Secure Software Modeling

Introduction to the CLASP methodology

Matthias Nissen

December 20, 2013

WS 2013/2014

Seminar: Sicherheitsmodellierung

Lecturer: Prof. Dr. Gerd Beuster

Table of Contents

1. Abstract

This seminar paper gives an introduction to the software methodology called CLASP – Comprehensible, Lightweight, Application Security Process – which is designed to cover security concerns in an existing or a new software development process. It gives an overview of how this methodology works and in what ways it can be applied to a specific software project. The focus lays on understanding and comprehending the process so that an insight to the core features of this method can be attained.

CLASP is still relatively new so this paper is an approach to present the methodology on a more understandable level by laying out the complete structure in a fresh and holistic way. The Paper closes with an evaluation of the practicability of this methodology.

2. Motivation

Security is an important part of today's Software Development. Files and business information are sent over public networks and are stored on many devices such as tablets, smartphones and notebooks or on servers which can be accessed from all over the world. None the less there are few widely established methods to implement security in a structured and understandable way. Furthermore there is little awareness of the consequences of fast and incomplete software development which puts security on a low scale and the customers' data at high risk. This paper gives an introduction to an approach called CLASP, which provides a guide for the whole process – from the people involved to common programming mistakes. It is a methodology, developed to give software engineers – especially experts in software security – a tool to introduce security into a new development life cycle or to apply it ad-hoc on an existing process.

3. The CLASP Project

CLASP has been developed within six years by working with several development teams to address security issues. It can be used in nearly any project and aims to focus on security in the early stage of the development process. It is the result of extensive field work by Secure Software inc.[1], a company known for analyzing security flaws in application-code. During the creation the authors used multiple best practices of software security that are documented in various books and included them into CLASP.

The result is a methodology which consists of several pieces which will be presented in short. It is published on the OWASP website (Open Web Application Security Project) under the Creative Common License 3.0. The project has produced a book – *CLASP v1.2* [1], which is the main source of information used in this paper

3.1 Overview

This chapter is dedicated to the question what the CLASP methodology is, and what features it offers. A main emphasis lies on the method's structure. This approach differs from other sources in so far, that it will not just name some features but rather draws a picture of the whole process. It is centered around the appliance of the process by showing how it is organized. This enables the reader to see strengths and weaknesses and apprehend the conclusions made. Later on will be given an overview of the methods features and how it can be applied.

3.2 Definition

The Best way to understand the CLASP project is to start with a simple description:
The CLASP process can be described as a set of resources or documentation that covers various aspects of software security in different sections throughout this documentation. These aspects can be followed if useful or ignored. There is no given sequence or order in which the different sections should be used, only some loosely proposals of how they could.

A more formal definition which gives insight of how he process works is as followed:

"CLASP - Comprehensible, Lightweight, Application Security Process - is an activity-driven, role-based set of process components whose core contains formalized best practices for building security into an existing or new-start software development lifecycles in a structured, repeatable, and measurable way." [1]

[1] Secure Software was acquired in 2007 by Fortify software which in turn was acquired by Hewlett-Packard in 2010.

In other words CLASP provides a huge amount of resources that are centered around a set of activities, whereas each activity is allocated to one specific role (or person). An activity is a task or a sequence of tasks with the aim of executing security measures. These activities can be used iteratively. Best practices are introduced to establish the necessary environment at the workplace to enforce these security measures.

CLASP is meant to be prescriptive and proactive, prescribing the activities that are used in a project. Those activities are centered around seven Best Practices. The CLASP process is meant to cover the whole software lifecycle rather than just the development. By its very open structure it can be adapted on any development process.

3.3 CLASP Structure

One of the keys to understand CLASP is to understand its structure. This has been one of the major challenges in evaluating the CLASP process in this paper. The CLASP-documentation briefly gives an Overview without too detailed explanation about the whole structure. The missing information in the documentation will be provided in the following in order to give a proper, valuable and understandable overview of the process described in this paper.
It is indispensable to clearly see how all the sections fit together in order to properly assess the CLASP process and give an answer to questions addressing comprehensibility, usability, applicability, completeness, etc.
To get started we will begin with a list of core features, which highlights the most important tools that CLASP provides. We will then continue with a detailed description of how the CLASP documentation is structured. Note that the following core features are just an overview; they are not a complete list, but only reflect, what is considered most important. This is helpful to get an overview and to understand what the core features are:

- 7 Best practices
- 24 Security related CLASP Activities
- 7 different abstract role types
- A vulnerability Lexicon with 104 problem types
- A set of 11 Resources to help planning, implementing and performing CLASP Activities
- CLASP Security Services

To break this down in a more structured way and to attain a more fundamental understanding, the following part gives an introduction to the CLASP method by describing how the process is designed. The key features mentioned above can be found on various locations throughout the documentation.
The proceeding is as follows: Firstly will be given an overview of how the CLASP documentation is build up , secondly the parts considered most important are explained in a more detailed way.

3.3.1 The document structure

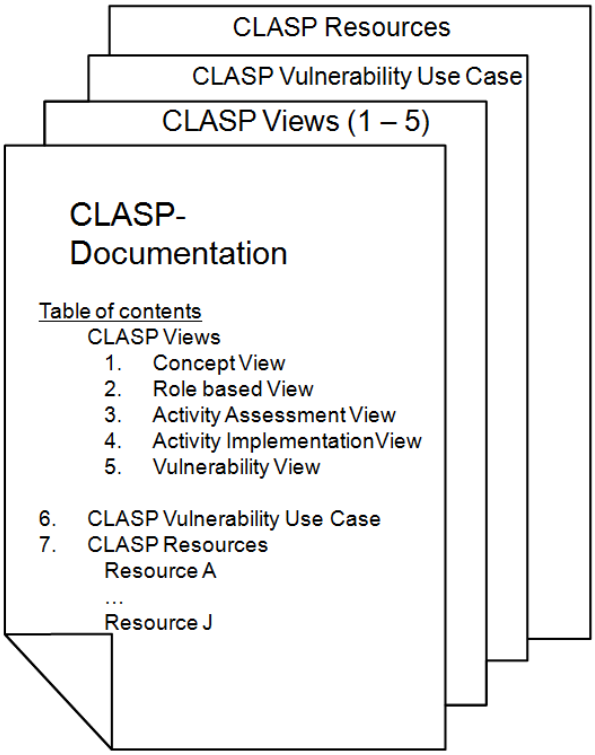

Figure 1 - CLASP document structure

The figure above shows how the documentation is built up. The CLASP documentation lacks a table of contents but it follows a consistent order; the figure is documenting this order and should help understanding the CLASP structure.
As Illustrated in the graphic, the CLASP documentation consists out of seven different sections. To simplify matters, the five sections concerning views are summed up to one section. This results in three main sections:

1. **CLASP Views:** The whole CLASP process is mainly presented in five Views. These Views are projections of the CLASP concept whereas each view represents a certain perspective from the concept. Each CLASP view has a certain purpose and most of them contain specific activities of the process.

   This can easily be illustrated by using a building as an example: The building can be seen as a whole when looking at it from the outside. On the other hand it is also possible to just look at the floor plan, to assess where the rooms and windows are or to look at the electric plan, to evaluate where the wires are.

2. **CLASP Vulnerability Use Case:** There are three very common component architectures based on how data between Clients and servers are processed. Based on these architectures CLASP defines interactions between components and users.

3. **CLASP Resources:** The 11 CLASP Resources can be seen as guidelines to apply basic security principles in a project. They address different goals and give detailed instructions to the most important concepts of the CLASP process.

3.3.2 CLASP Views

In the following the CLASP Views will be described, as they represent the heart of the CLASP process. A short description of the views will be provided and in a further step the first View (Concept View) is laid out, as it describes the underlying concept of the CLASP process.

These 5 CLASP Views are:

- **Concepts View**
  Gives an Introduction to the CLASP methodology.
- **Role-Based View**
  The goal is to understand the roles required in a security related project concerning the project team. There are 7 roles: Project Manager, Requirement Specifier, Architect, Designer, Implementer, Test Analyst and Security Auditor.
- **Activity Assessment View**
  This view provides an overview of the 24 CLASP activities, to assess which are necessary for the project. These activities are the most essential in the CLASP process. They range from “Institute security awareness program” to “Perform Code signing”.
- **Activity-Implementation View**
  This view provides detailed descriptions of the 24 CLASP activities for implementation.
- **Vulnerability View**
  This view is a catalog of 104 problem types also known as security bugs like “Buffer overflow” or “Uncaught exception”. Much of the information listed here can be enforced through the use of automated tools.

The following figure shows the CLASP views and their interactions. Note that each View has a milestone, which describes the goal of the view.

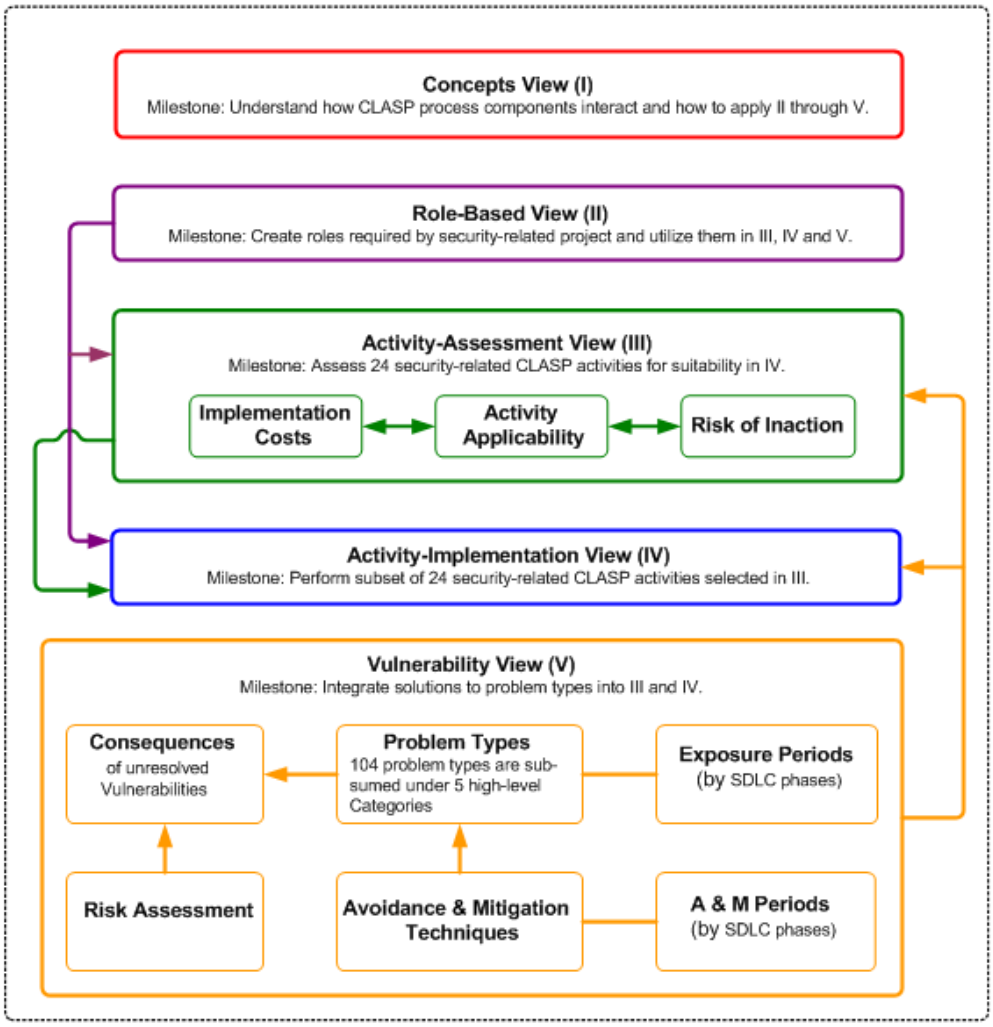

Figure 2 - CLASP Views and their interactions

Figure 2 reveals that the Concept View (I) is independent of the other views, as it only describes the concept behind CLASP. The Role-Based View (II) describes the roles required in the following views. The Activity-Assessment View (III) is intended to assess which of the 24 activities that CLASP provides are important for the project. It therefore gives information on what the activity offers, what it costs, and what the risks of inaction are. The Activity-Implementation View (IV) offers concrete instructions on how to implement the activities and the Vulnerability View (V) mainly provides 104 problem types that may occur in the activities described in view III and IV.

3.3.3 CLASP Concept View

The basic concept behind CLASP is described in the Concept View. Besides the description of the Views, it adds valuable information, providing a set of best practices and other important aspects of secure software. As this View is considered an introduction into the CLASP methodology and therefore covers the most essential topics, each of the 7 key points of this view are summarized in the following:

1. ***Overview of CLASP Process:*** Provides a description of the three main sections (see 3.3.1) in the CLASP Process: CLASP Views, CLASP Resources, Vulnerability Views.

2. ***CLASP Best Practices of Software Security:*** As mentioned earlier, the best practices are essential for the whole development process. They are a guideline for development teams to ensure good software quality. The seven best practices are:

   Institute awareness programs
   In order to enforce security measures, everyone involved in the process needs to be aware of risks and avoidance of his or her action. Therefore essential security concepts should be made known to developers and others involved in the process, especially the project manager.

   Perform application assessments
   Testing is highly important to expose unwanted behavior and security holes.

   Capture security requirements
   Who is using the application? What Resources are used? Security requirements will help planning and focus on the specific characteristics of where and how the software is used.

   Implement secure development practices
   Defined activities, guidelines and continuous reinforcements should be part of the organizations overall culture.

   Build vulnerability remediation procedures
   It is very important to identify mechanisms to assess vulnerabilities and to update the software.

   Define and monitor metrics
   Identify how to measure new security procedures to assess how effective these procedures are.

   Publish operational security guidelines
   Educate those who are managing and monitoring the application.

3. ***CLASP and Security Policies:*** CLASP can help increase awareness of the importance of implementing application security on organizational levels. This section describes the Interaction between CLASP and different levels of the organizations Security Policies.

4. ***What is Security Vulnerability:*** CLASP defines a security-vulnerability as a flaw in a software environment, so that an attacker may be granted unwanted privileges. This can be a consequence of a vulnerability in any 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

These services are highly important and therefore mentioned in the Concept View. One of the CLASP-activities also directly addresses the issue.

5. ***Overview of CLASP Taxonomy:*** Classification of the CLASP process for better evaluation of Security flaws in source code. It is divided into the classes:
   Problem types, problem type categories, exposure periods, consequences of exploited vulnerabilities, platforms and programming languages, resources required for attacking, risk assessment, avoidance and mitigation periods

6. ***Applying CLASP Components:*** This section describes a possible chronological sequence for applying CLASP components as seen below.
   - Read the CLASP Concepts View
   - Read the CLASP Sample Coding Guidelines (Resource E)
   - 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 which are most important to your project.
   - Familiarize yourself with the CLASP Role-Based View and assess and Implement the CLASP activities

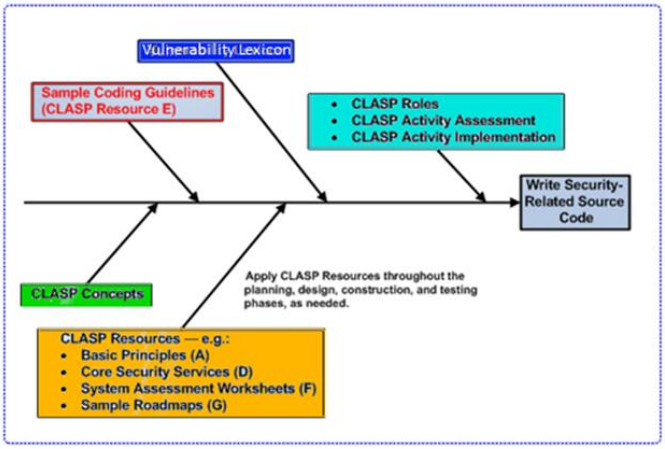

Figure 3 – Applying CLASP Components

7. ***CLASP and IT Internal Controls:*** Describes how CLASP can be used to meet IT internal controls.

3.3.4 Final notes

Many Details have been mentioned only briefly in order to give a short but rather complete overview without going too much into detail. The structure follows the book/documentation order. In contrast, the CLASP online-wiki [7] has a more feature-centered approach.

4. Example

This section covers a sample application to illustrate how the CLASP methodology can be applied. In this sample application, we imagine that clients will connect to a service over the web, whereas the service simply allows users to store information about their contacts. All contact information should be visible to the user only.

Figure 4 – Client – Server Application

There are variety of different options we could start with. We assume that this is a very small project with only one person involved. This simplifies matters. We begin by scanning the CLASP documentation and evaluate what measures address our project:

1. Concept View:
   Most of the resources and descriptions in the concept view address larger projects. However, some of the best practices could be applied e.g. *Capture security requirements*. Also basic security services like *authorization*, *confidentiality*, etc. should be addressed.
2. Role Based View:
   In this scenario only one person is involved. Therefore we will not have to define roles of this view as they are associated with a project team.
3. Activity-Assessment View:
   There are several activities that appear relevant for this project
4. Activity-Implementation View:
   The relevant activities can be applied here.
5. Vulnerability View:
   In the process of testing it might be useful to review possible Vulnerabilities and check if our code contains some common errors.
6. Vulnerability Use Case:
   We might check the use cases for monolithic UNIX which addresses how a terminal(client) connects to a UNIX server.
7. Resources:
   We can select several resources that provide introductions to important CLASP-concepts, for example *Resource A: Basic Principles in Application Security.*

Evaluating all of those measures takes a considerate amount of time. For demonstration purposes we will take a closer look at one activity from the Activity-Implementation-View, to get an impression of how a such a process takes place.

A very useful action in our example is to identify who might have access to the system. This includes the Server, the Internet and the Client. And we need to distinguish between legal and illegal access. The activity that covers this issue is called:

Identify User Roles and resources

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.

Figure 5 - Activity description from the Activity Assessment View

The purpose is to define system roles and the capabilities/resources that the role can access. The time needed is less than 80 hours for the initial iteration. In our one-man-project it will probably be much less.

Role	Generic	Description
User	Yes	These are valid users of the system who have already created an account.
System	No	The application server is represented by a role. This is separate from the "Admin" role, because there may be resources that the system will need to access that the admin should not have to be able to see.
Admin	No	These are users who have administrative access to the system. In this application, their role is restricted to account management, log monitoring and general availability.
Anonymous	No	This role represents people without accounts who may attempt to interact with the system. In this sample application, they have no capabilities, other than being able to sign up for an account.
Attacker	No	Anyone attempting unauthorized access to resources.

Figure 6 – Sample System Roles

We start the activity by enumerating system roles. They represent the user or the software that plays a role in accessing the system. We use a standard set of five common roles (Figure 6).

ID	Description	Owner role(s)	User capabilities
1.	User data	User	Varies (see below)
1.1	Name	User	User (cr) System (r) Admin (d)
1.2	Password	User	User (cw) System (v)
1.3	Contacts	User	User (crwd) System (User proxy)
2.	Compute resources	Varies	Varies
2.1	User CPU	User	User
2.2	User memory	User	User
2.3	User disk space	User	User
2.4	User machine admin info …	User	User
2.5 -	Similar information for middleware and DB servers	Admin	System (User proxy)
3.	Network resources		
3.1	Network between client and app server	Attacker	User, System (rw)
3.2	Network between app server and database (Direct cable connection)	Admin	System (rw, User proxy)
4	Web content (dynamic and static)		User (r) System (cw) Admin (r,w)
4.1	Static web content	Admin	User, Anonymous (r) Admin (r,w)
4.2	Dynamic web content	System	User (r), System (cw)
4.3	Cookies holding authentication information	System	System (crw)
5	Back-end executable content	Admin	User (rx) Admin (rw)
6	Database	Admin	
6.1	Table "user data"	Admin	System (crwd, User

Figure 7 – Example Resource List

A second measure this activity demands is to identify user capabilities in order to set the proper privileges for each role and to discover weaknesses. We will therefore need to identify resources and map the owning roles to the resource.
In our Example these resources include:

- User Information
  (User data, Name, Password, Contacts)
- Computer Resources
  (User CPU, User memory, Server CPU)
- Network Resources
- Web content
  (dynamic and static)
- Backend executable content
- Database

Now we can map the roles to the resources and define User capabilities (Figure 7). The Letters in the brackets represent the capabilities as listed:

r: read
w: write
c: create
d: delete

A third suggestion of the activity is to identify the attacker profile by taking a look at who might have an interest in obtaining data. Based on the profile, countermeasures can be evaluated to address critical spots.
Possible profiles are *competitors* who might want to steal the code, *script kiddies* who look for a challenge, or *activists* who want to expose how easy it is to compromise customer data. Based on those profiles, we can evaluate which attacker-type is probable and initiate customized counter-measures.

The activity is completed at this point. In a next step we could implement another activity, called *Document security-relevant requirements*, which applies basic security principles for each user-resource relation as described in the concept view.

5. Evaluation of the CLASP methodology

The CLASP methodology, which is based on ~ 550 page documentation, is an extensively large collection of useful resources that cover many aspects of security in the Software Development Lifecycle. Of course it cannot cover every possible security vulnerability, but it focuses on the most common mistakes made and thus eliminates the most probable errors on which an attacker often speculates when trying to break into a system. Since it does not offer any priority ordering of activities, nor does it mandate a specific course of action, it can be used in nearly any software project. The liberty of choice is also one of the biggest weaknesses of the process. The time needed to evaluate and implement all the necessary activities is highly dependent on the experience of the person responsible for implementing those security measures. Especially smaller projects or inexperienced agents can be easily overwhelmed by the number of options and by the required expert knowledge, to grasp the range of decisions that are necessary to be made. This could even lead to the ignorance of certain security issues, as they might not appear relevant at first sight.

There is no guideline on which information to focus on and a comparison of different sources [3; 4; 5] that deal with the CLASP project confirmed this problem as they introduce the methodology in different ways. In conclusion CLASP is "Good for experts to use as a guide, but hard for non-security folks to use off the shelf." [2] [5] Taking a step back and looking at the initial goal of the CLASP project to be comprehensive and lightweight, it seems as if this goal was not met by the project involved parties. In addition the specified activities are not exactly consistent. Also the documentation of the process lacks consistency and explanations that are necessary to understand the process. As mentioned earlier, the content is not indexed so it can be difficult and wearisome to evaluate it .

Recently the Project has changed its status to inactive. Therefore further development has come to a stop. Instead a new Project has been developed called OpenSAMM [2]. It focuses more on the specific organization and what security measures are needed rather than providing a general set of tools as in CLASP.

[2] Comment by Pravir Chandra, CLASP Project Lead

6. References

[1]. *CLASP v1.2.* s.l. : OWASP Books, 2007.

[2]. OWASP Software Assurance Maturity Model (openSAMM). [Online] https://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model.

[3]. **Chandra, Pravir.** CLASP PowerPoint presentation. [Online] www.owasp.org/images/5/53/OWASPAppSecEU2006_CLASP_Project.ppt.

[4]. *Build Security In - Official website of the Department of Homeland Security.* [Online] https://buildsecurityin.us-cert.gov/articles/best-practices/requirements-engineering/introduction-to-the-clasp-process.

[5]. **Chandra, Pravir.** Youtube - contains some statements of the CLASP Project Leader about CLASP. [Online] http://www.youtube.com/watch?v=XA-ERJRyOL4.

[6]. **Viega, John.** Building Security Requirements with CLASP. [Online] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.9620&rep=rep1&type=pdf.

[7]. OWASP CLASP Project Wiki. [Online] https://www.owasp.org/index.php/Category:OWASP_CLASP_Project.

[8]. Open Web Application Security Project (OWASP). [Online] https://www.owasp.org.

7. Figures