Applicability:	When exposure metrics are desirable and whenever using structured security analysis such as threat-modeling or source-code review.
Relative impact:	High
Risks in omission:	This is another activity that is often performed implicitly. Failure to document will generally result in an ad-hoc treatment or duplication of work in other activities where the data is needed and can result in a failure to consider important entry points.
Activity frequency:	As needed; usually once after design, and ongoing during elaboration.
Approximate man hours:	• Usually 5 to 20 man-hours in the initial iteration for small-to-medium sized software systems. • Up to 120 man-hours for complex systems containing many off-the-shelf components.

Apply security principles to design

Purpose:	• Harden application design by applying security-design principles. • Determine implementation strategies for security services. • Design secure protocols and APIs.
Owner:	Designer
Key contributors:	
Applicability:	All applications
Relative impact:	High
Risks in omission:	Unanticipated security problems introduced early in design — even if using an extensive set of security requirements.
Activity frequency:	Usually once in the initial iteration, with incremental changes as needed in subsequent iterations.
Approximate man hours:	• In the initial iteration, approximately 40 to 60 man hours for a small project, 80 to 120 for a medium project, and 200 to 300 for a large project. • Generally, no more than 15% of the cost in subsequent iterations.

Research and assess security posture of technology solutions

Purpose:	• Assess security risks in third-party components. • Determine how effectively a technology is likely to alleviate risks. • Identify lingering security risks in chosen security technologies.
Owner:	Designer
Key contributors:	Component Vendor
Applicability:	Any time third-party software is integrated into system development.
Relative impact:	High
Risks in omission:	• Security risks in third-party software can potentially compromise system resources, where compensating controls could have been identified or alternate technologies chosen. • Security flaws not introduced by your development organization can still lead to damage to your brand.
Activity frequency:	As necessary.
Approximate man hours:	Vendor-dependent; from 2 to 40 hours per acquired technology.

Annotate class designs with security properties

Purpose:	Elaborate security policies for individual data fields.
Owner:	Designer
Key contributors:	
Applicability:	Particularly useful in environments using mandatory access control enforcement technologies; is also useful for shops using UML class diagrams.
Relative impact:	Low
Risks in omission:	Implementer error in implementing access control policy.
Activity frequency:	Generally just once; then in iterations where the underlying data design of a class changes.
Approximate man hours:	Generally < 1 man-hour per class initially, with minimal as-needed maintenance.

Specify database security configuration

Purpose:	• Define a secure default configuration for database resources that are deployed as part of an implementation. • Identify a recommended configuration for database resources for databases that are deployed by a third party.
Owner:	Database Designer
Key contributors:	
Applicability:	Whenever a system can make use of a stand-alone relational database, but particularly when the system is to be deployed or managed internal to the developing organization.
Relative impact:	Medium to High
Risks in omission:	Operational security errors in database configuration. This is a very common occurrence.
Activity frequency:	As necessary, generally once per iteration.
Approximate man hours:	• 40 to 80 man-hours depending on the database. • There are existing tools to assist with automating this task.

Perform security analysis of system requirements and design (threat modeling)

Purpose:	• Assess likely system risks timely and cost-effectively by analyzing the requirements and design. • Identify high-level system threats that are not documented in requirements or supplemental documentation. • Identify inadequate or improper security requirements. • Assess the security impact of non-security requirements.
Owner:	Security Auditor
Key contributors:	Architect; Designer
Applicability:	Most applicable before software is implemented, but some sort of architectural analysis is a prerequisite to any effective security analysis.
Relative impact:	Very High
Risks in omission:	• No ability to assess likely level of security risk. • No ability to assess success of secure design efforts.
Activity frequency:	Generally, once after initial design and a significant revisit after implementation, with incremental modifications at regular checkpoints in development.
Approximate man hours:	• 120 hours for the initial model, with approximately 5 man hours per iteration of maintenance. • 40 man-hours for a significant revisit. • Automating technologies exist to support this task.

Integrate security analysis into source management process

Purpose:	Automate implementation-level security analysis and metrics collection.
Owner:	Integrator
Key contributors:	
Applicability:	Whenever using a source-control system and a programming environment supported by automating tools that can act as stand-alones. Automating tools are usually dependent on source languages and OS platform.
Relative impact:	Medium
Risks in omission:	• Regular metrics data will not be collected as specified. • Implementation reviews are more likely to be overlooked or deferred. • Manual labor can have a negative impact on project scheduling.
Activity frequency:	Once per project.
Approximate man hours:	Dependent on the automating technology and the process. Generally, 20 man hours total.

Implement interface contracts

Purpose:	• Provide unit-level semantic input validation. • Identify reliability errors in a structured way at the earliest opportunity
Owner:	Implementer
Key contributors:	
Applicability:	Performable on any well-defined programmer interface. Existing technologies provide slight automation for some OO languages (including Java).
Relative impact:	High
Risks in omission:	Incomplete input validation, particularly for security-critical data.
Activity frequency:	Ongoing throughout implementation.
Approximate man hours:	Generally, 5 minutes per parameter (per function or method), whenever a parameter is changed.

Implement and elaborate resource policies and security technologies

Purpose:	Implement security functionality to specification.
Owner:	Implementer
Key contributors:	
Applicability:	All software
Relative impact:	Very high
Risks in omission:	Arbitrary risk exposure.
Activity frequency:	Ongoing, as necessary.
Approximate man hours:	Widely variable, based on policy and technology.

Address reported security issues

Purpose:	Ensure that identified security risks in an implementation are properly considered.
Owner:	Designer
Key contributors:	Fault Reporter
Applicability:	All software
Relative impact:	High
Risks in omission:	Lack of process behind addressing reported problems often leads to incomplete fixes or introduction of additional security risks.
Activity frequency:	Any time an unanticipated risk is identified in the system.
Approximate man hours:	Generally, 8-16 hours in investigation, plus iteration time on other activities for remediation.

Perform source-level security review

Purpose:	Find security vulnerabilities introduced into implementation.
Owner:	Security Auditor
Key contributors:	Implementer; Designer
Applicability:	All software
Relative impact:	Very High
Risks in omission:	• Security risks introduced in implementation or those missed in design review will not be identified prior to deployment. • Health of secure software development effort can not be measured adequately, thereby leading to a lack of individual accountability.
Activity frequency:	Either on a regular (weekly or monthly) basis or on candidate-release builds.
Approximate man hours:	• Per man-hour, an auditor can generally review 100 to 400 lines of code. • Automating technologies exist that can reduce the cost to about one man-hour per 10,000 lines of code.

Identify, implement, and perform security tests

Purpose:	• Find security problems not detected by implementation review. • Find security risks introduced by the operational environment. • Act as a defense-in-depth mechanism, catching failures in design, specification, or implementation.
Owner:	Test Analyst
Key contributors:	
Applicability:	All development efforts.
Relative impact:	Medium for full-lifecycle CLASP implementation; high for other development.
Risks in omission:	Security risks that would have been identified during testing will instead be identified by others during deployment. Some risks might possibly manifest as actual exploitations during deployment.
Activity frequency:	Generally, once per testable requirement, plus ongoing regression testing.
Approximate man hours:	• 1 to 2 man-hours per requirement for test identification. • 2 to 5 man-hours per test identified for implementation. • Thereafter, ongoing costs associated with running the test. • Tools exist to automate parts of this activity.

Verify security attributes of resources

Purpose:	Confirm that software conforms to previously defined security policies.
Owner:	Tester
Key contributors:	
Applicability:	All software
Relative impact:	Medium
Risks in omission:	Configuration of the software's operational environment may leave unanticipated security risks, particularly to attackers with direct access to underlying resources that the software also uses directly — i.e., underlying machine or the network.
Activity frequency:	Once per candidate build.
Approximate man hours:	• 2-4 man hours for small and medium projects. • 10-20 man hours for large projects.

Perform code signing

Purpose:	Provide the stakeholder with a means of validating the origin and integrity of the software.
Owner:	Integrator
Key contributors:	
Applicability:	Particularly when software is being distributed via an untrusted medium — such as HTTP.
Relative impact:	Low
Risks in omission:	Customers receive a distribution of software that is illegitimate and includes malware.
Activity frequency:	Once per release build.
Approximate man hours:	• 4 man hours for credential acquisition. • 1 man hour per use.

Build operational security guide

Purpose:	• Provide stakeholder with documentation on operational security measures that can better secure the product. • Provide documentation for the use of security functionality within the product.
Owner:	Integrator
Key contributors:	Designer; Architect; Implementer
Applicability:	All software
Relative impact:	Medium
Risks in omission:	• Users may fail to install assumed or required compensating control for a known risk. • Users could accidently misconfigure software in a way that thwarts their security goals. • Users may not be exposed to security risks that they should understand, perhaps by right.
Activity frequency:	Ongoing, particularly during design and in preparation for deployment.
Approximate man hours:	40 man hours — in addition to documentation activities driven by other activities.

Manage security issue disclosure process

Purpose:	• Communicate effectively with outside security researchers when security issues are identified in released software, facilitating more effective pre-vention technologies. • Communicate effectively with customers when security issues are identified in released software.
Owner:	Project Manager
Key contributors:	Designer
Applicability:	All software with external exposure.
Relative impact:	Low
Risks in omission:	Security researchers finding problems in your software may damage your brand without adequate warning.
Activity frequency:	As necessary.
Approximate man hours:	Generally, 4 man-hours a week through the life of response.